PONTIFÍCIA UNIVERSIDADE CATÓLICA DO PARANÁ
ESCOLA POLITÉCNICA
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

ANDRÉ PINZ BORGES

UMA CONTRIBUIÇÃO PARA GERAÇÃO DE POLÍTICAS DE AÇÕES PARA CONDUÇÃO DE TRENS DE CARGA USANDO RACIOCÍNIO BASEADO EM CASOS

CURITIBA
2015

ANDRÉ PINZ BORGES

UMA CONTRIBUIÇÃO PARA GERAÇÃO DE POLÍTICAS DE AÇÕES PARA CONDUÇÃO DE TRENS DE CARGA USANDO RACIOCÍNIO BASEADO EM CASOS

Tese de Doutorado apresentado ao Programa de Pós-Graduação em Informática da Pontifícia Universidade Católica do Paraná, como requisito parcial para obtenção do título de Doutor em Informática.

Área de Concentração: Ciência da Computação

ORIENTADOR: Prof. Dr. Edson Emílio Scalabrin

CURITIBA

2015

Dados da Catalogação na Publicação
Pontifícia Universidade Católica do Paraná
Sistema Integrado de Bibliotecas – SIBI/PUCPR
Biblioteca Central

B732c
2015

Borges, André Pinz
Uma contribuição para geração de políticas de ações para condução de trens de carga usando raciocínio baseado em casos / André Pinz Borges ; orientador, Edson Emílio Scalabrin. – 2015.
xiv, 221 f. : il. ; 30 cm

Tese (doutorado) – Pontifícia Universidade Católica do Paraná, Curitiba, 2015
Bibliografia: f. 199-217

1. Inteligência artificial. 2. Raciocínio baseado em casos. 3. Transporte ferroviário de carga – Processamento de dados. 4. Informática. I. Scalabrin, Edson Emílio. II. Pontifícia Universidade Católica do Paraná. Programa de Pós-Graduação em Informática. III. Título.

CDD 20. ed. – 004

Pontifícia Universidade Católica do Paraná
Escola Politécnica
Programa de Pós-Graduação em Informática

ATA DE DEFESA DE TESE DE DOUTORADO
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

ÁREA DE CONCENTRAÇÃO: CIÊNCIA DA COMPUTAÇÃO

DEFESA DE TESE DE DOUTORADO Nº 030/2015

Aos 06 dias de Março de 2015 realizou-se a sessão pública de Defesa da Tese de Doutorado intitulada **"Uma Contribuição Para Geração de Política de Ações para Condução de Trens de Carga Usando Raciocínio Baseado em Casos"** apresentado pelo aluno **André Pinz Borges** como requisito parcial para a obtenção do título de Doutor em Informática, perante uma Banca Examinadora composta pelos seguintes membros:

Prof. Dr. Edson Emílio Scalabrin PUCPR (Orientador)	_____ *(assinatura)*	Aprov *(aprov/reprov.)*
Prof. Dr. Bráulio Coelho Ávila PUCPR	_____	Aprovado
Prof. Dr. Júlio Cesar Nievola PUCPR	Julio Cesar Nievola	APROVADO
Prof. Dr. Richardson Ribeiro UTFPR	_____	Aprov.
Prof. Dr. Mauri Ferrandin UFSC	_____	APROV.

Conforme as normas regimentais do PPGIa e da PUCPR, o trabalho apresentado foi considerado aprovado (*aprovado/reprovado*), segundo avaliação da maioria dos membros desta Banca Examinadora. Este resultado está condicionado ao cumprimento integral das solicitações da Banca Examinadora registradas no Livro de Defesas do programa.

Prof.ª Dr.ª Andreia Malucelli
Coordenadora do Programa de Pós-Graduação em Informática

Programa de Pós-Graduação em Informática PPGIa PUCPR

Dedico aos meus pais,
grandes incentivadores
dos meus estudos.

AGRADECIMENTOS

Primeiramente, a Deus. Graças a Ele pude percorrer todas as etapas desta longa jornada.

Agradeço especialmente aos meus pais que sempre me apoiaram e me proporcionaram condições para que eu pudesse chegar até aqui, me auxiliando de todas as formas e em todos os momentos. Esta tese é dedicada a vocês!

Um agradecimento muito especial ao meu orientador, professor Dr. Edson Emílio Scalabrin, que me acolheu ainda no mestrado e também agora no doutoramento. Graças a ele pude embarcar nesta viagem que iniciou ainda no mestrado em 2007. Aprendi muito com seus ensinamentos, com nossas longas conversas. Muito obrigado por transmitir a mim um pouco do seu vasto e admirável conhecimento. Obrigado também por sua amizade e pela oportunidade de ser seu discípulo, para mim um imensurável privilégio e honra. Espero que esta seja apenas uma parada em uma estação e que possamos continuar a viagem, agora com novas pesquisas. Que esta viagem dure por muitos anos, pois ainda tenho muito o que aprender com o senhor.

Não poderia deixar de agradecer também aos demais professores do grupo de agentes Dr. Bráulio Coelho Ávila e Dr. Fabrício Enembreck, importantes mestres. Ambos sempre estiveram próximos, contribuindo com esta pesquisa, com ideias, críticas, sugestões e valiosos ensinamentos. Meu muito obrigado! Aprendi muito com vocês também.

Quero agradecer também aos meus amigos de laboratório que estiveram comigo durante toda esta viagem: Osmar, Richard, Denise, Allan, Vanderson, Jean, Heitor, Mauri, entre outros "passageiros que entraram e saíram deste trem". Muito obrigado pelas dicas, conversas, sugestões, revisões e ajudas. Dentre eles, quero destacar meu amigo Osmar, que foi não somente um colega de viagem, mas um grande amigo também no âmbito pessoal, que esteve sempre comigo no laboratório por longos anos. Destaco também meu grande amigo Richard que me ajudou muito com valiosos conselhos e ensinamentos no meu mestrado, utilizados ao longo deste doutoramento e valiosos para toda a vida. Meus amigos, todos vocês foram importantes, espero manter este laço de amizade.

Não poderia jamais deixar de agradecer a minha namorada: Carla Cristiane Sokulski. Dedico esta tese a ela também! Juntamente com meus pais, embarcou comigo e sempre me apoiou, incentivou e me deu forças neste 'big bang' de emoções. Obrigado por todos os momentos que você teve que 'me suportar', aguentar minhas crises de humor e sempre me incentivou. Amo muito você!

Quero agradecer à PUCPR, ao Programa de Pós-Graduação em Informática (PPGIa) e à Co-

ordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) pela oportunidade de realização de trabalhos em minha área de pesquisa.

Por fim, quero agradecer a todos os membros da banca avaliadora que contribuíram com ideias, críticas e sugestões para o aprimoramento desta pesquisa.

“Seja você quem for, seja qual for a posição social que você tenha na vida, a mais alta ou a mais baixa, tenha sempre como meta muita força, muita determinação e sempre faça tudo com muito amor e com muita fé em Deus, que um dia você chega lá. De alguma maneira você chega lá.”

(Ayrton Senna da Silva)

RESUMO

Esta tese apresenta um modelo computacional capaz de melhorar as políticas de ações para um domínio bem-definido. Cada política bem-sucedida representa uma experiência. Cada nova experiência aprendida amplia, por meio do seu reuso, as capacidades do aprendiz em resolver novos problemas. Em termos técnicos, cada nova experiência pode ser articulada em uma estrutura particular de memória que facilita o seu reuso. Essa memória pode ser local — individual, colaborativa ou compartilhada, e explorada por meio de mecanismos canônicos de raciocínio baseado em casos; os casos são de um domínio bem-definido. Acerca deste trabalho de pesquisa, o domínio de estudo se encerra em torno do modal férreo, mais precisamente, da geração de políticas ou planos de ações para rebocar trens de carga em vias férreas realistas. Cada plano de ações *P* — ou conjunto ordenado de ações — quando aplicado deve rebocar um trem de um ponto *A* até um ponto *B* de uma via férrea. Todavia, a condução de trens de carga é uma tarefa complexa devido às variações nas formações dos trens, nas condições ambientais e, nos perfis verticais e horizontais das plantas das ferrovias, bem como o grande número de cálculos necessários para determinar cada ação de plano *P*; tal plano deve resultar em uma condução segura, rápida e econômica. Para fazer face a tal complexidade, assume-se que: (i) a aprendizagem baseada em casos é necessária para obter-se planos de ações viáveis e manter-se uma fraca dependência do especialista em condução; (ii) a distribuição do método de resolução de problemas em módulos de software facilita o compartilhamento de conhecimentos e o paralelismo dos cálculos. Essa distribuição estabelece três tarefas/papéis: manter uma memória de casos — *Memorizador*; gerar um plano viável de ações *P* — *Planejador*; e executar o plano de ações *P* — *Executor*. Esse último, embarcado no computador de bordo da locomotiva principal, realiza três diferentes tarefas para cada ação de *P*: (i) testa a aplicabilidade da ação; (ii) ajusta os parâmetros da ação (se necessário), e (iii) aplica efetivamente a ação. Ao final da missão, o plano aplicado com seus ajustes é integrado à base de experiências. A abordagem foi avaliada com diferentes métricas: (i) eficiência das tarefas de recuperação e adaptação dos casos, (ii) similaridades da execução dos planos frente ao planejamento e (iii) redução de consumo de combustível. Os resultados mostraram que a inclusão de novas experiências reduziu os esforços do *Planejador* e do *Executor* em suas tarefas, assim como uma redução no consumo de combustível. Além disso, o modelo permitiu mostrar que o aumento na diversidade na base de casos, aumenta o reuso das experiências em cenários-objetivo — ou diferente daquele que as experiências foram geradas — com características similares com reduzido esforço.

Palavras-chave: Inteligência Artificial, Raciocínio Baseado em Casos e Condução Férrea.

ABSTRACT

This work presents a computational model capable of improving the action policies for a well-defined domain. Each successful policy represents an experience. Each new learned experience expands, through its reuse, learner's capabilities to solve new problems. In technical terms, each new experience can be combined in a particular memory structure that facilitates its reuse. This memory can be local — individual, collaborative or shared, and explored through canonical mechanisms of case-based reasoning; the cases are in a well-defined domain. About this research work, the study area closes around the rail modal, more precisely, the generation of policies or action plans to haul freight trains in realistic railways. Each plan of actions *P* — or ordered set of actions — when applied have to tow a train from point *A* to point *B* in a railroad. However, driving freight trains is a complex task due to variations in formations of trains, environmental conditions, and the vertical and horizontal profiles of the plants of the railways, as well as the large number of calculations required to determine each action plan *P*; such a plan should result in a safe, fast and economical driving. To address this complexity, we assume that: (i) the case-based learning is needed to obtain viable measures and plans to remain a weak dependence on the driving expert; (ii) the distribution of problem-solving method in software modules facilitates the sharing of knowledge and the parallelism of the calculations. This distribution provides three tasks/roles: to maintain a case of memory — *Memorizer*; generate a workable plan of actions *P* — *Planner*; and implement the action plan *P* — *Executor*. This latter, embedded on-board computer of the main locomotive, performs three different tasks for each action of *P*: (i) tests the applicability of the action; (ii) adjusting the action of the parameters (if required), and (iii) effectively applies the action. At the end of the mission, the plan applied to your settings is integrated into the experience base. The approach was evaluated using different metrics: (i) efficiency of recovery tasks and adaptation of cases, (ii) similarities of the implementation of plans to forward planning and (iii) reduction of fuel consumption. The results showed that the inclusion of new experiences reduced efforts *Planner* and *Executor* in their tasks, and a reduction in fuel consumption. In addition, the model show that the increase in diversity in the case base increases the reuse of experiences in objectives-scenarios — or different from that which the experiments were generated — with similar characteristics with reduced effort.

Keywords: Artificial Intelligence, Case–Based Reasoning and Railway Driving.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

Esta pesquisa apresenta uma abordagem que visa reaproveitar e compartilhar políticas de ações definidas e aplicadas em um ambiente dinâmico. O ambiente estudado é o modal férreo. Esse ambiente é dinâmico à medida que pequenas variações nos objetos que o compõe faz com que as tomadas de decisões variem significativamente. Assim, a tarefa é complexa e exige experiência, principalmente quando as variações no ambiente aumentam. Em termos de esforços, a investigação se encerra na concepção e desenvolvimento de um sistema inteligente de condução assistida de trens de carga capaz de conduzir e apreender com as experiências.

A condução assistida de um trem de carga caracteriza-se por usar um programa de computador para auxiliar a tomada de decisão do maquinista ou conduzir o trem de forma automática. A tarefa de condução de um trem é uma atividade intelectual humana que envolve, a concepção dos elementos que compõe o modal, a percepção do estado do ambiente, a atuação e a retroalimentação. Um condutor deve possuir uma concepção dos objetos que compõe o mundo a ser explorado. Assim se a concepção já foi formada, logo é possível que ele perceba o estado atual do mundo, selecione uma ação adequada à condução e aplique-a. O resultado da aplicação é memorizado e utilizado para aprimorar a sua concepção do mundo.

Durante a condução, o maquinista deve estar atento à um grande número de informações do trem e da via férrea: velocidade, velocidade máxima permitida, potência, consumo, aderência das rodas ao trilho, tempo do percurso, eficiência da frenagem e da tração, procedimentos de segurança e sinalização, condições da via, entre outros. Todas estas informações devem ser monitoradas constantemente ao longo da condução, o que exige atenção e perícia do maquinista na execução das suas ações, as quais devem resultar em uma prática eficiente e segura. A eficiência pode ser medida pelo consumo de combustível. A segurança é alcançada quando não são gerados danos à via férrea permanente e ao trem, e são obedecidas irrestritamente as normas de segurança.

A automatização da tarefa de condução assistida pode ser vista como um problema complexo, à medida que há uma grande quantidade de variáveis de controle a serem monitoradas e cálculos a serem efetuados em *tempo real*. Além disso, a seleção da potência a ser empregada para rebocar um trem de forma eficiente e eficaz requer uma iniciativa que vise otimizar o deslocamento do percurso. A escolha desta potência não é uma tarefa trivial, em particular, quando o potencial de força de tração é formado por diferentes locomotivas, que podem ou não estarem distribuídas ao longo do trem. A força resultante é formada pela combinação das potências das locomotivas. A complexidade é elevada

devido a fatores como o volume de cálculos realizados e a dificuldade na previsão das ações face as condições do ambiente do momento e do futuro próximo.

Na condução assistida, antes de selecionar uma potência a aplicar, deve-se conhecer uma série de informações, tais como: velocidade após a aplicação da ação, potência de frenagem, resistência inercial/normal/acidental de todas as locomotivas e vagões, força de tração máxima permitida, percentual de rampa, ângulo de curva, coeficiente de atrito, entre outros valores. Essa série de informações permite calcular e compor a resistência total do trem em oposição ao movimento. O processo para encontrar a potência que supere tal resistência total é normalmente iterativo; não é permitido aplicar uma potência que gere patinagem. Cada potência aplicada resulta em um novo conjunto de informações: força de aceleração, força de tração, consumo, tempo e deslocamento. Todas estas informações, calculadas antes e após a seleção de uma ação, são obtidas várias vezes ao longo da viagem.

Além disso, para uma boa tomada de decisão, deve conhecer o trecho a percorrer além do seu ponto de visão local. Isto permite, de um lado, otimizar o uso da aceleração atual e o peso do trem e, de outro, evitar a falta de potência e consequente parada em determinado local. Uma situação como esta pode resultar em problemas, tais como: (i) aumento no consumo de combustível devido à parada; (ii) congestionamento no trecho de via férrea; (iii) necessidade de alocação de outra locomotiva para rebocar o trem (em caso extremo). Para evitar este tipo de problema, cada maquinista deve conhecer os pontos críticos do trecho de via férrea de sua missão de modo a prever corretamente as ações a empreender face cada situação crítica. Um sistema de condução, no entanto, pode armazenar e utilizar um grande conjunto de informações sobre um trecho de via férrea e sobre ações executadas com sucesso em missões anteriores e reutilizá-las missões futuras.

O modal férreo foi objeto de estudos por vários pesquisadores ao longo dos anos com diferentes abordagens, por exemplo: planejamento de tráfego intermodal (Khattak & Kanafani, 1996); controle da partida, velocidade e local de parada de um trem por meio de regras *fuzzy* (Oshima, Yasunobu, & Sekino, 1988); uso de modelos matemáticos para determinação de uma quantidade fixa de ações para um trecho de via férrea com gradiente continuamente variável (Howlett & Cheng, 1997); modelagem dos elementos percebidos durante a condução em redes Petri, onde os elementos são usados para controlar a movimentação ou não de um trem (Einer, Slovak, & Schnieder, 2000); abordagem multi-agente para reagendar trens em situações de colapso (Abbink, et al., 2010); modelagem de um modal férreo para simulação usando uma arquitetura *BDI* (Bhardwaj, Ghosh, & Dutta, 2013).

Também neste modal, vários trabalhos foram realizados com o objetivo de otimizar a condução de trens, por exemplo: controle da variação de velocidade — usando algoritmo genético (Chang & Sim, 1997); redução do consumo de energia — usando algoritmo genético, onde os indivíduos consistiam de ações (acelerar, desacelerar e frear) que representam estratégias de controle de um trem (Han, et al., 1999); condução confortável, pontual e econômica — algoritmo de evolução diferencial multiobjetivo combinado à lógica *fuzzy* — aplicado à trens de transporte de passageiros (Chang & Xu, 2000); programação dinâmica (Ko, Koseki, & Miyatake, 2004); otimização de potências aplicadas — usando algoritmo genético e redes neurais artificiais (Acikbas & Soylemez, 2008); eficiência energética — usando otimização de restrições distribuída (Leite, Giacomet, & Enembreck, 2009); controle da aceleração e desaceleração como forma de economia de energia (Alves & Pires, 2010); determinação da velocidade para cumprir horários preestabelecidos — usando algoritmo genético (Lechelle & Mouneimne, 2010); redução do consumo de energia e tempo de viagem — multiobjetivo usando pesos na função de fitness e algoritmo genético (Bocharnikov, Tobias, & Roberts, 2010); programação linear inteira (Wang, De Schutter, Ning, Groot, & Van den Boom, 2011); algoritmo para controle de velocidade baseado em seções de bloqueio (Andreotti, Martinis, & Torrieri, 2014).

Uma análise dos trabalhos supracitados mostra uma diversidade nas técnicas utilizadas na criação de sistemas inteligentes. Entretanto, os estudos realizados derivam ações futuras com base em regras ou algoritmos que não consideram as experiências de condução como método de aprendizagem das ações. Basicamente, os sistemas utilizam regras para decidir um comportamento e os algoritmos são aplicados para gerar conjuntos de ações para alcançar o comportamento desejado com base em valores gerados *aleatoriamente*. Tais técnicas oferecem poucas possibilidades de adaptação sem a intervenção do especialista de domínio. Tal dificuldade representa uma limitação importante quando há diversificação dos perfis de vias férreas e dos trens envolvidos.

A abordagem de condução desenvolvida nesta tese considera diretamente as experiências de condutores humanos (adaptando-as se necessário), assim como as experiências de outros sistemas automáticos de condução. Nesta linha, o que nos pareceu mais natural foi fazer uso da abordagem clássica de planejamento baseado em casos (Spalzzi, 2001), fundamentada no raciocínio baseado em casos (Aamodt & Plaza, 1994), cuja ideia principal é que, para solucionar um novo problema, uma entidade (pessoa ou *software*) não precisa criar um plano inteiramente novo passo a passo. Ao invés disso, ela relembra situações passadas similares, e as adapta de acordo com a nova situação. A solução do novo problema amplia a base de experiências. Tal capacidade incremental *online* de aprendizado,

mesmo em ambientes diferentes dos habituais, suaviza e facilita a aquisição e manutenção de conhecimentos. Ao contrário, por exemplo, de sistemas baseados em regras, onde a capacidade de aprendizagem não é inerente e há dificuldade de adaptação às mudanças, devido à necessidade de atualização das regras por especialistas de domínio. Isso torna um sistema baseados em regras complexo e de difícil manutenção (Kolodner, 1993).

O modal férreo, embora pouco utilizado no Brasil, é responsável por ~89% do transporte de cargas em países como a Rússia, ~67% no Canadá, ~48% nos EUA, 21% na China, países onde o uso deste meio de transporte é de fundamental importância para o desenvolvimento (International Energy Agency, 2012). No Brasil este número é de apenas 20% (BRASIL, 2007). A redução nos impactos ambientais também motiva o desenvolvimento de mecanismos que permitam a redução dos poluentes gerados pela emissão de combustíveis fósseis. Para exemplificar, o nível de dióxido de carbono por tonelada transportada (CO_2/TKU) emitido em ferrovias brasileiras corresponde a 34kg a cada 1000t transportada, contra 116kg a cada 1000t transportada do modal rodoviário (BRASIL, 2007) (Junior, et al., 2011). A baixa emissão de poluentes, somada à grande capacidade de carga e espaço para desenvolvimento do setor, motivaram o desenvolvimento desta pesquisa.

Embora o modal férreo seja um dos mais viáveis para o transporte de carga ainda há espaço para redução de custos, o que incentiva a pesquisa e inovação no setor. O emprego de qualquer recurso tecnológico capaz de reduzir, por exemplo, o consumo de combustível pode representar uma diminuição significativa nos custos anuais de operação. Para a maioria das empresas que operam com transporte de cargas ferroviárias, o gasto de óleo diesel corresponde a uma grande parte do orçamento anual. Por exemplo, a América Latina Logística, empresa responsável por transporte férreo de cargas no sul do Brasil gastou, somente no 3º trimestre de 2014, 172.9 milhões de reais em combustível (ALL Logística, 2014). Neste viés, este trabalho contribui com uma aplicação realista na área de condução de trens de carga assistida, onde os resultados em termos econômicos são bastante significativos.

Outro aspecto motivador desta pesquisa é a possibilidade de utilizar condutores experientes como geradores de experiências uteis na elaboração dos planos de condução. Pode-se, por exemplo, submeter maquinistas experientes a simuladores e, a partir dos dados gerados, capturar os traços e reaproveitar dados no formato de casos. Esta possibilidade estende-se também ao uso dos dados de quaisquer outros simuladores de condução. Obviamente, pode ser necessário uma conversão de formato de dados.

Ao longo dos últimos anos, vários estudos foram realizados no Laboratório de Pesquisa em

Agentes de Software do Programa de Pós-Graduação em Informática da PUCPR. O marco inicial destes estudos foi o Projeto PAI-L (**P**iloto **A**utomático **I**nteligente para **L**ocomotivas), onde foram realizados diferentes experimentos seguindo distintas abordagens para o desenvolvimento do piloto automático inteligente de locomotivas: aprendizagem de máquina (Borges, Ribeiro, Ávila, Enembreck, & Scalabrin, 2009) (Borges, et al., 2011), programação por restrição (Leite, Giacomet, & Enembreck, 2009), programação reativa (Dordal, et al., 2011b) (Dordal, et al., 2011a). Estas três pesquisas foram resumidos por Denise M. V. Sato et al (2012) na forma de um relato de lições aprendidas. Outro experimento foi realizado por Marcos R. da Silva (da Silva, et al., 2012), o qual definiu um gerador de planos de condução de trens de alto nível baseado em uma arquitetura BDI (*Believe Desire Intention*). Esses experimentos individualizados, em termos abordagens diferentes, permitiram, de um lado, estabelecer resultados significativos para o domínio de aplicação em questão, e de outro lado, encorajar a realização de outros esforços de pesquisa, seja pela combinação de abordagens já avaliadas individualmente pelo grupo de pesquisa, seja pelo estudo de outras abordagens ainda não avaliadas para o problema em questão.

A forma como o problema de condução de trens foi tratada nesta pesquisa pode ser enquadrada como uma inovação. O dicionário de Oxford define *inovar* como “mudar algo que está estabelecido, introduzindo novos métodos, novas ideias, ou produtos”. Já o Manual de OSLO, da *Organization for Economic Co-operation and Development*, define o termo ‘inovação’ sob o ponto de vista tecnológico como: a implementação de produtos (bens ou serviços) ou processos tecnologicamente novos ou substancialmente aprimorados (OECD, 2005). Neste sentido esta pesquisa propõe uma inovação tecnológica no setor de condução de trens de carga, na medida em que fazemos uso da abordagem de raciocínio baseado em casos para elaborar planos de condução e propomos uma arquitetura software que favorece a distribuição do controle e dos conhecimentos da aplicação.

Destaca-se também algumas contribuições desta pesquisa para a Ciência da Computação:

- Uma arquitetura de software que estrutura o reaproveitamento, a elaboração e o compartilhamento de políticas de ações entre agentes;
- Um estudo do impacto de tal arquitetura para operacionalizar a resolução de um problema complexo;
- Um framework que operacionaliza a arquitetura em questão, facilitando a indexação, a recuperação, o reuso e o armazenamento de experiências.

Esta pesquisa também apresenta contribuições para a área da Engenharia:

- Um estudo da aplicação de técnicas de Inteligência Artificial em um problema realista; e

- Um framework que permite melhorar a condução de trens de carga e, consequentemente, busca evitar problemas durante a condução de veículos de modo a garantir a fluidez do tráfego.

Neste trabalho, o elemento chave proposto foi a concepção de um sistema inteligente capaz de melhorar políticas de ações a partir de experiências na forma de casos. Assim, conjeturou-se que:

- As experiências passadas podem ser úteis na elaboração de políticas de ações em um sistema dinâmico com características ímpares — e.g., modal ferroviário;
- Um condutor inexperiente pode tornar-se mais rapidamente eficiente na sua tarefa reusando experiências oriundas de outros condutores experientes;
- O uso de técnicas de raciocínio baseado em casos na elaboração de políticas de ações pode facilitar a transferência e reuso de experiências, onde o destinatário das mesmas pode ou não usar o mesmo mecanismo de adaptação.

Para corroborar com as hipóteses da pesquisa, definiu-se como objetivo a concepção um modelo computacional capaz de melhorar a política de ações a partir do reuso de experiências. Para a consecução deste objetivo estabeleceu-se os seguintes objetivos específicos:

- Estudar uma forma de representação e raciocínio com baixa dependência do especialista e com capacidade de aprender à medida que novas situações acontecem;
- Definir uma arquitetura distribuída baseada em agentes para resolução de problemas e aprendizagem;
- Realizar um estudo referente ao domínio de aplicação do problema desta pesquisa;
- Definir um modelo matemático para operacionalizar simulações da arquitetura computacional desenvolvida; e
- Avaliar a arquitetura proposta no contexto do problema de condução de trens de carga.

A próxima seção apresenta a organização e o escopo do trabalho desenvolvido.

1.1 ORGANIZAÇÃO DO DOCUMENTO

Este trabalho está organizado da seguinte maneira:

Capítulo 2 – Ferrovias

Neste capítulo são introduzidos os conceitos acerca do modal férreo utilizados ao longo deste trabalho. *De modo resumido:* Uma ferrovia compreende de uma via permanente e estações. Um trem

é formado por uma ou mais locomotivas e um ou mais vagões. Cada locomotiva e vagão apresenta uma série de características que o maquinista deve considerar durante a condução. Algumas formas de sinalização existentes e usadas no gerenciamento da ferrovia. As principais atividades do maquinista durante a condução dos trens: aceleração e frenagem, incluindo os tipos de freios existentes. Para finalizar, o ciclo de condução é apresentado, seguido de um exemplo simplificado dos cálculos realizados para uma condução de 100m. O ciclo é sumarizado na forma de um algoritmo para mostrar a sequência dos cálculos realizados e facilitar a compreensão do leitor.

Capítulo 3 – Fundamentação teórica

Na fundamentação teórica é realizado um breve sobrevoo na área de inteligência artificial e agente inteligente. O interesse sobre agente inteligente limita-se a abstração que ele representa como unidade autônoma e que pode encapsular mecanismos computacionais — algoritmo genético e raciocínio baseado em casos — para a elaboração dos planos de ações. O algoritmo genético foi estudado como mecanismo principal de adaptação de casos. O ciclo canônico do raciocínio baseado em casos requer quatro etapas principais: recuperação, reuso, revisão e retenção. O esforço principal foi dado à etapa de reuso. São apresentados também alguns trabalhos que fizeram uso de tais abordagens.

Capítulo 4 – Metodologia

Inicialmente são apresentadas algumas definições acerca do domínio de aplicação e do método utilizado. Cada uma das etapas do ciclo do raciocínio baseado em casos aplicada nesta pesquisa desenvolvida é apresentada em detalhe, bem como os agentes responsáveis por ordenar o início do planejamento (Despachante), gerenciar o conjunto de experiências (Memorizador), elaborar os planos (Planejador) e executá-los (Executor). O fluxo de dados trocados entre os agentes é discutido, assim como o compartilhamento de planos.

Capítulo 5 – Experimentos e Resultados

Os experimentos e resultados discutidos são precedidos da forma como o modelo matemático utilizado foi validado. Este modelo é utilizado pelos agentes na experimentação de diferentes cenários.

Capítulo 6 – Conclusões e discussões finais

Este capítulo apresenta as considerações sobre o trabalho desenvolvido, os problemas encontrados e os resultados obtidos. Indica-se também quais trabalhos futuros podem ser elaborados para aprimorar este trabalho. São apresentadas também as direções de novas pesquisas.

Apêndice

O apêndice é subdividido em duas partes. A primeira parte contém um comparativo da execução de vários experimentos com o objetivo de analisar o comportamento do algoritmo genético frente ao problema desta pesquisa e algumas variações em alguns de seus parâmetros. A segunda parte é um artigo que publicamos fruto de estudos iniciais realizados ao longo desta pesquisa.

1.2 DELIMITAÇÃO

Este trabalho estuda e define uma arquitetura para elaboração e reuso de planos de ações, cujo domínio de aplicação é a condução assistida de trens de carga. Assume-se que cada locomotiva é equipada com um computador de bordo que recebe informações de um conjunto de sensores instalados ao longo do trem. A descrição da via férrea e do trem envolve uma grande variedade de fatores, no entanto, limitou-se a descrever os conceitos utilizados no processo de condução assistida de trens de carga. Para maior detalhamento acerca dos aspectos técnicos o leitor pode consultar (Hay, 1982) e (Loumiet, Jungbauer, & Abrams, 2005).

Os parâmetros de configuração de cada sensor e do computador de bordo, bem como o gerenciamento da malha não foram objeto de estudos e desenvolvimentos. Assume-se que tais parâmetros são dados. Durante a condução em um cenário realista podem haver obstáculos visíveis apenas para o maquinista e não percebidos pelos sensores do trem. Estes obstáculos podem requisitar ações emergenciais, as quais devem ser tomadas pelo *maquinista*.

2 FERROVIAS

As ferrovias nasceram na Grã-Bretanha. Elas evoluíram das linhas férreas usadas na mineração e pedreiras para meio de transporte. Muito dos principais elementos que compõe o modal férreo, tais como: via férrea, locomotiva, e práticas básicas de operação, foram estabelecidos e desenvolvidos na Grã-Bretanha (Solomon, 2003). As práticas desenvolvidas pelos britânicos foram estudadas e aprimoradas pelos norte-americanos e demais civilizações ao longo dos anos. Além disso, muitas das práticas seguidas atualmente continuam sendo aquelas estabelecidas pelos britânicos.

Uma ferrovia é um sistema de transporte sobre trilhos, constituído de vias férreas e outras instalações fixas, material rodante, equipamento de tráfego e demais equipamentos necessários para uma condução segura e eficiente de passageiros e cargas (DNIT, 2014). A ferrovia possui uma direção obrigatória determinada pela posição dos trilhos. A ferrovia é unidirecional à medida que a trajetória dos veículos é única e coincidente com o eixo longitudinal da via.

Ao longo deste capítulo serão abordados itens relativos a um sistema de transporte ferroviário importantes para a compreensão para o trabalho de pesquisa em questão, que são:

(a) elementos estruturais essenciais de uma ferrovia;
(b) funcionamento básico de locomotivas e vagões;
(c) conceitos fundamentais referentes ao gerenciamento de uma ferrovia; e
(d) condução de um trem, onde mostrar-se-á, por meio de um exemplo simples, a aplicação das equações de resistência, esforço, deslocamento, tempo e consumo.

2.1 ELEMENTOS ESTRUTURAIS DE UMA FERROVIA

Uma ferrovia é composta por vários elementos estruturais, tais como: trilhos, dormentes e estações (Chandra & Agarwal, 2007). Estes elementos são utilizados, de um lado, para dar suporte à movimentação dos trens, e de outro lado, para viabilizar o gerenciamento dos trens que trafegam na malha férrea. De forma analítica, a compreensão dos elementos essenciais da estrutura de uma ferrovia começa pela apresentação da via permanente.

2.1.1 *Via férrea*

A via férrea é o meio físico sobre o qual um trem é rebocado. Ela é, basicamente, composta por dois trilhos fixados em paralelo sobre dormentes com uma distância definida entre eles (cf. Figura

1). Cada componente da via férrea possui uma função específica. Os trilhos atuam como suporte para transmitir aos dormentes a carga sob as rodas do trem. Os dormentes e os fixadores mantêm os trilhos em paralelo e em uma distância correta; eles transferem a carga aos lastros. O sublastro absorve a carga total da via e a carga do trem em reboque (Chandra & Agarwal, 2007) (Kutz, 2011).

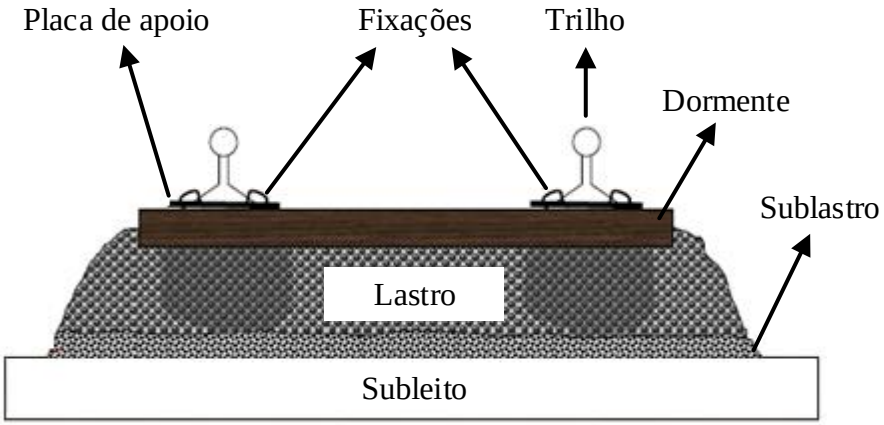


Figura 1 – Elementos estruturais de vias férreas.

A distância entre os trilhos é uma característica intrínseca denominada *bitola*; ela define a largura do veículo que poderá ser rebocado sobre os mesmos. A *bitola* é dada em metros e possui, normalmente, as seguintes medidas: 1.0m, 1.435m e 1.6m. No Brasil é utilizada a bitola de 1.0m na maioria das ferrovias. Uma via pode acomodar mais de uma *bitola*. Esta variação ocorre em função dos padrões utilizados em cada país, da organização e da velocidade da via férrea, uma vez que em vias de trens de maior velocidade a bitola é maior (Chandra & Agarwal, 2007).

A organização de uma via permanente (cf. Figura 2), inclui uma linha principal, assim como diferentes linhas secundárias, desvios, aparelhos de mudança de via e de passagem de nível. A linha principal atravessa os pátios, liga as estações e é por onde os trens são rebocados. As linhas secundárias são utilizadas para manutenção da linha principal; elas possuem uma intensidade de tráfego menor (Loumiet, Jungbauer, & Abrams, 2005). Um desvio é uma linha adjacente à linha principal. Ele é destinado às manobras de formação, ultrapassagem ou cruzamento de trens (Chandra & Agarwal, 2007). A mudança de um trem da linha principal para uma linha adjacente, ou vice-versa, é auxiliada por dispositivos instalados nos trilhos, denominados **A**parelhos de **M**udanças de **V**ia (AMV). Por fim, o cruzamento de uma via férrea com uma rodovia no mesmo nível é chamado de **P**assagem de **N**ível (PN).

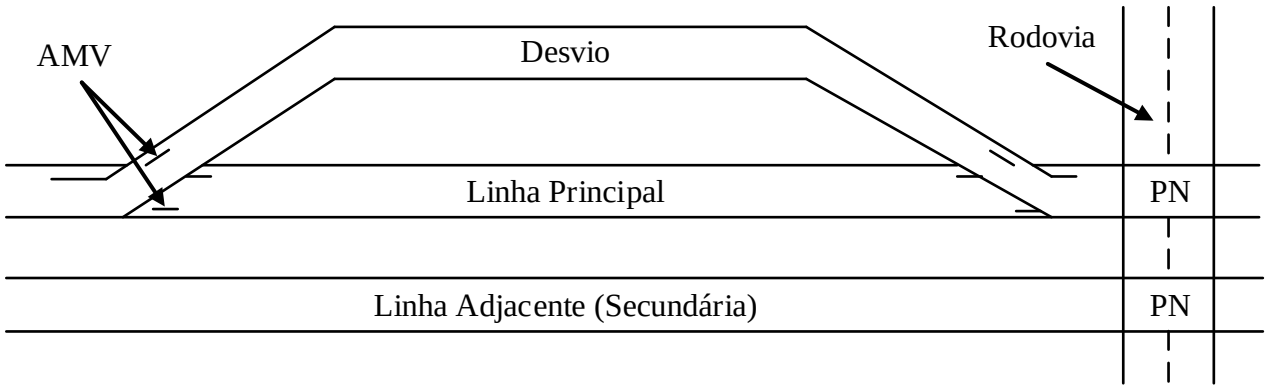


Figura 2 – Elementos de uma via permanente.

Uma via férrea requer diferentes projetos de engenharia. Cada projeto segue uma ou mais norma(s) bem definida(s). A norma brasileira NBR 13133/94 específica às condições necessárias à consecução do levantamento topográfico de uma via férrea. Adicionalmente, a **I**nstrução de **S**erviços **F**erroviários (ISF) 209, feita pelo **D**epartamento **N**acional de **I**nfraestrutura de **T**ransportes (DNIT, 2014) define e especifica os serviços que devem constar no projeto geométrico de engenharia ferroviária. Nestes termos, projeto geométrico de uma ferrovia é dividido em duas fases: projeto básico e projeto executivo (DNIT, 2014).

O projeto geométrico de uma ferrovia, de acordo com a ISF 203, faz parte dos estudos topográficos para os projetos básicos. Ele é formado pelo projeto em planta e pelo projeto em perfil. O primeiro contém as informações relativas às curvas horizontais pontuadas de 20m em 20m entre os pontos inferidos. O segundo contém as informações das curvas verticais pontuadas a cada metro. Tais informações são obtidas a partir do projeto planialtimétrico, que representam as informações planimétricas e altimétricas em uma única planta. A planimetria permite representar os acidentes geográficos (naturais ou artificiais) do terreno em função de suas coordenadas planas (x, y), *i.e.* as curvas horizontais e tangentes. A altimetria fornece outra coordenada (z) de pontos isolados do terreno ou de planos horizontais de interseção com o terreno (curvas de nível), sendo possível, por meio dele, obter as informações das curvas verticais (DNIT, 2014).

A partir das informações sobre a concordância horizontal do projeto é possível determinar, com base no centro da curva (O), os elementos que compõe cada *ponto*, ilustrado na Figura 3 (Kutz, 2011):

- **D**esenvolvimento (D) é o comprimento da curva entre o **P**onto de **C**urva (PC) e **P**onto de **T**angência (PT);
- **R**aio da curva (R) é dado pelo raio arco do círculo empregado na concordância;

- **Â**ngulo **C**entral (AC) é dado pelo ponto de intercepção dos raios que passam pelos pontos PC e PT;
- **G**rau da curva (G) é o ângulo central da curva que compreende de uma **C**orda (C). Normalmente, o comprimento de corda utilizada em ferrovias é de 20m[1].

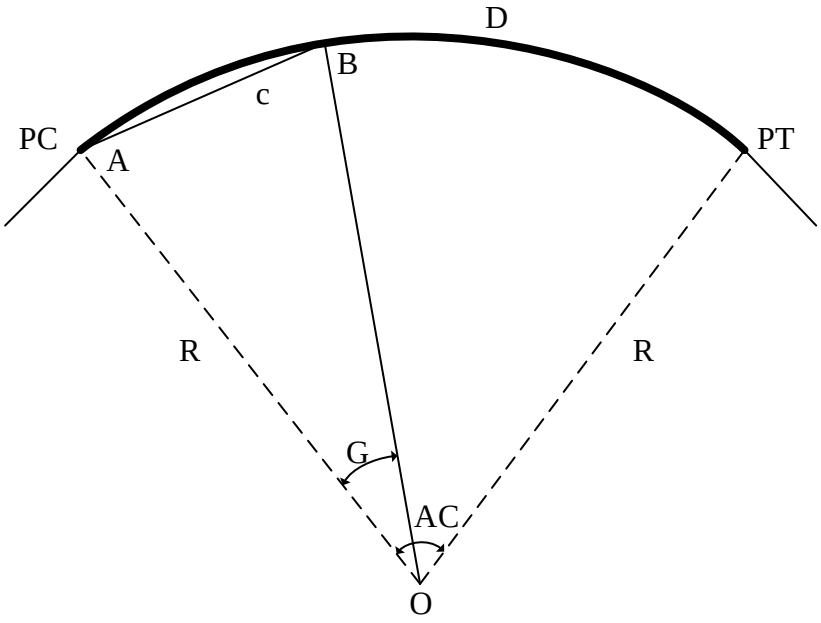


Figura 3 – Elementos da concordância horizontal (Kutz, 2011).

A concordância vertical é mapeada a partir do **P**onto de **C**urva **V**ertical (PCV) no início da curva e no **P**onto de **T**angência **V**ertical (PTV), ao final da curva (cf. Figura 4). Tais medidas permitem obter, entre outras informações, a **i**nclinação vertical (i) do terreno com base no **P**onto de **I**ntersecção **V**ertical (PIV). A inclinação, também chamada de *rampa*, corresponde ao percentual de inclinação do terreno e das eventuais tangentes verticais que o terreno abriga, medida pela diferença algébrica das declividades entre o PCV e PTV (Kutz, 2011).

[1] Por este motivo, esta medida é também chamada de 'grau 20' ou ‘g20’.

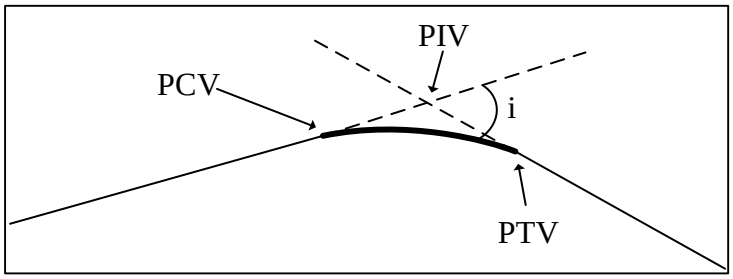


Figura 4 – Exemplo de uma curva vertical (Kutz, 2011).

As concordâncias horizontais e verticais são compostas por outros elementos. Para maiores detalhes o leitor pode consultar as seguintes fontes: (Chandra & Agarwal, 2007) (Loumiet, Jungbauer, & Abrams, 2005).

2.1.2 *Malha férrea*

Uma malha férrea, ou ferrovia, compreende de um sistema de transporte sobre trilhos. Este sistema é constituído de via férrea, instalações físicas, material rodante de tração, equipamentos de tráfego e outros equipamentos/materiais necessários para uma condução segura e eficiente (DNIT, 2014). Dentre as instalações físicas que compõe a malha férrea estão as estações férreas, descritas a seguir.

2.1.3 *Estações férreas*

Uma estação férrea é um local em uma linha férrea onde o tráfego é bloqueado e controlado. Algumas vezes apenas uma destas funções é realizada na estação e, portanto ela é classificada como *estação de bandeira* ou *estação de bloqueio*. Uma estação de bandeira é utilizada como um local de parada para os trens, onde é possível o descanso dos passageiros. Neste tipo de estação, a comunicação e controle limitam-se ao uso de rádios e equipamentos para informar a chegada e partida dos trens. Já uma estação de bloqueio é usada para gerenciamento do tráfego, onde o condutor obtém a permissão para entrar na próxima sessão. O tráfego é gerenciado pela concessão ou não de permissões. Contudo, a maioria das estações executam ambas as funções descritas (Chandra & Agarwal, 2007). A forma como o bloqueio e o gerenciamento dos trens são realizados nas estações cabe ao Centro de Controle de Tráfego, que será visto mais à frente.

Neste trabalho, o nosso interesse está apenas no conceito de estação de bloqueio, doravante denotada apenas como *estação*. Conforme dito, as estações são responsáveis por controlar o tráfego

de veículos como, por exemplo, locomotivas e vagões, descritos a seguir.

2.2 LOCOMOTIVAS E VAGÕES

A maioria das locomotivas em uso é equipada com motor diesel-elétrico ou elétrico, e utilizada para o transporte de cargas, pessoas, fins industriais, entre outros. As locomotivas diesel-elétrica geralmente pesam de 100t a 210t, medem de 16,7m a 22,5m, e possuem motores de 1500HP a 3600HP (em corrente contínua) e 6000HP (em corrente alternada). Os eixos das locomotivas modernas pertencem, normalmente, a classe B-B (dois truques independentes com dois eixos por truque tracionado) ou C-C (dois truques independentes com três eixos por truque tracionado), normalmente medindo de 2,7m a 3,2m de largura. Por exemplo, a locomotiva modelo SD-40-2, modelo similar ao que foi utilizado em nossos experimentos, possui seis eixos, pesa 184t, equipada com um motor elétrico de 3000HP – em corrente contínua (Loumiet, Jungbauer, & Abrams, 2005).

As locomotivas são frequentemente combinadas em múltiplas unidades. Elas são conectadas eletricamente para permitir controlar sincronamente as funções de tração e frenagem a partir de uma posição determinada em uma das locomotivas. Nesta configuração, várias locomotivas acopladas e encabeçando um trem podem operar como uma locomotiva única. Em certas situações, para trens muito longos, uma ou mais locomotivas são distribuídas ao longo do trem para reduzir as forças de acoplamento e melhorar o desempenho de frenagem. Esta configuração é chamada de *força distribuída*. No Brasil, o controle da distribuição de forças é feita por meio de comunicações de rádio. A distribuição da força pode ser realizada por computadores, capazes de controlar múltiplos conjuntos de locomotivas e prover operações com dados e opções de controle para aprimorar a eficiência do uso das locomotivas.

As locomotivas são equipadas, normalmente, com computadores de bordo responsáveis pela leitura de diversos sensores distribuídos ao longo do trem. O conjunto de sensores informa ao condutor a velocidade, a pressão no encanamento de freios, o ponto de aceleração aplicado, a posição da manopla de reverso (responsável por indicar se o trem está se deslocando para frente ou para trás), uso ou não do freio dinâmico, entre outras informações. Cada locomotiva é equipada também com rádio para comunicação com a Central de Controle de Tráfego e com GPS, utilizado para determinar a posição do trem na via. Com base na posição, o computador de bordo pode informar o maquinista a velocidade máxima permitida e o perfil da via alguns metros à frente da posição atual.

Os vagões são veículos utilizados para transporte de cargas que variam de 30t a 125t. Cada

vagão possui uma especificação de carga demarcada nele próprio. Cada demarcação específica a capacidade nominal de carga, a capacidade máxima de carga transportada pelo vagão e o peso do vagão vazio *ou tara*. Além disso, cada vagão deve ter uma placa indicando os ajustes do nível de freio. Esses ajustes, dados por um mecanismo chamado válvula tríplice – acoplada no vagão, permite, basicamente, configurar a distribuição de ar durante a frenagem. O mecanismo de frenagem do vagão é descrito na seção Frenagem.

2.3 GERENCIAMENTO DA FERROVIA

Nos primórdios das ferrovias, os sistemas de sinalização e gerenciamento não existiam. Com a prática, percebeu-se que alguma forma de controle de segurança deveria existir nas ferrovias, à medida que o campo visão do maquinista não era mais suficiente para fazer face as crescentes velocidades desenvolvidas pelos trens mais modernos, i.e., o risco eminente devido à grande distância necessária para que um trem efetuasse a parada por completo impôs novas práticas de sinalização e gerenciamento.

O primeiro sistema de sinalização usado foi uma “janela” de tempo, em inglês "headway". Por exemplo, só era permitido a um trem andar na velocidade máxima quando se passasse 10 minutos da saída do último trem. No decorrer do trecho, um operador com bandeiras verde, amarela ou vermelha informava o condutor a situação – vermelha tempo ≤ 5 min, amarela tempo > 5 min e tempo ≤ 10 min, e verde tempo > 10 min.

Com esse tipo de sinalização ainda era frequente a ocorrência de acidentes, à medida que o trem anterior poderia ter passado a mais de 10 minutos, mas poderia estar parado logo a frente por alguma razão e, com bandeira verde o condutor do trem que saiu depois estaria em velocidade máxima. Dessa forma, uma parada completa do trem seria praticamente impossível, levando em conta a precariedade dos freios dos trens no final do século XVIII (Solomon, 2003).

2.3.1 *Sinalização*

A primeira sinalização utilizada foi chamada de *sinalização fixa* que era disposta em postes localizados as margens das ferrovias. Mesmo com a utilização de sistemas modernos de controle de tráfego, as sinalizações fixas ainda são usadas em larga escala nas ferrovias mundiais.

Dentre os vários tipos de sinalização utilizados nas ferrovias serão abordados neste estudo apenas luminosos. Algumas ferrovias utilizam estas sinalizações combinando-as de formas distintas.

Estas combinações são conhecidas como múltiplos aspectos de sinais coloridos de luz, em inglês *Multiple Aspects Color Light Signal* (MACLS), e utilizam a sinalização luminosa nas cores: verde, amarela e vermelha. O *MACLS* possui inúmeras formas de sinalização, dentre elas: sinalização em dois aspectos (cf. Tabela 1) e sinalização em três aspectos (cf. Tabela 2) (NSW, 2013), (Goel, 2010), (Palumbo, 2013).

Tabela 1 – Sinalização com dois aspectos.

ASPECTOS	SIGNIFICADO	INDICAÇÃO PARA O CONDUTOR
Vermelho	Pare	Parar absolutamente
Amarelo	Cautela	Prossiga e se prepare para no próximo sinal de parada
Verde	Prossiga	Prossiga

Tabela 2 – Sinalização com três aspectos.

ASPECTOS	SIGNIFICADO	INDICAÇÃO PARA O CONDUTOR
Vermelho	Pare	Parar absolutamente
Amarelo	Cautela	Prossiga e se prepare para no próximo sinal de parada
Duplo Amarelo	Atenção	Prossiga e se prepare para passar pelo próximo sinal com velocidade restrita
Verde	Prossiga	Prossiga

O aspecto *duplo amarelo* vem com a informação a qual velocidade o condutor deve prosseguir. A informação é passada diretamente para a cabine do condutor por meio do *sistema direto*, em inglês *on-cabe signal* (Solomon, 2003).

2.3.1.1 Comunicação direta

Este sistema é responsável pelo envio de informações da **C**entral de **C**ontrole **O**peracional (CCO) de forma direta para a cabine da locomotiva. Esta comunicação acontece quando a CCO deseja enviar informações de velocidade restrita e possíveis situações de risco na ferrovia. Geralmente, a utilização deste sistema é realizada após o trem entrar em uma região da ferrovia em que a sua velocidade deva ser mais restritiva. A informação por esse sistema pode ser enviada em forma de comando de voz (e.g., rádio) ou sinais luminosos com informações de restrição no próprio painel da locomotiva. Isto é feito para que o trem que está mais à frente possa liberar o próximo trecho. Porém, esta forma de comunicação pode falhar quando o condutor não obtém as informações de restrição da CCO; em situações como esta ele deve conduzir o trem na velocidade mais restritiva para o trecho (Solomon, 2003).

Tanto na sinalização luminosa quanto no *sistema direto* verificou-se que as distâncias entre

os trens eram muito grandes e com isso a capacidade da via férrea era subutilizada. Dessa forma, sistemas para gerenciar aproximações de trens foram desenvolvidos, como por exemplo, *Distância a percorrer*, em inglês *distance to-go*.

2.3.1.2 Distância a percorrer

Neste sistema, as marcações na ferrovia são percebidas pelos sensores e a cada marcação que o trem passa um novo código de velocidade é passado para o condutor. A Figura 5 mostra tal controle de velocidade.

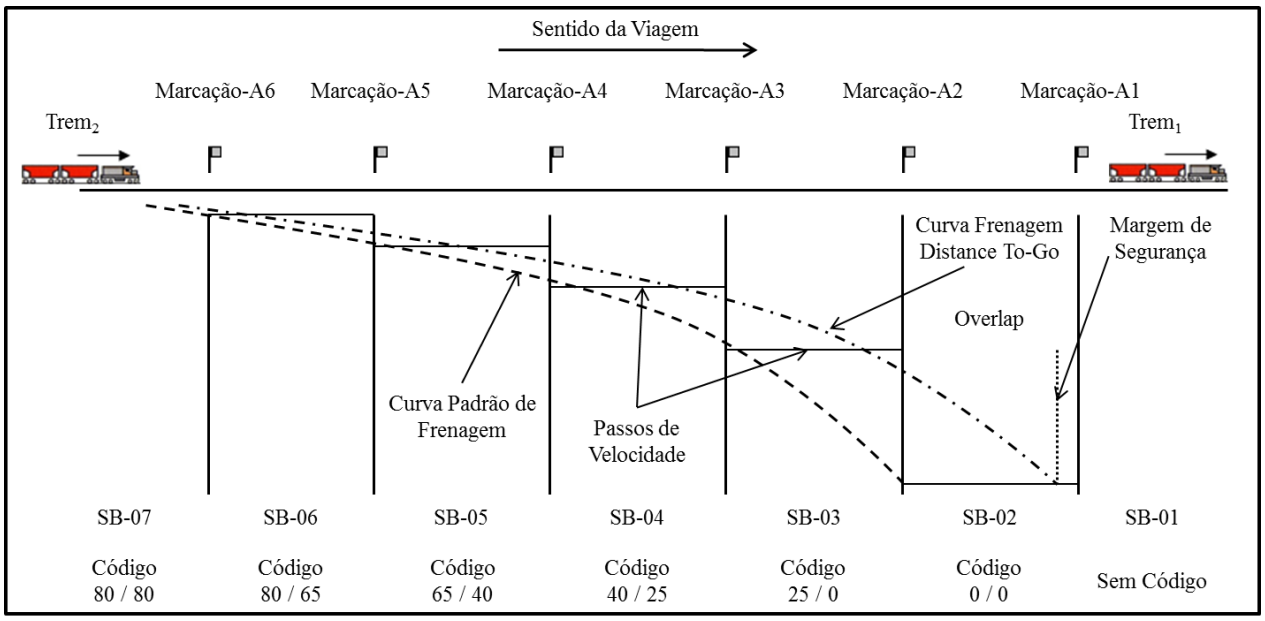


Figura 5 – Sistema de *distância a percorrer*.

Essa figura mostra as marcações com os códigos de entrada/saída que significa a velocidade que o condutor deve entrar na marcação e a velocidade que ele deve sair dela. A velocidade de entrada zero na Marcação-A2 é indicada pela curva de parada, e seria baseada no sistema *Segurança Padrão* que utiliza um bloco de segurança entre os trens, esse espaço de segurança é conhecido como sobreposição (*overlap*), e que geralmente possui a extensão de 183m (Davies, 2000). Já a outra curva de parada próximo ao local da Marcação-A1 é a curva de controle baseada no sistema *Distância a Percorrer*. Aqui, o sistema indica parada no bloco anterior a outro ocupado por um trem, mantendo sempre uma margem de segurança. Dessa forma, os trens poderiam se aproximar mais, reduzindo a su-

butilização da ferrovia. A divisão da ferrovia em blocos é uma prática utilizada para manter a segurança da ferrovia. Esses blocos são conhecidos como *seções de bloqueio* (Xi-Shi & Yong, 1999).

2.3.2 *Seções de bloqueio*

A seção de bloqueio ou bloco fixo foi uma das primeiras formas de evitar colisões entre os trens. Ela é um trecho de ferrovia com tamanho fixo. A função desta seção é impedir que trens diferentes ocupem o mesmo espaço ou espaços vizinhos ao mesmo tempo em uma ferrovia. Um trem, ao trafegar em uma seção de bloqueio, faz com que seja bloqueada a seção onde ele trafega, juntamente com a seção anterior e a seção posterior à seção atual. Este bloqueio é realizado por semáforos instalados ao longo da via férrea. O controle deste bloqueio pode ser automático por sensores na via ou pela central de controle. A Figura 6 mostra a utilização de seções de bloqueio e as sinalizações fixas (Solomon, 2003), (Bajpai, Karad, Pawar, & Pilakar, 2007).

Deve-se notar que após um trem passar por uma sinalização, ela imediatamente fica vermelha. Isto indica que a seção onde este trem está posicionado não pode, de forma alguma, ser usada por outro trem sob risco de ocorrer algum acidente na via férrea.

As seções de bloqueio tornam mais seguras as ferrovias. Entretanto, a consequência desta segurança adicional impacta negativamente na capacidade da via férrea e na utilização dos recursos. Desta forma, foi desenvolvido o sistema de *Seção de Bloqueio Móvel*.

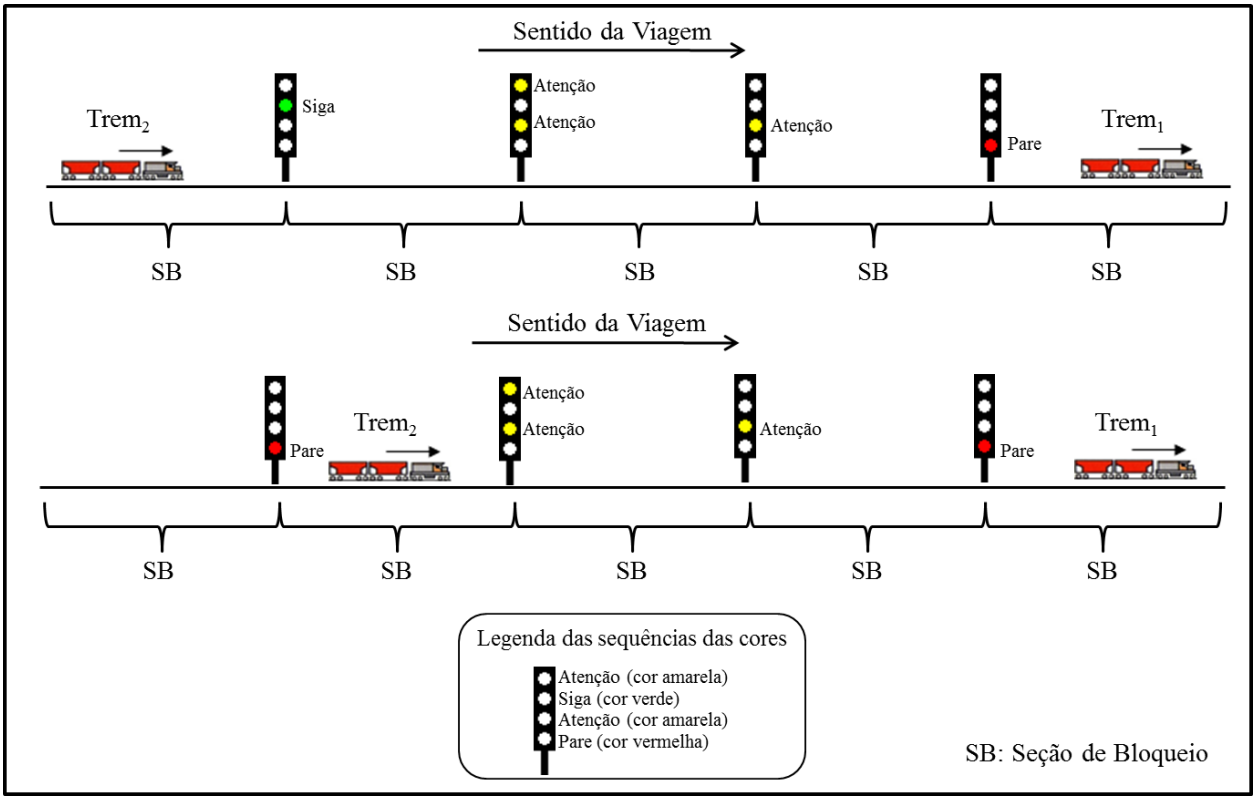


Figura 6 – Sinalização utilizando seção de bloqueio e sinalização com 4 aspectos.

2.3.2.1 Seção de bloqueio móvel

A seção de bloqueio móvel visa obter a totalidade da capacidade da via férrea, à medida que cada seção de bloqueio é formada pela distância entre trens e não mais pela seção de bloqueio fixa. Essa forma de controle considera o avanço tecnológico das comunicações e dos sistemas de posicionamento de trens em uma ferrovia. Dessa forma, os trens podem ocupar posições próximas umas das outras. Na Figura 7, os trens trafegam com um limite de segurança calculado entre eles. Este limite funciona como uma espécie de *bolha de segurança* entre os trens.

O limite de segurança é estipulado de acordo com a capacidade de frenagem do trem (valor calculado para o deslocamento à frente do trem até sua parada total), mais o tamanho do trem que está circulando a frente (Ke-Ping, Zi-You, & Bao-Hua, 2007), (Bajpai, Karad, Pawar, & Pilakar, 2007).

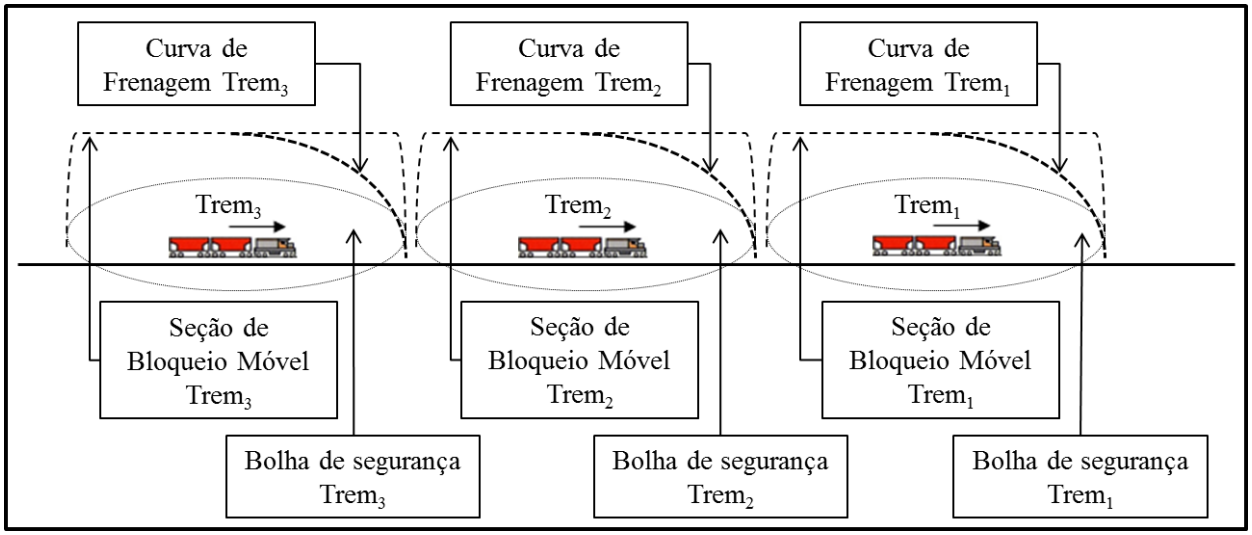


Figura 7 – Seção de bloqueio móvel.

2.3.3 *Licenciamento*

O licenciamento é uma autorização para trafegar em um trecho de ferrovia compreendido pelo bloqueio ou seção de bloqueio. A autorização pode conter informações adicionais como restrições de velocidade, precauções, serviços de manutenção na via férrea, dentre outras. A licença pode ser concedida por meio de um texto escrito, bastão de *staff*[2], autorização verbal gravada ou uma sinalização luminosa (Silva, 2008).

2.3.4 *Proteção Automática de Trem*

O sistema de **P**roteção **A**utomática de **T**rem (ATP) é responsável por controlar a velocidade de um trem e mantê-lo em uma velocidade de cruzeiro determinada. A condução é realizada de forma que o trem respeite a sinalização e as restrições da ferrovia. A restrição de velocidade máxima é informada pelo ATP ao condutor por meio do computador de bordo. Caso o condutor extrapole a velocidade máxima calculada ou não obedeça à sinalização de restrição do trecho, os freios são ativados automaticamente (cf. Figura 8). O ATP necessita de inúmeras informações para realizar os cálculos de controle do trem, que são: posicionamento na ferrovia, velocidade atual, rota a ser seguida e situação dos aspectos de sinalização. Estas informações são geralmente obtidas por sistemas de comunicação. O sistema também necessita de informações sobre o trem, tais como: tipo e peso de

[2] O *staff* consiste em um conjunto de dois equipamentos que continham diversos bastões metálicos. Cada um desses equipamentos localizado em uma das estações adjacentes e eram interligados eletricamente.

cada locomotiva e vagão. Todavia, muitas vezes tais informações são passadas pelos condutores a cada viagem, o que pode incorrer a erros humanos e, consequentemente, cálculos equivocados (Davies, 2000), (Palumbo, 2013) (Dong Hairong, 2010).

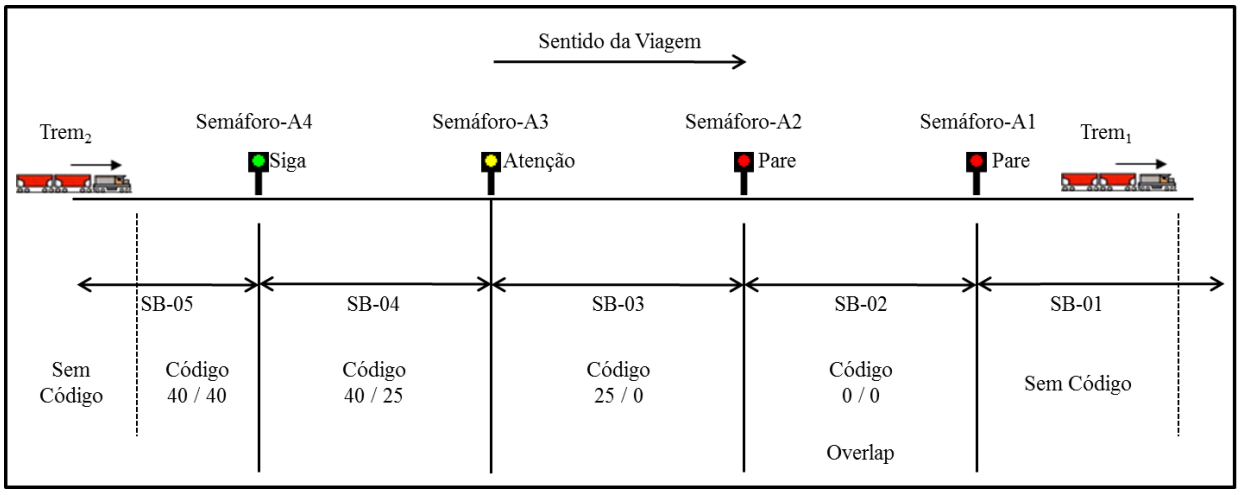


Figura 8 – Sistema ATP e suas velocidades de entrada/saída em cada seção de bloqueio com uma seção de sobreposição.

A restrição é baseada na sinalização. O ATP calcula as velocidades de entrada e saída, faz a frenagem automática caso essas velocidades extrapolem os seus limites ou a seção de sobreposição seja invadida.

2.3.5 *Operação Automática de Trem*

O sistema de **O**peração **A**utomática de **T**rem (ATO) realiza alguns controles sobre os trens que chegam e deixam as estações, e também sobre a distância exata de parada na plataforma. Esse sistema trabalha em conjunto com o ATP.

A execução do ATO começa com a aproximação do trem à plataforma, tanto para paradas quanto para partidas. No momento de uma parada, o ATO verifica se a plataforma está livre para a chegada do trem. Se a plataforma estiver livre, então são iniciados os cálculos de parada pelo ATO (Dong Hairong, 2010) (Wang, Ning, Cao, & De Schutter, 2011).

Na parada realizada pelo ATO, os cálculos são atualizados várias vezes para uma parada precisa e suave. A utilização do ATO é novamente requisitada para realizar a partida do trem da plataforma (Parkinson & Fisher, 1996).

2.3.6 *Supervisão Automática de Trem*

O sistema de **S**upervisão **A**utomática de **T**rem (ATS) se utiliza de uma tabela de tempo de chegada e saída de trens. Ele é capaz também de atualizar os horários em tempo real para verificar se os trens estão dentro de seus itinerários. Ele pode sobrepor-se ao sistema ATP em alguns ajustes (e.g., tempo de chegada à estação) dos trens no tocante às seções de bloqueios, o qual, na grande maioria das vezes, utiliza o sistema de seção de bloqueio móvel (Dong Hairong, 2010).

O ATP, mesmo sendo projetado com a finalidade de manter o itinerário dos trens, pode ser incapaz de atingir tal fim quando ocorrem situações de conflito (e.g., parada em função de um congestionamento na estação na qual o trem está programado para não parar). As situações de conflito podem ocorrer em desvios, cruzamentos e pátios onde não há uma forma de coordenação dos trens em operação. Isso pode causar descompassos no tempo de outros trens do mesmo trecho (Parkinson & Fisher, 1996).

2.3.7 *Controle Automático de Trem*

O sistema de **C**ontrole **A**utomático de **T**rem (ATC) visa automatizar as funções de controle de um trem a ponto de não haver a necessidade de intervenção humana. Ele é um sistema que utiliza os outros três sistemas já descritos (cf. Figura 9). O ATC utiliza a seguinte sequência de operação: (i) para manter uma distância segura entre as seções de bloqueio, o ATP lê os dados de sensores na via férrea que indica se há um trem na seção de bloqueio subsequente, e calcula velocidade limite para evitar invadir uma seção de bloqueio posterior ou ocupada. Essa informação é local e indica apenas a ocupação de uma seção de bloqueio; (ii) após isso, tal informação é repassada para um nível acima, no qual o ATS verifica se a velocidade aplicada ao trem está dentro do tempo programado de viagem e (iii) por fim, o ATO faz os cálculos de parada de estação ou pátio de manobra.

No ATC, a central de operações penaliza o condutor quando, por algum motivo, ele invade uma seção de bloqueio não autorizada. Essa penalização resulta na ativação de um sistema de **P**arada **A**utomática de **T**rem (*ATStop*) quando o condutor infringe uma norma. O *ATStop* então ativa automaticamente os freios de emergência até a parada completa do trem e só depois que o condutor relatar o que ocasionou a invasão ele poderá voltar a movimentar o trem (Dong Hairong, 2010).

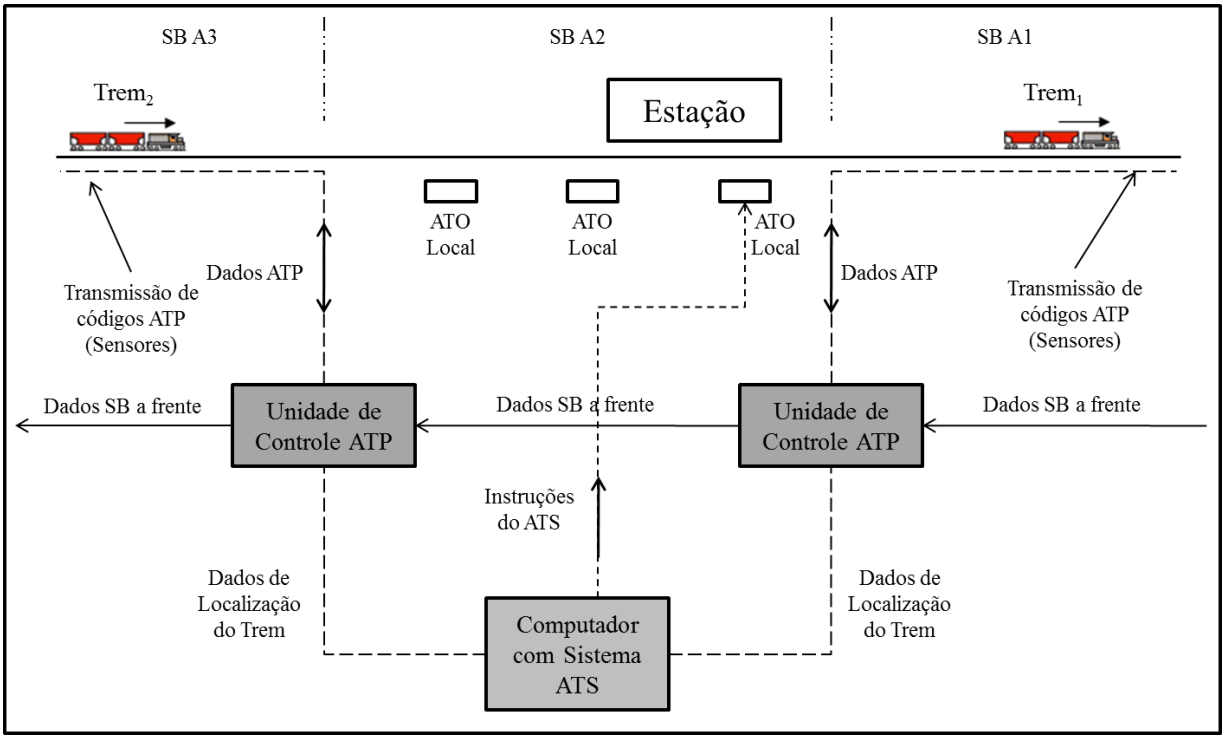


Figura 9 – Esquema do sistema de controle automático de trem (Dong Hairong, 2010).

2.3.8 *Controle de Trens Baseado em Comunicação*

O sistema de **C**ontrole de **T**rens **B**aseado em **C**omunicação (CBTC) foi implantado para evitar as falhas do sistema ATC, tais como: colisão e baixa ocupação da via férrea. Com isso algumas ferrovias americanas começaram a utilizá-lo para o controle do tráfego ferroviário (Hartong, Goel, & Wijesekera, 2006), (Allotta, Chisci, D'Adamio, Papini, & Pugi, 2013). Trata-se de uma tecnologia que evita a utilização de elementos de via férrea para o controle de tráfego, tais como: semáforo, seção de bloqueio e outros métodos de sinalização que por muitas vezes são danificados.

A comunicação é bidirecional entre trens e central de controle. As suas principais características são: baixo custo de implantação; baixo custo de manutenção; maior utilização da capacidade da via férrea; redução dos intervalos de tempos entre os trens; maior segurança e melhor controle das operações ferroviárias. Dessa forma, sua utilização foi se tornando cada vez mais comum em todos os tipos de ferrovias metropolitanas, suburbanas, regionais e nacionais (Kumar & Basha, 2013).

A ferrovia do Alasca utiliza o sistema CBTC. Ela faz uso de um sistema chamado *Global Navigation Satellite Systems* (GNSS), que é uma forma de navegação baseada em GPS. Na Europa uma série de ensaios está sendo realizados para a utilização de *GNSS* (Liu, Cai, & Wang, 2014), (Nguyen, Beugin, & Marais, 2014), (Albrecht, Luddecke, & Zimmermann, 2013). O CBTC tem todas

as outras funcionalidades dos sistemas anteriores porque sua comunicação é realizada por transmissão via satélite.

O sistema CBTC, com a evolução tecnológica de rastreamento via satélite, tem sido fonte de inspiração para os sistemas de condução de trens. Ele reduz o tempo e aumenta a qualidade das informações que podem ser enviadas aos sistemas autônomos de condução de trens.

O CBTC devido a algumas alterações na sua forma de comunicação pode também ser conhecido pelos nomes ***P**ositive **T**rain **C**ontrol* (PTC), ou ***W**ireless **P**ositive **T**rain **C**ontrol* (WPTC). Eles são variações do mesmo sistema, no qual são inseridas algumas formas diferentes ou melhorias em termos de sistemas de comunicação (Hartong, Goel, & Wijesekera, 2006), (Kumar & Basha, 2013), (Peters & Frittelli, 2012), (Petit, 2009).

O sistema PTC utiliza concomitantemente várias formas de comunicação sem fio (cf. Figura 10). Ele é capaz de monitorar a integridade de um trem e o controle de distribuição de mais de uma locomotiva ao mesmo trem (Abelleyro, 2009), (Hartong, Goel, & Wijesekera, 2012).

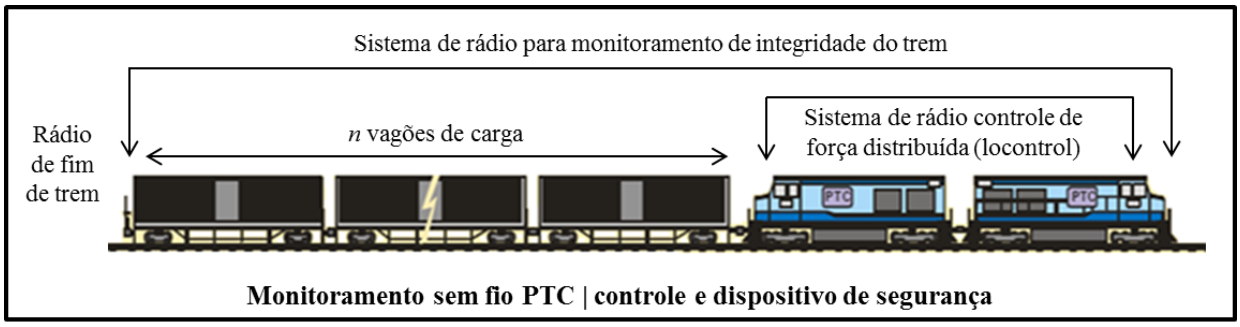


Figura 10 – Esquema do sistema de controle positivo de trem (Abelleyro, 2009).

O sistema de *fim de trem* instalado no último vagão possui a capacidade de notificar a posição do último vagão e também se houve algum desacoplamento. Assim, o PTC tem a informação completa da posição de cada vagão do trem. Um estudo mais aprofundado pode ser visto em (Petit, 2009), (Hartong, Goel, & Wijesekera, 2012).

O sistema de controle de trens baseado em comunicação e suas variantes de controle positivo de trem representam o *estado da arte* em termos de tecnologia.

2.4 CONDUÇÃO

A condução de um trem de carga é feita basicamente pela aplicação de uma força de aceleração ou de frenagem. Tal aplicação é usualmente controlada por uma manopla que possui diferentes

posições de força de aceleração em uma direção e uma posição costal ou neutra, assim como diferentes posições de frenagem na direção oposta. Tais posições de força de aceleração são denominadas de *pontos de aceleração*. Alguns trens utilizam um pedal associado à um sistema denominado *homem morto* orientado pelo pé esquerdo do condutor, um pedal de freio operado pelo pé direto, e um pedal de aceleração também operado pelo pé direito. Em cada caso, o número de posições depende das características da locomotiva e dos sistemas de controle do veículo.

Os elementos de controle incluem sensores utilizados para medir, por exemplo, a velocidade, pressão dos freios, posição da locomotiva líder. Os sinais destes sensores são processados por um computador de bordo, o qual provê funções de controle do motor e dos sistemas de frenagem (Kutz, 2011).

2.4.1 *Aceleração*

A aceleração é a variação da velocidade praticada pelo trem em relação ao tempo. Esta variação pode ser positiva ou negativa. Basicamente, todos os procedimentos de aceleração envolvem a aplicação de esforços tratores para a superar forças contrárias ou resistências. De modo geral, a aceleração positiva deve ocorrer sempre que a velocidade do trem é (i) menor que a velocidade mínima necessária para rebocar o trem; (ii) menor que a velocidade recomendada; (iii) decrescente e o trem está em uma rampa com percentual positivo. Para acelerar um trem, é necessário elevar a força de aceleração, aumentando assim a potência gerada pelo motor da locomotiva. Este aumento da potência se dá normalmente pela aplicação de um ponto de aceleração maior que o ponto de aceleração atual, movimentando a manopla de potência uma posição acima da atual. Porém, quando se está em rampa descendente, a aceleração pode ocorrer naturalmente devido às forças gravitacionais que impulsionam o trem no mesmo sentido.

Nada foi dito até o momento sobre as regras para indicar quando aumentar, manter ou reduzir a velocidade, ou frear um trem. Nesta linha de entendimento, o gráfico da Figura 11 fornece um esquema básico que relaciona a velocidade atual v, a velocidade inicial v_i, a velocidade cruzeiro v_c, a velocidade máxima v_s, a menor velocidade crítica $v_{critica}$ e a velocidade mínima projetada $v_{minProj}$ (Loumiet, Jungbauer, & Abrams, 2005). Esse esquema deve orientar a seleção das ações básicas para uma condução de regime.

Definição: condução de regime

> Uma condução de regime é obtida quando a velocidade atual é maior que a inicial e menor que a velocidade máxima, mantendo-se próxima da velocidade cruzeiro.

Em todas as situações de condução, a velocidade v deve ser superior à zero. O resultado das regras tem em vista quatro ações básicas: (i) aumentar a velocidade (ACELERAR), (ii) manter a velocidade (MANTER), (iii) reduzir a velocidade (REDUZIR) e (iv) frear o trem (FREAR).

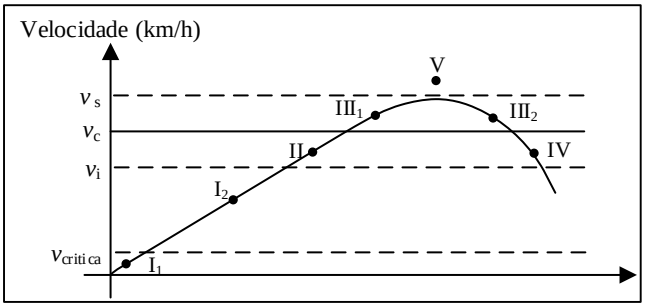


Figura 11 – Possíveis estados da condução de trens em função da velocidade (Loumiet, Jungbauer, & Abrams, 2005).

Os estados indicam ações que devem ser executadas para atingir uma condução de regime.

Regra 01. Se estado é I_1, então sugere-se ACELERAR; por que a velocidade atual é menor que a velocidade crítica ($v < v_{critica}$).

Regra 02. Se aceleração é negativa e velocidade atual é menor que a velocidade mínima projetada (ou margem de segurança), sugere-se ACELERAR; tendo em vista a dificuldade de movimentação de um trem abaixo desta velocidade e também o consumo de combustível ser mais elevado em velocidades menores $v \leq (v_{minProj} + v_c)$. A velocidade mínima projetada assumida é calculada pelo Algoritmo 7.

Regra 03. Se estado é *I_2*, sugere-se ACELERAR; por que a velocidade atual é menor que a velocidade inicial ($v < v_i$).

Regra 04. Se estado é *II*, sugere-se ACELERAR; por que a velocidade atual é menor que a velocidade de cruzeiro ($v < v_c$). Optou-se por usar nos experimentos MANTER a velocidade atual para economizar combustível.

Regra 05. Se estado é III_1, sugere-se FREAR; por que a velocidade é maior ou igual à velocidade de cruzeiro ($v \geq v_c$) e menor que a velocidade máxima ($v \leq v_s$) e a variação de velocidade é positiva ($\Delta v > 0$) e o percentual de rampa é negativo ($\%R < 0$).

Regra 06. Se estado é *III₂*, sugere-se REDUZIR a velocidade; por que a velocidade atual é maior ou igual à velocidade de cruzeiro ($v \geq v_c$), ela menor ou igual à velocidade máxima ($v \leq v_s$) e a variação de velocidade é positiva ($\Delta v > 0$).

Regra 07. Se estado é III_2 e percentual de rampa (%R) for inferior a zero, sugere-se MANTER; por que a velocidade atual é maior ou igual à velocidade cruzeiro ($v \geq v_c$), ela menor ou igual à velocidade máxima ($v \leq v_s$).

Regra 08. Se estado é *IV*, sugere-se ACELERAR; por que a velocidade atual é menor que a velocidade de cruzeiro ($v < v_c$). Optou-se por MANTER a velocidade para economizar combustível.

Regra 09. Se estado é *V*, sugere-se FREAR; por que a velocidade atual pode exceder a velocidade máxima permitida ($v \geq v_s$).

Regra 10. Se nenhuma regra anterior for avaliada, sugere-se FREAR; por motivo de segurança deve-se tentar minimizar riscos de acidentes.

Durante a condução é recomendado evitar acelerar demasiadamente o trem se logo em seguida há uma restrição de redução de velocidade ou uma parada obrigatória. Por questões de segurança, conservação da via férrea e do trem, deve-se acelerar o trem apenas quando todos os freios estejam completamente soltos para evitar arrastos e, consequentemente, danos materiais e acidentes. É recomendado também aumentar gradativamente os pontos de aceleração em intervalos de 2s a 3s, para o trem absorver melhor a potência do ponto de aceleração engrenado antes de avançar ao próximo ponto de aceleração (Loumiet, Jungbauer, & Abrams, 2005).

2.4.2 *Frenagem*

A frenagem de um trem é a operação realizada para o controle da velocidade, seja ela uma redução parcial ou total. Estas operações requerem maiores cuidados devido à complexidade na utilização dos três freios existentes nos trens: *(i) freio automático* ou *pneumático* formado por um compressor de ar em cada locomotiva que fornece ar para um encanamento. Esse sistema de freio começa no compressor de ar de uma locomotiva e passa por um encanamento ao longo do trem até atingir todos os vagões. Ele fornece ainda pressão para os cilindros de freio que consequentemente exercem força de pressão entre a sapata de freio e a roda do vagão. Esse freio atua apenas nas rodas dos vagões. Ele é um freio que se propaga das locomotivas para os vagões e depende de um intervalo de tempo para efetivação; esse tempo está relacionado ao comprimento do trem e da pressão máxima dos compressores; *(ii) freio dinâmico* ou *motor* é utilizado por meio da inversão de pontos de aceleração,

resultando assim em potência contrária ao movimento exercida pelas locomotivas. Nesse freio geralmente a mesma manopla dos pontos de aceleração é utilizada. O maquinista tem um ponto máximo e um ponto mínimo de utilização desse freio, dependendo da sua necessidade e; *(iii) freio independente* é aquele realizado nas rodas das locomotivas por meio da força aplicada em sapatas de freio (Loumiet, Jungbauer, & Abrams, 2005).

A frenagem em velocidade de cruzeiro deve ser iniciada a partir dos freios automáticos e gradativamente ser incrementada, caso a velocidade continue aumentando, pode-se optar por elevar ainda mais a frenagem automática ou utilizar de forma combinada a frenagem dinâmica. Se houver uma frenagem automática muito forte de forma prematura, deve-se soltar todo o freio automático para só depois iniciar novamente sua aplicação. Não há como reduzir a frenagem automática e elevá-la novamente. Isto não pode ser realizado pelo fato da necessidade de um tempo de espera da restauração da pressão pelos compressores das locomotivas. Assim, uma frenagem prematura próxima a um declive acentuado pode ocasionar na falta de freios do trem.

2.4.3 *Ciclo de condução*

A tarefa de conduzir um trem, seja por meio da aceleração ou da frenagem, é uma atividade intelectual que compreende de um ciclo formado por concepção, percepção, atuação e retroalimentação.

A concepção é uma característica humana que indica uma noção geral ou a capacidade de entender ou criar uma ideia, um modo de ver ou sentir. Assim, as ideias gerais e pensamentos criativos da imaginação podem ser qualificados como concepção. A concepção remete ao ato de elaborar conceitos, o qual começa com a compreensão da essência de um objeto e culmina na elaboração de um conceito. Muitos conceitos podem ser vistos como representações mentais estruturadas que codificam um conjunto de condições necessárias e suficientes para sua aplicação. Esta codificação ocorre por meios sensoriais ou pela percepção (Laurence & Margolis, 1999). A percepção corresponde às funções de organizar, identificar e interpretar os estímulos sensoriais para representar e compreender o ambiente (Goldstein, 2009).

A atuação é o processo em que o ser humano utiliza os recursos disponíveis para executar uma ação ou um comportamento. A atuação normalmente é regida pela concepção e pela percepção, na medida em que para atuar o ser humano inicialmente deve estar ciente do que o cerca antes de selecionar ou tomar uma decisão sobre qual ação irá executar (Harré, Clarke, & Carlo, 1985).

A retroalimentação é o procedimento de informar uma pessoa ou sistema acerca do seu desempenho ou da ação executada. A retroalimentação objetiva estimular ou reorientar algo ou alguém sobre comportamentos futuros mais adequados. A partir da retroalimentação o indivíduo pode memorizar as melhores ações a serem tomadas em cada circunstância para melhorar as próximas ações (Deterline, 2004).

Um condutor, tanto humano como automático, deve possuir uma competência básica que o possibilite aplicar tais conceitos. De forma simplificada, a condução de um trem é regida por um ciclo formado por ações que corroboram com os conceitos aqui apresentados. Um condutor deve possuir uma concepção dos objetos que compõe o mundo a ser percorrido. Assim se a concepção já foi formada, logo é possível que ele perceba o estado atual do mundo, selecione uma ação adequada à condução e aplique-a. O resultado da aplicação é memorizado e utilizado para aprimorar a sua concepção do mundo.

A seguir este ciclo será discutido do ponto de vista de condução manual e de condução assistida.

2.4.3.1 Condução Manual

A condução manual caracteriza-se pela tomada de decisão ser exclusiva do maquinista. Neste meio, para tomar uma decisão o maquinista faz uso de alguns equipamentos embarcados e as suas próprias percepções sensoriais.

A concepção inicial do maquinista é formada durante o seu treinamento teórico e prático. As informações iniciais para a condução dependem da forma como cada maquinista é treinado e das situações vivenciadas durante o seu treinamento. A percepção de um maquinista é assegurada pelo seu campo sensorial. Este campo utiliza principalmente as funções auditiva e visual. Na auditiva, o maquinista deve se basear nos sons internos e externos à cabine. Os sons podem indicar, por exemplo, um vagão descarrilando ou algum defeito mecânico. Já a percepção visual é utilizada para obter informações dos diversos equipamentos instalados na cabine, sinalizações na via férrea, e demais estímulos visuais recebidos que possam interferir na condução (e.g. cruzamentos, obstáculos e defeitos na via).

Com base na percepção, o maquinista deve selecionar uma ação que o permita atingir seu objetivo principal: aumentar, manter, reduzir a velocidade ou frear. Para executar qualquer uma das três primeiras ações, o maquinista normalmente seleciona um ponto de aceleração que presume satisfazer o objetivo e analisa o resultado da ação. Se o resultado for insatisfatório, ele seleciona outro

ponto de aceleração, até atingir o objetivo. Isto resulta em um processo de *tentativa-e-erro*; quando a condução é realizada por maquinistas inexperientes, este processo é observado com maior frequência. O mesmo se aplica na situação de frenagem, onde o maquinista faz a primeira aplicação de freios, que possui um valor padrão e, em seguida, seleciona ações de frenagem de acordo com a necessidade percebida, sem considerar, muitas vezes, o caminho à frente a percorrer.

A aplicação da ação depende do objetivo desejado. Para acelerar um trem, como já dito, normalmente o maquinista aplica um ponto de aceleração que gere uma potência maior que a atual. Maquinistas mais experientes tendem a aproveitar o perfil vertical da via para usar a força gravitacional em favor da aceleração. Para manter a velocidade constante basta manter a potência; em trechos em nível. Para reduzir a velocidade pode-se reduzir a potência ou frear. Estas são situações que variam de acordo com o perfil vertical da via; elas são semelhantes a condução de um veículo de passeio. A diferença está na maior influência da força da gravidade e na ausência de um acelerador na locomotiva, visto que a variação da velocidade é dada por alterações na posição da manopla dos pontos de aceleração e não por pedais.

Após aplicar uma ação, seja ela bem ou mal sucedida, o maquinista memoriza o seu resultado. Obviamente, nem todas as ações são memorizadas. Tende-se a memorizar as ações à medida que mais experiências são repetidas e novas situações vivenciadas.

A retroalimentação ocorre após a memorização do efeito da ação aplicada. Nela o maquinista capta pelas suas habilidades sensoriais o efeito da ação. Este efeito alimenta uma nova percepção e o ciclo de condução se repete, até chegar a seu destino.

2.4.3.2 Condução Assistida

A condução assistida caracteriza-se por usar um programa de computador para auxiliar a tomada de decisão do maquinista ou conduzir o trem de forma automática.

Aqui, a percepção ocorre pela leitura dos sensores instalados na locomotiva. Estes sensores alimentam ao computador de bordo em termos de dados e o sistema de condução os interpreta para compor a percepção do ambiente. Algumas destas informações podem ser obtidas no início da viagem por meio da ordem de despacho (e.g. número de veículos, origem, informações técnicas das locomotivas e perfil de toda a via) ou durante a missão, a partir da leitura de sensores em intervalos de tempo predefinidos (e.g., velocidade, posição). Outras informações podem ser deduzidas à partir da percepção e de dados contidos no computador de bordo, como é o caso do perfil de onde o trem está posicionado, resultado da inferência da posição percebida sobre o perfil total da via.

A leitura das medidas dos sensores por si só é insuficiente para o sistema selecionar uma ação correta. É necessário que vários cálculos sejam feitos para compor a situação atual e projetar a situação futura de um trem após a aplicação de uma determinada ação. Na condução manual, os cálculos são feito de maneira intuitiva e aproximada pelo maquinista.

Um sistema de condução automática faz uso dos cálculos do domínio para verificar situações de excessos, desperdícios, etc. A etapa de cálculo, a partir de um conjunto de dados de entrada, permite derivar informações relativas à situação de um trem, tais como: resistência total, força de tração necessária para superar as resistências, força de tração máxima aplicável, aceleração, consumo, dentre outras. A Tabela 3 enumera as equações. De forma resumida, o procedimento padrão consiste em aplicar uma força maior que a resistência total de um trem para rebocá-lo, respeitando em particular três classes de restrições: sinalização (e.g. seção de bloqueio), velocidade máxima permitida e força máxima aplicável. A sofisticação deste procedimento acrescenta restrições sobre o uso adequado dos recursos disponíveis e seus desdobramentos, objeto deste trabalho de pesquisa.

Tabela 3 – Equações para os cálculos de resistência, tração, descolamento, tempo e consumo (Loumiet, Jungbauer, & Abrams, 2005), (Profillidis, 2006), (Shabana, Zaazaa, & Sugiyama,, 2007).

EQUAÇÃO	FUNÇÃO	DESCRIÇÃO
$R_T = \sum_{i=0}^{n_l}(R_{nl} + R_{cl} + R_\gamma + R_i) + \sum_{i=0}^{n_v}(R_{nv} + R_{cv} + R_\gamma + R_i)$	$f_{Rt}(R_{nl}, R_{cl}, R_\gamma, R_i, R_{nv}, R_{cv})$ $f_{Rt}(T, ST, pos)$*	Resistência total do trem (em kgf.).
$R_{nl} = 1{,}3 + \frac{29}{w} + 0{,}03 \times v + \frac{0{,}0024 \times A \times v^2}{w \times n}$	$f_{R_n}(w, v, A, n)$	Resistência normal da locomotiva (em kgf.).
$R_{nv} = 1.3 + \frac{29}{w} + 0{,}045 \times v + \frac{0{,}0024 \times A \times v^2}{w \times n}$	$f_{R_v}(w, v, A, n)$	Resistência normal dos vagões em kgf.
$R_{cl} = 0{,}2 + \frac{100}{R} \times (br + b + 3{,}8)$	$f_{R_l}(R, br, b)$	Resistência de curva da locomotiva (em kgf.).
$R_{cv} = \frac{500 \times b}{R}$	$f_{R_{cv}}(b, R)$	Resistência de curva do veículo (em kgf.).
$R_i = 10 \times i$	$f_{R_i}(i)$	Resistência de rampa do veículo (em kgf.).
$R_\gamma = 4 \times \frac{v_F^2 - v^2}{\ell}$	$f_{R_\gamma}(v_F, v, \ell)$	Resistência inercial (em kgf.)
$F_t = \frac{273.24 \times 0.82 \times HP}{v}$	$f_{F_t}(HP, v)$	Esforço trator (em kgf.).
$F_{tm} = \frac{W \times f}{1 + (0{,}01 \times v)}$	$f_{F_{tm}}(W, f, v)$	Força tratora máxima (em kgf.)
$F_{ac} = F_t - R_T$	$f_{F_{ac}}(F_{tm}, R_T)$	Força de aceleração disponível (em kgf.).

$v_F = \sqrt{\left\lvert v^2 + \frac{(F_{ac} \times 20)}{(4 \times W)} \right\rvert}$	$f_{v_F}(v, F_{ac}, W)$	Velocidade final desejada para cada deslocamento de 20m (em km/h).
$\Delta\ell = 4 \times \frac{W \times (v_F^2 - v^2)}{F_{ac}}$	$f_{\Delta\ell}(W, v_F, v, F_{ac})$	Deslocamento (em metros).
$\Delta t = 7{,}2 \times \frac{\Delta\ell}{v_F + v}$	$f_{\Delta t}(\Delta\ell, v_F, v)$	Variação da duração da ação (em segundos).
$v_m = \frac{\Delta\ell}{\Delta t} \times 3{,}6$	$f_{v_m}(\Delta\ell, \Delta t)$	Velocidade média (em km/h).
$\Delta t_{final} = \sum_{k=1}^{n} \Delta t_k$	$f_{\Delta t_{final}}(\Delta t_k)$	Tempo gasto total da viagem (em segundos).
$\Delta\ell_{final} = \sum_{k=1}^{n} \Delta\ell_k$	$f_{\Delta\ell_{final}}(\Delta\ell_k)$	Deslocamento total da viagem (em metros).
$\gamma = F_{ac} \times \frac{g}{W}$	$f_{\gamma}(F_{ac}, g, W)$	Aceleração do trem.
$C = \frac{\Delta t}{60} \times cp$	$f_C(\Delta t, cp)$	Consumo nominal para um intervalo de tempo (em litros/minuto).
$LTKB = \frac{\sum C}{W \times \Delta\ell_{final}}$	$f_{LGTT}(\sum C, W, \Delta\ell_{\text{final}})$	Consumo de uma viagem (em litros por *tonelada* bruta *transportada*– LTKB).
$F_{fd} = pad \times v_F \times 1417{,}475$	$f_{F_{fd}}(v_F, pad), se\ v \leq 38{,}6km/h$	Força de frenagem dinâmica realizada pelos motores da locomotiva (em kgf) em velocidade abaixo de 38.6km/h.
$F_{fd} = pad \times v_F^{-1.06} \times 921526{,}6$	$f_{F_{fd}}(v_F, pad), se\ v > 38{,}6km/h$	Força de frenagem dinâmica realizada pelos motores da locomotiva (em kgf) em velocidade acima de 38.6km/h.
$F_{fa} = ar \times P_{cl} \times ra \times ef \times ca \times 10$	$f_{F_{fa}}(ar, P_{cl}, ra, ef, ca)$	Força de frenagem automática realizada pelos vagões de um trem (em kgf).
$F_f = F_{fd} + F_{fa} - R_i$	$f_{F_f}(ar, P_{cl}, ra, ef, ca, v_F, pad, R_i)$	Força de frenagem total.

Legenda:

A: área do veículo (em *pés-quadrados*)
ar: área do êmbolo do cilindro de freio (em m^2)
b: bitola da linha (em metros)
br: base rígida do veículo
ca: coeficiente de atrito entre sapata de freio e roda
cp: consumo de combustível em determinado ponto de aceleração (em litros)
ef: eficiência da timoneira de freio (em %)
f: coeficiente de aderência entre os trilhos e rodas
g: aceleração da gravidade (em m/s^2)
HP: potência do veículo (em cavalo de força)
i: porcentagem da inclinação da via a cada 100m
kgf: quilograma força
ℓ: deslocamento desejado (em metros)
n: número de eixos
n_l: número de locomotivas
n_v: número de vagões
pad: ponto de frenagem dinâmica
P_{cl}: pressão do cilindro de freio (em kPa)
pos : posição da cabeça do trem T sobre a ST.
R: raio da curva (em metros)
ra: relação da alavanca da timoneira de freio
ST: descrição de um trecho de via férrea onde o trem T está posicionado.
T: descrição de um trem em termos de locomotivas e seus vagões.
v: velocidade (em km/h)
v_F: velocidade desejada (em km/h)
w: peso do veículo (em toneladas)
W: peso do trem (em toneladas)

* essa função adapta a aplicação da função $f_{Rt}(R_{nl}, R_{cl}, R_{\gamma}, R_i, R_{nv}, R_{cv})$

Seja T um trem, F_{ac} a força de aceleração, R_T a resistência total que se opõe ao movimento de T, F_t a força de tração calculada para mover T em uma velocidade v, F_{tm} a força de tração máxima para rebocar T, F_f a força de frenagem, $velMin$ velocidade mínima para rebocar T, $velMax$ velocidade máxima permitida, pa ponto de aceleração responsável por gerar a potência atual de T; F_{tm} define o limite máximo de tração preservando a aderência das rodas de T sobre os trilhos. Para movimentar um trem com aceleração positiva — situação A — é necessário que (i) a força de tração seja superior à resistência total do trem e, ao mesmo tempo, seja inferior à força de tração máxima (para evitar patinagem); (ii) a força de aceleração seja positiva; (iii) a velocidade seja menor que a máxima permitida (caso contrário o trem deve parar por medida de segurança) e (iv) a potência gerada pelo ponto de aceleração seja maior que a potência mínima requerida para movimentar o trem.

Situação A: definida pela aceleração positiva

PRÉ-CONDIÇÕES			
Força de tração e Resistência	Aceleração	Velocidade	Ponto de aceleração
$F_t > R_T$ $F_t \leq F_{tm}$	$fa > 0$	$v > 0$ $v \leq velMax$ $v \geq velMin$	$pa \geq paMin$

Para movimentar um trem com aceleração negativa — situação B — é necessário que (i) a força de tração seja menor que a resistência total somada à força de frenagem e (ii) a força de aceleração seja negativa. Assim como na situação A, a velocidade deve ser maior que zero e a potência gerada pelo ponto de aceleração seja maior que a potência mínima requerida para movimentar o trem. Nota-se que na situação B a força de tração também deve ser menor que força de tração máxima (para evitar patinagem), a velocidade deve ser positiva (para garantir o deslocamento) e deve ser menor que a máxima (para garantir a segurança) e maior que a mínima (para garantir o deslocamento).

Situação B: definida pela aceleração negativa ou frenagem

PRÉ-CONDIÇÕES			
Força de tração e Resistência	Desaceleração	Velocidade	Ponto de aceleração
$F_t < R_T + F_f$ $F_t \leq F_{tm}$	$fa < 0$	$v \geq 0$ $v \leq velMax$ $v \geq velMin$	$pa \geq paMin$

A tarefa principal é selecionar uma força de tração que atenda a situação percebida, situação que deve resultar em uma aceleração positiva, negativa ou frenagem. Tal seleção pode ser feita por meio da aplicação das equações da cinemática de um trem (cf. Tabela 3). Esse processo é iterativo e sua complexidade é linear de ordem $O(n)$, onde n corresponde à quantidade de pontos de aceleração aplicáveis. Para exemplificar parte deste processo, considere a seguinte situação: um sistema computacional de condução deve selecionar e aplicar um ponto de aceleração em um trecho de resistência positiva ou aclive. Logo,

R1. se o ponto de aceleração ainda não é o máximo, então deve-se selecionar um ponto de aceleração superior ao atual.

R2. se essa nova seleção não obter a força aceleradora necessária ($F_t < R_T$) para que a velocidade seja elevada, então vá para R1—outro ponto de aceleração deve ser selecionado para a ação requerida ($F_t > R_T$).

A seleção pode resultar em um ponto de aceleração que gere excesso de força ($F_t > F_{tm}$), fazendo com que as rodas da locomotiva patinem. No exemplo descrito pode ser verificado na *Situação A*, onde $R_T < F_t \leq F_{tmax}$.

A vantagem de um sistema computacional sobre um condutor humano está em analisar a forma de condução com base em cálculos precisos da situação do trem. O sistema pode analisar em tempo de execução se a ação selecionada está ou não adequada a situação atual do trem. Essa avaliação pode ser realizada antes mesmo da ação ser efetivada, de forma que uma ação potencialmente equivocada seja revogada em detrimento de outra mais adequada.

A eficiência de uma missão é diretamente proporcional à redução do tempo e do consumo de energia empregado. Logo, o que é necessário para realizar uma missão de forma eficiente? Em termos práticos, a eficiência máxima deve ser alcançada quando o trem é rebocado em uma velocidade v próxima da $velMax$, empregando a menor potência necessária (Loumiet, Jungbauer, & Abrams, 2005). Em outras palavras, o problema principal consiste em encontrar, para uma dada resistência total R_T, a mais apropriada força de tração F_t capaz de rebocar um trem T de modo eficiente e eficaz.

Como já foi dito, a potência de cada locomotiva é graduada em pontos de aceleração $AP = \{ap_1, ap_2, \ldots, ap_{np}\}$. Cada ponto $ap_i \in AP$ é um par $\langle hp, cp \rangle$, onde hp é uma potência nominal e cp é o consumo nominal empregados no ponto ap_i. Tal informação de consumo é fundamental para determinar a eficiência de um trem em operação, em termos de consumo acumulado (e.g., litros de

diesel por tonelada bruta transportada ou LTKB), juntamente como: Δt um intervalo de tempo (em minutos), $\Delta \ell_{final}$ uma distância percorrida (em metros), cp o consumo de combustível associado a um $ap_i \in AP$ (em litros), W o peso do trem T (em toneladas). Ao longo deste texto é utilizada a chamada de função $f_{pot}(pa)$ para retornar uma potência associada à um ponto de aceleração pa de acordo com a Tabela 5.

Situação C: Eficiência de um trem em operação.

PRÉ-CONDIÇÕES	
Consumo Pontual	Consumo Acumulado
$\Delta t > 0$ $cp > 0$	$\Delta l_{final} > 0$ $W > 0$

A seguir é exemplificada a aplicação básica das equações da Tabela 3, cujo objetivo é determinar as resistências, as forças de tração, o deslocamento e o consumo associado à movimentação de um trem.

2.4.4 *Exemplos de cálculos*

Seja T um trem com 1 locomotiva e 5 vagões, cujas descrições estão resumidas nas próximas três tabelas. Elas descrevem: (a) uma locomotiva e um vagão padrão; (b) as diferentes potências e consumos nominais de uma locomotiva modelo C-30; cada potência está associada a um ponto de aceleração; e (c) um trecho de via férrea em termos de pontos de medidas.

Tabela 4 – Dados de configuração de uma locomotiva modelo C-30 e um vagão do tipo container.

CARACTERÍSTICA	LOCOMOTIVA C-30	VAGÃO CONTAINER
Comprimento	20m	20m
Base Rígida	2,4m	Não se aplica
Peso	169,7t	99,46 t
Número de Eixos	6	4
Eficiência de frenagem	0,3	0,65
Atrito da sapata de freio na roda	0,358	0,358
Área do êmbolo do cilindro de freio	78,58 pol^2	78,58 pol^2
Área frontal	120 pés	120 pés
Aceleração da gravidade	9.8 m/s^2	9.8 m/s^2
Coeficiente de aderência	0.22	0.22
Bitola	1.6m	1.6m

Tabela 5 – Potência e consumo de uma locomotiva modelo C-30.

PONTO DE ACELERAÇÃO	POTÊNCIA (CV)	CONSUMO (L/MIN)
Dinâmico	0	0,3168
Neutro	*0*	0,3168
1	*100*	0,5670
2	*275*	1,0668
3	*575*	1,9500
4	*960*	3,0330
5	*1440*	4,5330
6	*1930*	6,1830
7	*2500*	7,6998
8	*2940*	9,4002

Tabela 6 – Exemplos de 14 pontos do projeto topográfico de um trecho de via férrea.

ID	KM	VELMAX	RAMPA	RAIOCURVA	AC	G20	ALTITUDE	NROSB
1	349	60	0.95	1150.5	25.48	0.99	526.10	3
2	349	60	0.95	1150.5	25.48	0.99	526.10	3
3	349	60	0.95	1150.5	25.48	0.99	526.10	3
4	349	60	0.95	1150.5	25.48	0.99	526.10	3
5	349	60	0.95	1150.5	25.48	0.99	526.10	3
6	350	60	0.95	1150.5	25.48	0.99	526.10	4
7	350	60	0.95	1150.5	25.48	0.99	526.10	4
8	350	60	0.95	1150.5	25.48	0.99	526.10	4
9	350	60	0.95	1150.5	25.48	0.99	526.10	4
10	350	60	0.95	1150.5	25.48	0.99	526.10	4
11	351	60	0.95	1150.5	25.48	0.99	526.10	4
12	351	60	0.95	1150.5	25.48	0.99	526.10	4
13	351	60	1.20	818.52	66.20	1.40	534.26	4
14	351	60	1.20	818.52	66.20	1.40	534.26	4

Legenda:

id:	Identificador do ponto de medida
km:	Quilômetro de referência.
velMax:	Velocidade máxima permitida (em km/h).
rampa:	Inclinação da rampa (em %).
raioCurva:	Raio da curva (em metros).
ac:	Ângulo central da curva (em metros).
g20:	Grau da curva para uma corda de 20m.
altitude:	Altitude (em metros).
nroSb:	Identificador da seção de bloqueio

Para ilustrar os cálculos, foi considerado o trecho via férrea descrito na Tabela 6, onde cada linha denota um ponto de medida com 0,02km de comprimento. O trem está inicialmente posicionado como segue: o último vagão se encontra no ponto de medida 1 e a locomotiva se encontra sobre o ponto de medida 6. O procedimento consiste em rebocar o trem para o próximo ponto de medida. Assim, na primeira iteração a locomotiva está no ponto de medida 6, na segunda iteração ela estará

no ponto de medida 7 e o último vagão está sobre o ponto de medida 2, e assim por diante; cada novo ponto de medida vencido perfaz um deslocamento de 0,02km. Quando a locomotiva se encontrar no ponto de medida 11, o deslocamento total foi de 0,10km. Deve-se notar que as resistências são calculadas individualmente para cada componente do trem: locomotiva ou vagão. A Tabela 7 ilustra de forma simplificada as etapas dos cálculos e a Tabela 8 mostra os resultados.

Tabela 7 – Etapas dos cálculos para rebocar um trem. *Loop i* representa um ciclo completo. *Loop j* representa o processo iterativo para encontra uma potência aplicável.

ORDEM	ETAPA			VARIÁVEIS
1	Loop *i*	Percepção		pm, v
2		Cálculo de Resistências		R_T
3		Loop *j*	Seleção de Ponto de Aceleração	pa
4			Cálculo de Forças	F_{tm}, F_{ac}
5		Atuação		pa
6		Memória		$\Delta\ell, \Delta t, C$

Tabela 8 – Exemplo de cálculos para rebocar um trem com uma locomotiva e cinco vagões por 0,10km.

pm	v	R_T	F_{tm}	F_a	F_t	AP	$\Delta\ell$	Δt	C
7	18,43	7.738,25	31.501,49	3.879,05	11.617,31	4	0,02	3,88	0,196
8	18,60	7.740,17	31.460,35	3.780,69	11.520,86	4	0,02	3,85	0,194
9	18,74	7.749,91	31.254,91	8.838,35	16.588,26	5	0,02	3,70	0,279
10	20,16	7.768,28	30.881,34	12.927,08	20.695,36	6	0,02	3,44	0,355
11	21,63	7.785,44	30.547,80	11.680,21	19.465,65	6	0,02	3,24	0,333
Total							**0,10**	**18,11**	**1,357**

Seja $W_T = 667t$ o peso de trem, $\Delta\ell_{final} = 0{,}10km$ o deslocamento final, $\Delta t_{final} = 18{,}11s$ o tempo decorrido, $C_{final} = 1{,}357l$ o consumo total de combustível, o LTKB calculado é de 20,34. Este último obtido dividindo C_{final} pelo produto de $W_T \times \Delta\ell_{final}$. A velocidade do trem no ponto de medida 11 é de 21,63km/h.

O cenário supracitado, assim como os cálculos e seus desdobramentos ilustram uma situação bem simples. Essa situação inocente pode rapidamente mudar de patamar de complexidade. Para tal, considere um trem com 4 locomotivas, 100 vagões perfazendo 2,08km de comprimento. O método para os cálculos das resistências permanece inalterado. Entretanto, o método para a seleção da potência a ser empregada muda significativamente, na medida em que as potências das locomotivas podem ser combinadas para compor a força de tração necessária para fazer frente às resistências. Há duas formas de combinar as potências de um conjunto de locomotivas que rebocam um trem em conjunto, a saber: *Monoponto* e *Multiponto*.

Definição: método *Monoponto*

> Todas as locomotivas de um trem aplicam ao mesmo tempo, o mesmo ponto de aceleração ou potência.

Definição: método *Multiponto*

> Todas as locomotivas podem aplicar ao mesmo tempo pontos de aceleração ou potência diferentes.

A seleção da força de tração para rebocar o trem supracitado, quando ela é feita usando o método *Monoponto* o espaço de busca tem apenas 10 estados, e quando ela é feita usando o método *Multiponto* o espaço de busca tem 10k estados. Essa situação pode ser ainda mais complexa se tal seleção levar em conta os perfis horizontais e verticais da via férrea a percorrer como modo de antecipação de uma ação face, por exemplo, a um trecho sinuoso com aclive seguido de decline, i.e., parte do trem pode estar no aclive e outra parte no declive. Adicionalmente, o desgaste dos veículos pode fazer como que duas locomotivas, do mesmo modelo, tenham variações nas potências resultantes. Estas variações tornam a tarefa de condução de trens ainda mais complexa. Acrescenta-se ainda que tal tarefa também dependente da experiência do maquinista em trechos específicos, i.e., o conhecimento profundo das condições e das práticas de condução em um trecho são significativas para o bom desempenho de uma missão. Em face de tais complexidades, o desafio é desenvolver um método computacional capaz de gerar bons planos de condução que reuse as experiências de diferentes atores.

Mais à frente, é proposta uma arquitetura que coloca em prática a especialização da condução por trecho de via férrea. O método computacional em questão opera preferencialmente sobre as experiências bem sucedidas. Tais experiências são registradas e reusadas na forma de casos.

2.4.5 *Etapas dos cálculos*

As etapas dos cálculos mostrados na Tabela 7 são formalmente descritas no Algoritmo 1. Da linha 1 a linha 12 são realizadas as inicializações das variáveis. A linha 13 marca o início do laço principal que encerra quando a missão é cumprida. Entre as linhas 14 e 16 é obtida a percepção dos sensores. Na linha 15 é obtida a velocidade máxima para os próximos 5km — valor parametrizável. Na linha 16 é determinado o tipo de perfil a ser percorrido. Nas linhas 17 é calculada a resistência total do trem. A seleção de ações inicia na linha 18 com a seleção de um comportamento (i.e., ACELERAR, MANTER, REDUZIR ou FREAR). O comportamento é determinado de acordo o Algoritmo 3 que será visto adiante. Caso a decisão seja FREAR, a força de frenagem é calculada (linha 21) e

uma ação é selecionada — o ponto de aceleração tem valor atribuído como -1 para indicar o uso da frenagem automática. Com a frenagem em ação, os cálculos de aceleração entre as linhas 23 a 28 não são computados. Das linhas 30 a 36 são realizados os cálculos de forças: força aceleração (F_{ac}), tempo do deslocamento (Δt), tempo total de viagem (Δt_{total}), consumo do deslocamento (C), consumo total (C_{total}) e deslocamento total ($\Delta \ell$). Na linha 37 é realizado o ajuste da velocidade atual (v) para velocidade final (v_f) — representando a atuação. Por fim, são calculados e memorizados a quantidade de litros de combustível utilizado durante a missão (linha 39), do tempo total de viagem (linha 40) e do consumo final da viagem (linha 41).

Em um novo ciclo, a frenagem agora é verificada pela função *selecionarComportamento()*, se o retorno é continuar a frear, então a potência de frenagem deverá ser elevada. Porém, se a decisão for por parar de frear e iniciar uma aceleração, então o bloco com as linhas 30 e 37 é novamente executado, para só então ser iniciado na condição de aceleração. Deve-se notar que os freios de um trem não são relaxados imediatamente; isto pode levar alguns ciclos. Somente após a relaxação total dos freios, o ciclo de aceleração pode ser aplicado; essa aplicação corresponde o bloco formado pelas linhas 23 a 28, onde a função *selecionarPontoAceleracao()* seleciona um ponto de aceleração, e são calculadas as seguintes forças (linhas 25 e 26): força tratora (F_t) e força aderente (F_{tm}). A função *validar()* verifica se existe consumo excessivo e/ou patinagem. Este *laço* repete até que um ponto de aceleração aplicável seja determinado.

Nesta seção não serão detalhadas as funções *selecionarComportamento* e *validar*. Elas farão parte da seção 4.3.2. A função *selecionarComportamento* é descrita no Algoritmo 3 da página 119 deste trabalho. A função *validar* está descrita na Tabela 24 da página 162. Para atender a metodologia desta pesquisa, o ciclo entre as linhas 24 a 29 é sobreposto e pequenos ajustes são realizados no algoritmo.

2.5 CONSIDERAÇÕES FINAIS

A condução de um trem como foi ilustrada nesta seção é complexa. Tal complexidade está fortemente liga a relevância da experiência do maquinista para selecionar a potência correta em cada situação que se apresenta, visando uma condução segura e econômica. Em termos de técnica computacional, para tentar a virtualização do comportamento de uma maquinista em atividade, foram estudados e desenvolvidos vários experimentos baseados em temas específicos da área de *Inteligência Artificial*, a saber: raciocínio baseado em casos, algoritmos genéticos e agentes de software.

Algoritmo 1 – Ciclo de cálculos empregados para simular a movimentação de um trem durante a simulação.

procedure rebocar (T, ST_i)			
T: é um trem a ser rebocado. ST_i: trecho onde o trem será rebocado.			
01	I	$v \leftarrow 0$	{velocidade atual}
02		$v_f \leftarrow 0$ {velocidade final após 20m}	
03		$v_{max} \leftarrow 0$	{velocidade máxima do ponto de medida}
04		$v_{maxProj} \leftarrow 0$	{velocidade máxima projetada x km adiante}
05		$v_{critica} \leftarrow 10$	{velocidade crítica}
06		$v_i \leftarrow 0$	{velocidade inferior}
07		$ap \leftarrow 0$	{ponto de aceleração inicial *ap*}
08		$fa \leftarrow 0$	{força de aceleração}
09		$R \leftarrow 0$	{percentual de rampa}
10		$pm \leftarrow 0$	{ponto de medida atual}
11		$pm_{destino} \leftarrow p[pm[kmDestino]]$	{ponto de medida do destino}
12		$f \leftarrow 0$	{coeficiente de atrito trilho-rodas}
13		**while** (! FimDaMissão) **do**	{condição de parada}
14	P	p ← perceber()	
15		$v_{maxProj} \leftarrow velmax(p, 5)$; {obtem a maior velocidade máxima em 5 km}	
16		$perfil \leftarrow f_{perfil}(p[\%R])$	
17	CR	$R_T \leftarrow f_{R_T}(T, ST_i, pm)$	
18	S	comportamento ← **selecionarComportamento**$(v, v_{critica}, v_i, v_{max}, p[f_{ac}], \%R)$ {página 119}	
19		**if** comportamento = FREAR **then**	
20		$ap \leftarrow -1$	{indicação do uso do freio automático}
21	CF	$F_{Fa} \leftarrow f_{F_{fa}}(ar, P_{cl}, ra, ef, ca)$	
22		**else**	
23		**repeat**	
24	S	$ap \leftarrow$ **selecionarPontoAceleracao**$(ST_i, T, pm, pm_{destino})$	
25	CF	$F_t \leftarrow f_{F_t}(HP, v)$	
26		$F_{tm} \leftarrow f_{F_{tm}}(W, f, v)$	
27		$(p, c) \leftarrow$ **validar**(ap)	
28		**until** (p = não patine **or** c = não excessivo)	
29		**end if**	
30	CF	$F_{ac} \leftarrow f_{F_{ac}}(F_{tm}, R_T)$	{força de aceleração resultante}
31		$\Delta t \leftarrow f_{\Delta t}(\Delta\ell, v_F, v)$	{tempo gasto no deslocamento}
32		$\Delta t_{total} \leftarrow \Delta t_{total} + \Delta t$	{atualização do tempo total da missão}
33		$C \leftarrow f_C(\Delta t, cp)$	{consumo do deslocamento}
34		$C_{total} \leftarrow C_{total} + C$	{atualização do consumo total da missão}
35		$v_F \leftarrow f_{v_F}(v, F_{ac}, W)$	{cálculo da velocidade final}
36		$\Delta\ell \leftarrow \Delta\ell + 0{,}02$	{incremento de deslocamento de 20m}
37	A	$v \leftarrow v_F$	{atualização da velocidade atual}
38		**end while**	
39	M	$C_{final} \leftarrow f_{C_{final}}(C_{total})$	{consumo acumulado da missão}
40		$\Delta t_{final} \leftarrow f_{\Delta t_{final}}(\Delta t_{total})$	{tempo final da missão}
41		$LGTT \leftarrow f_{LGTT}(C_{final}, W, \Delta\ell)$	{consumo final da missão}
42		**end procedure**	

Legenda: A: Atuação; CF: Cálculo das forças; M: Memória; S: Seleção do ponto de aceleração; CR: Cálculo das resistências; I: inicialização; P: Percepção

3 FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresenta uma introdução sobre Inteligência Artificial, Agentes, Algoritmo Genético e Raciocínio Baseado em Casos. Agentes são utilizados neste trabalho como componentes de softwares autônomos para encapsular funcionalidades bem específicas: gerar planos de condução, aplicar planos de condução e organizar as experiências decorrentes da aplicação de tais planos. Os planos de condução são elaborados usando a técnica de raciocínio baseado em casos, técnica inspirada na forma como o ser humano resolve os problemas no dia a dia: relembrando situações passadas e adaptando-as para o problema-alvo. Para adaptar as soluções passadas aos novos problemas foi utilizado um algoritmo genético, por este ser uma técnica de otimização consolidada na literatura e apresentar resultados satisfatórios, mesmo quando aplicado individualmente. Ao longo deste capítulo também são apresentados trabalhos que utilizam estas técnicas, no modal de transporte de carga, bem como alguns trabalhos relevantes em âmbito geral.

3.1 INTELIGÊNCIA ARTIFICIAL E OS AGENTES INTELIGENTES

A inteligência artificial é definida como o estudo das faculdades mentais por meio da utilização de modelos computacionais capazes de tomar decisões, resolver problemas e aprender a partir de tarefas realizadas (Charniak & McDermott, 1985). Um sistema inteligente possui a capacidade de executar tarefas complexas que exigem inteligência quando executadas por seres humanos, como, por exemplo, jogar xadrez, resolver quebra-cabeças, planejar viagens e diagnosticar doenças (Kurzweil, 1999).

A inteligência artificial se subdivide em subáreas, dentre elas: a inteligência artificial distribuída. Ela se interessa em resolver problemas de forma coletiva a partir da interação de várias entidades. Ela se aplica principalmente na solução de problemas complexos, devido as seguintes características: o grande esforço computacional necessário para solucioná-los, espaço de busca exponencial, dinamicidade, complexidade de representação e de funcionalidade, entre outros. Nestes casos, um problema deve ser decomposto em problemas menores, os quais são controlados por entidades autônomas que interagem para trocar informações e compartilhar os conhecimentos adquiridos individualmente vis-à-vis a montagem de uma solução global para um determinado problema. Este é o processo que norteia o princípio da inteligência artificial distribuída (Durfee, Lesser, & Corkill, 1989).

A inteligência artificial distribuída pode ainda, ser subdividida em duas subáreas: resolução

distribuída de problemas e sistema multiagente. A resolução distribuída de problemas é uma subárea que centra seus esforços na decomposição de um problema em subproblemas. Além disso, ela busca estabelecer protocolos de cooperação e de compartilhamento de conhecimentos acerca do estado local de cada subproblema, que são utilizados pelos agentes para resolver o problema de modo global. A subárea multiagente preocupa-se com os meios computacionais (e.g. linguagens, protocolos, frameworks de desenvolvimento, arquiteturas de agentes) que viabilizam a coordenação e o comportamento inteligente de um grupo de agentes para coordenar seus conhecimentos, objetivos, planos e habilidades de forma conjunta (Wooldridge, 2002).

O restante deste capítulo versará acerca do conceito de *agente* do ponto de vista da comunidade de Sistema Multiagente. Ela deve auxiliar o leitor não-especialista em sua inicialização no assunto.

3.1.1 *Agentes inteligentes*

A inteligência artificial apresentou vários avanços ao longo dos anos em várias subáreas, tais como: sistemas especialistas, redes neurais artificiais, entre outras. Estes avanços originaram, nos anos 90, entidades denominadas *agentes racionais*, as quais compreendiam de programas autônomos capazes de executar determinados comportamentos (Russell & Norvig, 1995). Os anos se passaram e apesar dos desenvolvimentos na comunidade de inteligência artificial, cientistas ainda discordam acerca da definição conceitual do termo *agente*. Cada definição de *agente* é influenciada pela aplicação do agente em uma determinada área, não sendo aplicável em todos os casos. Várias definições de agente foram propostas na literatura (Maes, 1995) (Russell & Norvig, 1995) (Franklin & Graesser, 1997) (Poole, Mackworth, & Goebel, 1998) (Ferber, 1999).

A maioria dos pesquisadores considera um agente como uma entidade programada para habitar em um ambiente e que, nesse ambiente, a entidade seja capaz de resolver um ou mais problemas de forma inteligente e autônoma, com foco sempre no objetivo para o qual ela foi programada. Durante a resolução de um problema, o agente recebe como entrada um conjunto de informações sensoriais do ambiente e produz alguma ação que afetará o ambiente no qual o agente está situado. Esta interação, envolvendo o agente e o ambiente, normalmente repete-se até que o agente satisfaça os seus objetivos (Wooldridge, 2002). O termo *agente* aqui abordado corresponde à uma entidade capaz de atuar em um ambiente físico ou virtual, comunicar com outros agentes e tomar decisões para satisfazer os seus objetivos. Além disso, um agente também é capaz de perceber o ambiente onde está

inserido, se reproduzir, prestar serviços a uma comunidade de agentes e extrair do ambiente informações que o auxilie na sua tomada de decisão. O desenvolvimento de um sistema multiagente passa, primeiramente, pela análise de alguns aspectos do problema que são usados para definir a estrutura do agente. Dentre os aspectos analisados estão, por exemplo, a forma como cada agente percebe o ambiente, a maneira como ele interage com outros agentes e os comportamentos necessários para solucionar os problemas (Ferber, 1999).

Uma visão simplificada da estrutura de um agente, com alguns aspectos internos, como objetivos e serviços, pode ser resumida da seguinte forma: um agente é capaz de receber mensagens com informações e requisições de informações modelados em termos de performativas—ou tipos de mensagens. Cada performativa (e.g., *ask-all, tell*) da linguagem KQML (Finin, et al., 1994) deve ser expressa em uma linguagem compartilhada pelo agente para que ele possa interpretá-lo. Ao interpretar a performativa, um serviço do agente é executado; obviamente, o agente deve ter no seu repertório de ações e de habilidades os recursos necessários para fazer o acordado. Deve-se salientar que cada agente busca alcançar seus objetivos de maneira autônoma, o que faz com que ele possa recusar um serviço se não possuir as habilidades para fazê-lo; i.e., ele pode recursar a demanda por estar ocupado realizando outra tarefa, ou por algum outro motivo, como insuficiência de recursos. Feita a tarefa, o agente retorna ao solicitante, usando uma performativa adequada (e.g., *tell*), o resultado da tarefa solicitada. Cada performativa pode codificar, por exemplo, a execução de uma ação ou simplesmente a comunicação de informação. Nestes termos pode-se dizer que um agente, na sua versão computacional mais básica, é um objeto com objetivo.

Para que um agente possa ser considerado inteligente não basta que ele perceba o ambiente, escolha uma dada ação dentre as ações disponíveis e produza alguma saída. Ele deve possuir um conjunto de capacidades que inclui: reatividade, proatividade e habilidade social. Reatividade corresponde a capacidade do agente perceber o ambiente onde está situado, respondendo às percepções de maneira oportuna para atender seus objetivos. A proatividade faz com que os agentes sejam capazes de possuir comportamentos dirigidos por seus objetivos, tomando a iniciativa de empreender as ações necessárias para alcança-los. Por fim, a habilidade social corresponde à capacidade de um agente comunicar com outros agentes ou seres humanos de modo a satisfazer os seus objetivos. O comportamento de um agente em determinado contexto é definido por um papel ou função (Wooldridge, 2002). Cada agente em uma comunidade pode desempenhar papéis sociais (e.g., papel de coordenação, execução de tarefas). Em (Biddle, 1986) (Shmeil, 1999), um papel social define um conjunto de normas, direitos, deveres e expectativas que condicionam o comportamento de um agente dentro de

uma organização. Esta distribuição de papéis dentro de um ambiente permite ver a organização como um conjunto de unidades funcionais; cada unidade é uma entidade cognitiva capaz de executar atividades (e.g., aprender, decidir, executar tarefas) para o alcance do objetivo global. Pode-se utilizar os papéis para definir os conceitos de agentes envolventes, os quais são formados de uma base de conhecimento e um conjunto de agentes (Shmeil, 1999). O conjunto de agentes formam dois subconjuntos distintos de agentes: agentes gestores e agentes executores. Um agente gestor é formado por uma base de conhecimento individual e um conjunto de capacidades que forma a competência do agente e compreende capacidades de planejamento, coordenação, comunicação e aprendizagem. O agente executor, por sua vez, é formado por uma base de conhecimento individual e um conjunto de capacidades. As capacidades de um agente executor são a especialidade/perícia, comunicação e aprendizagem.

Os módulos que implementam as capacidades de cada um dos agentes dependem da metodologia que o desenvolvedor deseja implementar. Os agentes gestores e executores apresentam capacidades semelhantes, como a aprendizagem e a comunicação. Todavia, pelo fato de desempenharem papéis diferentes na comunidade, eles possuem capacidades distintas: planejamento no caso do gestor e especialidade/perícia no caso do agente executor. A base de conhecimento do agente envolvente é chamada de base de conhecimento corporativa. Ela registra o conhecimento sobre o perfil do agente envolvente e seus planos e representa o conhecimento que não pertence a nenhum dos agentes gestores ou executores em particular. A aquisição, aprendizado e manutenção do conhecimento corporativo pelos agentes internos ao agente envolvente, ocorre em função das suas relações com outros agentes. O conhecimento coorporativo é o orientador do comportamento dos agentes gestores e executores no desempenho de suas tarefas, tendo como finalidade que as atividades serem executadas estejam de acordo com os objetivos da organização. O acesso e a manutenção do conhecimento coorporativo ocorrem por meio de permissões para que os agentes da organização tenham uma visão compatível com suas competências e atividades pelas quais são responsáveis (Shmeil, 1999).

A base de conhecimento individual de cada agente gestor e executor é formada por um conjunto de conhecimentos usados para o exercício das competências do agente no domínio ao qual ele está inserido. O acesso ao conhecimento individual de cada agente é interessante à medida que outros agentes gestores ou executores podem utilizar tais conhecimentos no exercício de papéis semelhante. Visualizar uma organização apenas como um conjunto de entidades distribuídas, tratando problemas de forma centralizada e com decisões concentradas a um único componente, iria restringi-las quanto à autonomia que cada agente deve possuir. Do ponto de vista de aplicação, a visualização de uma

entidade por meio de agentes inteligentes é adequada, porque se aproxima da forma como as organizações operam, diminui a complexidade dos componentes porque cada unidade funcional constitui um módulo, aumenta a flexibilidade das unidades, permite a divisão de tempo das atividades, entre outras vantagens (Shmeil, 1999). Além disso, a dimensionalidade, bem como a complexidade dos ambientes motiva a utilização e o desenvolvimento de agentes inteligentes dotados de técnicas de aprendizagem para a tomada de decisão em tempo real, não somente em ambientes organizacionais, mas também em áreas como robótica, Web, entre outras (Panayiotopoulos & Zacharis, 2001).

A visão supracitada de (Shmeil, 1999), do uso de papéis como gestor, executor e envolvente inspiraram a arquitetura do sistema proposto neste trabalho de pesquisa, discutido mais à frente; onde usamos os papéis similares: planejador, executor e memorizador.

3.1.2 *Agentes aplicados ao modal férreo*

Em (Abbink, et al., 2010) é apresentado um método para solucionar o problema de agendamento de trens de passageiros em caso de grandes interrupções. O método assume que a tabela de tempo e o material circulante foram reagendados baseados em incidentes, onde cada maquinista de trem é representado por um agente condutor. O agente condutor, cuja condução se tornou inviável por causa da interrupção, inicia um processo de troca de tarefas para resolver a inviabilidade. Esse processo é gerenciado por um agente analisador da via férrea, que determina se a troca é ou não viável, com base em vários parâmetros visando encontrar um conjunto de trocas com custo total mínimo. O custo é calculado em função do tempo decorrido, disponibilidades de outros condutores, número de trens afetados e preferências individuais de cada trem.

O paradigma reconhece atores humanos e agentes artificiais como membros equivalentes de um time, cada um cumprindo suas respectivas regras de modo a alcançar os objetivos do time, onde as capacidades cognitivas dos agentes são complementares às capacidades humanas (Iacob, Nieuwenhuis, Wijngaards, Pavlin, & van Veelen, 2009). Tal paradigma provê ao sistema uma abordagem descentralizada na qual agentes usam conhecimento local, visões do mundo e interações para contribuir com a arquitetura aberta do sistema. Esta abertura favorece a fácil reconfiguração e/ou adaptação dos requisitos do sistema. Além disso, a combinação de agentes e atores humanos na arquitetura do sistema permite que eles interajam em níveis apropriados de abstração. Tanto os times de agentes atores como as comunidades de agentes são comunidades de especialistas; os agentes artificiais possuem capacidades cognitivas complementares centradas em determinados problemas. Segundo (Abbink, et al., 2010), o reagendamento é resolvido fazendo uso de níveis de responsabilidade:

despachantes humanos no nível estratégico/gerencial; condutores humanos no nível de defesa de seus interesses; agentes artificiais no nível de implementação das decisões estratégicas/gerenciais e resolução de conflitos nas escalas de agendamento.

Quando se fala em gerenciamento de ferrovia, Bhardwaj, Ghosh e Dutta (2013) propõem o uso de agentes BDI (*Belief Desire Intentions*). Atuando de forma autônoma, os agentes BDI podem substituir sinalização fixa, à medida que são capazes de comunicar uns com os outros para garantir o melhor gerenciamento da malha, a qual foi modelada como um grafo. O objetivo é evitar colisões e otimizar o espaço entre os trens. Foram modelados individualmente como agentes: trem, estação e junção. Eles tomam suas as próprias decisões. Um sistema baseado em regras também é usado como um sistema *backbone*, o qual pode ajudar os agentes na tomada de decisão (Ghosh, Dutta, & Alam, 2013).

Uma revisão sobre o uso de agentes em sistemas de tráfego e transportes é apresentada por (Chen & Cheng, 2010), os quais apresentam várias abordagens de controle. Eles pontuam que a maioria das aplicações baseadas em agentes têm como foco a modelagem e a simulação sem considerar cenários reais. Frente a nossa proposta, trabalhou-se com dados reais e a abordagem usada não faz parte dos trabalhos analisados pelos autores em questão.

É apresentado (Borges, Ribeiro, Ávila, Enembreck, & Scalabrin, 2009) e (Borges A. P., 2009) o uso de árvore de decisão como mecanismo de inferência de regras de condução de trens com dois focos distintos. Os autores analisaram as velocidades praticadas durante a condução e o número de ações executadas frente à condução com dados reais. O objetivo era comparar a similaridade entre a condução real e a condução simulada a partir de regras extraídas na forma de uma árvore de decisão. Resultados esses que mostraram-se satisfatórios em termos de similaridade das ações, porém, os resultados ficam aquém quando se fala de abordagens relacionadas ao consumo de combustível.

Uma vez analisados alguns trabalhos que utilizam agentes inteligentes no modal férreo, a próxima seção introduz os principais conceitos do método que será utilizado para adaptar os planos de condução gerados pelos agentes: algoritmo genético.

3.2 ALGORITMO GENÉTICO

Algoritmos genéticos fazem parte da família dos algoritmos evolutivos, que são métodos estocásticos de busca baseados em mecanismos de seleção natural e genética estudados inicialmente por Darwin. O argumento é que na natureza os indivíduos mais aptos sobrevivem por apresentarem a capacidade de adaptação às mudanças do ambiente onde vivem (Darwin, 1859). Neste contexto, os

algoritmos evolutivos são métodos capazes de representar os dados de um problema do mundo real e fazer evoluir a sua solução. Por este motivo, os algoritmos evolutivos são considerados uma abordagem na busca por soluções subótimas, principalmente em problemas de otimização. Ela envolve a busca por valores de parâmetros ótimos pela área mais provável do espaço de busca baseado no *fitness* (i.e., função de avaliação dos melhores indivíduos), da solução proposta para o domínio do problema e restrições físicas consideradas. Apesar de gerarem soluções aleatórias, os algoritmos evolutivos utilizam informações anteriores na busca no espaço de soluções (Goldberg, 1989) (Mitchell M. , 1996).

O Algoritmo Genético foi utilizado neste trabalho para otimizar a seleção da ação a ser tomada por um condutor, à medida que tal seleção pode ser modelada como um problema de otimização, onde o consumo de combustível e o tempo de viagem devem ser minimizados; além disso, o processo completo deve satisfazer um conjunto de restrições de controle, como velocidades máximas e critérios de segurança.

As etapas de um algoritmo evolutivo são: geração de indivíduos da população inicial; reprodução; avaliação de cada indivíduo gerado; seleção dos melhores indivíduos; cruzamento e mutação a fim de criar uma nova população. Cada indivíduo corresponde a uma solução candidata para um determinado problema. Mais formalmente, podemos definir os elementos de um algoritmo genético como: uma população pop, de tamanho $npop$, formada por um conjunto de indivíduos. Cada indivíduo possui um conjunto de características (i.e., genes) de tamanho n, cujos valores (i.e., alelos) são avaliados por uma função de *fitness*. Essa última é responsável por medir o grau de aptidão do indivíduo no ambiente. O grau de aptidão é a qualidade da solução candidata à resolução do problema (Viana, 1998).

O esquema definido no Algoritmo 2, mostra em detalhe o procedimento canônico de um algoritmo genético. O processo começa com a leitura dos parâmetros necessários para a execução do algoritmo: tamanho da população ($npop$), número de indivíduos selecionados para a próxima geração ($nsel$), número máximo de gerações ($maxGer$), probabilidade de cruzamento ($prob_c$) e probabilidade de mutação ($prob_m$). Em seguida uma população inicial de tamanho $npop$ é gerada. Neste exemplo a condição de parada é controlada pelo número de gerações (g). Enquanto ela não for atingida, o ciclo é executado. Tal ciclo inicia com a seleção de $nsel$ indivíduos da geração anterior para formar a geração atual. Uma nova população é gerada pela etapa de reprodução denotada pela função

$reproduzir(pop_g, npop)$. Os indivíduos gerados são avaliados pela função $fitness(pop_g)$. Em seguida, a população passa por operações como trocas de partes do material genético (i.e., *cruzamento*) denotada pela função $cruzar(pop_g, prob_c)$, escolhidas de acordo com uma certa probabilidade de cruzamento ($prob_c$). Os indivíduos também passam pelo processo de substituição aleatória de partes do material genético, denominado mutação. A mutação também ocorre dada certa probabilidade ($prob_m$), denotada pela função $mutar(pop_g, prob_m)$. Por fim, ao atingir a condição de parada, a melhor solução, ou melhor individuo, é selecionado ($selecionar(pop_g)$).

Algoritmo 2 – Aprendizagem utilizando a técnica evolucionária do algoritmo genético canônico (Back, Fogel, & Michalewicz, 2000).

function algoritmoGenetico($npop, nsel, maxGer, prob_c, proc_m$) : SOLUCAO		
npop: número de indivíduos da população. *nsel*: número de indivíduos a serem selecionados. *maxGer*: número máximo de gerações. $prob_c$: probabilidade de cruzamento. $prob_m$: probabilidade de mutação.		
01	$pop_0 \leftarrow$ gerarPopulacao(npop)	{geração da população inicial}
02	$g \leftarrow 1$	{controle de gerações}
03	**while**($g \leq$ maxGer) **do**	{condição de parada}
04	$pop_g \leftarrow$ selecionar(pop_{g-1}, nsel)	{seleção dos melhores indivíduos para a próxima geração}
05	$pop_g \leftarrow$ reproduzir(pop_g, npop)	{reprodução}
06	$pop_g \leftarrow$ fitness(pop_g)	{cálculo de fitness dos indivíduos da população}
07	$pop_g \leftarrow$ cruzar(pop_g, $prob_c$)	{cruzamento}
08	$pop_g \leftarrow$ mutar(pop_g, $prob_m$)	{mutação}
09	$g \leftarrow g + 1$	
10	**end while**	
11	$s \leftarrow$ selecionar(pop_g)	{seleção do melhor indivíduo da última geração}
12	**return** s	{melhor solução}
13	**end function**	

Ambas as técnicas (algoritmo evolutivo e algoritmo genético) são inspiradas na genética natural. Como resultado, uma nova população pop é gerada, com novos indivíduos correspondendo aos antecessores da população, os quais serão responsáveis por substituir os indivíduos atuais (i.e., pais). Todo este processo evolutivo é repetido até que um critério de parada seja satisfeito, como por exemplo, um número máximo de gerações ($maxger$) seja atingido ou até que uma solução considerada satisfatória seja encontrada (Mitchell M. , 1996). Existem vários tipos de algoritmos evolutivos e o algoritmo genético é um dos mais utilizados na resolução de problemas de otimização, em particular por ser um método consolidado matematicamente (Freitas A. A., 2008). Considerando g o número de gerações, p o tamanho da população e n o número de genes em cada configuração, a ordem de complexidade do algoritmo genético é O($g \times p \times n$). De modo geral, $g = c_1 \times n$, para $c_1 > 0$, a complexidade é de ordem o($c \times n^2$), equivalente, no pior caso, a $O(n^3)$ (Viana, 1998).

A seguir serão descritas, em detalhes, as características fundamentais para a compreensão do funcionamento de um algoritmo genético, o qual será utilizado neste trabalho como técnica para adaptação da solução para um problema-alvo.

3.2.1 *Representação*

A representação de um indivíduo é o primeiro passo de um algoritmo evolutivo, correspondendo à ligação entre o problema real e o espaço de busca a ser explorado pelo algoritmo. As possíveis soluções no contexto do problema são denominadas *fenótipos*. A codificação dos indivíduos é chamada de *genótipo*. O *fenótipo* deve ser encontrado por meio da decodificação do melhor *genótipo* localizado pelo algoritmo evolutivo após a sua execução (Eiben & Smith, 2003). Em outras palavras, a solução de um problema-alvo é obtida após decodificar os indivíduos do algoritmo evolutivo.

Na representação do indivíduo, o *cromossomo* corresponde a uma cadeia de caracteres, cujos valores (i.e., *alelos*) possuem posições específicas (i.e., *lócus*) e podem ser representados de diferentes formas. A escolha de uma forma de representação adequada dos indivíduos é uma tarefa decorrente do domínio de aplicação do problema (Mitchell M. , 1996). E dentre as formas citadas estão às representações por valores binários, inteiros e reais.

A representação binária é a forma mais utilizada. Ela conta com diversas teorias de convergência dos algoritmos genéticos e as operações de *cruzamento* e *mutação* são fáceis de programar. Todavia, quando o problema de otimização é complexo, necessitando de *cadeias* significativas, a representação binária não é vantajosa (Mitchell M. , 1996). A Figura 12 apresenta um exemplo de um indivíduo codificado por uma *cadeia* binária, onde os caracteres que representam esse indivíduo são bits 0 e 1.

1	1	0	1	0	0

Figura 12 – Indivíduo codificado como uma *cadeia* de bits (Mitchell M. , 1996).

Em outras situações, quando o problema é encontrar um conjunto de variáveis cujos domínios sejam números inteiros, a solução binária não é a mais adequada. A representação da solução como uma *cadeia* de inteiros é recomendada, na qual o tamanho da *cadeia* é igual ao número de variáveis e cada posição (gene do indivíduo) representa o valor da variável em questão (cf. Figura 13).

1	6	3	0	2	5

Figura 13 – Indivíduo codificado como uma *cadeia* de inteiros (Mitchell M. , 1996).

Outras duas formas de representação de indivíduos podem ser utilizadas: representação por valores reais e por árvores. A primeira pode ser utilizada quando o domínio das variáveis do problema são números reais, o que acontece quando se deseja obter pesos de uma rede neural, por exemplo. Todavia, as árvores podem se tornar muito grandes à medida que a população evolui e possui caminhos incontroláveis. A representação por árvores é utilizada com mais frequência na programação evolutiva, onde cada indivíduo da população é um programa com dados, operadores e funções.

3.2.2 *População inicial*

Os primeiros indivíduos, em um algoritmo genético, são gerados de modo aleatório. O tamanho da população (i.e., número de indivíduos da população inicial) é predeterminado pelo projetista. É aconselhável que os indivíduos sejam os mais diferentes possíveis entre si, de forma que estejam distribuídos no espaço de busca de soluções para o problema em questão, para dar maior diversidade à população (Eiben & Smith, 2003) (Mitchell M. , 1996).

O procedimento mais comum para geração da população inicial é preencher os genes dos indivíduos com valores aleatórios. Por exemplo, se o *cromossomo* de um indivíduo é uma *cadeia* de caracteres no formato binário, o preenchimento aleatório dos genes pode assumir uma probabilidade p para cada gene receber o valor 0 e uma probabilidade $(1 - p)$ para receber o valor 1, garantindo a aleatoriedade da *cadeia* de caracteres (Eiben & Smith, 2003).

Espera-se, com a geração de uma população inicial, obter um conjunto de indivíduos que representem soluções alcançáveis para o problema em questão. Uma forma de facilitar a geração de bons indivíduos é utilizar os conhecimentos do domínio do problema. Por exemplo, no caso do preenchimento de uma *cadeia* de caracteres composta por valores inteiros, não é qualquer valor do domínio dos números inteiros que pode ser atribuído para qualquer gene do indivíduo em um domínio de problema; pode haver restrição nos valores das variáveis representadas pelo indivíduo (Eiben & Smith, 2003).

3.2.3 *Função de fitness*

A função de *fitness* é responsável por medir o grau de aptidão dos indivíduos da população e,

juntamente com a etapa de codificação dos indivíduos, é uma das partes do algoritmo diretamente relacionada com o domínio do problema abordado (Eiben & Smith, 2003) (Mitchell M. , 1996). O papel desta função é apresentar os requerimentos para adaptar e otimizar os indivíduos, uma vez que forma a base para o processo de *seleção*. Na perspectiva da resolução de problemas, a função de *fitness* representa a tarefa de selecionar o contexto evolutivo, onde é possível otimizar um ou mais objetivos, dependendo do domínio e objetivo do problema tratado (Eiben & Smith, 2003).

3.2.3.1 Um único objetivo

Em determinados problemas a solução procurada visa minimizar um valor $u(x)$ ou maximizar uma determinada função $g(x)$. Nestes casos, é utilizado diretamente $u(x)$ ou $g(x)$ como função de *fitness*, onde a primeira visa minimizar e, a segunda, maximizar de um determinado problema (Eiben & Smith, 2003).

Todavia, em algumas soluções o valor da função *fitness* não deve ser negativo, em particular, quando é utilizado o método da seleção por roleta. Neste caso, é necessário mapear o valor obtido e o valor de *fitness* para não que este não possua valores negativos. Para tal função de *fitness* do indivíduo x inclui uma constante C_{max}, definida *a priori*, e subtraída desta constante o valor a ser minimizado $g(x)$, logo o valor de *fitness* $f(x)$ do indivíduo é dado por $f(x) = C_{max} - g(x)$. O valor de C_{max} pode ser escolhido de várias formas. Ele pode receber o maior valor observado de $g(x)$, o maior valor de $g(x)$ na população atual ou o maior de valor de $g(x)$ nas últimas k gerações. Isto para garantir que $f(x)$ não seja negativo para qualquer valor x em hipótese alguma. Quando o problema é de maximização, o mapeamento é direto. Contudo, alguns valores podem ser negativos e o mapeamento deve resolver esta situação. Analogamente a minimização, é inserida uma constante C_{min} definida *a priori* e somada à $u(x)$. A determinação do valor de C_{min} é a mesma utilizada para determinar C_{max}, porém, ao invés de escolher os maiores valores, são escolhidos os menores valores de u (Eiben & Smith, 2003).

3.2.3.2 Vários objetivos

Os problemas com multiobjetivos surgem quando deseja-se otimizar simultaneamente duas ou mais funções objetivo f_1, f_2, ..., f_n (ou uma função objetivo vetorial $f = (f_1, f_2, \ldots, f_n)$) sujeitas provavelmente a restrições $g_i(x) \leq 0, i = 1, \ldots, r$. Assim, um problema de otimização multiobjetivo

pode ser escrito na seguinte forma $\max_x f(x) = (f_1(x), f_2(x), \ldots, f_n(x))$, sujeito a $g_i(x) \leq 0, i = 1, \ldots, r$ (Eiben & Smith, 2003).

Em geral, os objetivos a serem otimizados $(f_1, f_2, \ldots, f_n)$ são conflitantes, i.e., dificilmente uma mesma escolha de parâmetros x otimiza todos os objetivos ao mesmo tempo. Por este motivo, a busca pela melhor solução está relacionada ao conceito de dominância (Eiben & Smith, 2003). Segundo tal definição, seja $f(x) = (f_1(x), \ldots, f_n(x))$ uma função definida em um ponto do espaço X. Um ponto $x_1 \in X$ domina outro ponto $x_2 \in X$ (denota-se $x_1 > x_2$) se $f_i(x_1) \geq f_i(x_2), i = 1, \ldots, n$ e se existe pelo menos um índice $k \in \{1, \ldots, n\}$ tal que $f_k(x_1) > f_k(x_2)$. Em outras palavras, um ponto x_1 domina um ponto x_2 se a avaliação de x_1 for melhor que a avaliação de x_2 em um objetivo e não for pior em nenhum outro objetivo. Este conceito é aplicado para problemas de maximização. Caso o problema seja de minimização, a definição de $x_1 < x_2$ vale, trocando os sinais $\geq$ e $>$ por $\leq$ e $<$, respectivamente (Pareto, 1896).

Os problemas multiobjetivos podem ser manejados por três diferentes abordagens: convencional *fórmula-ponderada*, lexicográfica e de Pareto (Freitas A. , 2004), conforme veremos a seguir.

— *Abordagem fórmula-ponderada*

Trata-se de uma abordagem comumente usada na literatura para lidar com problemas multiobjetivos. Nela, um problema multiobjetivo é transformado em um problema com objetivo único por meio da atribuição de pesos para cada objetivo, sendo que cada objetivo corresponde a um critério de avaliação da solução. Os valores dos critérios são combinados por meio de operações como adição e/ou multiplicação, conforme mostrado na Equação (1), onde w_1 e w_2 são os coeficientes de ponderação, representando a importância de cada critério, e $A(i)$ e $B(i)$ são os critérios avaliados (Freitas A. , 2004).

$$fitness(i) = w_1 \times A(i) + w_2 \times B(i) \quad (1)$$

A principal vantagem desta abordagem é sua simplicidade. Entretanto, os pesos são frequentemente determinados de forma intuitiva pelos usuários da aplicação a respeito dos critérios avaliados. Além disso, uma vez definidos os pesos e a função *fitness*, o algoritmo busca a melhor solução usando apenas os pesos definidos, não explorando pesos diferentes (problema conhecido como “oportunida-

des perdidas"). Outro problema desta abordagem envolve a soma/subtração (ao invés de multiplicação/divisão) de termos representando diferentes critérios, os quais podem ter escalas muito diferentes em suas unidades de medida. Para resolver este problema, normalmente são aplicadas técnicas de normalização, tornando seus valores na escala $[0, 1]$ (Freitas A. , 2004).

— *Abordagem de Pareto*

A ideia da abordagem de Pareto é usar um algoritmo multiobjetivo para solucionar o problema multiobjetivo na sua forma original ao invés de transformar um problema em um único objetivo. Na dominância de Pareto, uma solução s_1 é dita dominar uma solução s_2 se, e somente se, (i) s_1 é estritamente melhor que s_2 em pelos menos um dos critérios (objetivos) a serem otimizados e (ii) s_1 não é pior que s_2 em todos os outros critérios, ou seja $s_1(c_1) \leq s_2(c_1)$ para todo critério c_i sendo $i = 1, \dots, N_{obj}$, onde N_{obj} corresponde ao número de objetivos. Se não há uma solução que domine s_i a solução s_i é dita não-dominada para o critério c_i (Coello, Lamont, & Veldhuizen, 2006).

Na abordagem de Pareto é possível combinar medidas com escalas diferentes e também combinar medidas não mensuráveis para avaliação da qualidade de um modelo, na medida em que não há uma combinação de diferentes critérios em uma única fórmula, visto que todos os critérios são tratados separadamente. Além disso, ao contrário das outras abordagens, a abordagem de Pareto retorna ao usuário um conjunto de soluções ao invés de uma única solução. Esses fatores resultam em uma maior complexidade dos algoritmos de otimização multiobjetivo baseados na abordagem de Pareto (Coello, Lamont, & Veldhuizen, 2006).

A Figura 14 ilustra um exemplo de curva de Pareto para um problema de minimização de dois critérios (objetivos) A e B. A solução ótima é aquela com menores valores para A e B e as soluções não-dominadas são aquelas mais próximas de cada eixo do gráfico, por possuírem os menores valores dos critérios.

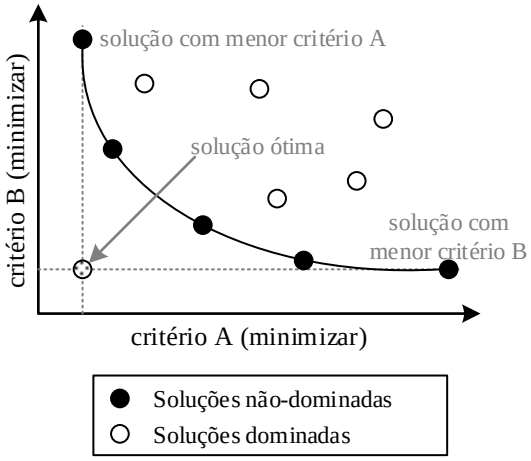


Figura 14 – Exemplo de curva de Pareto (Coello, Lamont, & Veldhuizen, 2006).

Na abordagem *fórmula-ponderada* muitas vezes cabe ao usuário informar os pesos para os critérios a serem otimizados com base em seus conhecimentos no domínio de aplicação. Na abordagem de Pareto, há interação com o usuário no final da execução do algoritmo. Conforme dito anteriormente, essa abordagem retorna um conjunto de soluções não-dominadas e cabe ao usuário a tarefa de escolher, com base em seus conhecimentos no domínio da aplicação, a melhor solução. Todavia, quando não se tem conhecimento prévio do problema em questão, a utilização da abordagem de Pareto pode apresentar desvantagens, como por exemplo, a dificuldade na escolha da melhor solução a ser utilizada na prática devido ao fato do usuário ter de escolher apenas uma solução dentre as disponíveis. Outra dificuldade da abordagem de Pareto é a dificuldade de trabalhar com diferentes níveis de prioridades de objetivos, quando um objetivo é significativamente mais importante que outro, considerando que a abordagem de Pareto assume que os critérios são igualmente relevantes (Freitas A. , 2004).

— *Abordagem lexicográfica*

Ao contrário da abordagem *fórmula-ponderada*, na abordagem lexicográfica o principal objetivo é atribuir diferentes prioridades a objetivos distintos e, então, focar na otimização dos objetivos de acordo com suas respectivas prioridades. Por exemplo, considere x e y como dois indivíduos da população, e a e b como dois objetivos (i.e., critérios de avaliação) e que a tem prioridade sobre b e que t_a e t_b são os limiares de tolerância t associados aos objetivos a e b, respectivamente. A abordagem lexicográfica trabalha com a seguinte análise: se $|a_x - a_y| > t_a$, então é possível escolher o melhor indivíduo apenas com base no critério a. Se não for possível escolher o indivíduo apenas com

base no critério a, a medida b deve ser avaliada. Caso $|b_x - b_y| > t_b$, então o melhor indivíduo pode ser escolhido considerando o critério b. Se ainda assim continuarem empatados, utiliza-se o valor absoluto de a para determinar o melhor indivíduo. Dependendo do critério de escolha, o melhor valor tanto para a como para b pode ser o maior ou o menor deles. Se os indivíduos representarem métodos computacionais para resolução de um problema, a e b podem representar a precisão e o número de atributos analisados, sendo priorizado o indivíduo com maior precisão e, em caso de empate, aquele indivíduo que utilizou menos atributos. Considere, por exemplo, dois métodos computacionais m_1 e m_2 utilizados na solução de um problema qualquer. Sabendo que m_1 possui uma precisão de 90% e considera 20 atributos, e m_2 possui uma precisão de 70% e considera 15 atributos, o método m_1 seria preferível pela maioria, por apresentar uma maior precisão. Contudo, segundo a abordagem de Pareto, os dois métodos são equivalentes, ou seja, o método m_1 não domina o método m_2 (Freitas A. , 2004).

Um problema que surge nesta abordagem é a definição dos limiares utilizados, o que não é uma tarefa trivial. Uma solução comum é usar métodos estatísticos para definir os limiares, como por exemplo, o desvio padrão, que permite rejeitar uma hipótese nula de diferença não significante entre dois valores com certo grau de confiança. Todavia, é necessário que o usuário defina o grau de confiança, normalmente definido como 95% de confiança estatística (Freitas A. , 2004).

3.2.3.3 Seleção

Algoritmos genéticos são técnicas de otimização que minimizam ou maximizam uma dada função. A etapa de seleção é importante para determinar os espaços de busca evolucionários, à medida ela é utilizada para melhorar as chances de sobrevivência dos indivíduos mais aptos. Assim, são selecionados os indivíduos que formarão a próxima geração da população, pela avaliação do grau de aptidão dos indivíduos, medido pela função de *fitness*. Os indivíduos mais aptos possuem maior probabilidade de serem escolhidos como reprodutores. Cabe ressaltar que nem sempre o melhor indivíduo, com maior aptidão, será um dos selecionados para formar a próxima geração da população pelo fato da escolha do indivíduo ser probabilística.

A garantia que os melhores indivíduos de uma população sejam mantidos na próxima geração é dada pelo *elitismo*. Esse conceito foi introduzido por Kenneth De Jong (Jong, 1975) e pode ser utilizado em vários métodos de *seleção*. Ele garante que os melhores indivíduos não sejam perdidos após as operações de *cruzamento* e *mutação*.

Segundo Mitchell (Mitchell M. , 1996), vários pesquisadores observaram melhoras nas performances de seus algoritmos quando utilizam a técnica do *elitismo*. Todavia, o número de melhores indivíduos deve ser atribuído com cautela, pois, se ele for muito grande pode resultar num domínio dos indivíduos mais aptos desde as primeiras gerações, levando a uma perda rápida de diversidade da população. Vários métodos de *seleção* de indivíduos podem ser utilizados, como por exemplo, os métodos roleta e torneio, descritos a seguir, dentre outros específicos para determinados problemas (Abuiziah & Shakarneh, 2013).

— *Roleta*

Neste método cada indivíduo possui um intervalo de números de uma roleta, proporcional ao seu valor de *fitness*. A roleta é então girada α vezes, onde α corresponde ao número de reprodutores. A cada novo giro um número β é marcado. Cada número possui igual probabilidade de ser marcado. O indivíduo selecionado é aquele que possuir o número dentro do intervalo β. Ao final do processo são obtidos α indivíduos reprodutores selecionados. Por exemplo, supondo uma população de indivíduos exemplificada na Tabela 9, formada por 10 indivíduos $(a, b, \dots, j)$, com seus respectivos valores de *fitness* e a proporção do intervalo de números da roleta. A proporção de cada indivíduo é calculada dividindo seu valor de *fitness* pela soma do valor de *fitness* de todos os indivíduos.

Tabela 9 – Exemplo de uma população de indivíduos com seus respectivos valores de fitness.

INDIVÍDUO	**A**	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**	**J**
Valor de *Fitness*	6	5	10	8	8	3	9	6	1	7
Proporção	10%	8%	16%	13%	13%	5%	14%	10%	2%	11%

A vantagem deste método é privilegiar um pequeno grupo de indivíduos, por possuírem maior aptidão e, consequentemente, maiores chances de serem selecionados. Contudo, isto pode aumentar a pressão seletiva. Tal pressão impede uma maior diversidade da população nas gerações iniciais. Esta restrição na diversidade da população leva o algoritmo a convergir rapidamente para ótimos locais ao invés de ótimos globais, o que pode não ser o esperado pelo usuário.

— *Ranking*

A convergência prematura para ótimos locais no espaço de solução do problema é diminuída no método de seleção por *ranking*, onde os indivíduos da população são ordenados de acordo com

seu *fitness* e a probabilidade de escolha é atribuída conforme a posição do indivíduo no *ranking*. O problema de grandes diferenças entre as chances de escolha dos indivíduos é eliminado, porque a probabilidade de escolha dos indivíduos de posições subsequentes no *ranking* (i e $i+1$) é independente das diferenças entre os valores de *fitness*, i.e., ela é feita, tendo como base, apenas nas posições dos indivíduos no *ranking* (Mitchell M. , 1996).

Um ponto positivo deste método é diminuir a alta possibilidade de seleção de um grupo pequeno de indivíduos altamente aptos em relação aos demais, reduzindo assim a pressão da seleção quando a variação na aptidão é muito elevada. Em alguns casos, porém, desconsiderar o valor absoluto do *fitness* de um indivíduo é desvantajoso, pois pode ser importante saber a diferença entre os valores de aptidão de indivíduos consecutivos no *ranking*, principalmente nas gerações finais da população, quando é esperado que eles estejam nas melhores soluções para o problema.

— *Torneio*

No método de *torneio* um número t de indivíduos é escolhido aleatoriamente na população, com probabilidades iguais, para participarem de um *torneio*. Quanto maior o valor de t, maior a chance dos indivíduos mais aptos serem selecionados, o que aumenta a pressão seletiva. Normalmente é utilizado o valor $t = 2$ a 5. Neste método, não é preciso escalonar a aptidão dos indivíduos e nem ordená-la. Existe também um parâmetro $k = [0, 1]$ que indica a probabilidade do melhor indivíduo do *torneio* ser selecionado. Se o valor de k for o maior, i.e., $k = 1$, o método de *torneio* é dito determinístico, por selecionar sempre o melhor indivíduo da população. Após cada rodada do *torneio*, todos os participantes, incluindo o vencedor, voltam à população original onde todos podem ser selecionados novamente para os próximos torneios. Um novo *torneio* é realizado até que o número de reprodutores seja alcançado (Eiben & Smith, 2003).

O método de *seleção por torneio* é muito utilizado na literatura, devido a sua simplicidade e pelo fato de ser possível controlar a chance dos indivíduos mais aptos serem selecionados, modificando apenas o valor do número de indivíduos do *torneio*. Além disso, não é necessário normalizar o *fitness* dos indivíduos (assim como no método de *ranking*), porque o fato de um indivíduo ser várias vezes o mais apto que outro, não determina a sua escolha; tem-se apenas qual deles é o melhor (Eiben & Smith, 2003).

3.2.4 *Operadores genéticos*

Os operadores genéticos *cruzamento* e *mutação*, descritos a seguir, são os mais conhecidos na literatura. Eles são também os principais mecanismos de busca dos algoritmos genéticos usados para explorar regiões desconhecidas do espaço de busca (Mitchell M. , 1996).

3.2.4.1 *Cruzamento*

O *cruzamento* combina dois cromossomos (i.e., pais) para produzir um novo cromossomo (i.e., filho), sendo, por este motivo, importante na produção de uma nova geração (Mitchell M. , 1996). A ideia por trás do operador de *cruzamento* é que a produção de um novo cromossomo pode ser melhor que os cromossomos dos pais, se forem utilizados, na reprodução, as melhores características de cada pai. Cada par de pais pode gerar um par de filhos, onde cada filho é formado pela combinação dos genes dos pais, passados dos pais para filhos pela operação de *cruzamento*. O *cruzamento* é aplicado de acordo com uma probabilidade, denominada taxa de *cruzamento*, que varia de 60% a 90%. Caso esta taxa seja igual a 0%, os filhos gerados serão iguais aos pais. Na literatura são apresentados vários métodos de *cruzamento*, dentre eles o *cruzamento* de *um ponto* e *multiponto* (Otman & Jaafar, 2011).

O *cruzamento* de *um ponto* é aquele onde é determinada uma única posição p, e o primeiro filho (F1) recebe os genes do primeiro pai (P1) até a posição p, e os genes do segundo pai (P2) a partir da posição p. O segundo filho (F2), por sua vez, recebe os genes do segundo pai até o ponto p e os genes do primeiro pai de p em diante. A Figura 15 apresenta um exemplo no qual, dois indivíduos (linhas P1 e P2) representam os pais e os dois indivíduos (linhas F1 e F2) representam os filhos.

P1	0	1	1	0	0	0	0	1	0	0
P2	1	0	0	1	1	0	1	1	1	1
F1	0	1	1	0	1	0	1	1	1	1
F2	1	0	0	1	0	0	0	1	0	0

Figura 15 – *Cruzamento* de *um ponto*.

É possível realizar a operação de *cruzamento* com vários pontos de *cruzamento* entre pais e filhos, conhecido como *cruzamento* multiponto (Figura 16), ou trocar genes de maneira uniforme e

aleatória em cada execução, *cruzamento* conhecido como uniforme (Figura 17).

P1	0	1	1	0	0	0	0	1	0	0
P2	1	0	0	1	1	0	1	1	1	1
F1	0	1	1	0	1	0	1	1	0	0
F2	1	0	0	1	0	0	0	1	1	1

Figura 16 – Cruzamento multiponto.

P1	0	1	1	0	0	0	0	1	0	0
P2	1	0	0	1	1	0	1	1	1	1
F1	0	0	1	0	1	0	0	1	0	1
F2	1	1	0	1	0	0	1	1	1	0

Figura 17 – *Cruzamento* uniforme.

3.2.4.2 Mutação

A *mutação* é dada pela alteração aleatória no valor dos genes de um indivíduo com determinada probabilidade denominada *taxa de mutação*. Normalmente, a probabilidade é pré-determinada pelo usuário e aplicada em cada um dos filhos após a operação de *cruzamento* e antes da população final ser definida (Eiben & Smith, 2003). A alteração pode ser feita pela troca do valor de um gene qualquer por outro, pela adição ou subtração do valor do gene por algum outro valor aleatório ou simplesmente pela troca de valores de alguns genes por valores de outros genes do próprio indivíduo. A Figura 18 apresenta um exemplo de *mutação*, em que quatro genes aleatórios de um indivíduo são alterados.

Antes	0	0	1	0	1	0	0	1	0	1
Após	0	1	1	0	1	1	0	0	1	0

Figura 18 – *Mutação* gerada aleatoriamente.

A *mutação* visa, de um lado, melhorar a diversidade dos cromossomos da população, de outro lado, ela destrói a informação contida no cromossomo. Esta destruição faz com que sejam usados valores pequenos de taxa de *mutação*, normalmente entre 0.1% e 5%. Ela é suficiente para assegurar a diversidade da população (Eiben & Smith, 2003).

Tendo em vista sua capacidade de otimização, sólida fundamentação matemática, o algoritmo genético pode ser aplicado em vários domínios de aplicações.

3.2.5 *Aplicações do algoritmo genético*

Na literatura são apresentados vários trabalhos que utilizam algoritmo genético na otimização

de problemas (Kumar, Husian, Upreti, & Gupta, 2010) (Nielsen, Danoy, & Bouvry, 2013). Dentre eles, alguns autores já mostraram a utilização do algoritmo genético em modais de transporte, foco de nosso interesse, descritos a seguir.

Em (Han, et al., 1999), tem-se um estudo voltada a definição de uma estratégia ótima para a condução de um trem de passageiros. O objetivo é otimizar o consumo de energia. Em termos de modelagem do problema, cada cromossomo foi considerado uma estratégia de controle e seus valores referem-se às posições relativas entre as estações, para um percurso de teste com aproximadamente 920m. Como o objetivo era otimizar o consumo de energia, o valor da função de *fitness* foi subtraído de uma constante C_{max}. Os resultados obtidos mostraram uma melhora de tempo e consumo, quando aplicado o algoritmo genético, comparado a comandos preestabelecidos pela central de operações.

Em (Chang & Sim, 1997), tem-se outro trabalho cujo objetivo é gerar uma tabela de ações de condução de trens, considerando pontualidade (p), conforto dos passageiros (c) e consumo de energia (E). Na modelagem do problema, cada cromossomo é formado pelo conjunto de informações contidas na tabela de ações com número fixo de bits. Os genes representam a posição relativa entre as estações. Por exemplo, um cromossomo de tamanho 4, possui 4 decisões a serem tomadas entre duas estações, sendo que a ordem dos genes no cromossomo indica a ordem das decisões. As ordens foram separadas em dois grupos: genes ímpares e genes pares. Os genes ímpares representam o comando para redução de velocidade, enquanto que os genes pares representam os comandos para empregar força de tração para aumentar a velocidade. Os parâmetros do algoritmo genético foram utilizados: método de seleção por torneio e $fitness = 1/(E \times p \times c)$. Resultados, de cenários de 1000m, mostraram que o método pode ser aplicado em problemas de controle considerando múltiplos objetivos na função de *fitness*.

Ao otimizar o conjunto de equações que determinam os ângulos das rodas de um veículo para aprimorar a segurança e melhorar o controle de um veículo tracionado. O critério de parada do algoritmo utilizado foi baseado no tempo (7s). Foram utilizadas 20 gerações, cada geração contendo 10 indivíduos, com percentuais de *cruzamento* e *mutação* em 80% e 20%, respectivamente. Uma solução ótima foi encontrada após 300 iterações. Os resultados obtidos mostraram a efetividade do algoritmo genético em cenários reais com poucas gerações e poucos indivíduos (Amdouni, Jeddi, & Amraoui, 2013).

O algoritmo genético, conforme mencionado, é utilizado nesta pesquisa para adaptação dos casos. A adaptação é uma parte do método do *Raciocínio Baseado em Casos*, descrito na próxima seção.

3.3 RACIOCÍNIO BASEADO EM CASOS

Estudos iniciais do *Raciocínio Baseado em Casos* têm como marco a pesquisa de Schank & Abelson (Schank & Abelson, 1977), a qual propôs que os conhecimentos humanos sobre situações fossem registrados na forma de *roteiros* (Schank R. , 1975) (Watson I. , 1997). O conceito de *roteiros* foi descrito na teoria de dependência conceitual, usada para representar o significado de sentenças. Esta teoria tem como principal elemento a ideia que todos os conceitos podem ser representados em termos de poucos atos primitivos realizados por um ator ou objeto. Assim, toda memória é baseada em *episódios*, denominados *roteiros* e organizada em torno de experiências individuais. Já as memórias específicas são armazenadas como sugestões para formação dos *roteiros*, e permitem que os indivíduos façam as inferências necessárias para compreender certo episódio, preenchendo a informação que está faltando com base nos conceitos armazenados (Schank & Abelson, 1977)

A teoria de *roteiros* foi utilizada como base para um modelo de *Memória Dinâmica*, uma das mais importantes contribuições de Roger Schank (1983). O modelo sugere que os eventos sejam compreendidos em termos de *roteiros*, planos e outras estruturas de conhecimentos, assim como experiências relevantes. Uma *Memória Dinâmica* é capaz de organizar automaticamente as informações contidas em memória com base em novas experiências. Por este motivo, ela é considerada um sistema de aprendizado flexível e sem término. É deste modo que a memória humana funciona. Ela é capaz de organizar as estruturas de conhecimento, abstraindo, das informações recebidas, generalizações significantes a partir de nossas experiências e também é capaz de armazenar exceções para as generalizações criadas. Este dinamismo na organização da memória permite alterar a própria organização da memória quando novas experiências exigem (Schank R. C., 1983) (Schank R. C., Dynamic Memory Revisited, 1999). Ele também propôs o conceito de *Pacotes de Organização de Memória*, que utilizam a lembrança de experiências passadas associadas a estereótipos de situações para a solução de problemas e aprendizagem (Schank R. C., 1986).

Em 1989, continuando os estudos desenvolvidos, Riesbeck e Schank (Riesbeck, 1989) desenvolveram uma nova forma de aprendizagem, denominada *Raciocínio Baseado em Casos*. O raciocínio baseado em casos denota, além de um método de raciocínio particularmente independente de como os casos são adquiridos, um paradigma de aprendizagem de máquina que permite manter o conhecimento, ao atualizar a base de casos após um problema ser solucionado (Pant & Joshi, 2012). Alguns estudos caracterizam o raciocínio baseado em casos como um método de aprendizagem incremental, sendo o desenvolvimento de uma ciência e de uma tecnologia que usa a simulação computacional do pensamento e do raciocínio humano para mudar e compreender o mundo (Lin, Deng, & Peng, 2000).

No raciocínio baseado em casos, todo o esforço feito em uma situação passada para resolver um problema é reaproveitado em uma nova situação. Outro ponto relevante é que a reutilização prévia de uma experiência não se restringe apenas ao sucesso passado como um guia para a obtenção de novo sucesso, à medida que é possível também reutilizar insucessos de forma a antever futuras situações de falhas e evitá-las. Assim, não há necessidade de gastar recursos para corrigir esforços de potenciais falhas na situação atual (Riesbeck, 1989).

No que diz respeito à qualidade de uma solução baseada em casos, deve-se considerar em relação ao raciocinador a sua experiência, a capacidade de interpretar novas situações em termos de experiências prévias e a aptidão em adaptar e avaliar a situação recuperada. À medida que novas situações surgem, experiências anteriores são recuperadas e adaptadas para atender as novas situações, diversificando o conjunto de casos do raciocinador, o que resulta em um aumento da sua base de experiências. Quanto maior a base de experiências, maior é a capacidade do raciocinador em interpretar, adaptar e avaliar novas situações. Este ganho de experiências por meio de casos não ocorre em sistemas baseados em regras, por exemplo, onde o conhecimento do sistema é gerido por um conjunto de regras fixas, as quais necessitam ser alteradas, por especialistas para representarem novas situações/conhecimentos (Riesbeck, 1989).

Vários aspectos tornam o raciocínio baseado em casos aplicável na condução de trens de carga:

- Ao analisar as características do processo de condução de trens de carga, é observado que não há um conjunto de regras capaz de prever com exatidão quais ações devem ser executadas durante a condução, devido às inúmeras variações de características que formam um trem e as diversidades na condução de uma via férrea. Logo, os condutores tendem a buscar a exatidão das suas ações com a vivência e a memorização das experiências ao longo dos anos. Tais experiências, em sistemas que seguem a abordagem do raciocínio baseado em casos, são reorganizadas de forma similar;
- Durante uma viagem, o condutor relembra as ações praticadas em viagens anteriores para aplicá-las novamente, seja no mesmo trecho ou não. Eles fazem deste modo para tentar realizar uma viagem mais eficiente e reduzir os erros cometidos;
- Ao executar uma ação que impeça a continuidade de uma viagem (e.g. iniciar uma subida com uma potência que não permita o trem percorrê-la), um erro é cometido. Este erro pode ocasionar danos materiais e punições. Ao falhar uma vez, o condutor tentará não falhar novamente, i.e., aprenderá com a falha cometida e terá uma tendência de não cometê-la

novamente;

- Com o acúmulo de experiências, o número de tentativas e erros para encontrar a potência ideal a ser empregada durante uma viagem tende a diminuir, devido também à capacidade de adaptação do condutor a trens e vias com diferentes características.

Analisados os aspectos motivadores do uso de raciocínio baseado em casos na condução de trens, o próximo passo é discutir os conceitos e abordagens utilizadas em cada uma das etapas do raciocínio.

3.3.1 *O que é um caso?*

Um caso é a principal forma de representar o conhecimento em um sistema de raciocínio baseado em casos. Um caso registra um episódio em que um problema ou situação problemática foi total ou parcialmente resolvido (Kolodner, 1993). É possível também considerar um caso como uma abstração de fatos e ações (Althoff & Wess, 1991).

Há ainda a possibilidade de observar um caso de forma mais complexa, como uma conexão entre subcasos menores (i.e., um subconjunto de problemas), facilitando a resolução de problemas (Barletta R. , 1991). Por exemplo, supondo que um caso que representa a solução do problema de mover um trem, partindo de um ponto *A* até um ponto *E* de uma ferrovia, passando pelos pontos *B*, *C* e *D*. Este caso pode ser decomposto em subcasos menores, que levam a uma sequência de movimentações iniciando do ponto *A* até o ponto *B*, do ponto *B* até o *C*, do *C* até o *D* e, por fim, do ponto *D* até o ponto *E*.

A forma como um caso é representado, define os formalismos com os quais são estabelecidos os conhecimentos utilizados para solucionar um problema. A representação de itens do mundo real, possui normalmente uma complexidade elevada (em número de variáveis e restrições) e qualquer representação pode captar os detalhes apenas parcialmente. Assim, aspectos específicos a serem representados dependem dos objetivos do processo de raciocínio (von Wangenheim, Wangenheim, & Rateke, 2013).

Um caso pode ser representado como um conjunto de pares de atributo-valor, fixos para todos os casos ou variando entre casos individuais (de Mántaras, Bridge, & Mcsherry, 1997). Cada atributo é geralmente associado a um domínio (i.e., um tipo de dado). Esta abordagem é simples quando comparada às demais formas de representação e permite simplificar a programação de medidas de simi-

laridade, além de facilitar o armazenamento e recuperação de casos. Todavia, esta forma não representa informações estruturais ou relacionais, sendo recomendada basicamente para tarefas de diagnóstico que manipulam grandes bases de dados (de Mántaras, et al., 2005).

Outra forma de representação de um caso é orientada a objetos, onde casos são representados como objetos e cada objeto representa uma entidade ou conceito de um domínio de aplicação, com informações e serviços para entidades externas (Göker & Roth-Berghofer, 1999). Os objetos são definidos por classes, contendo operações e estados assim como na orientação a objetos. Tal abordagem é valiosa em domínios de aplicações complexos, onde casos podem ter estruturas variadas, e permite: *(i)* relacionar objetos de diferentes tipos (e.g., taxonomias, composições, associações e agregações); *(ii)* organizar os casos em classes e subclasses, utilizando o conceito de hierarquia de classes; *(iii)* armazenar os casos de forma mais compacta que a abordagem atributo-valor; *(iv)* representar informações estruturais e relacionais diretamente, por ser uma abordagem natural. Por outro lado, o cálculo da similaridade e a recuperação de casos de uma base tornam-se tarefas mais complexas e custosas, comparadas à abordagem atributo-valor, por necessitar de troca de mensagens entre objetos para obter informações usadas no cálculo da similaridade.

Outras formas de representação de casos, tais como: frames e textos são discutidos em (Bergmann, Kolodner, & Plaza, 2005) e (von Wangenheim, Wangenheim, & Rateke, 2013).

Até o momento não há um consenso de qual informação deve estar contida em um caso. Há um consenso entre os trabalhos desenvolvidos de que um caso deve ser representado de tal forma que seja funcional e permita adquirir informações úteis de maneira fácil, permitindo assim que o fluxo do raciocínio baseado em casos, descrito na próxima seção, seja o mais eficaz possível.

3.3.2 *Funcionamento do Raciocínio Baseado em Casos*

A elaboração de um novo caso para solucionar um problema-alvo é obtida pela recordação de um conjunto de casos que são adequados para revolver o problema-alvo (Kolodner, 1993). Tal processo envolve quatro diferentes estágios (cf. Figura 19), os quais utilizam conhecimentos específicos do domínio de aplicação, como oposto dos conhecimentos específicos contidos nos casos:

1. Recuperar o caso mais similar à descrição do problema-alvo;
2. Reutilizar os conhecimentos do caso recuperado para solucionar o problema-alvo por meio de adaptação ou justificativa de uma decisão;
3. Revisar a solução obtida no passo 2, tanto na forma de avaliação como na forma de crítica; e
4. Reter a experiência (ou parte dela) na tentativa de solucionar problemas futuros.

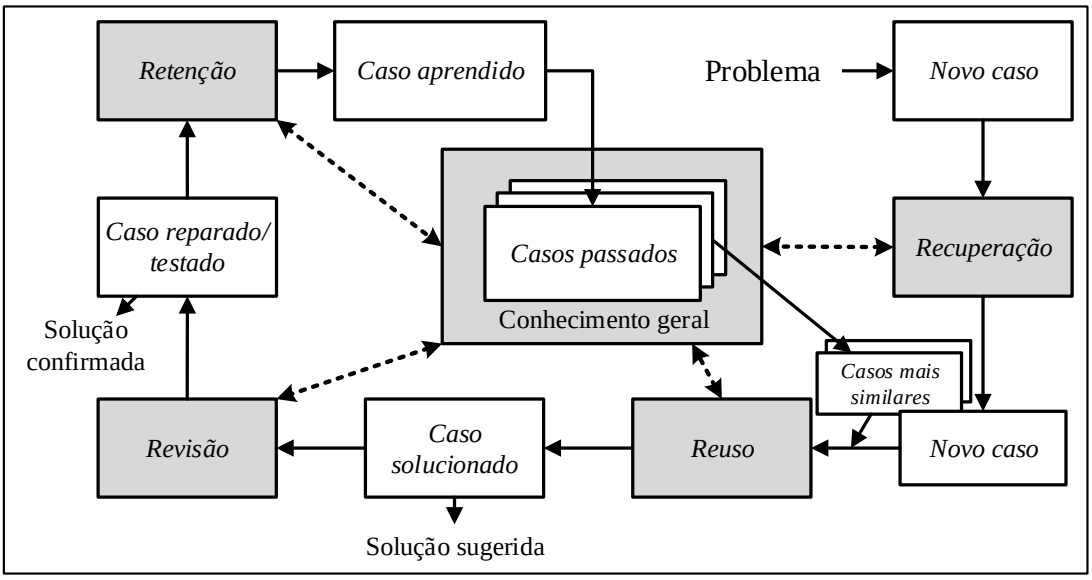


Figura 19 – Passos do raciocínio baseado em casos (Aamodt & Plaza, 1994).

Na área de jogos, o ciclo do raciocínio baseado foi aplicado como forma de aprendizagem de agentes de software na elaboração de planos em tempo real (Ontañón, Mishra, Sugandh, & Ram, 2010). A arquitetura apresentada difere-se por possuir camadas adicionais, percepção e atuação (cf. Figura 20), que permitem ao agente monitorar e executar planos, visto que, em problemas complexos a solução de um problema é subdividida em soluções menores — subplanos. Neste cenário, a execução de um único ciclo do raciocínio baseado em casos é insuficiente para solucionar o problema-alvo como um todo. Logo, o agente deve possuir a capacidade de raciocinar à medida que interage com um dado ambiente.

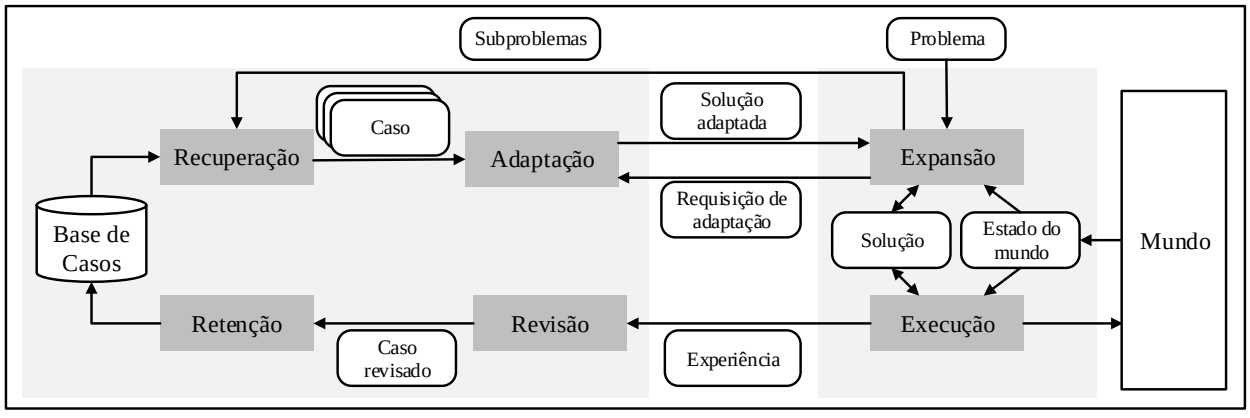


Figura 20 – Passos do raciocínio baseado em casos (Ontañón, Mishra, Sugandh, & Ram, 2010).

Apresentado o ciclo básico de funcionamento do raciocínio baseado em casos, nas próximas

seções, serão discutidas as etapas desse ciclo e as questões ligadas a cada algoritmo que pode ser usado em cada etapa, bem como as vantagens e desvantagens de cada algoritmo. Tal discussão é iniciada com foco na primeira etapa: recuperação de casos.

3.3.3 *Recuperação de casos*

Recuperar um caso é um processo supostamente similar a relembrar uma experiência passada registrada em memória. É um processo usado para encontrar na memória, ou na base de casos, um conjunto de casos que possam conter uma solução útil para o problema-alvo (Riesbeck, 1989). O conjunto de casos recuperados pode ser composto de um ou mais casos similares ao problema-alvo que servirão como entrada para o processo de adaptação.

O processo de recuperação pode ser descrito como um conjunto de tarefas que devem ser realizadas para recuperar um caso: assessoramento da situação, *matching* e seleção. O assessoramento da situação busca formular uma consulta representada por um conjunto de descritores relevantes da situação ou problema-alvo e o *matching* identifica um conjunto de casos suficientemente similares à consulta. Por fim, a seleção escolhe o melhor caso identificado na etapa de *matching* (Smyth & McKenna, Footprint-Based Retrieval, 1999) (von Wangenheim, Wangenheim, & Rateke, 2013).

O processo de recuperação tem por objetivo encontrar casos úteis no contexto da descrição do problema-alvo. Entretanto, é difícil determinar se um caso é útil ou não para resolver um determinado problema-alvo ou determinar o que torna um caso mais apropriado que outro presente na base de casos. A adequação de um caso para resolver um problema-alvo pode ser determinada apenas depois de ter sido elaborada uma solução ao problema-alvo com base no caso selecionado. Cabe então, ao mecanismo de raciocínio, elaborar uma solução com base em hipóteses para todos os casos. Porém, a elaboração de hipóteses em grandes bases de dados é muitas vezes impraticável devido ao elevado número de comparações possíveis, sendo que, para uma base de casos com *n* casos, o processamento dos n os casos em memória teria complexidade $O(n)$ (Kolodner, 1993) (von Wangenheim, Wangenheim, & Rateke, 2013). Uma alternativa para solucionar este problema é calcular a similaridade dos atributos do problema-alvo com os atributos de um caso recuperado. Esta alternativa parte da hipótese que problemas similares possuem soluções similares (de Mántaras, et al., 2005). Com base nesta hipótese, o critério *a posteriori* da utilidade da solução passa a ser reduzido ao critério *a priori* de similaridade de problemas. Esta forma de proceder é justificada visto que, em muitos domínios de aplicações, a similaridade de problemas implica na aplicabilidade da solução de um caso anterior sobre outro caso, se: *(i)* permitir resolver o problema-alvo de alguma forma; *(ii)*

evitar a repetição de um erro anterior; *(iii)* permitir resolver o problema-alvo de forma eficiente, i.e., mais rápida que uma forma heurística, por exemplo; *(iv)* oferecer a melhor solução ao problema-alvo de acordo com um dado critério; *(v)* oferecer ao usuário uma solução cuja lógica seja compreensível para ele.

A avaliação do desempenho da recuperação de casos pode ser calculada segundo a *velocidade* de recuperação de um caso ou de acordo com o impacto na *qualidade da solução* recuperada. Uma avaliação em termos da *precisão da classificação* é possível somente se as classes preditas no conjunto de testes estiverem representadas no conjunto de treinamento, o que não acontece em casos onde há uma única instância daquela classe na base de casos. Algumas medidas apropriadas para analisar o desempenho da etapa de recuperação de casos envolvem *precisão* e *recall* (Aha, Breslow, & Muñoz-Avila, Conversational Case-Based Reasoning, 2001). Contudo, muitas aplicações de raciocínio baseado em casos utilizam conhecimentos acerca do domínio de aplicação para avaliar o desempenho da etapa de recuperação de casos (de Mántaras, et al., 2005).

Na literatura há várias abordagens utilizadas na recuperação de casos, como uso de computação paralela, a organização das bases de casos baseado na similaridade entre os casos dividindo-os em grupos, busca em populações de casos distintas, inclusão de procedimentos de validação de casos na etapa de recuperação, entre outras (de Mántaras, et al., 2005). Uma pesquisa na base de dados de artigos da IEEE contendo os termos “case-based reasoning” + retrieval, realizada no ano de 2014, apresenta como resultado aproximadamente 450 artigos, uma parte deles descrevendo técnicas de recuperação de casos, outros descrevendo estudos acerca da recuperação de casos em problemas específicos. Nestes últimos, é possível notar que, dentre os trabalhos estudados ao longo desta tese, a maioria deles utiliza o algoritmo *k*-nn para recuperação de casos, com a similaridade calculada utilizando a distância Euclidiana. Esta conclusão é mencionada também em (Zia, Akhtar, Mughal, & Mala, 2014).

Uma forma de diminuir a carga computacional empregada na recuperação dos casos é criar vocabulários de indexação de casos. Os vocabulários devem conter as informações explícitas acerca dos casos e capturar as características determinantes para a recuperação de casos (Schank, et al., 1990) (Leake D. , 1992). Para criar este vocabulário, o primeiro passo é definir quais atributos, chamados de *índices*, serão usados na comparação entre o problema-alvo e o caso recuperado. O objetivo da indexação é determinar índices para os casos a partir de um conjunto de características do problema que melhor representam um caso; para orientar a avaliação da similaridade entre os casos. Os índices de um caso são combinações de seus atributos mais importantes. Eles devem permitir distinguir o

problema-alvo dos demais casos e também permitem identificar casos úteis segundo a descrição do problema-alvo. Estes atributos são chamados de *informações indexadas.* Já os atributos de um caso não utilizados na recuperação, mas presentes na descrição do caso são denominados de *informações não indexadas* (Barletta & Mark, 1988).

O conjunto de índices deve antecipar possíveis resultados, endereçando os propósitos dos casos que serão usados. Eles devem permitir ampliar a base de casos com a inclusão de novos casos e também permitir que os casos sejam reconhecidos e utilizados em futuras tarefas (Mckenna & Smyth, 2001) (Watson I. , 1997). Para atingir estas características, os índices devem ser representados por características relevantes à descrição dos casos e relevantes para orientar os cálculos de similaridade (Kolodner, 1993). De modo formal, o conceito de similaridade é a definição de uma medida numérica de distância. Uma medida de similaridade global de dois casos *A* e *B*, sobre um universo *U*, é uma função do tipo $sim(A, B)$: $(U \times U \rightarrow [0,1])$ que determina a similaridade entre uma pergunta e um caso, considerando todos os *n* índices do caso $A = (a_1, \dots, a_n)$ e do caso $B = (b_1, \dots, b_n)$.

A seleção apropriada de uma função de cálculo distância tem impacto direto no valor da similaridade (Chen & Jiang, 2013). Como já dito, a maioria dos trabalhos encontrados na literatura utiliza o algoritmo *k*-nn, descrito em (Duda, Hart, & Stork, 2000), e usando a distância Euclidiana para computar a similaridade entre um problema-alvo e os casos contidos na base de casos (Aha, Kibler, & Albert, 1991) (Corchado & Laza, 2003), (Pant & Joshi, 2012), (Chakraborty, Ghosh, Ranjan, Garnaik, & Debnath, 2010), (Watson & Rubin, 2008), (shik Shin & Han, 1999) (Gómez & Plaza, 2012). Essa abordagem apresenta bons resultados na etapa de recuperação (Pant & Joshi, 2012). O algoritmo *k*-nn também pode ser usado combinando uma medida de similaridade própria do domínio de aplicação (Tiako, Jayaweera, & Islam, 2011) e (Tiako, Jayaweera, & Islam, 2012).

A recuperação de casos pode ser feita também utilizando árvore de decisão combinada com o algoritmo *k*-nn (Chen & Jiang, 2013). A árvore de decisão é utilizada para minimizar as informações redundantes e encontrar o menor subconjunto de casos que descreve completamente o conjunto de instâncias de treinamento, baseado no cálculo de ganho de informação (Quinlan, 1993) (Borges A. P., et al., 2012). Quando recuperado um caso da base, é feita uma busca na árvore de decisão e selecionam-se casos que casem com o problema-alvo, cuja similaridade é calculada com base na distância Euclidiana. Todavia, o método demanda muito tempo para gerar a árvore de decisão quando a base é muito grande.

Nos últimos anos, vários mecanismos de cálculo de similaridade foram desenvolvidos na ten-

tativa de melhorar a etapa de recuperação, dentre eles, tem-se os mecanismos baseados em: (i) métricas diretas (e.g., Euclidiana, Manhatan, Minkowski), (ii) transformação (e.g., distância de grafos, recuperação guiada por adaptação), (iii) teoria da informação (e.g., baseado em compreensão de conteúdo), e (iv) métricas emergentes (e.g., técnicas de clusters, florestas aleatórias) (Cunningham, 2009). A busca de mecanismos que permitam aprimorar o cálculo da similaridade é justificada pelo fato de que, quanto mais similar ao problema-alvo for o caso recuperado, melhor será o resultado do raciocinador. Tendo em vista a diversidade de métricas, uma taxonomia foi proposta por Pádraig Cunningham em 2009, para categorizar as novas técnicas no contexto das técnicas consolidadas; nesta proposta é possível encontrar vários trabalhos que utilizam estas métricas. Num primeiro momento, tal variedade de métricas disponíveis, permite, por exemplo, expandir a etapa de recuperação — do ciclo do raciocínio baseado em casos — em camadas, permitindo o uso de várias métricas de recuperação de casos.

Recuperado o conjunto de casos mais similares, o raciocinador reutiliza um ou mais casos similares para resolver o problema-alvo.

3.3.4 *Reuso de casos*

Dado um caso recuperado, a solução sugerida dele é objeto de uma tentativa de reutilização para solucionar o problema-alvo. É considerada uma tentativa visto que nem sempre uma solução recuperada e aplicada no passado se encaixa perfeitamente ao problema-alvo. A reutilização consiste em adaptar o caso recuperado ao problema-alvo (Kolodner, 1993) (Aamodt & Plaza, 1994) (Spalzzi, 2001) (Mitra & Basak, 2005).

Uma adaptação pode ser considerada como um par situação/ação. A situação contém as diferenças entre o caso recuperado e o problema-alvo. A ação captura as atualizações para o caso recuperado. Esta captura pode ocorrer na forma de substituição (i.e., novos valores para o caso reutilizado), na forma de transformação (i.e., componentes do caso podem ser adicionados, removidos ou alterados), na forma de aplicação de heurísticas com conhecimentos mais aprimorados para executar a atualização (Kolodner, 1993) (Craw, Wiratunga, & Rowe, 2006).

Os métodos de adaptação desenvolvidos foram classificados quanto à necessidade de conhecimento sobre o domínio de adaptação, capacidades de adaptação e tipo de conhecimento utilizado (Mitra & Basak, 2005).

3.3.4.1 Classificação dos métodos de adaptação

Os métodos de adaptação de casos podem ser classificados quanto aos requisitos de conhecimentos do domínio (pouco ou muito), as capacidades dos métodos de adaptação de casos (método *adaptativo* ou método *não adaptativo*), e o tipo dos conhecimentos necessários à adaptação (*estático* ou *dinâmico*) (Mitra & Basak, 2005).

Os métodos de adaptação pouco dependentes do conhecimento (em inglês, *Knowledge-Lean*), dependem pouco ou não dependem do conhecimento do domínio da aplicação para realizar as adaptações. Nesta categoria estão os algoritmos genéticos, à medida que o espaço de representação pode ser guiado pelo domínio, mas a evolução não é necessariamente guiada pelo domínio da aplicação. Estes métodos podem ser classificados em métodos estocásticos ou métodos determinísticos. No primeiro, a tarefa é modelada como um problema de satisfação de restrições e as soluções são obtidas usando algoritmos de busca ou algoritmos evolucionários, como por exemplo: as redes bayesianas, os algoritmos genéticos e os algoritmos de satisfação de restrições. No segundo, os métodos determinísticos utilizam técnicas de adaptação baseadas em substituições e *replay* derivacional.

Os métodos de adaptação dependentes do conhecimento — ou de conhecimentos intensivos, apresentam maior acoplamento aos conhecimentos de domínio quando comparados com os métodos de adaptação anterior. Três categorias distinguem estes métodos de adaptação: *(i)* métodos de substituição — onde às partes selecionadas dos casos recuperados são substituídas por partes do problema-alvo, sendo necessários conhecimentos de domínio para determinar as substituições adequadas; *(ii)* métodos transformacionais — que alteram a estrutura da solução, necessitando conhecimentos de domínio para tal; *(iii)* recuperação de casos baseada em *ranking* de casos relevantes — no qual a adaptação é feita com base na média dos pesos dos valores dos atributos com maior medida de relevância e os conhecimentos de domínios são necessários para *ranquear* os casos recuperados.

A classificação baseada na capacidade de adaptação é baseada em algoritmos de aprendizagem de máquina e se dividem em métodos *adaptativos* e *não adaptativos*, e dependentes da implementação. Os métodos adaptativos utilizam aprendizagem de máquina e se dividem em: adaptação baseada em modelo de substituição, *replay* derivacional e busca orientada em memória. Nos métodos *não adaptativos*, a adaptação de um caso é feita por meio de regras estáticas, sem a incorporação de um mecanismo de aprendizagem. Nesta categoria estão os algoritmos genéticos, os *frameworks* de raciocínio bayesianos e a programação por restrição. Por último, os métodos dependentes da implementação podem ser programas de modo *adaptativo* e *não adaptativo*. Entre os métodos estão: adap-

tação baseado em substituição, *ranking* de casos recuperados e método de transformação. Por exemplo, se um sistema armazena o último processo de adaptação ou transformação utilizado, ele é considerado adaptativo.

Por fim, a classificação pode ser baseada nos conhecimentos utilizados na adaptação: *estático* ou *dinâmico*. No primeiro, os conhecimentos expressos na forma de regras ou modelos, são inseridos no sistema por um especialista e não é alterado após a inserção. Porém, nem todos os sistemas que utilizam regras ou modelos são estáticos. Os sistemas baseados em regras podem utilizar conhecimentos de domínio ou conhecimentos abstratos. Já os conhecimentos dinâmicos podem ser alterados a partir das experiências adquiridas na adaptação de casos. Algumas formas de adaptar casos baseado em conhecimento são:

- **Baseado em regras**: converte regras abstratas em regras específicas de domínio;
- **Busca em memória de casos**: à medida que os casos são resolvidos com sucesso, o sistema armazena os passos da solução em uma memória para reuso futuro;
- **Adaptação de casos**: os casos são usados para armazenar informações sobre a adaptação como forma de adaptar um caso, incluindo técnicas de busca em memória; e
- **Pesos**: eles podem ser usados na adaptação baseada em modelos — que utiliza um modelo de domínio para ajudar na adaptação das soluções — ou no ranking de recuperação de casos. Há dois tipos de pesos: pesos de *matching* e pesos de características. No primeiro, os pesos são usados para determinar a similaridade dos casos. No segundo, os pesos representam uma importância relativa. Os valores precisos dos pesos não são fornecidos por especialistas; eles são gerados pelo sistema de forma indutiva.

Contudo, antes de adaptar efetivamente um caso, é necessário identificar o que precisa ser adaptado em um caso.

3.3.4.2 Identificando o que precisa ser adaptado

Dado um caso recuperado, para que ele seja usado em uma nova situação, precisa-se identificar o que deve ou não ser adaptado no caso recuperado para atender o problema-alvo. A identificação pode ser realizada de três formas: *lista de verificação*, *sistema de manutenção de raciocínio* ou um *propagador de restrições* (Kolodner, 1993).

A *lista de verificação* permite ao sistema fazer um conjunto de comparações entre o problema-alvo e o caso recuperado, identificando desta forma o que precisa ser alterado. Criada a *lista de verificação*, os itens identificados são ajustados. Uma vez feita à adaptação, o sistema pode usar outra *lista de verificação* para identificar possíveis problemas e auxiliar na identificação de problemas que irão frequentemente ocorrer, criando uma espécie de padrão comportamental.

Outra forma de identificar o que necessita ser adaptado é por meio da utilização de um *sistema de manutenção de raciocínio* ou por um *propagador de restrições*. Nestes casos, deve-se determinar o que não está de acordo com o problema-alvo face o caso recuperado. No *sistema de manutenção do raciocínio* é realizada uma inferência sobre o que precisa ser adaptado. Já nos sistemas que utilizam *propagação de restrições*, as restrições violadas no caso recuperado são adaptadas.

Ainda segundo Janet Kolodner (1993), há seis etapas para identificar o que precisa ser corrigido em uma solução passada, comparando-a com o problema-alvo. As etapas são:

1. Verificar as diferenças nas especificações com o problema-alvo, conectando os aspectos especificados do problema-alvo com os aspectos da solução, o que vai auxiliar na identificação das mudanças necessárias. Este método permite identificar características que faltam na nova solução, mas estão na solução antiga, e vice-versa. Normalmente, sistemas que usam as diferenças entre a especificação de um problema-alvo para escolher o que adaptar, também usam estas diferenças para escolher a estratégia de adaptação a usar;
2. Aplicar uma *lista de verificação* capaz de identificar problemas recorrentes que, se presentes, devem ser corrigidos. Esta forma ocorre quando o método anterior se torna computacionalmente inviável, (i.e., casos com muitas descrições, quando características importantes que guiam a adaptação requerem inferências complexas). A *lista de verificação* registra um conjunto de testes que verificam a adequação de uma solução; ela é formada por um ou poucos padrões que podem ser aplicados para identificar potenciais problemas;
3. Comparar a solução proposta com os objetivos e restrições do problema-alvo, e identificar as inconsistências. Um modo de fazer isto é transferir a solução passada para o novo caso, propagar as restrições do problema-alvo na estrutura da solução, identificar violações e adaptar a solução para eliminar as violações;
4. Fazer uma projeção dos resultados da solução, visando identificar o que pode dar errado e então reparar tais problemas. A projeção pode ser feita usando um modelo, o qual simula a solução com base em um modelo causal (similar a um modelo mental de como o sistema fun-

cionará) de como ela deve se comportar. Porém, em alguns domínios os modelos causais podem não ser precisos, nestes, uma projeção baseada em casos é recomendada. Na projeção baseada em casos, casos com situações passadas são lembrados, e seus resultados examinados para verificar se serão consistentes com o objetivo do problema-alvo. Se não forem, a solução sugerida é reparada. Uma terceira forma de projetar os resultados é por simulação computacional, como em sistemas reais ou complexos, onde a simulação real é inviável;

5. Realizar uma análise causal dos resultados pode mostrar as falhas que precisam ser reparadas. Na projeção é buscado antecipar o que poderia dar errado e compensar antes de executar. Uma forma de identificar erros é simular a execução da solução, analisar a saída, e raciocinar sobre o que poderia ter sido feito melhor. Se os resultados da simulação forem insatisfatórios, faz-se uma análise causal do que pode ter causado a falha;
6. Identificar alterações já feitas, na medida em que elas podem requer novas adaptações para compensar as mudanças na solução.

Ao identificar que o caso precisa ser alterado, um método de adaptação deve ser aplicado. Na literatura existem inúmeros métodos de adaptação de casos, cabe ao raciocinador identificar como alterar o caso, seguindo os passos: *(i)* identificar exatamente o que precisa ser alterado. *(ii)* encontrar uma estratégia de adaptação ou reparo apropriada e *(iii)* escolher entre vários métodos de adaptação.

A identificação do que precisa ser alterado deve iniciar pela tentativa de fazer modificações fáceis. É preferível *substituir ao invés transformar*, à medida que é mais fácil encontrar substitutos para determinados itens que são pouco conectados do que aqueles que são mais conectados a outras partes de uma solução (Kolodner, 1993). De modo geral é mais fácil trocar itens que possuem poucas restrições conectando-os com outros componentes de uma solução, do que trocar itens muito conectados. Quanto mais similares forem dois itens, mais simples será a substituição. Por outro lado, quanto mais central for o componente do qual ele faz parte, menos apropriado é substituí-lo e adaptar a estrutura do caso é mais apropriado.

Devido a todos estes critérios que devem ser considerados na decisão do que adaptar e devido à complexidade como os critérios interagem uns com os outros, criar um algoritmo capaz de decidir o que adaptar e aplicar a qualquer tipo de problema é uma tarefa complexa. Anos antes da classificação proposta por (Mitra & Basak, 2005), Janet Kolodner já havia identificado os critérios para determinar o que deve ser adaptado em um caso, com base em fatores como:

- **Conectividade** de um item com outras partes de uma solução, o que afeta a facilidade de adaptação, pois, quanto mais conectado um item é, mais difícil será de adaptá-lo;

- **Centralidade** de um item para com o componente de uma solução, o que afeta tanto a facilidade quanto a conveniência de uma adaptação, pois itens centrais tendem a ser altamente conectados com outras partes de uma solução. A forte conectividade torna mais difícil de encontrar substitutos para tais itens e é menos apropriado trocar um item central do que transformá-lo em não central. Se um item central não pode ser transformado para corrigir as inconsistências, é apropriado trocar toda a solução da qual ele faz parte do que encontrar um substituto;
- **Proximidade do substituto em atenuar** os efeitos de ambos em outras dimensões, das supracitadas. Uma substituição que pode ser difícil de realizar por causa da densidade de conexões. Porém, ela pode ser facilitada pela disponibilidade de um substituto próximo.

Já a escolha de uma estratégia de adaptação apropriada para adaptar ou reparar um caso pode ser feita (i) ao associar os modos de executar a adaptação aos métodos de identificação do que precisa ser adaptado; (ii) ao associar alguns itens com substitutos próximos; e (iii) ao associar a estratégia às circunstâncias em que são aplicadas heurísticas de adaptação indexadas de acordo com as situações.

Por último deve ser feita a escolha do que alterar (Kolodner, 1993). Esta escolha pode ser baseada em um conjunto de diretrizes para guiar o processo de escolha de qual método de adaptação utilizar, devido à grande quantidade de métodos existentes e a não existência de um algoritmo genérico de adaptação, dentre elas:

- Tentar aplicar estratégias de adaptações mais específicas do que gerais. Isto permite tentar soluções melhores antes. As adaptações mais específicas são aquelas indexadas por descritores concretos do problema-alvo e aqueles descritores apontados pelos métodos que escolheram o que adaptar. Estes tendem a utilizar heurísticas de adaptação com propósitos especiais e heurísticas de ajuste de parâmetros. Mais especificamente, adaptações aplicáveis tenderão a resultar em soluções mais apropriadas;
- Tentar estratégias fáceis antes das difíceis, o que leva o raciocinador a gastar menos esforços na adaptação. Exemplos de estratégias simples com pouca análise: busca local, ajuste de parâmetros e busca especializada;
- Escolher estratégias com maior probabilidade de sucesso do que aquelas que são mais propicias a falhas caso as estratégias anteriores sejam iguais;
- Aplicar estratégias complexas somente quando estratégias específicas e simples falharem;
- Usar estratégias propícias a falhas somente: (i) quando estratégias propícias a boas soluções tenham um potencial de solução muito menor que estratégias propícias a falhas; (ii)

quando as consequências de falhas são baixas; ou (iii) quando há boas chances da potencial falha ser capturada com menos esforço ou custo antes da solução ser implantada.

Em nosso trabalho, a identificação do que é necessário alterar em um caso é realizado por meio de uma *lista de verificação*, formado por um conjunto de normas que garantem a movimentação do trem, como identificação de (i) restrição de velocidade máxima, (ii) patinagem, (iii) falta de força e (iv) consumo excessivo. Recuperado um caso, o primeiro passo é verificar se cada ação sugerida é passível de aplicação via *lista de verificação*. Esta verificação permite modificar o caso para eliminar quaisquer umas das violações de normas existentes. Os resultados da adaptação são projetados via simulação computacional, visto que uma validação real é custosa e, em caso de falhas, pode ocasionar danos à via férrea e ao trem.

Analisado este conjunto de diretrizes para a escolha de qual método de adaptação utilizar, realizou-se um levantamento na literatura com o objetivo de identificar quais métodos de adaptação de casos já foram ou ainda estão sendo estudados.

3.3.4.3 Métodos de Adaptação

Os métodos de adaptação podem ser classificados em uma estrutura multidimensional, de acordo com a forma com que a adaptação é guiada, a tarefa executada na adaptação e onde a adaptação é realizada. A seguir são descritos brevemente alguns métodos de adaptação apresentados na literatura.

— *Adaptação nula*

É a forma mais simples de adaptação, onde é feita uma busca por um caso similar. A solução contida no caso recuperado é executada (total ou parcialmente), **sem qualquer adaptação**, i.e., a solução do problema-alvo é simplesmente copiada do caso recuperado (Pal & Shiu, 2004). Na adaptação nula, cabe ao usuário modificar a solução, se necessário. A adaptação nula é útil em tarefas como classificação simples, onde há raciocínio complexo e soluções simples.

Esta forma de adaptação foi usada em comércio eletrônico (von Wangenheim, Wangenheim, & Rateke, 2013), para predição de um conjunto de serviços automatizados em *casas inteligentes* (Tinghuai, Yong-Deak, Qiang, Meili, & Weican, 2005), no *framework* proposto para o desenvolvimento de aplicações personalizadas (Coutand, 2009) e na escolha do melhor número de telefone ou e–mail para redirecionar uma mensagem no sistema CAMS (Nakanishi, Tsuji, & Hakozaki, 1999) e

em sistema médico (Pant & Joshi, 2012).

— *Adaptação transformacional e substitucional*

A solução do caso similar recuperado é transformada em uma nova solução para satisfazer o problema-alvo com a aplicação de conhecimentos específicos ao domínio da aplicação. A transformação envolve mudanças na estrutura da solução (Kolodner, 1993) (Mitra & Basak, 2005). A adaptação transformacional é usada quando não existe um substituto apropriado e permite que, em certas condições, elementos da solução sejam reorganizados (Pal & Shiu, 2004). Na reorganização, alguns elementos podem ser adicionados ou removidos (parcialmente ou totalmente), manipulando operadores de adaptação e/ou regras transformacionais que modificam a solução de acordo com as diferenças entre os atributos de um caso recuperado e do problema-alvo (von Wangenheim, Wangenheim, & Rateke, 2013).

Na adaptação transformacional, o foco não está em como o problema é resolvido, mas na equivalência de soluções. Isto requer um conhecimento significativo e bem-definido da aplicação sob a forma de operadores transformacionais associados a um regime de controle para organizar a aplicação destes operadores. Dependendo de quanto um caso precisa ser adaptado, a adaptação transformacional pode ser diferenciada em adaptação substitucional e adaptação estrutural.

A adaptação substitucional troca partes dos atributos da solução antiga que são inválidas, instanciando-os novamente, à medida que eles conflitam ou contradizem os requerimentos do problema-alvo (Kolodner, 1993) (Pal & Shiu, 2004). A adaptação significa, por exemplo, recalcular diversos parâmetros que dependem dos relacionamentos entre o problema-alvo e o caso recuperado. Esta forma de adaptação define uma estratégia promissora quando o caso recuperado é muito semelhante ao problema-alvo e demanda pequenas modificações, sem alterações estruturais. Este modelo caracteriza-se por ser independente de domínio de aplicação (Mitra & Basak, 2005). A adaptação substitucional pode ser de seis tipos: reinstanciação, ajuste de parâmetros, busca local, memória de consultas, busca especializada e substituição baseada em casos. Pesquisas como (Jurisica & Glasgow, 1995), (Craw, Wiratunga, & Rowe, 2006) e (Chakraborty, Ghosh, Ranjan, Garnaik, & Debnath, 2010) utilizaram a adaptação substitucional de casos.

Na adaptação estrutural, toda a estrutura do caso é adaptada, o que implica em uma modificação mais profundamente no caso, em particular devido que à solução de um problema é menos estereotipada. Ela suporta a reorganização de elementos e a solução permite a adição e/ou a remoção desses elementos em certas condições. A escolha em modificar ou não a estrutura de uma solução

depende de relações entre a descrição do problema-alvo e do caso recuperado. Esta forma é usada, por exemplo, no sistema DÉJÀ VU (Smyth & Cunningham, 1992).

A Tabela 10 ilustra alguns métodos de adaptação baseados em transformação e substituição, discutidos em (Kolodner, 1993), sendo que os métodos baseados em substituição são os mais utilizados (Mitra & Basak, 2005).

Tabela 10 – Métodos de Adaptação (Kolodner, 1993).

MÉTODO	GUIADO POR	TAREFA	OPERA SOBRE
Reinstanciação.	Estrutura.	Substituição.	Valores.
Ajuste de parâmetros.	Senso comum.		Valores.
Busca local.	Especialização. Senso comum.		Valores. Estrutura.
Consulta em memória.	Especialização.		Valores. Estrutura.
Busca especializada.	*Ad hoc*. Senso comum.		Valores. Estrutura.
Substituição baseada em casos.	Casos.		Valores.
Transformação de senso comum.	Senso comum.	Transformação.	Valores. Estrutura.
Reparação guiada por modelo.	Causal.	Transformação. Substituição.	Valores. Estrutura.
Adaptação especializada e uso de heurísticas de reparação.	Especialização. Senso comum. Causal.		Valores. Estrutura.
Replay derivacional.	Casos.		Valores. Estrutura.

Reinstanciação é usada para instanciar uma solução antiga com novos objetos. Como no caso da elaboração de pratos no sistema CHEF (Hammond, 1986), onde um novo prato é criado a partir da reinstanciação de um prato anterior. Por exemplo, CHEF utiliza a reinstanciação para criar um prato de frango e ervilhas, ao substituir estes ingredientes por bife e brócolis em uma nova solução.

Ajuste de parâmetros é uma heurística para ajustar parâmetros numéricos de uma solução antiga. Este método recai em heurísticas especializadas que relacionam as diferenças na especificação da entrada com as especificações da saída do planejador. JUDGE faz uso desta abordagem na análise de sentenças criminais, ao analisar a gravidade de um novo crime, comparando-a com crimes anteriores. Logo, para um problema-alvo (crime) com menor gravidade, JUDGE atribui sentenças menores às aplicadas anteriormente (Bain, 1986). Uma utilização mais recente desta forma de adaptação é apresentada em (Vong & Wong, 2010) no ajuste de motores de veículos.

A *reinstanciação* substitui um conjunto inteiro de regras em um caso a cada vez, mas algumas vezes é necessário substituir apenas algumas partes de uma solução antiga para torná-la aplicável ao problema-alvo. A *Busca local*, por sua vez, provê um modo de buscar em uma hierarquia de abstração de alguns conceitos, uma estrutura de conhecimento auxiliar para substituir um valor ou estrutura que não atenda o problema-alvo. Em CHEF, por exemplo, se em um cardápio antigo eram servidas laranjas para sobremesa, mas não há mais laranjas disponíveis, a busca local permite buscar em uma rede semântica ingredientes para encontrar frutas semelhantes à laranja que possam servir como substitutas.

A *busca local* é eficiente quando o conceito procurado existe na hierarquia de memória e pode ser facilmente encontrado. Contudo, em cenários onde não é possível prever os relacionamentos entre os conceitos na hierarquia da memória, o método de *consulta em memória* é mais indicado, por buscar, nas estruturas de conhecimento ou na memória dos casos do raciocinador, conceitos descritos parcialmente. Por exemplo, ao buscar um cardápio para um jantar onde o prato principal é um ensopado, o raciocinador, pode retornar uma sugestão. Contudo, se esta sugestão não agradar o paladar dos convidados, nova busca local é realizada. Se não for encontrado na memória do raciocinador nenhum prato contendo os termos específicos desejados (no caso ‘ensopado’), a busca local deve procurar por termos mais amplos, o que pode ser custoso computacionalmente em termos de memória e tempo. A consulta em memória soluciona este problema da busca local por permitir a construção de uma descrição parcial de um item novo que substituirá o item buscado anteriormente; a busca é feita diretamente pela descrição. No exemplo dado, o sistema pode buscar por pratos cujos ingredientes sejam carne e batatas preparadas por estufa — processo usado para preparar um ensopado — retornando uma sugestão de sopa irlandesa, um tipo de ensopado.

Na *busca especializada* é feita uma consulta tanto nas estruturas dos conhecimentos como na memória de casos. Adicionalmente, nesse método é utilizada uma heurística especializada para guiar a busca na memória, indo além da busca local, por fornecer à memória dos casos instruções de como buscar o conceito necessário. Por exemplo, no sistema SWALE (Kass, Leake, & Owens, 1986), ao buscar as causas da morte de uma pessoa por ataque cardíaco inesperado, o sistema buscará um caso semelhante de uma pessoa morta pela mesma causa. Na análise da morte (de um caso antigo) de um corredor, os médicos concluíram que houve um problema cardíaco não detectado, estressando o coração e levando a morte do paciente. Logo, a morte antiga foi causada por um problema cardíaco durante uma corrida. Contudo, o novo caso não é de um corredor, contradizendo a solução recuperada (morte anterior), e o sistema procurará outro motivo que possa estressar o coração e levar ao ataque

cardíaco. Para controlar esta busca, é incluída uma heurística denominada *ação substituta: tema do ator* para guiar a busca, procurando na memória lugares onde respostas para esta heurística possam ser encontradas. No exemplo, a busca heurística será guiada para buscar em *temas* (esportes praticados pelo ator do novo caso), para tentar encontrar nestes temas ações que possam ter levado à morte, uma pessoa não esportista, por ataque cardíaco.

A *substituição baseada em casos* busca por um substituto para o problema-alvo em casos antigos que possam sugerir alternativas. Por exemplo, ao buscar um prato vegetariano para substituir uma lasanha, ao invés do raciocinador buscar no conjunto de macarrão para vegetarianos um prato similar à lasanha (como a busca em memória faria), o conjunto de casos buscados que incluem pratos com macarrão é consultado para encontrar uma alternativa para o problema-alvo. Nesta situação, por exemplo, o raciocinador pode sugerir a troca da lasanha por panqueca.

A substituição é apropriada quando existe na memória um item ou um conceito que pode ser substituído por um novo valor. Contudo, os métodos baseados em substituição, descritos até o momento, não podem ser utilizados se não existir um valor requerido na memória e nem se for necessário inserir ou apagar partes da solução antiga. Para estas situações são indicados os métodos baseados em transformação (Kolodner, 1993).

Na *transformação de senso comum* são usadas heurísticas para substituir, apagar ou adicionar componentes na solução. As heurísticas são determinadas com base no conhecimento comum acerca do problema e do conhecimento sobre a importância de cada item que compõe a solução do problema-alvo. Para funcionar, a transformação por senso comum requer (i) que os sistemas sejam capazes de identificar o componente do caso que está sendo transformado e precisa de reparo, (ii) as representações devem separar os componentes do artefato em primários e secundários de modo a permitir priorizar um componente sobre outro durante a troca e (iii) as representações devem manter relacionamentos internos entre os elementos dos problemas que estão sendo solucionados. Por exemplo, em uma situação onde é necessário criar um prato para o jantar de um cardápio judaico, e o raciocinador recupera a um prato de lasanha como o mais similar. Lasanha viola os princípios judaicos por conter leite e carne na sua composição. Para transformar este prato, é necessário identificar o que precisa ser transformado, o que requer, em primeiro, identificar a violação nas restrições e, em segundo, identificar o que precisa ser feito para remover a violação. Neste exemplo, a violação é composta da combinação de ingredientes que não podem ser utilizados, e removê-la consiste em apagar tais ingredientes. Para finalizar o prato, lasanha no caso, os ingredientes retirados serão substitu-

ídos por outros analisando as possibilidades com base no senso comum (e.g., textura dos ingredientes).

Na *reparação guiada por modelo* o conhecimento acerca de conexões causais sobre algum tipo de sistema ou situação é usado para guiar a adaptação. A reparação guiada por modelo é uma coleção de heurísticas que acessam modelos causais para transformar uma solução antiga aplicável no problema-alvo. Primeiro, o problema recuperado e o problema-alvo são comparados para extrair as diferenças de ambos. Em seguida, as diferenças são avaliadas usando algum modelo que permita caracterizar tais diferenças. Por fim, para cada diferença, uma heurística de reparação guiada por modelo é aplicada na solução antiga para criar uma nova solução.

Os métodos descritos até agora são métodos gerais para executar adaptações. Eles são considerados métodos independentes do domínio de aplicação, embora dependam com frequência de conhecimentos acerca do domínio de aplicação. Este conjunto de métodos é considerado, portanto, métodos fracos para uma adaptação efetiva.

Uma melhor eficiência da etapa de adaptação é obtida quando são utilizados conhecimentos acerca do domínio de aplicação, o que resulta em melhores resultados (Kolodner, 1993). É o caso dos métodos que se enquadram na *adaptação especializada e heurísticas de reparação*, onde são usadas heurísticas específicas do domínio de aplicação para adaptar os casos, as quais são indexadas de acordo com as situações em que são aplicáveis. Entre os métodos presentes nesta categoria estão: (i) adaptação específica de domínio; (ii) modificação de estrutura e (iii) reparação de propósito geral.

A adaptação específica de domínio utiliza regras de substituição e transformação específicas de domínio do problema, por exemplo, o sistema CHEF possui regras de domínio culinário que especificam, entre elas, que orégano pode ser substituído em pratos por uma combinação de manjerona e tomilho.

Na modificação da estrutura podem ser utilizadas heurísticas de adaptação com propósitos especiais para transformar a estrutura de soluções antigas, similarmente a transformação baseada em senso comum. Por exemplo, o sistema JULIA (Hinrichs, 1989) utiliza regras específicas para apagar partes redundantes da composição de um prato. E em seguida é utilizado à estratégia *dividir e conquistar* para expandir a estrutura de uma solução antiga e adicionar partes de componentes extras que duplicam alguma parte do prato que já esteja na receita, mantendo todas as restrições satisfeitas.

Quando uma solução proposta pelo procedimento de adaptação não é aplicável, uma reparação é executada. A reparação fornece uma realimentação do resultado da adaptação. Ela é utilizada para avaliar COMO a solução em andamento deve ser modificada para tornar-se mais adequada. Por

exemplo, quando um prato é provado e a pessoa diz que ele está azedo, o prato deve ser adaptado para mudar o seu teor de doçura. A adaptação pode ocorrer por uma ou mais regras especificadas no raciocinador; algumas delas aplicadas apenas durante a reparação e não durante a primeira adaptação.

Por último, *replay* derivacional consiste no processo de usar um método que permita calcular uma nova solução parcialmente ou total, usando os mesmos métodos pelos quais a solução antiga foi gerada. No exemplo do sistema JULIA, a elaboração de um prato vegetariano para uma pessoa alérgica a derivados de leite deve ser realizada. Por ser primavera, o raciocinador sugere servir um suflê de aspargos, porém, suflê contem queijo. JULIA raciocina sobre como adaptar o suflê para torna-lo apto ao vegetariano intolerante à lactose e lembra de uma refeição na qual adaptou um prato a base de tomate com queijo como ingrediente secundário. Neste prato, o queijo foi removido. Logo, JULIA repete este processo de inferência. i.e., o problema consistiu de um caso recuperado não condizente com as necessidades do problema-alvo e, para solucionar o problema-alvo, obtém um elemento substituto para o novo caso, recalcula como o valor antigo foi derivado e reproduz tal cálculo.

— *Adaptação gerativa e derivacional*

A adaptação gerativa replica o método, derivando a solução recuperada para o problema-alvo. Ela é usada em tarefas complexas, onde é preciso estender os conhecimentos contidos nos casos, utilizando uma forma de representação dos conhecimentos detalhados (Carbonell, 1985). O caso recuperado, além da descrição da solução, armazena os passos do processo derivacional que levou a uma solução no passado, incluindo: informações sobre decisões tomadas e suas justificativas, justificativas dos operadores aplicados, subobjetivos levados em consideração, alternativas geradas, caminhos de busca que falharam, entre outras informações relevantes.

A forma como o problema foi solucionado no passado é repassada para o contexto do problema-alvo, passo chamado de *replay*. Isto inclui as mesmas decisões ou decisões similares testadas na solução do problema-alvo. Se as decisões não puderem ser transferidas para o contexto atual, o mecanismo de solução de problemas gerativos tentará modificar o processo, buscando gerar uma nova sequência para a solução do problema-alvo.

A adaptação gerativa dá suporte para o controle de conflitos. Ela é mais adequada quando é necessário considerar as dependências entre os componentes de uma solução. Esta forma de adaptação requer que seja integrado ao raciocinador baseado em casos um solucionador de problemas (i.e. máquina de inferência), capaz de adaptar soluções sem o auxílio de um sistema de raciocínio baseado em casos. Devido à explosão combinatória que a máquina de inferência pode gerar quando aplicada

em problemas de configuração ou planejamento, é evitado o uso de uma estratégia que solucione o problema *do zero*. Portanto, é tarefa do sistema de raciocínio baseado em casos recuperar uma solução similar e adaptá-la ao problema-alvo. *One shot replay* e *replay* são duas estratégias usadas na reativação de uma solução.

A técnica de *one-shot replay* identifica quais partes da solução podem ser reutilizadas e são reativadas pelo solucionador de problemas (passo denominado *replay*). Depois, o solucionador toma as decisões por conta própria com base no modelo de domínio, podendo montar o restante da solução com partes diferentes daquelas contidas no caso recuperado. O principal desafio desta técnica consiste em decidir quando optar por um *replay*, pois se houver muitas interações e interdependências entre os passos do raciocínio, então será difícil decidir se um determinado passo poderá ser repetido na nova situação ou não (von Wangenheim, Wangenheim, & Rateke, 2013).

— *Adaptação composicional*

Na adaptação composicional, os novos componentes adaptados de vários casos passados são combinados para a geração de uma nova solução graças ao poder de recuperação, adaptação e composição de múltiplos casos (Redmond, 1990) (Sycara & Navinchandra, 1991).

A entrada consiste de uma representação do procedimento de operação. Os casos envolvem a carga e descarga de lotes de material e sua realocação em áreas de armazenamento. A base de casos é organizada de tal forma que cada solução completa é representada por um caso de alto nível que descreve a solução do problema-alvo em linhas gerais. Estes casos apontam para vários casos-detalhados que representam os componentes da solução. Esta forma de organizar a base de casos é denominada *hierarquia panorâmica*, onde a base de casos contém soluções completas representadas por um caso de alto nível descrevendo a solução em linhas gerais. Cada caso de alto nível possui referências para outros casos-detalhe que representam os módulos que ajudam a elaborar a solução completa.

O sistema de controle de veículos autônomos de chão de fábrica DÉJÀ VU (Smyth & Cunningham, 1992) utiliza esta forma de adaptação.

— *Adaptação baseada em ranking*

Aqui, todos os casos relevantes são selecionados como entrada e a adaptação é feita com base no peso médio dos valores dos atributos, i.e., os casos recuperados são ordenados de acordo com uma medida de relevância e a média das medidas é calculada. Sistemas de interpretação de imagens

de satélites e sistemas de controle em robótica utilizam esta forma de adaptação. A desvantagem deste método é, em geral, requerer que sejam utilizados apenas numéricos atributos para adaptação (Mitra & Basak, 2005).

— *Adaptação hierárquica*

A adaptação hierárquica permite reutilizar um único caso ou diferentes casos em vários níveis de abstração para refinar diferentes aspectos da solução. Os casos são armazenados em vários níveis de abstração e a adaptação é realizada de cima para baixo. O primeiro passo consiste em adaptar a solução no nível mais alto, onde os detalhes menos relevantes são omitidos. Em seguida, a solução é refinada passo a passo, com adição dos detalhes necessários até que se complete a solução (Mitra & Basak, 2005).

Um exemplo da utilização desta forma de adaptação é PARIS (*Plan Abstraction and Refinnement in an Integrated System*) (Bergmann & Wilke, 1996), onde os casos de planejamento no nível concreto são abstraídos em diferentes níveis de abstração. A abstração resulta em um conjunto de casos abstratos armazenados em uma base de casos. Desta forma, quando um novo problema surge, um caso abstrato, que satisfaça exatamente a descrição do problema-alvo é recuperado. Na fase de reuso, a solução abstrata é refinada por um planejador que realiza uma pesquisa dirigida em um espaço de estados, adicionando os detalhes não contidos na descrição recuperada para compor a solução do problema-alvo. O planejamento busca por sequencias de operadores concretos que levem a um estado também concreto. Contudo, para funcionar de modo adequado, este método de adaptação necessita de um modelo concreto e abstrato dos conhecimentos do domínio da aplicação.

— *Adaptação guiada por satisfação de restrições*

Esta técnica consiste em adaptar casos com base em escolhas que precisam ser feitas (variáveis), cada qual associada com um número de opções (o domínio das variáveis) e um conjunto de relacionamentos entre as escolhas (restrições). Uma solução válida é a atribuição de um valor para cada variável. Cada valor atribuído pertence ao domínio de uma variável. Um conjunto de atribuições é feito respeitando todas as restrições do problema. A adaptação é baseada no conceito de permutabilidade entre valores da solução de um problema. O método é capaz de determinar como uma atribuição de valor se propaga no conjunto de soluções e gera um conjunto mínimo de escolhas, o qual necessita ser alterado para adaptar uma solução existente a um problema-alvo (Purvis & Pu, 1995).

Técnicas de satisfação por restrição exigem esforços para extrair e interpretar os conhecimentos a respeito de um problema-alvo. O conhecimento pode ser modelado na forma de relações lógicas, expressões matemáticas ou até mesmo domínios de validação. O problema-alvo é submetido a um processo de raciocínio que manipula as variáveis para que obedeçam as restrições. Apesar de proverem todas as soluções possíveis e rapidamente, este método requer muito esforço para identificar e modelar os conhecimentos (Purvis & Pu, 1995).

Este modo de adaptação compreende de sete passos: (i) recuperar as restrições de um caso; (ii) selecionar o conjunto inicial de parâmetros para as restrições; (iii) resolver as restrições com um método de satisfação de restrições; (iv) determinar quando o sistema possui muitas/poucas ou só uma restrição; (v) se o sistema possuir muitas restrições deve-se repetir o conjunto inicial de parâmetros, quando o sistema tiver poucas restrições deve-se aplicar um filtro; (vi) mostrar a solução para um especialista e se não for boa modificá-la e voltar ao passo (iii); (vii) armazenar a solução se for aceitável.

A adaptação guiada por satisfação de restrições foi usada em (Roldán, Neágny, Lann, & Cortés, 2010) no processo de desenho de uma nova estação de gás comprimido. Apesar de obter bons resultados, os autores esperam usar, em trabalhos futuros, técnicas de aprendizagem de máquina para extrair conhecimentos da base de casos e usar este conhecimento na etapa de adaptação.

— *Adaptação usando aprendizagem de máquina*

Nas técnicas tradicionais de adaptação, os conhecimentos utilizados na adaptação de casos são obtidos por meio de entrevistas com especialistas do domínio de aplicação. Este processo pode ser demorado e exaustivo, além de tornar difícil a manutenção dos conhecimentos de adaptação devido à dependência do especialista (Mitra & Basak, 2005).

A vantagem da adaptação baseada em aprendizagem de máquina é o aprendizado automático dos conhecimentos necessários para adaptação a partir dos casos, sendo potencialmente um método robusto. As tarefas de manutenção e gerenciamento também se tornam melhor controladas por meio da reaprendizagem e retreinamento com novos casos. Além disso, a geração de heurísticas capazes de relacionar as diferenças entre os casos recuperados e a solução podem ser utilizadas para determinar a quantidade de mudanças requeridas para certos casos (Mitra & Basak, 2005). Contudo, a obtenção de bons resultados significativos requer um esforço computacional para identificar as heurísticas. Em estudos anteriores foram utilizadas técnicas de aprendizagem de máquina combinada à técnica de raciocínio baseado em casos (Borges A. P., et al., 2012) e os resultados, em termos de eficiência na

recuperação dos casos, foram os esperados para o nível de esforço empregado.

Diversos autores fizeram uso desta abordagem ao longo dos últimos anos, empregando diferentes técnicas de aprendizagem de máquina. Algumas destas pesquisas serão descritas a seguir.

- Árvores de decisão

As árvores de decisão facilitam a adaptação. Entretanto, elas são inicialmente abordagens custosas. O porte da base de casos pode ter uma implicação direta na geração de uma árvore, que pode demorar além de requer de muitos recursos computacionais.

O C4.5 foi utilizado na formulação de medicamentos por (Craw, Wiratunga, & Rowe, 2006), com objetivo de adaptar os componentes utilizados na elaboração dos medicamentos. Eles foram comparados diferentes métodos de adaptação: (i) a simples cópia da solução recuperada sem nenhuma adaptação (*Retrieve-Only*), (ii) a adaptação do caso mais similar (NN_A) e (iii) a incorporação de características adicionais durante a adaptação (CB_A). O C4.5 foi aplicado para cada conjunto de exemplos de adaptação e os conjuntos mais precisos são usados para formar a base de casos usada por CB_A. Logo, CB_A tem acesso somente aos casos adaptados que possuem as características selecionadas do problema para discriminar as ações da adaptação. A escolha dos componentes sofre diferentes influências relacionadas às quantidades de cada componente, dependendo do princípio ativo usado na elaboração dos medicamentos e também pelo fato de algumas propriedades competirem umas com as outras. Devido a estas influências, não houve um método de adaptação, entre (i e iii) que se sobressaísse em todos os experimentos, o que mostrou a utilidade do conhecimento capturado dos exemplos de adaptação; isto indica a importância de escolher a forma de aprendizagem que melhor generalize os dados de treinamento. Os resultados mostraram também que a escolha do método de adaptação depende das características do problema e da tarefa de adaptação. Além disso, a adaptação deve explorar os limiares dos atributos dos casos para garantir um equilíbrio entre eles, quando um atributo influência outro de maneira oposta.

A extração dinâmica de regras de adaptação usando árvores de decisão foi empregada no desenvolvimento de um sistema capaz de prever o perfil de compra de consumidores em *shoppings* e também na escolha das características do problema usadas na adaptação dos casos (Gouttaya & Begdouri, 2012) (Qi, Hu, & Peng, 2012).

O algoritmo C4.5 também foi utilizado como forma extrair regras de condução de trens (Borges, Ribeiro, Ávila, Enembreck, & Scalabrin, 2009) e como mecanismo de organização da base

de casos (Borges A. P., et al., 2012). A base de casos foi organizada em diferentes níveis especializações. Em um primeiro momento, um caso é recuperado. Se ele possuir especializações, então utilizava-se de um mecanismo baseado em aprendizagem para selecionar a especialidade a considerar. Contudo, os resultados, em termos de percentuais de acerto na tomada de decisão mostraram-se aquém do esperado.

- Lógica fuzzy

A lógica Fuzzy é uma lógica aproximada na qual os valores verdadeiros são subconjuntos *fuzzy* de um intervalo de unidades rotuladas, como alta, média e baixa. Ela possui três etapas principais, fuzzificação, inferência e defuzzificação. A fuzzificação é o processo encarregado de determinar o grau de aptidão de um valor no conjunto fuzzy. A inferência busca formular um mapeamento de uma entrada em uma saída usando operadores lógicos, funções e regras no formato *se-então*. A terceira etapa, defuzzificação, tenta reduzir o resultado inferido para um valor ou algo único (Zadeh, 1975).

A abordagem Fuzzy é utilizada na adaptação de casos para um problema de recomendação de itinerários baseado na preferência dos usuários (Mahdi, Soui, & Abed, 2014). As preferências dos usuários são expressas na forma de variáveis linguísticas. Um problema-alvo é solucionado por meio da combinação de partes de diferentes soluções. A adaptação recebe uma lista de casos similares, cujos valores são usados como entrada do processo de fuzzificação. A lista compreende de três casos mais similares, um caso contendo as preferências do usuário, um caso contendo as características do ambiente e outro contendo as características técnicas do dispositivo utilizado pelo usuário. O processo de inferência tenta acessar todas as regras ativas, as quais expressam relacionamentos entre as variáveis de entrada, e geram como saída uma única variável que corresponde à relevância da solução. Na defuzzificação, os resultados inferidos da similaridade são traduzidos em resultados numéricos específicos que indicam a relevância da solução. Contudo, esta abordagem necessita que as regras de inferência sejam implementadas com conhecimentos específicos do domínio de aplicação.

A lógica Fuzzy foi utilizada de modo similar em (Behbood, 2011), com o objetivo de tomar decisões no mercado financeiro. Um diferencial frente ao trabalho anterior foi a criação de regras de inferência a partir de redes neurais *fuzzy*. Todavia, o treinamento da rede neural contou com todas as características do problema para obter a rede. Essa abordagem é inadequada para o problema tratado nesta tese, à medida que há atributos (e.g., resistência total) cujos valores calculados são conhecidos após a determinação dos valores de outros atributos (e.g., potência, velocidade).

- Redes neurais

As redes neurais podem ser utilizadas para treinar algoritmos e obter conhecimentos necessários para adaptação de casos. Para isto, recuperado um caso, a adaptação é feita usando os valores dos casos recuperados para treinar a rede neural. Dado um problema-alvo, um caso similar é recuperado da base de casos, e são calculadas as diferenças entre ambos. As diferenças formam a entrada da rede. Em seguida deve ser obtida a quantidade de ajustes da rede. Isto é feito com base nas diferenças dos valores dos atributos do problema-alvo vis-à-vis o caso similar. Finalmente, os ajustes devem ser desnormalizados e usados para adaptar a solução antiga (Pal & Shiu, 2004).

As redes neurais *back propagation* já foram testadas de diferentes formas na etapa de adaptação (Lodhi, et al., 2003): (i) usando toda a base de casos para treinar a rede; (ii) utilizando apenas alguns casos mais similares ou (iii) usando três redes neurais, em uma espécie de comitê de especialistas do problema. Nos experimentos realizados, a abordagem formada por um comitê de especialistas obteve os melhores resultados. Contudo, dentre as conclusões obtidas, consta (i) a limitação de que os dados devem ser pré-processados antes de serem submetidos à rede neural para garantir bons resultados e também (ii) há a necessidade de executar os experimentos repetidamente para encontrar a melhor arquitetura da rede neural, i.e., aquela que não resulta em *overfiting* e (iii) que redes neurais, que não são capazes de generalizar a partir das instancias de treinamento, não adaptam os casos de modo efetivo.

Outros trabalhos que utilizaram redes neurais na adaptação de casos são (Craw, Wiratunga, & Rowe, 2006), (Jung, Lim, & Kim, 2009), (Butdee, 2011) e (Henriet, P-E., Laurent, & Salomon, 2014).

- Modelo bayesiano

O modelo bayesiano também pode ser utilizado como alternativa para adaptação de casos. O modelo também recebe como entrada um conjunto de casos recuperados, os quais são submetidos a um modelo *bayesiano*. O treinamento da rede deve ser feito utilizando os casos armazenados na base de casos. Dentre as vantagens deste método está o fato dele prover uma boa generalização e poder alcançar um alto nível de classificação mesmo com pouco número de amostras de treinamento. Porém, se o número de casos disponível for limitado, este método não apresenta bons resultados por não conseguir gerar um modelo genérico eficiente (Pal & Shiu, 2004).

— *Algoritmo genético*

O algoritmo genético pode ser utilizado para ajudar na adaptação de casos, em particular em situações onde o volume de casos é pequeno. Em termos práticos, ele pode ser utilizado para obter os conhecimentos de adaptação, ao contrário de algumas estratégias descritas anteriormente, que têm os conhecimentos de adaptação obtidos diretamente dos casos (Pal & Shiu, 2004).

A ideia principal do uso de um algoritmo genético é modificar a solução antiga. A população inicial é formada por casos recuperados da base de casos, mapeados para uma representação de genótipos. A modificação é feita por meio da aplicação de operadores de cruzamento e mutação. Ela pode ser expressa na forma de um vetor. Esse vetor codifica um cromossomo. A geração de descendentes é feita por meio da aplicação de operadores genéticos sobre um conjunto de cromossomos. Após aplicar várias operações genéticas sucessivamente sobre tais vetores, modificações graduais são observadas. Finalmente, os genótipos gerados são mapeados para seus fenótipos/casos correspondentes, inferindo valores de atributos e contextualizando a nova situação. Os cromossomos otimizados podem ser usados na adaptação, onde a solução adaptada é testada. A resposta do desempenho compreende no cálculo do valor de *fitness*. Cada gene representa uma característica diferente do indivíduo, sendo que a coleção de atributos usados para descrever um caso pode ser comparada com uma coleção de genes que formam um genótipo dos indivíduos. O valor de *fitness* pode ser avaliado por meio de um modelo específico de domínio ou por testes no mundo real, dependendo da aplicação. Este processo é repetido por muitos ciclos até que uma solução satisfatória seja obtida (Mitra & Basak, 2005).

A vantagem de utilizar algoritmo genético na adaptação dos casos é o mecanismo artificial de busca no espaço de soluções. Primeiramente, a busca é simultaneamente conduzida por uma população de pontos com igual probabilidade, ao invés de ponto a ponto. A evolução da população é aleatoriamente guiada por regras de transição probabilísticas, ao invés de usar métodos matemáticos determinísticos que não permitem a livre movimentação dentro do espaço de busca. O algoritmo genético não necessita de informações adicionais específicas (e.g., tamanho do passo, informação de gradiente, valores faltantes iniciais) para funcionar de modo apropriado. Estas características contribuem tornando-o mais robusto e flexível que outros mecanismos de busca. A sua principal limitação consiste na dificuldade de determinar o critério de parada, por ser difícil de decidir quando uma solução encontrada é aceitável ou ótima. Critérios como o tempo de CPU disponível para cálculos, número de gerações ou um limiar de distância entre os resultados simulados e os resultados desejados são critérios de parada comumente utilizados.

Vários trabalhos utilizaram algoritmo genético na adaptação de casos com sucesso em diferentes contextos:

- Na área ferroviária para determinar o custo de movimentação de um trem elétrico entre estações, elaborando uma tabela de caminhos antes do início da viagem (Chang & Sim, 1997);
- No planejamento de escalas de ônibus, em que cada gene do indivíduo correspondia à um determinado local de trabalho e alelos os funcionários que iria trabalhar neste local (Dias, de Sousa, & Cunha, 2002);
- Na adaptação de casos de elementos hidrodinâmicos para criação de modelos numéricos para estuários (ambiente aquático de transição entre um mar e um rio), combinando o poder de busca do algoritmo genético com pouco conhecimento específico do domínio da aplicação (Passone, Chung, & Nassehi, 2006);
- Para adaptar ações emergenciais no distrito de Shangai, onde foi empregado uma função de *fitness* multi-fator para elaborar uma solução composta por uma série de medidas emergenciais (Liao, Mao, Hannam, & Zhao, 2012).
- Para obter soluções ótimas no problema de escalonamento de tripulação em ônibus (Liu, Ma, Guan, Song, & Fu, 2012).
- O algoritmo genético combinado com raciocínio baseado em casos, mostrou-se (i) eficiente para lidar com modelagem numérica de aplicações que requerem a substituição de valores para um grande número de parâmetros e (ii) mais efetivo e mais rápido que a abordagem clássica do algoritmo genético, mesmo quando usadas 10% das informações contidas na população inicial, poucas gerações (15 ao todo), 50% de taxa de cruzamento e 1% de taxa de mutação (Passone, Chung, & Nassehi, 2006).

Ao analisar os trabalhos desenvolvidos com algoritmo genético, foi possível observar que eles apresentam bons resultados em cenários diversos. Apesar de ser uma técnica desenvolvida a vários anos, ela continua sendo utilizada em vários trabalhos por apresentar pontos que a colocam em destaque frente a outras técnicas de adaptação. Um destes pontos está no fato do algoritmo genético tornar dispensável o envolvimento do usuário durante a etapa de adaptação. Outro fator positivo surge na combinação do algoritmo genético com conhecimento específico do problema, o que o torna eficiente e robusto, com uma grande vantagem em termos de precisão e velocidade (Passone, Chung, & Nassehi, 2006).

— *Outras formas de adaptação*

Conforme vimos até o momento, há inúmeras formas de adaptação de casos propostas na literatura. Algumas foram destacadas ao longo deste trabalho, por aproximarem-se dos métodos comumente utilizados em pesquisas na área de *Inteligência Artificial*. Entre os métodos de adaptação estudados ao longo deste trabalho e não abordados com a mesma ênfase estão: adaptação de casos numéricos por meio de fórmulas análogas ao cálculo diferencial (Fuchs, Lieber, Mille, & Napoli, 2014); adaptação guiada por fluxos de tarefas (Minor, Bergmann, & Görg, 2014); adaptação guiada pela técnica de *Support Vector Machine* (Sharifi, Naghibzadeh, & Rouhani, 2013); a adaptação pela combinação de partes (Manzano, Ontañón, & Plaza, 2011); adaptação guiada um conjunto de adaptações por regressão (Jalali & Leake, 2013).

3.3.4.4 Considerações

Como já dito, há várias maneiras de adaptar um caso. A adaptação efetiva depende da natureza e da estrutura do conhecimento a ser implementado no módulo de adaptação. Depende também do conhecimento acerca das modificações válidas e de como selecionar modificações apropriadas e eficazes em determinada situação (Passone, Chung, & Nassehi, 2006). As técnicas tratadas na reutilização de casos tentam resolver os problemas envolvidos na adaptação de casos, dentre eles: quais aspectos da situação devem ser adaptados? quais modificações devem ser realizadas para esta adaptação? qual método aplicar para realizar a adaptação e como controlar este processo? (Leake D. , 1996).

A forma tradicional de adaptação aborda especialistas humanos por meio de entrevistas e codifica manualmente os conhecimentos obtidos representados por meio de uma tabela decisória, uma árvore semântica ou um conjunto de regras no formato *SE-ENTÃO*. Porém, além de ser um processo trabalhoso e demorado, a manutenção dos conhecimentos de adaptação adquiridos é complexa.

A escolha de quais partes da solução devem ser modificadas depende da facilidade da modificação de uma parte da solução, de quanto esta modificação irá satisfazer os requisitos da modificação, dos efeitos colaterais desta modificação e também de possuir os conhecimentos para realizar a modificação. Além disso, para permitir a adaptação automática de casos, dois aspetos são relevantes. Primeiro, quais são as diferenças entre o caso recuperado e o problema-alvo e, segundo, o que pode ser transferido do caso recuperado para a solução do problema-alvo. Para atender estes aspectos

normalmente utiliza-se estratégias gerais que são especializadas para cada domínio de aplicação (Althoff & Wess, 1991).

O uso de estratégias gerais de adaptação não é a melhor forma de adaptação (Neagu & Faltings, 2001). O método de adaptação escolhido varia de problema para problema. A adaptação pode ser *a simples substituição de um componente*, ou até mesmo a *modificação de toda a estrutura da solução recuperada*. Assim, a adaptação recai novamente sobre o domínio do problema e sobre conhecimentos específicos.

Observou-se que, em sistemas reais, se um raciocinador é altamente preciso, falta a ele características de independência do domínio de aplicação (Mitra & Basak, 2005), i.e., para fazer uma adaptação precisa, os conhecimentos de adaptação precisam ser muito específicos, como ocorre em sistemas médicos. Esta falta de independência limita a capacidade de adaptação. É observado que alguns processos de adaptação são altamente adaptativos, enquanto outros são altamente específicos (ou precisos), o que revela a impossibilidade de desenvolver mecanismos com grande poder de adaptação e fraca dependência dos conhecimentos do domínio da aplicação.

Apresentadas algumas formas de adaptação, em seguida a próxima etapa do ciclo do raciocínio baseado em casos é a revisão.

3.3.5 *Revisão de casos*

Uma solução gerada na fase de adaptação nem sempre é correta e aplicável. Surge então a oportunidade para reparar tal falha. Esta reparação consiste de duas tarefas. A primeira é avaliar a solução adaptada. Se ela está correta, o ciclo do raciocínio baseado em casos continua e a solução adaptada é armazenada na base de casos como um novo caso. Por outro lado, se a solução adaptada não é aplicável, a segunda tarefa é reparar tal solução (Kolodner, 1993).

A reparação de uma solução pode empregar conhecimentos específicos do domínio de aplicação ou informações fornecidas por um usuário. Esta etapa captura o resultado da aplicação da solução no ambiente por meio da monitoração automática de resultados ou pela interação com o usuário. Cabe ao sistema projetado oferecer uma interface que permita esta monitoração automática. A etapa de revisão pode ser demorada em alguns casos, em função, por exemplo da dependência da análise de um especialista ou pela exigência de um grande tempo computacional. Mesmo assim, o sistema pode aprender com o caso: desde que ele seja armazenado na base como um caso não avaliado, no formato ‘consulta + resposta’. Quando a resposta estiver disponível, o caso é armazenado como avaliado. Esta necessidade pode ocorrer por exemplo em sistemas médicos, onde a avaliação de um novo caso pode

necessitar que seja esperado dias até a recuperação de um paciente (Kolodner, 1993). Segundo essa referência, a reparação das falhas, quando detectadas, envolve a detecção de que partes da solução proposta contem falhas e a recuperação ou geração de explicações para estas falhas. Em geral, quando um sistema inteligente precisa explicar uma falha, ele já resolveu o problema, tentou aplicar a solução no mundo real e obteve alguma resposta sobre a falha. Caso contrário, o sistema pode ter simulado a aplicação da solução e ter concluído que a solução não irá gerar os resultados esperados.

As explicações de falhas são usadas para modificar a solução ou a forma como o sistema chegou a solução, para que o caso presente possa ser resolvido com sucesso e para evitar que a falha ocorra novamente no futuro. O sistema CHEF, por exemplo, faz uso de conhecimentos causais de domínio de aplicação para gerar automaticamente uma explicação do porquê determinadas falhas ocorreram, levando o sistema a não atingir as metas do plano ou da solução. Assim, CHEF aprende com as situações que poderão causar as falhas, usando uma técnica de aprendizado por explicação. Este subsistema está acoplado em uma memória de falhas que é usada na fase de adaptação para evitar que falhas se repitam no futuro.

Em tarefas de uso do raciocínio baseado em casos para o planejamento de ações, onde um caso é tido como um plano, ou parte de, a reparação é normalmente mais eficiente do que planejar novamente desde o início. Esta reparação pode ser feita pela adição ou remoção de ações via refinamento reverso (van der Krogt & de Weerdt, 2005). A estratégia de refinamento reverso remove primeiramente do plano parcial as ações que impedem o plano de atingir seus objetivos. A segunda fase corresponde refinar o plano parcial para satisfazer os objetivos. Por exemplo, imagine a seguinte situação: uma pessoa tem um jantar marcado e, na hora de sair de casa vê que um pneu do carro está furado. Um reparo simples no plano seria: adicionar ações para trocar o pneu e ir ao jantar. Neste caso, trocar o pneu pode custar muito tempo. Ao invés disso, a ação de dirigir até o jantar é removida e trocada pela ação de usar um taxi. Assim, a reparação não consistiu em adicionar ações, mas remover ações que iriam obstruir a solução. Para fazer a busca por alternativas, pode-se utilizar o algoritmo *tabu-search* (Glover & Laguna, 1997).

A reparação de casos pode ser feita apenas com extensões simples dos casos adaptados para ser eficiente e permitir a aplicação dos casos. A utilização de extensões simples, em problemas práticos, é vista como uma alternativa que apresenta melhores resultados do que a elaboração de todo o plano, tendo em vista que, o planejamento ‘do zero’ levaria muito tempo e despenderia muitos recursos computacionais (Kolodner, 1993).

Conforme dito anteriormente, a solução revisada deve ser retida para servir como novo conhecimento.

3.3.6 *Retenção de casos*

A retenção de um caso é a forma de estender os conhecimentos de um sistema baseado em casos. A nova solução obtida — ou o novo caso adaptado e revisado — é incorporada na base de casos existente, o que permite incrementar os conhecimentos a disposição do raciocinador de forma contínua (Kolodner, 1993) (von Wangenheim, Wangenheim, & Rateke, 2013).

Dentre os principais aspectos a serem considerados durante a retenção de casos estão a seleção: (i) de informações adequadas a serem armazenadas conjuntamente com o novo caso; (ii) da estrutura da informação e dos conhecimentos; (iii) da estrutura de índices para o acesso à informação; (iv) do tipo de integração para realizar na estrutura de conhecimentos existentes (Kolodner, 1993).

Basicamente, o gerenciamento de casos pode ser feita das seguintes formas (von Wangenheim, Wangenheim, & Rateke, 2013):

- **Sem retenção de casos:** forma usada nos sistemas de raciocínio baseado em casos mais simples, principalmente em domínios de aplicação bem compreendidos, que podem ser modelados de forma satisfatória já durante o desenvolvimento do sistema de raciocínio baseado em casos, sendo desnecessária a inclusão de novos casos para que o desempenho do sistema melhore. Exemplo: sistema de diagnóstico de problemas em certa linha de modelos de carros, onde o domínio é bem-conhecido.
- **Retenção de soluções de problemas:** tão logo um novo problema é resolvido, ele é armazenado para auxiliar na solução de problemas similares no futuro. Assim, sempre que um novo problema é resolvido, este pode ser incorporado à base de casos como um novo caso. Todo caso solucionado é armazenado na base e seus índices são atualizados, método chamado de força-bruta. Na retenção inteligente, o novo caso é filtrado e o conhecimento é avaliado por meio de técnicas inteligentes.
- **Retenção de documentos:** os conhecimentos são adquiridos independente da operação do sistema, de forma assíncrona ao processo de solução de problemas. Um exemplo é o uso de documentos em um sistema de gerência onde a retenção é separada do processo de solução, estando a retenção dependente da disponibilidade de novos conhecimentos sobre o domínio de aplicação, que são adicionados quando disponíveis.

A retenção de casos pode ser refinada em três fases: extração de conhecimentos, indexação de casos e integração na base de casos (Kolodner, 1993). A extração de conhecimentos consiste em selecionar as estruturas de conhecimentos que serão usadas para capturar as informações, como documentos, bases de dados, entre outras fontes. Com base em uma nova informação adquirida, um caso pode ser integrado em um caso existente, um novo caso pode ser construído ou ainda um caso similar pode ser generalizado para incluir a nova experiência. Contudo, em todas as três formas é necessário decidir de que forma usar a fonte de aprendizado. A indexação implica em decidir quais índices devem ser usados para recuperar e estruturar o espaço de busca. Uma solução trivial é o uso de todos os atributos do caso como índices, como aplicado em métodos baseado em sintaxe e raciocínio baseado em memória. Outra alternativa consiste na indexação em duas fases, onde são atribuídos índices primários ao modelo para explicar um caso e, quando um novo problema aparece, as características deste são propagadas no modelo e os estados que explicam estas características são usadas como índices para uma memória de casos, como no sistema CASEY (Koton, 1989). O passo final para reter novos casos é a integração, onde os conhecimentos adquiridos são unidos aos conhecimentos já existentes por meio da adição de novos casos à base de casos ou pela modificação ou remoção de casos antigos com uso de técnicas de aprendizagem.

A retenção equivale ao aprendizado de novos casos ou conhecimentos relevantes abstraídos de casos individuais. Para ser efetivo, o aprendizado requer a utilização de algoritmos de aprendizagem de máquina para extrair conhecimentos relevantes de experiências passadas, indexar estes conhecimentos e integrá-los nas estruturas existentes. No raciocínio baseado em casos, os casos são armazenados em uma base de dados, ao contrário de técnicas de aprendizado indutivo, em que os casos que levaram à formação de uma hipótese válida são esquecidos. Há vários métodos de aprendizado de casos: pode-se usar, por exemplo, somente exemplos válidos ou somente inválidos, ou ambos. Estes exemplos podem ser apresentados a todos de uma só vez, caracterizando um *aprendizado não-incremental*, ou aos poucos, no *aprendizado incremental* (von Wangenheim, Wangenheim, & Rateke, 2013).

A escolha de quais casos devem ser inseridos ou não em uma base de casos e quais critérios são usados e como definir a similaridade são tarefas que podem ser feitas com uso de algoritmos de *aprendizagem baseada em instâncias* (Aha, Kibler, & Albert, 1991) (Mitchell T. , 1997). Estes algoritmos aprendem a categorizar um conjunto de classes de objetos de forma incremental com base em exemplos de instâncias destas categorias, partindo do princípio que instâncias similares pertencem a categorias similares, e criam as categorias com base em similaridades detectadas.

Outro benefício de algoritmos de aprendizagem baseados em instâncias é que eles fornecem uma descrição conceitual do problema, a qual mapeia casos a categorias: dado um caso, a descrição prove uma classificação que é o valor predito para o atributo de categoria desse caso. A descrição conceitual inclui um conjunto de casos armazenados, e pode incluir também o desempenho classificatório no passado (erros e acertos). Ao contrário do que acontece na aprendizagem simbólica, nos métodos baseados em casos, os conceitos estão descritos de forma implícita, por meio da função de similaridade, da função de classificação e dos casos armazenados na base de casos (Richter, 1992).

Uma vez que o conceito foi aprendido, ele não pode ser lido diretamente a partir da classificação de um caso, à medida que ele está implícito no comportamento do sistema e não explicitamente representado. Nem o conhecimento da medida da similaridade e nem o conhecimento da base de casos é suficiente para a realização da classificação. O conceito aprendido é a soma das relações da base de casos mais a medida de similaridade. Sempre que o sistema altera seu conhecimento (adicionando ou removendo casos) de forma constante há um aprendizado (Kolodner, 1993). E o desempenho do algoritmo de aprendizagem pode ser medido de acordo com a generalidade, precisão, taxa de aprendizagem, custos de incorporação e requisitos de armazenamento.

A generalidade representa as classes de conceitos que podem ser aprendidos e descritos pelo algoritmo. A precisão representa a exatidão da classificação provida pela descrição conceitual. A taxa de aprendizagem é a velocidade com a qual a precisão aumenta durante o aprendizado e é um indicador de desempenho melhor do que a precisão de treinamento com um conjunto finito de exemplos. O custo de incorporação decorre da atualização da descrição conceitual por meio da inclusão de uma instância única, incluindo custos de classificação. Por fim, os requisitos de armazenamento correspondem ao tamanho da descrição conceitual, que em algoritmos de aprendizagem baseados em instâncias é dada como o número de instâncias que precisam ser salvas para ratificar uma classificação adequada (von Wangenheim, Wangenheim, & Rateke, 2013).

Uma descrição dos principais algoritmos de aprendizagem baseada em instâncias é mostrada em (Aha, Kibler, & Albert, 1991), onde são apresentados com detalhes os algoritmos IBL1, IBL2 e IBL3.

3.3.7 *Colaboração em Raciocínio Baseado em Casos*

Jacques Ferber, em (Ferber, 1999), apresenta uma classificação das interações entre os agentes de um sistema multiagente. As interações são classificadas por tipos de situações que podem ocorrer,

analisadas de acordo com os objetivos, recursos e habilidades dos agentes. Dentre as formas de interação apresentadas, a colaboração simples é a que melhor se encaixa no contexto da troca de planos entre os agentes. Na colaboração simples, os objetivos dos agentes são compatíveis, há recursos disponíveis para ambos, porém faltam habilidades para executar as tarefas. A falta de habilidade faz com que os agentes compartilhem conhecimentos e não necessitem de uma forma de coordenação entre as ações envolvidas. Outro exemplo é sistema multi-especialista, onde os agentes compartilham conhecimentos para solucionar um problema.

Diferentemente da colaboração simples, a colaboração coordenada supõe que o agente não possui recursos suficientes para executar a tarefa (Ferber, 1999). Da falta de recursos emerge a necessidade de coordenação dos agentes durante a execução da tarefa, o que não é o caso do problema desta tese. Assume-se que há comunicação entre agentes, parte-se do pressuposto que um dado agente necessite apenas dos casos de outros agentes para o planejamento de sua política de ações e não do auxílio deles durante o planejamento da política de ações.

A colaboração entre agentes no raciocínio baseado em casos foi estudada por (Prased & Plaza, 1996), onde os autores discutiram como memórias corporativas modeladas na forma de bibliotecas de casos distribuídas podem aprimorar a descoberta e exploração de experiências passadas. Os autores apresentaram duas técnicas desenvolvidas no contexto de raciocínio baseado em casos para acessar e explorar experiências passadas. Na primeira técnica, chamada *Recuperação Negociada*, os agentes recuperam e montam partes de casos locais a partir de diferentes recursos para formar um caso global útil na solução do problema-alvo. Na segunda abordagem, denominada *aprendizagem federada por pares*, a cooperação entre raciocínio baseado em casos se estabelece de duas formas: distribuída e colaborativa, que permitem aos agentes explorar as experiências e as especialidades de outros agentes para executar uma tarefa local.

A memória corporativa é a soma de todas as informações e fontes de conhecimento de uma organização, vistas como bases de casos distribuídas em um formato específico. Na recuperação negociada, a descoberta e exploração de soluções entre diferentes agentes ocorre de forma dinâmica e incremental durante a solução do problema-alvo e necessita da comunicação dos resultados parciais locais dos agentes para formar uma solução para o problema-alvo. Eles comunicam por meio de mensagens e negociam para resolver possíveis conflitos. Cada agente recupera subcasos da sua base de casos local e todos os agentes juntos elaboram um caso geral, a partir das partes recuperadas, para solucionar o problema-alvo estabelecido pelo usuário.

Na abordagem *aprendizagem federada por pares* há dois modos de cooperação entre os agentes introduzidos pelos autores: raciocínio distribuído baseado em casos (*DistCBR*) e raciocínio coletivo baseado em casos (*ColCBR*). Em *DistCBR* um agente a_i delega o controle para resolver o problema-alvo para outro agente a_j quando não é capaz de solucioná-lo. Já em *ColCBR* um agente a_i mantem o controle sobre a solução do problema-alvo e envia a um agente a_j o método de solução do problema e o problema-alvo, i.e., o agente usa a experiência acumulada por outros agentes enquanto mantém o controle de como o problema-alvo é resolvido. Em outras palavras, o agente a_i usa a memória dos outros agente como uma extensão da sua memória — como uma memória coletiva. Em ambas as abordagens, *DistCBR* e *ColCBR*, se o agente a_j fracassar em solucionar o problema-alvo, ele envia uma resposta negativa (i.e., sinal de falha) ao agente a_i e este deve tentar cooperar com outro agente de sua preferência (Ginty & Smyth, 2001).

Contudo, em ambas as abordagens há a necessidade de comunicação entre os agentes à medida que novos casos devem ser solucionados pelo agente durante a sua execução. A abordagem *ColCBR*, ilustrada na Figura 21, mostra-se interessante por permitir que o agente utilize o conhecimento de outros agentes na solução de seus problemas.

No modelo de *aprendizagem federada por pares*, quando um agente a_i recebe um problema p, primeiramente ele determina se ele conseguirá ou não solucionar p de acordo com seus conhecimento especializados. Tais conhecimentos correspondem, por exemplo, à capacidade de planejar uma viagem em um certo território, usando a sua própria base de casos para gerar um plano. Se não for capaz, um protocolo de colaboração entra em ação visando localizar casos, a partir de um conjunto de agentes similares, que possuam as experiências de planejamento necessárias para a_i, e solucionem p usando os casos emprestados. Especificamente, a_i faz um *broadcast* do problema-alvo p para cada agente colaborador e seleciona aquele que apresentar a melhor a solução, utilizando assim a capacidade de aprendizagem e experiência de outros agentes. A Figura 21 apresenta a arquitetura proposta pelos autores, onde U_n corresponde à n usuários que utilizam o sistema para resolver problemas p_n, com o auxílio de agentes A_n, onde cada agente possui a sua base de conhecimento CB_n. Experimentos mostraram que a abordagem de raciocínio baseado em casos colaborativa permitiu que um agente elabore planos em um ambiente desconhecido emprestando casos de outros agentes, transferindo planos de rotas entre usuários. Outro ponto mostrado pelos autores foi que o método de raciocínio baseado em casos colaborativo estabeleceu planos de rotas mais rápido que o algoritmo A*.

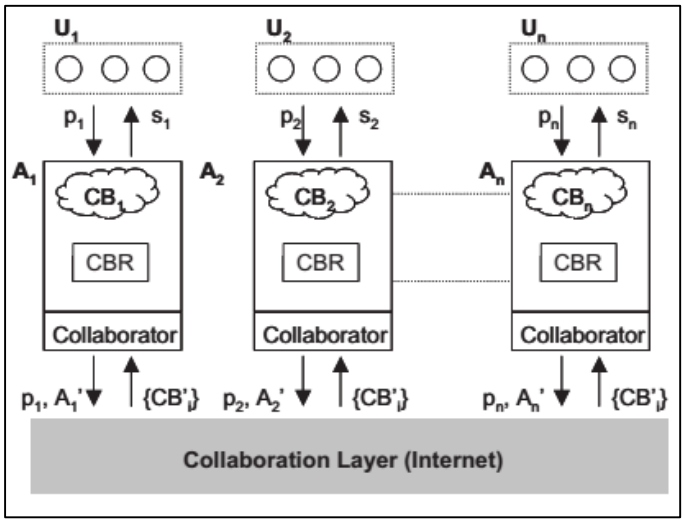


Figura 21 – Exemplo de uma arquitetura de raciocínio baseado em casos colaborativa (Ginty & Smyth, 2001).

Uma abordagem semelhante foi proposta por (Leake & Sooriamurthi, 2003), chamada *múltiplo raciocínio baseado em casos*. Em tal abordagem, um problema-alvo é submetido a várias bases de casos externas e o caso mais similar é recuperado. Esta abordagem estende o raciocínio baseado em casos a múltiplas bases de casos, onde um agente complementa automaticamente sua base de casos conforme necessário, enviando problemas para bases de casos externas. O uso de múltiplas bases de casos tem sido defendido como forma de melhorar o desempenho de grandes bases de casos, por meio da manipulação de subconjuntos de bases de casos separadamente. A divisão da base de casos pode acelerar o processo de recuperação de casos e melhorar a cobertura dos casos, quando necessário (Leake & Sooriamurthi, 2002). Resultados mostraram que a abordagem *múltiplo raciocínio baseado em casos* apresentou um desempenho melhor da taxa de acerto quando comparada à abordagem padrão, onde a base de casos contém todos os casos.

Outras pesquisas sobre colaboração em raciocínio baseado em casos têm estudado como e sob que circunstâncias conhecimentos oriundos de casos de um domínio *fonte* podem ser utilizados em novos problemas (Klenk, Aha, & Molineaux, 2011). Por consequente, as pesquisas estão ligadas a transferência de aprendizagem que ocorre quando, após adquirir experiências de aprendizagem na solução de problemas em certo domínio (chamado de problema *fonte*), o mesmo aprendiz explora esta experiência para melhorar seu desempenho e aprender em novos domínios (problema-objetivo). A transferência de aprendizagem consiste em treinar um sistema sob um conjunto de tarefas e condições e medir o efeito da aprendizagem em um diferente, mas relacionado, conjunto de tarefas e condições. Nesta tese, a transferência de aprendizado ocorre quando um agente é treinado para conduzir em determinado trecho ou com determinada configuração de trem e é submetido a novas situações

com características de trens e trechos diferentes.

A aprendizagem de máquina tradicional assume que a tarefa a ser executada (i.e., classes e funções objetivo da classe) e o domínio de execução (i.e., características do espaço e distribuição das instâncias) permanecem inalterados entre problemas passado e o problema-alvo. Na transferência de aprendizagem, estas afirmações são flexibilizadas, uma vez que os problemas passados e objetivo podem envolver diferentes tarefas e domínios. Esta transferência de domínio e tarefa, se encaixa no contexto de raciocínio baseado em casos em três diferentes categorias: para transferir aprendizagem, para aprender o problema e para transferir conhecimentos (Klenk, Aha, & Molineaux, 2011).

Raciocínio baseado em casos pode ser usado diretamente como um método de transferência de aprendizagem, onde o ciclo do raciocínio baseado em casos considera os três passos da transferência de aprendizagem. Aqui, o conhecimento aprendido e transferido pode ser a própria base de casos após o treinamento nos problemas *fonte*. Durante a aprendizagem de novos problemas, o mesmo ciclo do raciocínio baseado em casos pode ser usado, atualizando a mesma base de casos. Assim, o raciocínio baseado em casos não fará distinção entre os casos passados e os casos novos (Klenk, Aha, & Molineaux, 2011). Este conceito se enquadra também nos trabalhos a serem desenvolvidos nesta tese, na medida em que um agente condutor pode utilizar a base de casos de outros agentes para aprender a agir em um ambiente desconhecido para ele, mas de conhecimento de outro agente.

Em raciocínio baseado em casos voltado ao aprendizado de problemas há distinção entre os casos de origem e os casos do novo problema. Nesta abordagem, o raciocinador utiliza a saída de uma tarefa de origem (e.g., classificação) como entrada para uma nova tarefa (e.g., aprendizagem por reforço). Logo, o raciocínio baseado em casos deve ser integrado com outro componente para fazer a transferência do conhecimento entre problema fonte e problema-alvo (Klenk, Aha, & Molineaux, 2011).

Por fim, na terceira abordagem, raciocínio baseado em casos para transferir conhecimentos, métodos de raciocínio baseado em casos são usados para transferir conhecimentos de uma fonte para o domínio do novo aprendiz (Klenk, Aha, & Molineaux, 2011). Estas abordagens utilizam o ciclo do raciocínio baseado em casos para modificar as instâncias que serão usadas pelo algoritmo de aprendizagem, que pode ser, por exemplo, algoritmo de aprendizagem por reforço *Q-Learning* (Watkins, 1989).

Todo este conhecimento, seja ele armazenado na base de casos de um único agente ou de vários, deve sofrer processos de manutenção para ser útil.

3.3.8 *Manutenção da base de casos*

Assim como a maioria dos sistemas construídos para funcionarem por longos períodos de tempo e desenvolvidos para lidar com grande quantidade de informações, sistemas que utilizam a abordagem raciocínio baseado em casos também pode sofrer com uma grande exigência de armazenamento e tempo de consulta na fase de recuperação de casos. Para evitar este tipo de problema e garantir a qualidade, a manutenção do sistema faz-se necessária, incluindo a manutenção da base de casos.

A manutenção da base de casos tem por objetivo garantir um bom funcionamento do sistema com relação à redução do tempo de processamento para buscar informações e facilitar futuros raciocínios para um determinado conjunto de objetivos (Wilson & Leake, 2001). Uma visão geral dos métodos de manutenção é apresentada por (Smiti & Elouedi, 2011). A maioria dos trabalhos existentes em manutenção da base de casos são baseados na atualização da base de casos, adicionando ou removendo casos para otimizar e reduzir o número de casos contidos na base, envolvendo operações como apagar casos antigos, redundantes ou insistentes; unir grupos de casos para eliminar redundâncias e melhorar a precisão; descrever novamente casos para reparar inconsistências.

Para avaliar a necessidade de manutenção são utilizadas as medidas de competência e desempenho. A competência é a variedade de problemas que podem ser solucionados de modo satisfatório. Ela possui duas propriedades básicas: cobertura e acessibilidade. A cobertura de um caso é o conjunto de problemas que são solucionados pelo caso. A cobertura total de uma base de casos em relação à um conjunto de consultas é dada pelo número total de consultas atendidas pela base, dividido pelo número total de consultas que compõe o conjunto de consulta. A acessibilidade corresponde ao conjunto de casos que podem ser usados para solucionar o problema. Uma base de casos com boa competência é aquela que possui taxa de cobertura alta e acessibilidade baixa (Smiti & Elouedi, 2011).

O desempenho é o tempo necessário para calcular uma solução para determinado problema e está diretamente relacionado aos custos e aos resultados de adaptação. O desempenho depende da precisão e dos casos armazenados na base de casos. A precisão considera a correção da solução atual do caso recuperado mais similar. A precisão de uma base de casos em relação ao conjunto de consultas é o número de casos classificados corretamente dividido pelo número total de consultas. O desempenho também depende do espaço de armazenamento, que leva em consideração a velocidade de recuperação de casos, definida como o número total de casos na base (Smiti & Elouedi, 2011).

Algumas estratégias de manutenção da base de casos envolvem o particionamento da base de casos, seleção baseada em métodos de redução e otimização.

A primeira estratégia, particionamento, consiste em criar coleções de bases de casos distribuídas, onde cada elemento contido na estrutura da base de casos é um *cluster* criado. Para cada *cluster* é gerado um caso representativo, o qual possui um subconjunto de atributos. Os atributos com as informações mais valiosas são selecionados, e pode ter um maior potencial para cobrir uma ampla estrutura da base de casos. Tais políticas permitem adicionar e remover casos em cada pequena base de casos, sem usar toda a base de casos ao mesmo tempo (Smiti & Elouedi, 2011).

A técnica de seleção baseado em métodos de redução inicia com um conjunto vazio, seleciona um subconjunto de instancias do conjunto original e as adiciona à vários métodos, como *Condensed Nearest Neighbor Rule* (CNN) (Chou, Kuo, & Chang, 2006), *Reduced Nearest Neighbor Rule* (RNN) (Manry & Wilson, 2005), *Edited Nearest, Neighbor Rules* (ENN) (Wilson D. L., 1972) e *Instance Base Learning* (IBL) (Aha, Kibler, & Albert, 1991) com o objetivo de reduzir uma base de casos, selecionando os casos mais representativos da base de casos treinada (Smiti & Elouedi, 2011).

Na otimização de uma base de casos várias estratégias avaliam os casos de acordo com o critério de suprimir e conduzir a base de casos para um número específico de casos. As estratégias mais importantes visam preservar a competência (quantidade de problemas objetivo que um sistema pode solucionar com sucesso) da memória dos casos através da eliminação de casos. Como já dito, a competência da base de casos pode ser medida de acordo com a cobertura e acessibilidade. O método de preservação de competência por eliminação, por sua vez, categoriza os casos de acordo sua competência, apagando os casos com cobertura baixa (Smiti & Elouedi, 2012).

As políticas desenvolvidas até hoje são custosas quando aplicadas em grandes bases de casos e sofrem com a queda do desempenho do raciocinador baseado em casos, especialmente quando existem casos ruidosos. Para suplantar esta deficiência, pode-se utilizar técnicas de agrupamento para identificar quais casos apagar da base sem reduzir a precisão do raciocinador (Smiti & Elouedi, 2013). Para isto foram definidos dois tipos de casos que podem existir em uma base de casos: *outlier* e caso interno. *Outliers* são casos isolados que não são atingidos por nenhum outro caso da base e sua eliminação reduz a competência da base de casos uma vez que nenhum outro caso poderá solucionar o problema-alvo por um *outlier*. Caso interno é aquele pertencente a um grupo de casos similares onde cada caso provê uma cobertura similar aos demais no grupo. Apagar qualquer membro do grupo não afeta a competência visto que outros casos tem a mesma cobertura, mas apagar o grupo todo é o mesmo que apagar um caso *outlier*. Assim, deve-se manter pelo menos um caso do grupo de casos similares. O método desenvolvido consiste em manter casos internos e *outliers*, primeiro criando vários pequenos grupos a partir da base de casos, cada grupo contendo casos relacionados uns com

os outros por meio de *clusters*. Esta técnica permite a criação de pequenas bases de casos e mais fáceis de manter. Em seguida, *outliers* e casos próximos ao centro de cada *cluster* são selecionados e os demais casos removidos. Resultados mostraram a eficiência da técnica em termos de rapidez na recuperação de casos, classificações satisfatórias e manutenção da competência da base de casos.

Experimentos realizados com o objetivo de avaliar as técnicas de manutenção da base de casos mostraram que nenhum método foi superior em todos os conjuntos de dados avaliados (Smiti & Elouedi, 2011). Esta constatação leva a inferir que a escolha do melhor algoritmo de manutenção de bases de casos depende do domínio de aplicação. Portanto, cabe ao especialista do domínio selecionar a técnica mais apropriada com base em seu conhecimento e com base nas necessidades do sistema em termos de efetividade e restrições de armazenamento.

3.3.9 *Vantagens do uso do Raciocínio Baseado em Casos*

O raciocínio baseado em casos elimina a necessidade de expressar os conhecimentos em modelos ou em conjuntos de regras, como é necessário nos sistemas/modelos baseados em regras. A aquisição de conhecimentos para tarefas de raciocínio baseado em casos consiste em uma coleção preliminar de casos/experiências passadas e sua representação e armazenamento. Ele evita a repetição de erros cometidos no passado, na medida em que o sistema armazena tanto os sucessos como as falhas (e explicações das falhas) e utiliza estas informações para prever potenciais falhas (Pal & Shiu, 2004).

Devido a rigidez na formulação e na modelagem de um problema, um sistema baseado em modelos não pode solucionar um problema que depende dos seus conhecimentos quando há dados incompletos ou faltantes. Em contraste, raciocínio baseado em casos usa experiências passadas como conhecimentos e pode oferecer uma solução sensata por meio da adaptação de tais experiências, garantindo flexibilidade na modelagem dos conhecimentos (Barletta R. , 1991).

À medida que um sistema com raciocínio baseado em casos é utilizado, mais e mais soluções são criadas para problemas. Se a solução para um problema é testada subsequentemente no mundo real e um nível de sucesso é determinado para esta solução, essa solução pode ser adicionada à base de casos e usada para ajudar na solução de um novo problema. Tal adição à base de casos, dotará o sistema da capacidade de raciocinar sobre várias situações com um dado grau de refinamento e sucesso, assim, um sistema de raciocínio baseado em casos aprende à medida que o tempo passa (Kolodner, 1993).

A base de conhecimentos é formada por casos ao invés de regras: os conhecimentos de um

sistema baseado em regras são mais difíceis de construir e manter comparados aos conhecimentos de um sistema baseado em casos. Há evidência que especialistas solucionam problemas baseando-se em casos, i.e., usando suas experiências armazenadas como casos históricos, enquanto que os não-especialistas são mais inclinados a aplicar regras para solucionar problemas (Mitra & Basak, 2005).

Por último, a aplicação do raciocínio baseado em casos é possível em um domínio com pouco conhecimento. Quando a situação não fornece muitos casos disponíveis, um raciocinador baseado em casos pode iniciar com poucos casos e aumentar os conhecimentos à medida que novos casos são adicionados. A adição de novos casos pode expandir um sistema em direções que são determinadas por casos aplicados na resolução de novos problemas (Mitra & Basak, 2005).

3.3.10 *Aplicações do Raciocínio Baseado em Casos*

A técnica de raciocínio baseado em casos foi aplicada nos mais diversos domínios ao longo dos anos.

No controle de robôs, em (Jurisica & Glasgow, 1995), com objetivo de planejar movimentos. Cada posição do braço foi representada por 9 atributos e foram armazenados cerca de 2000 movimentos. Os casos foram recuperados por similaridade baseada no contexto. Esta técnica de recuperação permitia atribuir pesos para os atributos mais relevantes do problema. Os casos eram ordenados de acordo com sua relevância, e atribuídos pesos mais altos aos casos mais relevantes e então a média era calculada. Casos eram adaptados automaticamente por meio do cálculo da média ponderada dos valores dos atributos dos casos recuperados. Esta pesquisa mostrou que é possível utilizar o raciocínio baseado em casos com um pequeno conjunto inicial de casos. Contudo, segundo os autores, para aplicações de alta precisão a combinação do raciocínio baseado em casos com um método de otimização é plausível.

Raciocínio baseado em casos foi usado para selecionar comportamentos apropriados na condução de veículos em cenário complexos (Vacek, Gindele, Zollner, & Dillmann, 2007). Vacek modelou a aplicação com cinco componentes básicos: *(i) conjunto de sensores, (ii) percepção, (iii) base de dados* sobre situações passadas, *(iv) interpretador de situação* e *(v) atuador*. Cada caso era definido com: informações generalizadas da cena (e.g., carros e caminhões eram generalizados como veículos); o comportamento de um veículo e de outros participantes da cena; uma avaliação da cena; violações de diretrizes (e.g., velocidade acima da recomendada); violação de regras de trânsito e acidentes. A base de casos foi construída organizando os casos em três dimensões: (i) casos são organizados hierarquicamente de acordo com a especialização do caso; (ii) casos com o mesmo nível de

especialização compartilham uma ligação com suas diferenças e (iii) casos com evoluções temporais. A recuperação dos casos é feita percorrendo a hierarquia por casos mais similares, priorizando especializações. Para selecionar o melhor caso, todas as consequências da situação são consideradas, analisando os sucessores temporais do caso recuperado, de acordo com uma função heurística particular do problema. Não foram apresentados resultados comparativos. Uma dúvida deixada em aberto pelos autores: o que poderá acontecer se o veículo se deparar com uma situação que não estiver presente na base de casos?

O raciocínio baseado em casos no planejamento de ações foi discutido também em ambientes estocásticos (não-determinísticos), e com informações imperfeitas, como no caso do jogador de pôquer CASPER, cujo o objetivo é encontrar a melhor jogada (Watson & Rubin, 2008). A busca era feita em diferentes bases de casos para cada estágio do jogo, utilizando vizinho mais próximo para o cálculo de similaridade dos casos. CASPER utilizava duas métricas de similaridade, dependendo do tipo de característica: distancia Euclidiana ou função exponencial *de decaimento*. O uso de duas métricas foi motivado porque a distância Euclidiana produz mudanças suaves e contínuas na similaridade, contudo, para algumas características, diferenças menores nos valores leva a maiores mudanças nas similaridades, justificando o uso da função exponencial *de decaimento* (Rubin & Watson, 2007). CASPER obteve resultados melhores que jogadores que tomavam decisões aleatoriamente.

Raciocínio baseado em casos foi usado para garantir o fornecimento de energia por meio de um sistema eólico de tempo real com objetivo de calcular o tempo máximo de sincronismo do gerador após ocorrer uma falha (Tiako, Jayaweera, & Islam, 2011) e (Tiako, Jayaweera, & Islam, 2012). Eles optaram pelo uso do raciocínio baseado em casos frente a outras abordagens, como árvore de decisão e redes neurais, devido ao menor esforço computacional empregado na resolução dos problemas com raciocínio baseado em casos. Além disto, o raciocínio baseado em casos mostrou-se muito robusto e eficiente para ser aplicado em sistemas de tempo real, devido à busca de soluções passadas *offline*. Cada caso contém informações do domínio de operação do sistema: força do vento, força da rede, força no gerador. A adaptação é feita usando uma função objetivo simples baseada na estimativa de tempo, calculada por meio de uma equação própria. Além de efetivo (acerto de ~94%), o raciocínio baseado em casos se mostrou muito rápido na solução do problema em tempo real.

Raciocínio baseado em casos é utilizado juntamente com raciocínio baseado em regras para solucionar problemas de escalonamento de tripulação em ônibus (Liu, Ma, Guan, Song, & Fu, 2012). Este problema consiste em atribuir motoristas e condutores à rotina diária de ônibus de uma companhia pública durante um período de tempo. Primeiramente, um mecanismo de inferência integrando

ambas as abordagens é instanciado para obter soluções próximas de ótimas. O algoritmo de busca tabu foi usado repetidamente para encontrar um grupo de soluções próximas da ótima para o problema. Com base nessas soluções, um algoritmo genético é utilizado para produzir soluções ainda melhores. O sistema utilizou dados reais (i.e. rotas, horários, agenda de veículos, etc.) e um conjunto de restrições (i.e., horas trabalhadas por dia, leis de trabalho, etc.).

O planejamento de ações de um robô é feito utilizando raciocínio baseado em casos (Min, Huang, & Gan, 2012). As partes do robô foram modeladas como agentes, com funções específicas: percepção, interação, reação, ação e controle e deliberação. O raciocínio baseado em casos foi usado pelo agente deliberativo para manipular informações complexas do mundo, oriundas do agente percepção. As informações permitiam que fossem recuperadas ações passíveis de execução, adaptadas para emitir sinais que permitiam ao robô interagir de modo coerente com a percepção. O autor utiliza um limite de tempo para que um caso seja recuperado: se o robô recuperar uma ação no tempo estimado, ela é modificada e revisada. Um novo limite de tempo é estipulado para as ações de modificação e revisão da ação, de modo a confirmar uma solução. Se o tempo de recuperação acabar e não for recuperado nenhum caso, o robô fica sem ação. Se o tempo de confirmar a ação acabar, é disparada a ação como estava no momento que o tempo acabou. Este limite é imposto justamente pela rápida interação que ocorre entre a entidade planejadora (robô) e o mundo onde ele habita, necessitando de respostas rápidas para novas situações encontradas. A forma como os casos recuperados eram adaptados foi omitida pelo autor.

Outros trabalhos que utilizaram raciocínio baseado em casos em diferentes contextos: planejamento de ações para combate à incêndios (Chakraborty, Ghosh, Ranjan, Garnaik, & Debnath, 2010); planejamento com aprendizagem baseada em casos (Kavuluri & Kumar, 2011); na construção de agentes BDI deliberativos (Corchado & Laza, 2003); combinado com abordagem híbrida aplicada na classificação de títulos de créditos empresariais no mercado financeiro (shik Shin & Han, 1999).

3.4 CONSIDERAÇÕES FINAIS

A inteligência artificial visa proporcionar aos sistemas computacionais capacidades e habilidades naturais dos seres humanos. Uma forma de realizar isto é por meio da programação de agentes. Agentes são entidades autônomas, capazes de executar um ciclo que compreende em perceber o ambiente onde estão inseridos, raciocinar, executar ações e aprender com as ações executadas.

O aprendizado pode ocorrer, por exemplo, pela observação de fatos, situações ou casos do

ambiente. Neste sentido, o acúmulo de experiências é importante, pois permite ao agente tomar decisões baseado no sucesso de soluções passadas. A escolha de uma forma de dotar os agentes com capacidades de aprendizado é uma tarefa delicada. Existem várias alternativas que foram estudadas ao longo dos anos, dentre elas o uso de estratégias evolutivas, como o algoritmo genético. O algoritmo genético é baseado nos princípios evolutivos dos seres vivos e quando aplicado computacionalmente visa otimizar o domínio tratado. Ele pode ser utilizado como uma técnica de adaptação de conhecimentos passados para solucionar novos problemas. A adaptação é uma das etapas do raciocínio baseado em casos; ela é importante para reaproveitar as experiências adquiridas, mesmo em cenários diversos.

Neste trabalho defende-se a ideia de que as experiências podem ser úteis na elaboração de planos de ações em um sistema realista com características únicas. E quando um agente não possui um conjunto próprio experiências, ele pode fazer uso de experiências passadas de outros agentes, mesmo em características diferentes da atual. Todas estas características vêm ao encontro as necessidades do domínio de aplicação, objeto desta tese, a condução de trens. Em outras palavras, um agente deve ser capaz de planejar uma sequência de ações, executá-las e aprender com os seus efeitos.

O sucesso de uma condução de trens pode ser medido, por exemplo, pela eficiência energética. Alcançar esta eficiência é uma tarefa complexa, que demanda horas de treinamentos, principalmente para condutores iniciantes. Neste cenário, o desenvolvimento de agentes capazes de aprender e auxiliar os condutores mostra-se vantajoso e interessante. Em termos computacional, esta tese utiliza o algoritmo genético para otimizar plano de ações e reduzir a utilização de recursos diretos (e.g. combustível) e indiretos (e.g. uso de locomotivas e vagões). Cada plano de ações é elaborado por um agente que faz uso da técnica de raciocínio baseado em casos na solução de um problema-alvo.

4 MÉTODO

A inspiração da arquitetura adotada vem da área inteligência artificial distribuída, cuja proposta se fundamenta em uma abordagem de construção progressiva e modular de sistemas, privilegiando a alta coesão e o baixo acoplamento; isto favorece a distribuição do controle e dos conhecimentos de uma aplicação. Neste sentido, o conceito de agente é fundamental, tanto para permitir realizar efetivamente a distribuição dos conhecimentos de uma aplicação, quanto para fornecer ao projetista um alto nível de abstração, necessário para modelar sistemas complexos e abertos.

A apresentação do problema aqui tratado inclui (i) uma descrição dos estudos de campo realizados antes e durante esta pesquisa; (ii) uma visão geral de um sistema formado por um conjunto de agentes especializados na condução de trem de carga; (iii) um conjunto de definições do referido domínio da aplicação; (iv) um método de geração de políticas ou de planos de condução de trens de carga; e (v) o detalhamento de cada agente especializado e do fluxo operacional entre agentes.

4.1 ESTUDOS REALIZADOS PARA A COMPREENSÃO DO PROBLEMA

Estudos de campo foram realizados para obtenção de uma descrição do fenômeno estudado e compreender o contexto do uso das tecnologias (Plowman, Rogers, & Ramage, 1995). Ao longo desta pesquisa foram realizados diferentes estudos:

- O primeiro estudo foi feito para compreender o funcionamento de uma locomotiva e principais comandos utilizados pelos maquinistas ao longo das conduções;
- O segundo estudo foi realizado para compreender o funcionamento e a gestão básica de uma ferroviária, cujos principais temas foram: elementos da via permanente, licenciamento, lotação e condução de trens de carga;
- O terceiro estudo foi feito para extrair conhecimentos básicos relativos a condução—iteração com maquinistas experientes; e
- O quarto estudo foi realizado para vivenciar a atividade de um maquinista em ação. Esta experiência mostrou-se bastante útil para compreender como é realizada a condução *in loco* e analisar, do ponto de vista do condutor, quais características são importantes durante a condução dos mais diversos trens.

As experiências adquiridas ao longo dos estudos realizados foram importantes para compreensão do domínio do problema e descrição da solução proposta, descrita na próxima seção.

4.2 VISÃO GERAL DA SOLUÇÃO PROPOSTA

O sistema proposto reúne um conjunto bem-definido de funcionalidades, a saber: executar ordens de viagens decorrentes do gerenciamento de uma malha férrea, elaborar planos de condução de trens para locomotivas monoponto, gerenciar experiências passadas em condução e executar planos previamente gerados. Cada uma destas funcionalidades é operacionalizada por um agente. A malha é formada por um conjunto de estações conectadas por vias férreas sob as quais trafegam um ou mais trens. A malha é gerenciada por um agente Despachante, presente no centro de controle operacional de um sistema ferroviária, responsável por autorizar a partida e a parada de trens.

A condução de um trem de carga é realizada normalmente por apenas um maquinista e o trecho de atuação desse maquinista depende da sua experiência em vencer as dificuldades e minimizar os riscos do percurso. Em outras palavras, os maquinistas menos experientes são alocados para trechos mais simples, onde há uma maior margem para ajustes no estilo de condução, sem colocar em risco às regras de segurança. A simplicidade de um trecho mede-se em função das inclinações verticais (aclives e declives), principalmente, e dos ângulos de abertura das curvas. Assim, um trecho simples é predominantemente plano e retilíneo ou com valores altos para ângulos centrais das curvas. Consequentemente, as restrições de velocidades máximas permitidas em trechos simples possuem pequenas variações ao longo de sua trajetória. Um trecho complexo inclui muitas curvas e inclinações variadas; o que demanda, por exemplo, diferentes restrições de velocidades máximas, assim como o número de aplicações de freios. Logo, a condução de um trem de carga em tal trecho deve ser feita por um maquinista experiente. Tanto em trechos simples quanto em trechos complexos, a condução de um trem possui como principal política a correta aplicação de pontos de aceleração e de freios. Obviamente, durante a viagem, o maquinista pode executar outras ações, tais como: aplicar areia nos trilhos para aumentar a aderência trilho-roda, emitir avisos sonoros em passagens de nível. Contudo, este conjunto de outras ações não é tratado como foco deste trabalho.

A elaboração de planos de condução por entidades especializadas pode auxiliar os maquinistas em suas tomadas de decisões. Tal ajuda inclui a redução de esforços empregados para determinar as ações aplicáveis ao longo de uma viagem, além de promover melhor uso dos recursos, tais como: combustível, locomotivas e vagões.

A demanda de planejamento inicia quando uma ordem de movimentação de trem é gerada para um dado trecho. Tal ordem inclui as estações de origem *O* e destino *D*, assim como outros locais de paradas ou cruzamentos intermediários de trens. Um plano é elaborado para cada ordem. Cada ordem começa sua execução quando o trem passa pela estação *O* e termina quando o trem chega à

estação *D*. Caso o trem prossiga além da estação *D*, um novo plano é elaborado e aplicado para o trecho que começa na estação *D*. O plano aplicado para mover o trem de *O* para *D* retorna à base de experiências passadas do sistema de geração de planos da estação de origem *O*. Frisamos que a geração do plano é feita por um agente localizado na estação de origem *O*, e a aplicação do mesmo é feita por outro agente embarcado no computador de bordo da locomotiva mestra. Oportunamente, o agente que aplicou o plano retorna à sua estação de origem *O* e repassa o plano aplicado ao sistema de planejamento local como forma de enriquecer a sua base de experiências. O trafego de agentes de estação em estação é semelhante às redes DTN (Voyiatzis, 2012).

4.3 DEFINIÇÕES DA APLICAÇÃO E DO MÉTODO

As diferentes definições apresentadas nesta seção são relativas à aplicação, como por exemplo, a definição de uma malha férrea, descrita como um grafo, onde as estações são descritas como vértices e os possíveis caminhos como arestas ligando os vértices. São apresentadas também definições acerca do método, utilizadas na elaboração de cada etapa do planejamento, como plano e caso.

O método computacional em questão opera preferencialmente sobre as experiências bem sucedidas. Tais experiências são registradas e reusadas na forma de casos.

4.3.1 *Definições do domínio de aplicação*

Uma malha férrea pode ser definida como um grafo. Um grafo é um par $G = (V, E)$, onde V assume um conjunto de n vértices $\{v_1, v_2, \dots, v_n\}$ e E é um conjunto de m arestas $\{e_1, e_2, \dots, e_m\}$; cada aresta é formada por par de vértices $v = (u, v)$ onde v e $u \in V$.

Para este trabalho, V é um conjunto de estações férreas, i.e., $V = \{s \mid s \text{ é uma estação}\}$, e E é um conjunto de caminhos que conectam as estações, logo, $E = \{(u, v) | \langle u \text{ está ligada à } v \rangle\}$. A Figura 22 exemplifica tal situação, sendo *V={Da Luz, Central do Brasil, São João del-Rei, Bento Gonçalves}* e *E={(São João del-Rei, Da Luz), (Da Luz, São João del-Rei), (Central do Brasil, Da Luz), (Da Luz, Central do Brasil), (São João del-Rei, Bento Gonçalves), (Bento Gonçalves, São João del-Rei), (Central do Brasil, Bento Gonçalves), (Bento Gonçalves, Central do Brasil)}*.

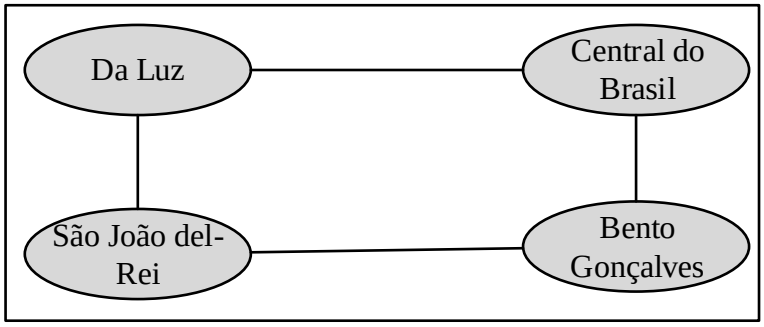


Figura 22 – Exemplo de parte de uma malha férrea representada graficamente.

Neste exemplo, a relação $\langle u$ está ligada à $v\rangle$ é simétrica, uma vez que se $\langle u$ está ligada à $v\rangle$ então $\langle v$ está ligada à $u\rangle$.

O grafo G que representa uma malha férrea é rotulado, à medida que cada vértice possui um rótulo associado e ele é estabelecido por uma função de mapeamento $\alpha: V \rightarrow A$, onde A é um conjunto de rótulos. Analogamente, as arestas de G também são rotuladas por uma função $\beta: E \rightarrow B$, onde B é um conjunto de rótulos. Assim, A é formado pelos nomes das estações férreas e B é formado por um conjunto de rótulos. Cada rótulo de B identificará um conjunto de dados a respeito do trecho de via férrea que ele representa, denominados *pontos de medidas (PM)*, conforme Figura 23. Tais pontos de medidas serão detalhados mais à frente.

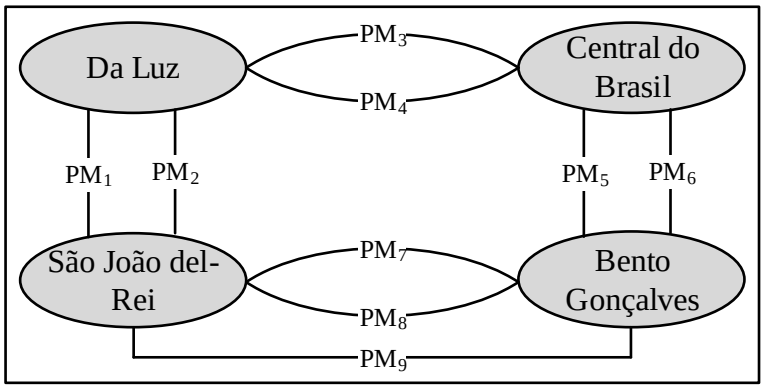


Figura 23 – Exemplo de malha férrea representada por um grafo: (a) cada vértice tem rótulo e (b) cada aresta tem rótulo e cada rótulo identifica um conjunto de dados relativos ao perfil de um trecho de via férrea.

O grafo G que representa uma malha férrea além de ser rotulado, é orientado (i.e., dígrafo) (Rosen, 2012). Os vértices são conectados por arestas orientadas. Isto permite representar o sentido do caminho percorrido por um trem na malha férrea. Um dígrafo é um par $G_u = (V, E)$, onde V é o conjunto de vértices e E é o conjunto de arestas. Cada elemento de E é par ordenado: a aresta do

vértice u ao vértice v é escrita como (u, v) e o par (v, u) é a aresta na direção oposta. Um caminho em dígrafo pode ser escrito como: $v_{i_0}, e_{j_1}, v_{i_1}, e_{j_2}, \ldots, e_{j_k}, v_{i_k}$, onde v_{i_l} é o vértice inicial e $v_{i_{l-1}}$é o vértice final da aresta e_{j_l}. Logo, neste trabalho assume-se que $G = G_u$.

O grafo G que representa uma malha férrea além de ser um dígrafo rotulado, ele é um multigrafo (Rosen, 2012). Além do sentido entre estações de uma malha férrea pode haver dois caminhos diferentes que conectam duas estações (cf. Figura 24). Um multigrafo orientado permite que múltiplas arestas m conectem pares de vértices de G, onde cada aresta possui uma orientação. Formalmente, um multigrafo orientado é uma tripla $G = (V, E, f)$, onde V é o conjunto de vértices, E é o conjunto de arestas e $f: E \to V \times V$ é uma função (Rosen, 2012). Nesta representação não foi considerada a possibilidade de laços, i.e., um caminho tendo como ponto de partida e chegada a mesma estação férrea.

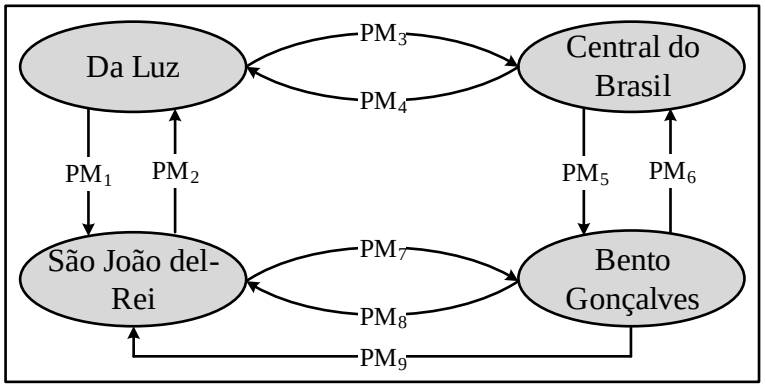


Figura 24 – Exemplo de malha férrea representada por um multigrafo orientado: (a) cada vértice tem rótulo e pode ter mais de uma conexão a um mesmo vizinho e (b) cada aresta tem rótulo e cada rótulo identifica um conjunto de dados relativos ao perfil do trecho de uma via férrea.

A seguir serão discutidos os componentes dos vértices e arestas no contexto da aplicação.

4.3.1.1 Um vértice é uma estação férrea

O conjunto de vértices V de G corresponde às estações férreas S, logo $V = S$, onde S é um conjunto de n estações s_{id}, e cada estação é identificada univocamente por índice id. Assim sendo, $S = \{s_{id} \mid id = 1, 2, \ldots, n\}$, onde $n = |V| = |S|$.

4.3.1.2 Uma aresta é um caminho ou trecho de via férrea

O conjunto de arestas E de G corresponde aos caminhos trafegáveis por trens, de estação em estação. A orientação de cada aresta indica o sentido trafegável, que parte do vértice v_i para o vértice v_j. Como G é um multigrafo, o tráfego pode ocorrer por qualquer caminho k dado por $f: E \rightarrow V \times V$. Para expressar tal relação, fez-se uso da notação $E_{i,j,k}$, onde E é igual à um trecho/caminho de via férrea St, i a estação de origem, j a estação de destino e k o caminho a ser percorrido, assim temos que $E_{i,j,k} = St_{i,j,k}$. A Figura 25 ilustra diferentes caminhos entre diferentes estações.

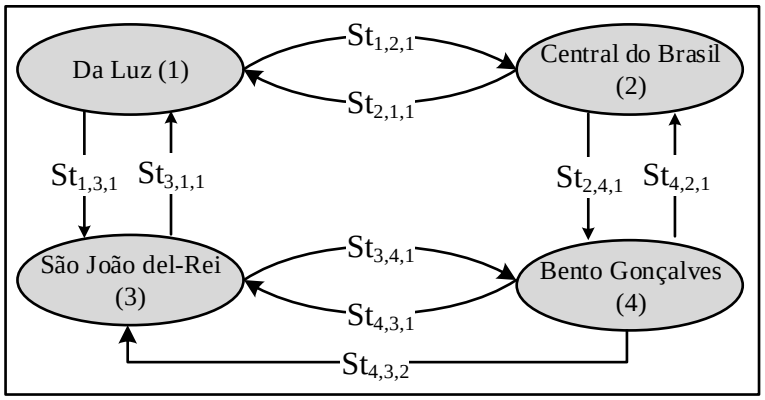


Figura 25 – Exemplo de definição das arestas.

Cada trecho físico de via férrea $St_{i,j,k}$ é descrito por um conjunto ordenado de pontos $p \in PM_k$, obtidos do projeto topográfico da ferrovia considerada (Projeto Planimétrico) e normalmente calculados de 20m em 20m (Chandra & Agarwal, 2007). Cada ponto — ou **p**onto de **m**edida (pm) — é uma *tupla (id, km, velMax, rampa, raioCurva, AC, g20, altitude, nroSb)*. A Tabela 11 fornece uma descrição sucinta para cada componente desta *tupla* e ilustra um trecho $St_{1,2,1}$ que conecta as estações 1 e 2 pelo caminho 1, cuja extensão é de 280m.

Tabela 11 – Exemplos de 14 pontos do projeto topográfico de um trecho de via férrea.

ID	KM	VELMAX	RAMPA	RAIOCURVA	AC	G20	ALTITUDE	NROSB
1	349	60	0.95	1150.5	25.48	0.99	526.10	3
2	349	60	0.95	1150.5	25.48	0.99	526.10	3
3	349	60	0.95	1150.5	25.48	0.99	526.10	3
4	349	60	0.95	1150.5	25.48	0.99	526.10	3
5	349	60	0.95	1150.5	25.48	0.99	526.10	3
6	350	60	0.95	1150.5	25.48	0.99	526.10	4
7	350	60	0.95	1150.5	25.48	0.99	526.10	4
8	350	60	0.95	1150.5	25.48	0.99	526.10	4
9	350	60	0.95	1150.5	25.48	0.99	526.10	4
10	350	60	0.95	1150.5	25.48	0.99	526.10	4
11	351	60	0.95	1150.5	25.48	0.99	526.10	4
12	351	60	0.95	1150.5	25.48	0.99	526.10	4
13	351	60	1.20	818.52	66.20	1.40	534.26	4
14	351	60	1.20	818.52	66.20	1.40	534.26	4

Legenda:

id:	Identificador do ponto de medida.
km:	Quilômetro de referência.
velMax:	Velocidade máxima permitida (em km/h).
rampa:	Inclinação da rampa (em %).
raioCurva:	Raio da curva (em metros).
AC:	Ângulo central da curva (em metros).
g20:	Grau da curva para uma corda de 20m.
altitude:	Altitude (em metros).
nroSb:	Identificador da seção de bloqueio a qual o ponto de medida pertence.

Um caminho k que liga duas estações s_i e s_j, é designado por trecho de via férrea $ST_{s_i,s_j,k}$, onde s_i é a estação de origem, s_j é a estação de destino; lembra-se que entre s_i e s_j podem haver mais de um caminho k. Em termos de notação de grafo, nota-se que o conjunto de vértices V de G corresponde às estações férreas S, logo $V = S$, e S é um conjunto de n estações s_{id}, onde cada estação é identificada univocamente por índice id. Assim sendo, $S = \{s_{id} \mid id = 1, 2, \dots, n\}$, onde $n = |V| = |S|$. Doravante, usaremos ST_k para designar um trecho de via férrea $ST_{i,j,k}$.

Cada estação s_{id} dispõe de um conjunto de recursos computacionais. Esses recursos são organizados em unidades denominadas de *contêineres*. Cada contêiner pode hospedar um ou mais agentes de software (Bellifemine, Caire, & Greenwood, 2007). Cada agente realiza uma tarefa bem-definida. A configuração básica de cada estação s_{id} é dada por um contêiner CM e um conjunto de contêineres CO (cf. Figura 26). Cada contêiner CM hospeda um único agente Memorizador, especializado em gerenciar as experiências passadas sobre os planos de condução de trens já aplicados. Cada contêiner $co_k \in CO$ hospeda um agente Planejador especializado em gerar planos de condução de trens para um trecho k bem-definido, e outro agente Executor especializado em aplicar o plano gerado, sobre o mesmo trecho k; *ressalta-se que k identifica ao mesmo tempo um contêiner operacional*

co_k *e caminho* ST_k. A cardinalidade de CO é dada pelo grau de saída de cada estação (ou vértice de G), logo $|CO| = d(s_{id})$. Assim, para cada aresta que parte da estação s_{id} é atribuída um contêiner co_k. Esse contêiner hospeda um Planejador e um Executor que atuam em binômio em um único caminho k. Essa especificidade deve propiciar, com o passar do tempo, um aumento de eficiência do binômio por meio do acúmulo e reuso eficiente de experiências passadas.

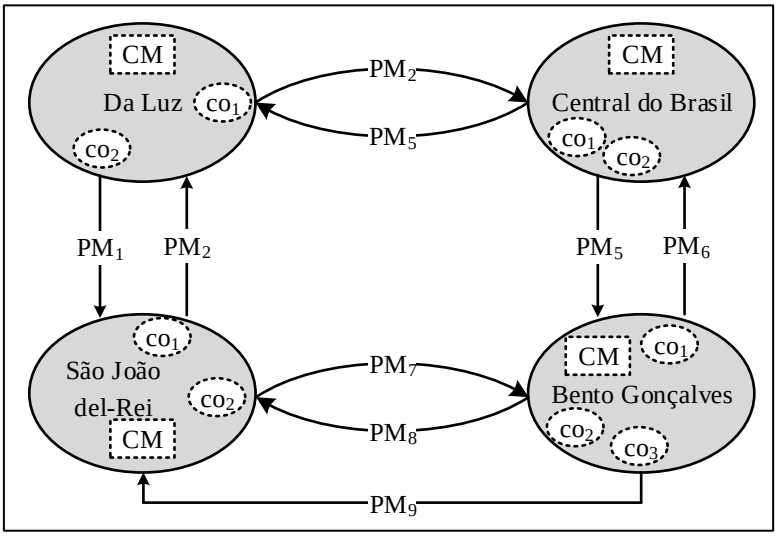


Figura 26 – Exemplo de malha férrea representada por um multigrafo orientado: cada vértice é uma unidade computacional que hospeda contêineres de agentes especializados.

Dado que a operacionalização básica de cada $co_k \in CO$ requer um Planejador e um Executor, assume-se que (i) cada Executor é uma entidade de software móvel; e (ii) cada trem, que parte de uma estação s_{id} , hospeda, no *contêiner* do seu computador de bordo, um Executor. Desta forma, o número de executores pertencentes a co_k embarcados em diferentes trens é igual ao número de trens que partem da estação s_{id} para outra estação s_{id+1} ou s_{id-1}. É importante notar que a realimentação da base de experiências, situada em cada estação s_{id}, é feita pela incorporação dos planos de condução aplicados por um ou mais executores que iniciaram as suas atividades na estação s_{id}. Assim, cada Executor ao concluir sua missão deve retornar a estação de origem e repassar, ao Memorizador de casos residente no contêiner *CM,* o plano que ele efetivamente aplicou.

A Figura 27 ilustra o ciclo de vida dos agentes: Despachante, Planejador, Executor e Memorizador. De forma resumida, o Despachante envia uma ordem de despacho *O* ao Planejador. Este elabora um plano *P* que atenda *O*, repassa o plano *P* para o Executor. O Executor aplica o plano *P* e adiciona as modificações Δ ao plano *P*, dando origem a um plano modificado *P'*, que é repassado ao Memorizador, que o armazena localmente.

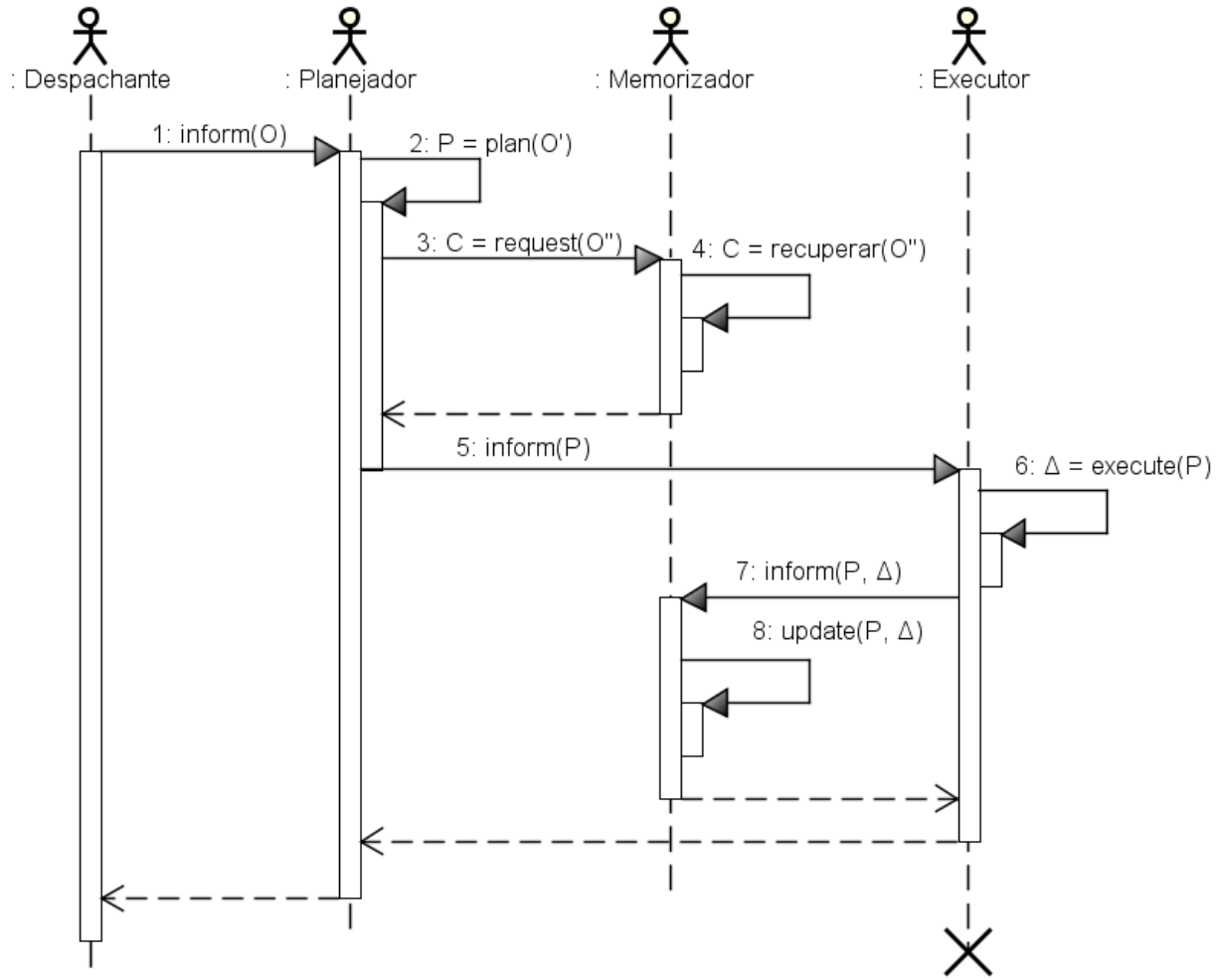


Figura 27 – Ciclo de vida dos agentes Despachante (situado na central de controle), Planejador (hospedado em contêiner co_k da estação s_i), Executor (instanciado no contêiner co_k e migrado para o contêiner do computador de bordo da locomotiva em missão) e Memorizador (hospedado no contêiner CM da estação s_i).

A visão ilustrada neste diagrama de sequência é trivial. Entretanto, ela nos remete a uma situação-colaboração, onde os agentes revolvem suas situações-problemas localmente e compartilham esforços e/ou experiências. Como já dito, cada nova experiência é gerida como um caso ou um conjunto de casos.

4.3.2 *Definições do método*

A principal abordagem de resolução de problema usada segue os princípios um sistema de raciocínio baseado em casos clássico. Adicionalmente, como dito anteriormente, um conjunto de

agentes especializados é instanciado para reduzir a complexidade do problema — decomposição funcional, usufruir eventualmente do paralelismo, aumentar a disponibilidade e robustez do sistema.

> A principal unidade de informação operacional manipulada por cada agente especializado, seja na resolução de um problema-alvo, na aplicação da solução, na transferência ou no compartilhamento de conhecimento, é o *caso*.

Em geral, um caso representa um objeto do mundo real ou episódio em um esquema de representação particular. Assume-se que um caso *C* é representado como um conjunto finito de n pares de atributo/valor $\mathrm{C} = \{\langle a_0 : CV_0\rangle, \ldots, \langle a_k : CV_k\rangle\} = \{\langle a_i : CV_i\rangle\}_0^n = \{A_i\}_0^n$, onde $\langle a_i : V_i\rangle$ é um par atributo/valor (A_i). Um conjunto finito de casos compõe uma base de casos CB. Um contexto $\Omega = \{\langle a_0 : CV_0\rangle, \ldots, \langle a_k : CV_k\rangle\} = \{\langle a_i : CV_i\rangle\}_0^k = \{A_i\}_0^k$ é definido como um conjunto finito de atributos e restrições, onde a_i é um nome de atributo e CV_i é uma restrição que determina o conjunto de valores admissíveis para a_i.

Os esforços canônicos do *raciocínio baseado em casos* concernem em decidir:

(i) qual estrutura usar para descrever o conteúdo do caso;

(ii) o que reter de um caso; e

(iii) como organizar e indexar os casos para facilitar a recuperação e reuso.

Aqui, um caso é um *snapshot* de cada situação executada por um condutor de trem durante uma missão ou viagem. Os principais componentes deste caso dizem respeito à descrição física do trem (e.g., número de locomotivas, número de vagões), a descrição planimétrica e altimétrica parcial da via férrea em uso (e.g., posição atual, percentual de rampa, espaço do *snapshot* em metros) e a descrição do comportamento do trem (e.g., velocidade inicial, velocidade máxima, ação aplicada para rebocar o trem). Um conjunto de *snapshot* forma a base de casos inicial.

Em face de uma situação de reuso destes casos, para o caso recuperado similar ao problema-alvo, a *principal atividade* é inferir o conjunto de pontos de aceleração $SP = \{ap_0 : m_0, \ldots, ap_n : m_n\} = \{\langle ap_i : m_i\rangle\}_0^n = \{SP_i\}_0^n$ para atender o problema-alvo, onde $\langle ap_i : m_i\rangle$ é um par atributo/valor, ap_i é um ponto de aceleração e m_i é a posição de aplicação de ap_i. Cada posição m_i alcançada representa um deslocamento. A Figura 28 mostra uma instância de SP, onde a aplicação de cada ap_i, da esquerda para a direita, deve rebocar o trem por 403m.

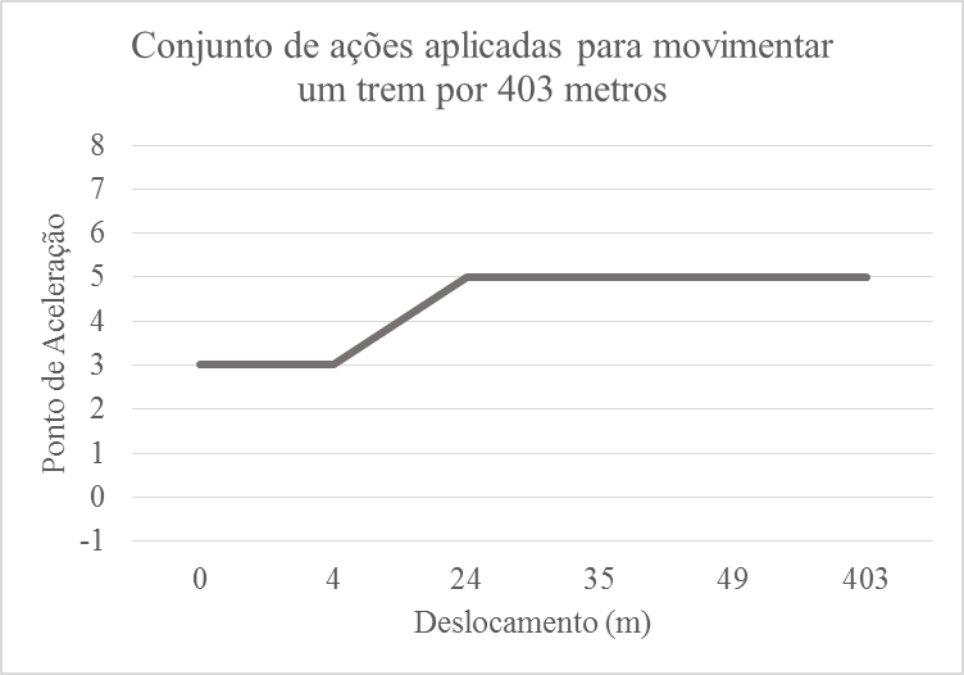


Figura 28 – Representação parcial aplicação de um plano de ação para rebocar um trem.

A Figura 29 mostra um exemplo de caso recuperado e formado pelos seguintes atributos: ação executada (*EAC*), tipo do perfil a percorrer (Perfil), quilômetro inicial (*KM*), número de locomotivas (*NL*), número de vagões (*NW*), velocidade inicial (*IS*) em km/h, velocidade final (*FS*) em km/h, velocidade máxima (*MS*) em km/h, percentual de rampa (*%R*) e deslocamento total (*CL*) em metros. Embora tais atributos estejam sendo apresentados como ilustração, eles representam as características determinantes e dominantes na movimentação de um trem.

$$cr_1 = \begin{Bmatrix} \langle EAC, Acelerar\rangle, \langle Perfil, Em\ Nível\rangle, \langle KM, 339.404\rangle, \langle NL, 3\rangle, \langle NW, 58\rangle, \langle IS, 10\rangle, \langle FS, 11\rangle, \langle MS, 45\rangle, \langle \%R, -0.41\rangle, \langle CL, 0.403\rangle, \\ \langle SP, \{\langle 3{,}0\rangle, \langle 3{,}4\rangle, \langle 5{,}24\rangle, \langle 5{,}35\rangle, \langle 5{,}49\rangle\}\rangle \end{Bmatrix}$$

Figura 29 – Exemplo de um caso.

Obviamente, a situação supra descrita é trivial e pouco representa a complexidade de geração de um plano de movimentação de um trem efetivo. Em outras palavras, dada à extensão e variação de perfil de um trecho de via férrea, a consecução de um plano completo passa pela abordagem: *dividir para conquistar*.

4.3.2.1 Método: Segmenta e Planeja

A Figura 30 ilustra um trecho de via férrea $ST = \{st_1, st_2, \ldots, st_n\}$, onde cada st_i é um segmento cujos limites st_i de início e fim são determinados pelo perfil vertical, por exemplo, aclive, declive e em nível. Cada st_i é tratado de forma individual. Assim, para cada situação pr_i — definida sobre o conjunto de pontos de medidas do segmento de via férrea st_i —, um caso c_i é recuperado da base de casos CB e um problema-alvo $pb_i = \langle pr_i, c_i \rangle$ é instanciado. A solução de cada pb_i é dada por meio da adaptação do conjunto de pontos de aceleração de $ap_j \in c_i$. As últimas etapas consistem em aplicar os pontos de aceleração $j \in AP'$ e armazenar pb_i em CB_{local}; essa última funciona como uma memória de trabalho. Finalmente, $P = \{\langle st_1, p_1 \rangle, \langle st_2, p_2 \rangle, \ldots, \langle st_n, p_n \rangle\}$ é o plano completo para rebocar um trem *T*, sobre o trecho ST, usando o plano *P*. Cada p_i descreve um conjunto de ações com os comportamentos (e.g., manter, acelerar, reduzir ou frear) mais previsíveis para um perfil vertical (e.g., aclive, declive e em nível). A estrutura de pr_i é dada pelo seguinte conjunto de valores observados (*obs*) ou percebidos:

$$pr_{i=st1} = \{ação_{obs}, perfil_{obs}, km_{obs}, NL_{obs}, NW_{obs}, IS_{obs}, FS_{obs}, MS_{obs}, \%R_{obs}, CL_{obs}\}$$

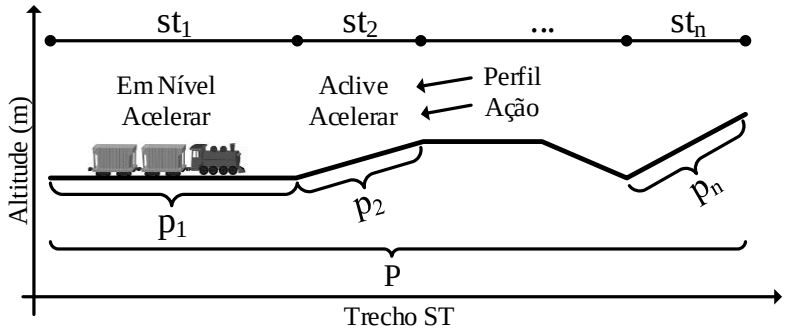


Figura 30 – Trecho de via férrea subdividido em segmentos em função do perfil vertical.

Para aumentar, manter ou reduzir a velocidade do trem são utilizadas as regras descritas no Capítulo 2. O conjunto de regras foi operacionalizado em uma função *selecionarComportamento* (cf. Algoritmo 3) que retorna, a partir das velocidades: atual, máxima e mínima projetada, da força de aceleração da ação anterior e do percentual de rampa, uma das seguintes classes ou ações: ACELERAR, REDUZIR, MANTER, FREAR.

Algoritmo 3 – Função de seleção do comportamento a ser executado pelo condutor.

function *selecionarComportamento* (v, v_s, $v_{minProj}$, f_{ac}, $\%R$) : AÇÃO		
v: velocidade; v_s: velocidade superior; $v_{minProj}$: velocidade mínima projetada. f_{ac}: força de aceleração; $\%R$: percentual de rampa.		
01	VAR ← 2 {variação de velocidade desejada, em km/h}	
02	$v_{critica}$ ← 10 {menor velocidade praticável por segurança}	
03	$\mathbf{v_c} \leftarrow v_s - \text{VAR}$	
04	$\mathbf{v_i} \leftarrow v_s - (2 \times \text{VAR})$	
05	**switch** v, $v_{critical}$, v_i, v_c,v_s, f_{ac}, %R)	
06	**case** ($v < v_{critical}$) : **return** ACELERAR	{R01}
07	**case** ($f_{ac} < 0$ and $v \leq v_{minProj} + 10$) : **return** ACELERAR	{R02}
08	**case** ($v < v_i$) : **return** ACELERAR	{R03}
09	**case** ($v < v_c$) : **return** MANTER	{R04}
10	**case** ($v \geq v_c$ and $v \leq v_s$ and $\%R < 0$) : **return** FREAR	{R05}
11	**case** ($v \geq v_c$ and $v \leq v_s$) : **return** REDUZIR	{R06}
12	**case** ($v \geq v_c$ and $v \leq v_s$) : **return** MANTER	{R07}
13	**case** ($v < v_c$) : **return** MANTER	{R08}
14	**case** ($v \geq v_s$) : **return** FREAR	{R09}
15	**default : return** FREAR	{R10}
16	**end switch**	
17	**end function**	

As definições já apresentadas dão suporte mínimo para representar casos simples e orientar o processo de adaptação cujas regras supracitadas sugerem ações básicas, tais como: acelerar, manter, reduzir a velocidade de um trem, ou frear. Essa estrutura mínima pode ser enquadrada em um ciclo canônico de um raciocinador baseado em casos.

Relembrando, o processo canônico de raciocínio baseado em casos tem cinco etapas. A primeira etapa consiste, a partir de uma percepção (e.g., valores lidos de sensores) e de uma representação do domínio de aplicação, gerar uma formulação robusta do problema-alvo. Essa formulação assume muitas vezes o formato de um conjunto de índices (Formular). Dada à formulação do problema-alvo, a segunda etapa consiste em recuperar, a partir de uma memória, casos relevantes para resolvê-lo (Recuperar). Um caso é constituído de um problema, sua solução, e, em geral, as anotações sobre a maneira de como a solução foi derivada. Por exemplo, supomos que o maquinista Bari quer manobrar trens de carga com 80 vagões. Sendo ele um maquinista iniciante, a experiência mais relevante que ele pode recordar é aquela em que ele fez com sucesso, a saber: manobra de trens em pátio de classificação. O procedimento que ele seguiu para fazer as manobras em pátio de classificação em conjunto com as justificativas para as decisões tomadas ao longo do tempo, constitui o caso recuperado por Bari.

Depois de encontrar o caso mais similar, a próxima etapa consiste em mapear a solução do caso recuperado para o problema-alvo (Reusar). Isto pode envolver a adaptação da solução para adequá-la à nova situação. No exemplo manobras de trens, Bari deve adaptar a solução que ele recuperou

de modo a rebocar um trem de carga com vários vagões. Para tal, ele deve determinar qual é a diferença entre o problema-alvo e o caso recuperado, e modificar a solução recuperada para levar em conta essas diferenças.

Tendo sido mapeada a solução recuperada para a situação-alvo, a próxima etapa consiste em testar a nova solução no mundo real (ou numa simulação) e, se necessário, revê-la (Revisar). Supondo-se que Bari adaptou sua solução manobra em pátio de classificação adicionando mais uma locomotiva ao trem. Após a adição, ele descobre que a força de tração não é suficiente — um efeito indesejado. Isto sugere a seguinte revisão: acrescentar uma locomotiva para cada 20 vagões do trem para ter a disposição à força de tração necessária para superar a resistência de inércia inicial. Quantas adaptações precisam ser feitas? Isto depende da natureza das diferenças entre o problema-alvo e o caso recuperado.

Depois que a solução foi adaptada com sucesso ao problema-alvo, a próxima etapa, consiste em armazenar a experiência resultante como um novo caso na memória (Reter). Bari, por conseguinte, registra o procedimento de seu recém-achado para rebocar trens com 80 vagões, enriquecendo assim o seu conjunto de experiências armazenadas, e se preparando melhor para as futuras demandas de manobras de trens.

4.3.2.2 Formulação

Dada uma percepção parcial do mundo e uma representação do domínio, a tarefa é gerar uma formulação robusta da situação observada ou *problema-alvo*. Essa formulação assume muitas vezes o formato de um conjunto de índices ou restrições.

A Tabela 12 enumera os valores que compõe a formulação do *problema-alvo* e define as fontes, que são: sensores, ordem de despacho e cálculos. Nesta pesquisa, partimos do pressuposto que não ocorre alterações no número de locomotivas ou vagões ao longo da viagem. A alteração das características principais do trem ou da via férrea a percorrer, depois que o plano de condução já foi estabelecido, pode afetar a eficiência da missão, em particular, se o trem já se encontra em missão e o plano não pode mais ser refeito/otimizado; alterar as principais características ligadas à movimentação do trem pode ter impacto direto na resistência total e nos esforços.

Tabela 12 – Conjunto de índices usados para formular um problema-alvo.

OBTENÇÃO	ÍNDICE	SIGLA	DESCRIÇÃO
Cálculo	Ação a executar	EAC	Ação a executar. x=*selecionarComportamento*(v, v_s, $v_{minProj}$, f_{ac}, $\%R$).
	Velocidade final	FS	Velocidade desejada para um deslocamento de 20m. $y = f_{v_F}(v, f_{ac}, W)$.
	Deslocamento total	CL	Deslocamento total, em km, ainda não percorrido do segmento de trecho st_i. $z = \lvert kmMudança - km \rvert$
Ordem de despacho	Número de locomotivas	NL	Número de locomotivas que rebocam o trem.
	Número de vagões	NW	Número de vagões rebocados pelo trem.
Leitura de sensor	Perfil	Perfil	Tipo de perfil vertical a percorrer (subida/descida/em nível)
	Quilômetro	km	Posição atual do trem em km
	Velocidade	IS	Velocidade atual do trem.
	Velocidade máxima	MS	Velocidade máxima do trecho a percorrer.
	Percentual de rampa	%R	Inclinação (%) no ponto de medida onde o está a locomotiva líder.

Seja $p = \{\langle EAC: Acelerar\rangle, \langle Perfil: Em\ Nível\rangle, \langle KM: 339{,}400\rangle, \langle NL: 3\rangle, \langle NW: 58\rangle, \langle IS: 9\rangle, \langle FS: 11\rangle, \langle \%R: -0{,}41\rangle, \langle CL: 0{,}400\rangle\}$ um conjunto de pares atributo/valor. Os valores dos atributos IS, Perfil, KM, %R são obtidos de sensores, os valores dos atributos NL e NW são extraídos da ordem de despacho, e finalmente, os valores dos atributos EAC, FS e CL são derivados de cálculos específicos. No exemplo, assume-se um trem de peso total 6277,8t com o ponto de aceleração atual igual a 5, localizado no quilômetro 339400 e que a distância da próxima alteração no perfil vertical está localizada no quilômetro 339800. Assumindo-se também que as variáveis x, y, e z são os respectivos valores de atributos EAC, FS e CL, logo os valores dessas variáveis são:

x = $selecionarComportamento\ (v = 9, v_s = 45, v_{minProj} = 2,\ f_{ac} = 114303, \%R = -0.41)$

$y = f_{v_F}(v = 30,\ f_{ac} = 114303, W = 6277.8t)$

$z = |339800 - 339400|$

Tal conjunto p define a formulação do *problema-alvo* para um trecho de via férrea st_i. O script para obter p é definido pela função *formular* descrita no Algoritmo 4.

Algoritmo 4 – Função formular.

function *formular* (p, T) : PROBLEMA	
p: objeto que descreve as medidas obtidas dos sensores – percepção. T: objeto que descreve o trem.	
	{Leitura de dados do objeto p}
01	pIS ← p[IS] {velocidade inicial}
02	pMS ← p[MS] {velocidade máxima}
03	pKM ← p[KM] {quilômetro atual}
04	p%R ← p[%R] {percentual de rampa}
05	pf_{ac} ← p[f_{ac}] {força de aceleração}
06	$pv_{minProj}$ ← p[$v_{minProj}$] {velocidade mínima projetada}
	{Leitura de dados do objeto T}
07	dNL ← T[NL] {número de locomotivas}
08	dNW ← T[NW] {número de vagões}
09	pW ← T[W] {peso do trem}
	{Formulação do problema }
10	pPerfil ← f_{perfil}(%R) {obtenção do perfil do trecho aonde se encontra a locomotiva líder}
11	x = selecionarComportamento$(pIS, pMS, pv_{minProj}, f_{ac}, p\%R)$ {obtenção do comportamento}
12	y = $f_{v_F}(pIS, f_{ac}, W)$ {obtenção da velocidade desejada}
13	z = pKM − pPerfil[kmMudanca] {obtenção do deslocamento}
14	pf = {⟨EAC: x⟩, ⟨Perfil: pPerfil⟩, ⟨KM: pKM⟩, ⟨NL: dNL⟩, ⟨NW: dNW⟩, ⟨IS: pIS⟩, ⟨MS: pMS⟩, ⟨FS: y⟩, ⟨%R: p%R⟩, ⟨CL: z⟩}
15	**return** pf {retorna formulação do problema pf}
16	**end function**

O próximo passo é submeter à formulação do *problema-alvo* p ao gerenciador de casos.

4.3.2.3 Recuperação

Dada à formulação do problema-alvo p, a próxima tarefa é medir a similaridade entre p e $c_i \in CB$ e gerar um conjunto ordenado com os casos mais similares $R^+ = \{c_0 : s_0, \dots, c_n : s_n\} = \{\langle c_i : s_i \rangle\}_0^n = \{R_i^+\}_0^n$, onde $\langle c_i : s_i \rangle$ é um par caso/similaridade, $s_i \geq s_{i+1}$. A similaridade pode ser medida usando as distâncias: Euclidiana (2), Manhattan (3), Minkowski (4), dentre outras medidas. Nesta tese é utilizado a distância Euclidiana para medir a similaridade entre casos.

$$d_{c,c_i} = \sqrt{\sum_{k=1}^{m} \left(w_k (c_k - c_{ik})\right)^2} \tag{2}$$

$$d_{c,c_i} = \sum_{k=1}^{m} w_k |c_k - c_{ik}| \tag{3}$$

$$d_{c,c_i} = \left(\sum_{k=1}^{m} |c_k - c_{ik}|^p \right)^{\frac{1}{p}} \tag{4}$$

Alguns atributos utilizados na recuperação dos casos possuem valores discrepantes entre si,

como por exemplo, o quilômetro e percentual de rampa. O primeiro pode assumir qualquer valor positivo, enquanto o segundo apresenta valores restritos à uma faixa entre [-4, 4]. Esta discrepância torna os atributos com valores menores quase que insignificativos no cálculo da similaridade. Uma solução para este problema consiste em normalizar os valores de cada atributo a_j que compõe p para harmonizar as escalas dos atributos. A base de casos é normalizada no processo de atualização da base de casos, quando novos planos são obtidos.

— *Normalização da base de casos*

Aqui, definido o processo de normalização dos valores de p e dos casos da base CB. Seja X uma matriz com A colunas e n linhas representando a base de casos CB. Cada coluna representa um atributo $a_i \in A$ de um caso e cada linha um caso $c_i \in CB$. Seja U uma matriz com A colunas e uma linha, representando um problema-alvo p.

$$X = \begin{bmatrix} x_{1,1} & \cdots & x_{1,k} & \cdots & x_{1,A} \\ x_{j,1} & & x_{j,k} & & x_{j,A} \\ x_{n,1} & \cdots & x_{n,k} & \cdots & x_{n,A} \end{bmatrix}$$

$$U = [u_{1,1} \quad \cdots \quad u_{1,k} \quad \cdots \quad u_{1,A}]$$

Seja M uma matriz formada pela justaposição de X e U. A normalização se utiliza de M, sendo $M = \begin{bmatrix} X \\ U \end{bmatrix}$, onde seja $Z(x, k, M)$ uma função de normalização de atributo, onde a coluna k de X representa um atributo, a linha j (caso) os valores $x_{i,k} \in [0,1]$ de um caso, conforme (7). Os valores mínimos $(\min(M_k))$ e máximos $(\max(M_k))$ utilizados na normalização são determinados em função do domínio de valores de cada atributo da aplicação.

$$Z(x, k, M) = \frac{x - \min(M_k)}{\max(M_k) - \min(M_k)} \tag{5}$$

Após a aplicação da função $Z(x, k, M)$ os valores normalizados entre $[0,1]$, onde û representa os valares normalizados que descrevem o problema-alvo e $\hat{X}$ os valores normalizados que descrevem a base de casos.

$$\hat{U} = [Z(u_{1,1}, 1) \quad \cdots \quad Z(u_{1,k}, k) \quad \cdots \quad Z(u_{1,A}, A)]$$

$$\hat{X} = \begin{bmatrix} Z(x_{1,1}, 1) & \dots & Z(x_{1,k}, k) & \dots & Z(x_{1,A}, A) \\ Z(x_{j,1}, 1) & & Z(x_{j,k}, k) & & Z(x_{j,A}, A) \\ Z(x_{n,1}, 1) & \cdots & Z(x_{n,k}, k) & \dots & Z(x_{n,A}, A) \end{bmatrix}$$

— *Cálculo da distância Euclidiana*

O cálculo da distância Euclidiana, utilizada como métrica de proximidade entre o problema-alvo p e cada caso $c_i \in CB$ é realizado com os valores û e $\hat{X}$. Tais valores representam, respectivamente, o problema p normalizado e da base de casos CB normalizada. A distância é dada pela função $d(\hat{u}, \hat{x})$, onde k corresponde à um atributo $a_i \in A$, cujo peso w_k é dado, conforme Equação (6).

$$d(\hat{u}, \hat{x}) = \sqrt{\sum_{k=1}^{|\hat{x}|} w_k (u_k - \hat{x}_k)^2} \quad (6)$$

O resultado de cada cálculo é armazenado em um vetor v representado como um conjunto finito de n pares identificador/distância $v = \langle i, d \rangle$, onde i representa o índice do caso em $\hat{X}$ e d a distância de û para $\hat{x}$.

$$v = [\langle i, d(\hat{u}, \hat{x}) \rangle]_{i=1}^{n}$$

O par $\langle i, d \rangle$ que contém o caso mais similar ao problema descrito por û pode ser obtido pela função $c = \min(v, qte)$, onde v corresponde ao vetor de pares identificador/distância e qte indica a quantidade de pares mais similares a recuperar. Por exemplo, para qte igual a 1 tal função retornará um caso mais similar, para qte é igual a 50 tal função retorna os 50 casos mais similares.

4.3.2.4 Reuso

Seja $c_0 \in R^+$ o caso mais similar ao problema-alvo p. Resumindo o que já foi dito, a tarefa consiste em mapear a solução do caso recuperado c_0 para o problema-alvo p. O resultado é um novo caso c'; isto pode envolver a adaptação da solução para adequá-la à nova situação. Há vários algoritmos que podem ser usados para tal mapeamento, a saber: métodos de adaptação independentes do

conhecimento do domínio (e.g., algoritmo genético, redes Bayesianas, satisfação de restrições, *replay* derivacional), ou métodos de adaptação baseados no conhecimento do domínio (e.g., substituição, *ranking* e transformação). Observa-se que alguns dos processos de adaptação de casos são adaptáveis e alguns deles são específicos ou exatos. Porém, nenhum deles possui uma elevada independência do domínio de aplicação. Isto revela que a concepção de um algoritmo genérico para adaptação de casos que seja altamente independente do conhecimento do domínio ainda é um desafio. Neste trabalho é utilizada, para adaptar um caso em face de um problema-alvo, a abordagem evolucionária, em particular, algoritmo genético, detalhada mais à frente.

4.3.2.5 Revisão

Seja c' a solução mapeada para o problema-alvo p. A tarefa consiste em testar tal solução c'no mundo real (ou numa simulação), e, se necessário, revê-la. Em caso de falha, deve-se analisar o motivo da falha, repará-la e então aplicar a solução. A análise e reparação normalmente são feitas com base em regras do domínio de aplicação ou em informações fornecidas pelo usuário, e deve ser o mínimo possível para não prejudicar as vantagens inerentes à abordagem. A reparação de falhas envolve a identificação de uma cada das partes da solução proposta que falhou e a recuperação ou geração de uma explicação para cada uma delas. Em geral, quando um sistema inteligente precisa explicar uma falha, ele já resolveu o problema, tentou aplicar a solução no mundo real e obteve alguma resposta sobre a falha. Caso contrário, o sistema pode ter simulado a aplicação da solução e ter concluído que a solução não irá gerar os resultados esperados. No caso de sucesso, deve-se aprender com o sucesso obtido e então reter a nova solução na base de casos. Uma abordagem sem a intervenção humana é utilizada para revisar uma solução em face de um problema-alvo, onde as regras usadas são dependentes do domínio e encapsuladas em funções. Essas últimas serão detalhadas nas próximas seções.

4.3.2.6 Retenção

Depois que a solução c' foi adaptada com sucesso ao problema-alvo p, a próxima etapa, consiste em armazenar a experiência resultante como um novo caso na memória. O registro da nova solução enriquece o conjunto de experiências armazenadas e, consequentemente, amplia as possibilidades de resolução de novos problemas com o passar do tempo e sua utilização.

4.4 ABORDAGEM

O nosso principal objetivo é conceber um sistema inteligente capaz de conduzir um trem de carga. A consecução desse objetivo pode ser alcançado por partes, a saber: (i) gerar um plano ação/condução P; (ii) executar P, fazendo as correções Δ durante sua aplicação; e (iii) armazenar P e Δ na memória de experiências para ser usado como substrato em uma nova situação semelhante. Cada uma destas partes/atividades é atribuída a um agente de software distinto, cuja competência é a própria atividade. A Tabela 13 coloca em correspondência competências, agentes, tarefas de cada agente e os efeitos das tarefas. A distribuição dos agentes e comunicação entre eles será abordada mais à frente.

Tabela 13 – Correspondência entre tarefas e competências.

COMPETÊNCIA	NOME DO AGENTE	LOCALIZAÇÃO	TAREFA	EFEITO
Informar licença.	*Despachante*	Central de Controle	Informar licença.	Ambiente
Gerar plano.	*Planejador*	Estação	Adaptar, validar, reparar (se necessário), executar caso.	Memória de trabalho
Executar plano.	*Executor*	Computador de Bordo	Reparar (se necessário), executar o plano.	Ambiente
Armazenar plano.	*Memorizador*	Estação	Recuperar, registrar caso.	Memória de casos

A Figura 31 a seguir ilustra graficamente as responsabilidades e as trocas entre cada agente. A linguagem de comunicação agente limite-se a duas performativas: *request e inform*. A primeira codifica uma demanda e a segunda codifica uma resposta a uma demanda ou validação de uma informação.

A competência do *Despachante* é informar as licenças para as movimentações de trens da malha férrea. Cada licença encerra um contexto e serve como fronteira para cada agente *Planejador* e *Executor* atuar autonomamente. As licenças mais comuns são: liberação de movimentação de um determinado trem — ela pode atuar como gatilho para início o planejamento; nova velocidade máxima permitida em determinado ponto da via; permissão ou restrição de passagem em determinado setor.

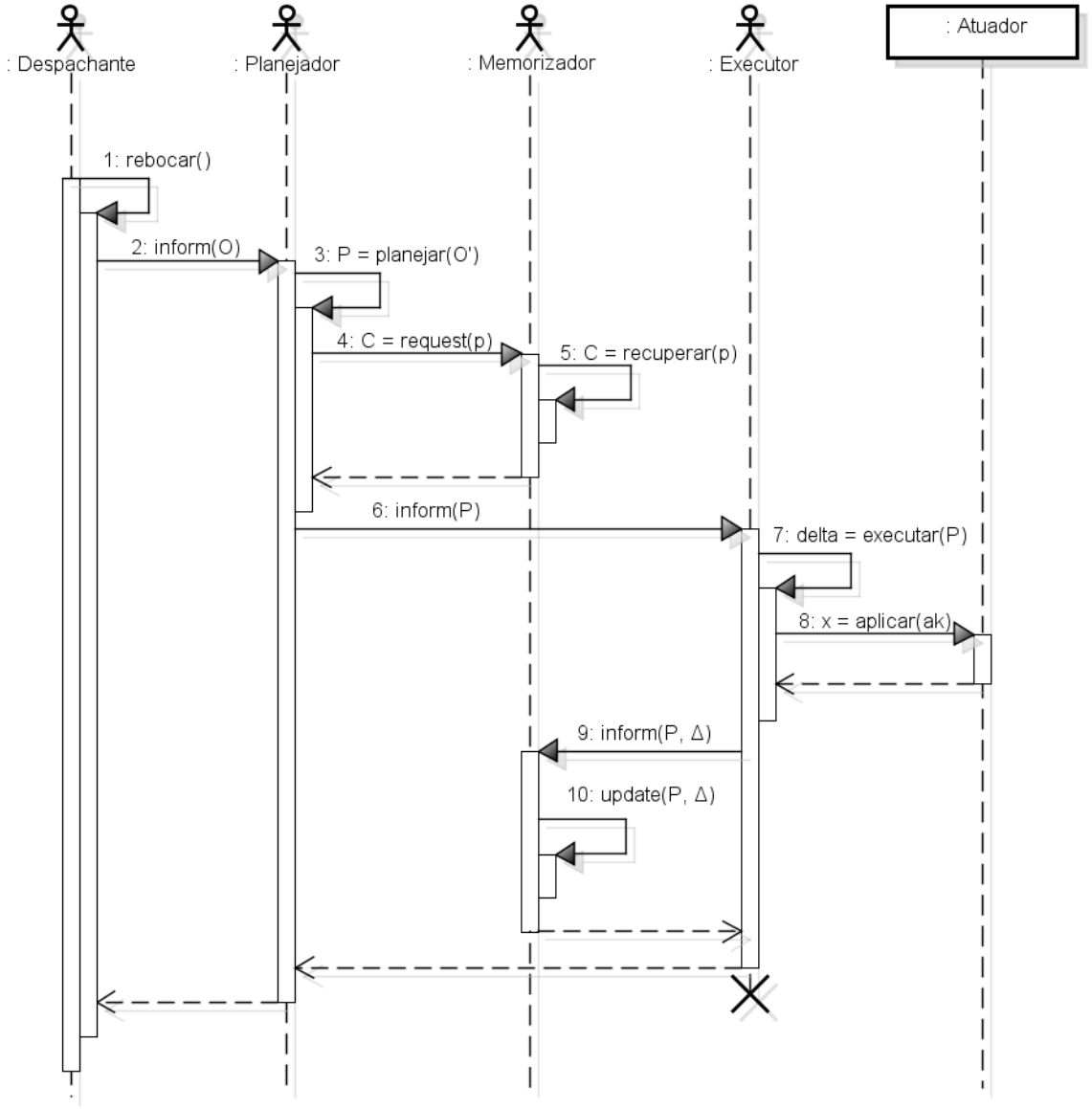


Figura 31 – Atribuição de responsabilidades aos agentes.

A competência do *Planejador* é gerar um plano de condução *P* para atender a demanda do Despachante, que consiste em rebocar o trem T_i de um ponto A até um ponto B. Cada ação a_k de *P*, descrita na solução de um problema-alvo p, pode assumir um dos seguintes comportamentos: acelerar, manter ou reduzir a velocidade do trem T_i. Como já dito, cada comportamento é ajustado de acordo com uma determinada potência resultante da aplicação de um ponto de aceleração. Logo, os comportamentos acelerar e reduzir correspondem, respectivamente, a aumentar e diminuir pontos de aceleração. Em uma locomotiva, cada ponto de aceleração, tipicamente [1; 8], gera uma potência positiva capaz de mover o trem. A redução de velocidade pode ser feita gerando uma potência menor que a soma das resistências ou aplicando freios. A frenagem consiste na aplicação de pressão no

encanamento de freios, medida em *psi* (*pressure* in pounds per *square inch*) e é representada aqui pelo ponto de aceleração -1. Para rebocar o trem deve-se selecionar um ponto de aceleração adequado para uma determinada posição do trecho da via férrea a percorrer, de modo a evitar patinagem e superar a soma das resistências.

Para garantir a movimentação do trem em todos pontos da via se faz necessário também conhecer a potência mínima requerida. O cálculo considera a força de tração mínima do conjunto das locomotivas destinadas a mover um trem face a maior resistência do trecho a percorrer. Essa potência mínima é calculada pelo *Planejador*; os detalhes serão mostrados mais adiante. Além disso, o conjunto de ações resultante deve atender vários objetivos: realizar uma viagem rápida, reduzir o consumo de combustível e respeitar as restrições de segurança — os dois primeiros estão em oposição. Assume-se a hipótese que, para reduzir o consumo de combustível e o tempo de viagem, deve-se planejar ações que desloquem o trem em velocidades próximas da velocidade cruzeiro, por exemplo, 5 km/h abaixo da velocidade máxima. O conjunto resultante é um plano de ação *P* para rebocar um trem T_i em um trecho ST_k.

A competência do *Executor* é aplicar o plano *P*. Ele o faz executando as seguintes tarefas básicas: testar a aplicabilidade de cada $a_j \in P$ com base nas condições atuais do trem T_i, ajustar os parâmetros de a_j (se necessário) e aplicar a_j. Até a completa execução de *P*, *P* pode sofrer vários ajustes Δ. Por exemplo, em caso de não aplicabilidade de uma ação a_j, o *Executor* deve ajustar a_j com base em seus conhecimentos locais em condução de trem de carga. A não aplicabilidade de uma ação por ser em função de condições adversas, tais como: mudança climática que altera o coeficiente de atrito, mudança de velocidade máxima permitida diferente daquela prevista no plano *P* original, entre outras. Tais condições percebidas por meio de vários sensores, que são lidos em intervalos de tempo predeterminados, conforme especificação do computador de bordo das locomotivas.

A competência do *Memorizador* é armazenar casos. Ele o faz executando duas funções básicas: (i) atualizar a base casos/planos executados; (ii) recuperar *(quando solicitado)* um conjunto de planos similares para cada problema-alvo devidamente formulado. A estrutura interna da base de casos segue um modelo em tabela, onde cada linha é um caso.

A Tabela 14 mostra os casos organizados de acordo com as ações tomadas (*ação*) e os constituintes da via férrea (*perfil, km, % de rampa*), deslocamento do trem coberto pelo plano/caso (*cl*), ponto de aceleração aplicado ap_i e posição de sua aplicação (m_i). Esta organização permite recuperar os casos mais similares ao trecho da via em questão e a ação a ser aplicada. Conforme visto anteriormente, os casos são normalizados para tornar viável o cálculo da distância.

Tabela 14 – Representação da base de casos local — Memorizador.

CASO	AÇÃO	PERFIL	KM	%R	CL	m_1	ap_1	m_2	ap_2	m_i	ap_i
1	Acelerar	Em Nível	339414	0.28	0.009	0	5	5	3	...	...
2	Acelerar	Em Nível	339404	-0.41	0,403	0	3	25	5		
3	Acelerar	Em Nível	277677	-0.03	0.915	0	4	10	4	...	...
...	...	...	...	...	...	...	...	...	...	...	

Na próxima seção é apresentado um detalhamento de cada agente da aplicação.

4.5 DETALHAMENTO DOS AGENTES DA APLICAÇÃO

A arquitetura envolve quatro agentes que implementam serviços bem-definidos. O cenário de funcionamento segue o seguinte enunciado: o *Planejador* recebe de um sistema externo uma demanda do *Despachante* (ordem de despacho) para planejar a tarefa de rebocar um trem sobre o trecho ST_k. Ele aplica o método Segmenta e Planeja de via férrea, apresentado anteriormente. Para cada $st_i \in ST_k$, um problema-alvo p_i é formulado e enviado, na forma de uma solicitação de casos similares, ao Memorizador; ele devolve os casos mais similares R^+. R^+é reduzido à um único caso c_i, adaptado para ser a solução do problema-alvo p_i . c_i é adicionado a P. Ao final, $P = \{\langle st_1, c_1\rangle, \langle st_2, c_2\rangle, \ldots, \langle st_n, c_n\rangle\}$ é o plano completo para rebocar o trem T no trecho ST_k. P é repassado ao *Executor*, que embarca no trem T e aplica P.

Durante a aplicação de P não há comunicação entre os agentes, aspecto diferencial frente a outras abordagens (Eldredge & Houpt, 2011) (Gu, Cao, & Tang, 2012) (Hengyu & Hongze, 2012). Todavia, como as condições de execução do plano P podem mudar, subplanos de P podem sofrer ajustes Δ. A capacidade do *Executor* em refazer subplanos de P garante uma condução viável e sem a necessidade de requerer um novo plano P' ao *Planejador*. Esse é um aspecto importante para tornar menos relevante o uso de canais de comunicação. A comunicação existente entre trem-estação é utilizada apenas para monitoramento e controle do posicionamento de cada trem. Isto ajuda a reduzir custos de transporte de carga e acoplamentos entre os módulos de software do sistema. É importante salientar que o plano aplicado e os ajustes retornam a base de experiências, gerida pelo *Memorizador*.

A distribuição básica dos agentes é feita usando a própria estrutura lógica e física da rede ferroviária. Em outras palavras, cada estação da rede ferroviária possui uma central computacional capaz hospedar dois tipos de contêineres de agentes, respectivamente, contêiner operacional e contêiner memória. Cada contêiner operacional hospeda um *Planejador* e um *Executor*. Cada contêiner memória hospeda apenas um *Memorizador*. Para cada trecho de via férrea, partindo de uma estação,

um contêiner operacional é instanciado. A Figura 32 ilustra um cenário com dois trechos $ST_1 = \langle S_{i,i-1,k} \rangle$ e $ST_2 = \langle S_{i,i+1,k} \rangle$, onde i corresponde à estação atual, $i-1$ corresponde à uma estação anterior (à esquerda), e $i+1$ corresponde à uma estação posterior (à direita).

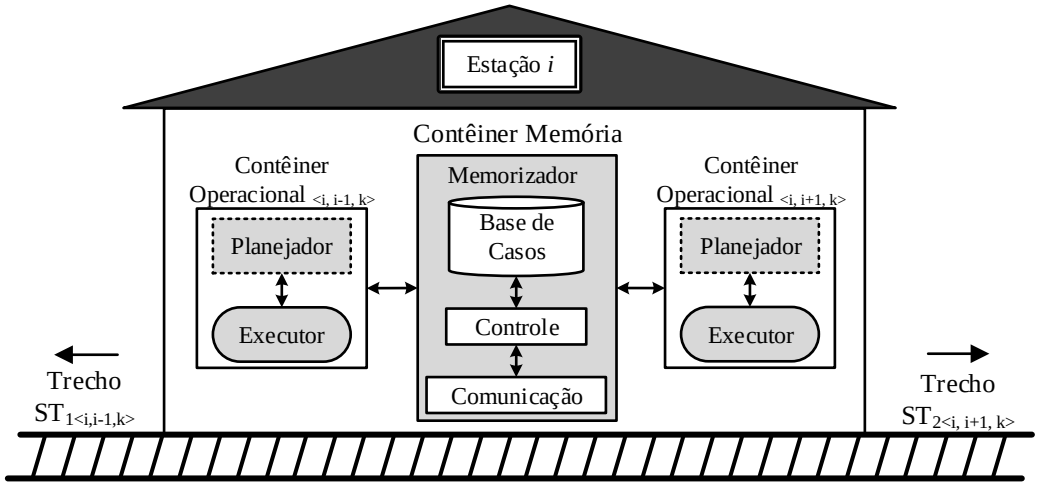


Figura 32 – Representação de uma estação como contêineres de agentes.

A seguir é discutido o detalhamento interno de cada agente presente no sistema, iniciando pelo agente *Despachante*.

4.5.1 *Agente Despachante*

O *Despachante* é o responsável pela veiculação comunicação entre as partes de uma rede ferroviária. O nosso interesse limita-se as ordens de movimentação de trens. Essas ordens são os gatilhos para o início das atividades do Planejador. Cada ordem de despacho pode ter informações, tais como: data, número da ordem, identificação do trem, identificação da estação, identificação do operador, número de locomotivas, número de vagões, locais de parada, tabela de horários, velocidades máximas permitidas em cada seção de bloqueio, seções com acesso permitido, entre outras. O formato de representação de uma ordem pode variar de acordo com o sistema emissor. Optou-se em usar o padrão XML (*eXtensible Markup Language*).

A Tabela 15 ilustra o esquema XML para uma ordem despacho.

Tabela 15 – Esquema de uma ordem de despacho em XML.

INFORMAÇÃO	TAG
Identificação do trem	`<TrainID></TrainID>`
Número de locomotivas	`<NL></NL>`
Número de vagões	`<NW></NW>`
Identificação do operador	`<OperatorID></OperatorID>`
Identificação da estação de origem	`<StationID></StationID>`

Locais de Parada	`<Stop>`
	`  <km> </km>...`
	`</Stop>`

Tabela de horários	`<Timetable>`
	`  <place>`
	`  <km></km>`
	`  <hour></hour>`
	`  </place>`
	`</Timetable>`

Velocidades máximas	`<MaxSpeed>`
	`  <place>`
	`  <km></km>`
	`  <speed></speed>`
	`  </place>`
	`</MaxSpeed>`

Sessões permitidas	`<SectionAllowed>`
	`  <id></id>`
	`</SectionAllowed>`

Assume-se que cada ordem de despacho é preestabelecida pelo *Despachante* da malha ferroviária, pois a criação e gerenciamento destas ordens não fazem parte do foco desta pesquisa.

4.5.2 *Agente Memorizador*

As suas competências são persistir e recuperar experiências. Nesta pesquisa optou-se por separar a etapa de recuperação do restante do ciclo do raciocínio baseado em casos, atribuindo-a a um agente. Esta atribuição permite ao agente, por exemplo, utilizar a métrica de cálculo de similaridade que considerar mais apropriada a situação.

A organização das experiências é dependente do domínio. A Tabela 16 mostra a organização interna da memória de casos. O esquema da tabela descreve três descritores diferentes, a saber: ação tomada, perfil do segmento da via férrea aonde a ação foi aplicada e índices da assinatura. Cada elemento do descritor de cada caso possui um peso. Cada linha i é um caso c_i.

Tabela 16 – Esquema parcial de armazenamento dos casos e pesos dos atributos das assinaturas. Neste exemplo, a solução dos casos é omitida.

i	CASO								
	AÇÃO	PERFIL	ASSINATURA						
			KM	NL	NW	IS	FS	MS	%R	CL
1	Acelerar	Em Nível	339414	3	58	9	10	40	0.28	0.009
2	Acelerar	Em Nível	339404	3	58	10	11	45	-0.41	0,403
3	Acelerar	Em Nível	277677	3	47	34	43	53	-0.03	0.915
4	Acelerar	Aclive	302883	4	59	24	25	53	1.54	0.243
5	Acelerar	Declive	335872	3	58	3	23	45	1.24	0.310
		Peso (w)	0.4	0.1	0.1	0.0	0.1	0.0	0.3	0.0

Legenda:
KM**:** Quilômetro inicial**,** NL: Número de locomotivas**,** NW: Número de vagões, IS: Velocidade inicial, FS: Velocidade final, MS: Velocidade máxima, %R: Percentual de rampa, CL: Deslocamento total (em km).

A parte relativa ao perfil é determinado de acordo com o percentual de rampa. A classificação do perfil foi realizada com base nos estudos publicados no livro "Estradas – Rodovias e Ferrovias" (Perreira, 1958). Esses estudos são ainda utilizados pelo Departamento Nacional de Infraestrutura de Transportes (DNIT). Segundo Pereira, para trens com bitola de 1.60m, um trecho plano possui um percentual de rampa (%R) máxima de 0.6%. Portanto, trechos planos são aqueles cujo percentual de rampa (%R) está $[-0.6, 0.6]$. Trechos de aclive são aqueles com $\%R > 0.6$ e trechos em declive aqueles com $\%R < -0.6$.

A função $f_{perfil}(x)$ encapsula as regras para determinar o perfil de um trecho via férrea com base no percentual de rampa. Essa função é usada pelo *Planejador* para compor parte da formulação de um *problema-alvo*. Ela também é usada *Memorizador* durante a preparação da base de casos (cf. coluna Perfil da Tabela 16).

$$f_{perfil}(x) = \begin{cases} Declive, & if\ x < -0.6 \\ Aclive, & if\ x > +0.6 \\ Em\ Nível, & padrão. \end{cases}$$

A rampa tem influência direta no cálculo de resistência para rebocar um trem, i.e., ela impacta na força-motor para vencer determinada velocidade e tempo de viagem, número de locomotivas, e, consequentemente, nos custos de operação.

4.5.2.1 Procedimento de recuperação de casos

Seja rq o *problema-alvo*. A tarefa de recuperar o caso mais similar em resposta a solicitação

em questão segue as seguintes etapas:

a) Interpretar o conteúdo da mensagem: nesta etapa, o agente Memorizador lê e interpreta o conteúdo da mensagem recebida, convertendo cada *tag* que compõe a *tag* '<caso>' para um atributo. Cada atributo convertido irá compor a assinatura do caso. Por exemplo, a tag <KM>339.400</KM> será convertida no atributo KM.

b) transformar rq na assinatura c_T para o problema-alvo; após ser feita a conversão do conteúdo da mensagem nos atributos que compõe o caso, o mesmo poderá ser normalizado.

$$c_T = \begin{Bmatrix} \langle EAC, Acelerar\rangle, \langle Perfil, Em\ Nível\rangle, \langle KM, 339.400\rangle, \langle NL, 3\rangle, \langle NW, 58\rangle, \\ \langle IS, 10\rangle, \langle FS, 11\rangle, \langle MS, 45\rangle, \langle \%R, -0.41\rangle, \langle CL, 0.403\rangle \end{Bmatrix}$$

c) selecionar o conjunto de casos a serem utilizados para o cálculo da similaridade: com base na ação a ser executada presente na assinatura de c_T o agente Memorizador seleciona a base de casos que contem apenas casos com a mesma ação executada em c_T. No caso do exemplo, o agente seleciona, dentre as bases de casos disponíveis ($BC_{acelerar}, BC_{manter}, BC_{reduzir}, BC_{frear}$) a base de casos $BC_{acelerar}$.

d) selecionar R da $BC_{acelerar}$ somente os casos que possuam um perfil de via igual ao perfil a ser planejado, no exemplo, casos cuja assinaturas possuem o valor 'Em Nível' para o atributo 'Perfil'. Lembrando, a opção de selecionar um conjunto de casos de acordo com a ação a ser executada e o perfil da via tem por objetivo reduzir o escopo de busca na etapa de recuperação de casos;

e) normalizar c_T (cf. Algoritmo 5): os valores numéricos que compõe c_T são normalizados obedecendo os mesmos valores mínimos e máximos de cada atributo que descreve os casos da base de casos.

Algoritmo 5 – Procedimento de normalização.

function *normalizarBC*(BC : MATRIZ) : BC NORMALIZADA		
BC: base de casos.		
01	**for** $k \leftarrow 0$ **to** $\lvert BC[A]\rvert$ **do**	{ atributos/coluna}
02	**for** $n \leftarrow 0$ **to** $\lvert BC[N]\rvert$ **do**	{ casos/linha}
03	$temp \leftarrow Z(BC_{n,k}, k, BC)$	{cálculo do valor normalizado}
04	$\widehat{BC}_{n,k} \leftarrow temp$	{armazenamento do valor normalizado}
05	**end for**	
06	**end for**	
07	**return** $\widehat{BC}$	
08	**end function**	

Para o exemplo do caso c_T anterior, o resultado da sua normalização é c_{Tn}.

$$c_{Tn} = \begin{Bmatrix} \langle EAC, Acelerar\rangle, \langle Perfil, Em\ Nível\rangle, \langle KM, 0.972\rangle, \langle NL, 0.75\rangle, \langle NW, 0.983\rangle, \\ \langle IS, 0.145\rangle, \langle FS, 0.183\rangle, \langle MS, 0.005\rangle, \langle \%R, 0.448\rangle, \langle CL, 0.013\rangle \end{Bmatrix}$$

f) determinar o conjunto de casos mais similares R^+ para rq a partir de c_T (cf. Algoritmo 6). Independente da quantidade de casos solicitados, são recuperados sempre os casos mais similares.

Algoritmo 6 – Procedimento de Recuperação.

function *recuperar* (u, q) : CONJUNTO DE CASOS	
u: problema-alvo. q: número de casos similares solicitado (ex. q=1 para o caso mais similar).	
	{Normalização}
01	$M \leftarrow \begin{bmatrix} X_{m,n} \\ u_{1,n} \end{bmatrix}$ {m é o número de casos da base; n é o número de atributos}
02	$\widehat{M} \leftarrow normalizarBC(M)$
	{Cálculo de distância entre o problema-alvo $\hat{u}_{m+1}$ e cada elemento da base $\widehat{M}$}
03	$y \leftarrow \emptyset$
04	**for** $i \leftarrow 0$ **to** $\lvert\widehat{M}\rvert$ **do** {$\lvert\widehat{M}\rvert$ é o número de linhas}
05	$y \leftarrow y \cup \langle i: d(\hat{u}_{m+1}, \widehat{M}_i)\rangle$ {cálculo e armazenamento da distância}
06	**end for**
	{Recuperação dos q elementos mais similares}
07	$R^+ \leftarrow \min(v, q)$
08	**return** R^+
09	**end function**

Para comparar o caso c_{Tn} com dois casos quaisquer c_1 e c_2 descritos abaixo, a distância entre c_{Tn} e c_1 é igual a 0.061, e a distância entre c_{Tn} e c_2 é igual a 0.299.

$$c_1 = \begin{Bmatrix} \langle EAC, Acelerar\rangle, \langle Perfil, Em\ Nível\rangle, \langle KM, 0.972\rangle, \langle NL, 0.75\rangle, \langle NW, 0.983\rangle, \\ \langle IS, 0.145\rangle, \langle FS, 0.183\rangle, \langle MS, 0.005\rangle, \langle \%R, 0.335\rangle, \langle CL, 0.005\rangle \end{Bmatrix}$$

$$c_2 = \begin{Bmatrix} \langle EAC, Acelerar\rangle, \langle Perfil, Em\ Nível\rangle, \langle KM, 0.504\rangle, \langle NL, 0.75\rangle, \langle NW, 0.98\rangle, \\ \langle IS, 0.15\rangle, \langle FS, 0.3\rangle, \langle MS, 0.1\rangle, \langle \%R, 0.4\rangle, \langle CL, 0.03\rangle \end{Bmatrix}$$

Devido a distância entre c_{Tn} e c_1 ser a menor, c_1 é determinado como o caso mais similar.

Se a base de casos estiver vazia, então é utilizado como solução para o *problema-alvo* um único par de acordo com a ação a ser executada, a saber: (i) para a ação ACELERAR, retorna-se um ponto de aceleração igual ao ponto de aceleração atual acrescido de uma unidade — respeitando o

limite máximo; (ii) para a ação REDUZIR, retorna-se um ponto de aceleração igual ao ponto de aceleração atual menos uma unidade — respeitando o limite mínimo que é 2, caso contrário é retornado 1; e (iii) por fim, para a ação MANTER, retorna-se o ponto de aceleração atual.

4.5.2.2 Procedimento de preparação da base de casos.

O procedimento de preparação da base de casos consiste da aplicação dos seguintes passos, executados no final de cada missão ou viagem:

1- Dados de viagem: eles são mantidos em uma memória temporária m_t, cuja estrutura é similar a relação da Tabela 17, onde o atributo q determina a ordem do evento.

Tabela 17 – Dados relativos ao plano de condução aplicado durante uma missão.

q	KM	NL	NW	VELOCIDADE	VELOCIDADE MÁXIMA	%R	PSI	AP
1	339414	3	58	9	40	0.28	85	3
2	339416	3	58	10	35	0.28	85	3
3	339423	3	58	9	40	0.28	85	5
4	339404	3	58	10	40	-0.41	85	3
5	339414	3	58	10	45	-0.41	85	3
6	339429	3	58	11	45	-0.41	85	5

2- Identificação de mudança de perfil: a partir do conjunto de dados de m_t são criados agrupamentos $g \in Grupo$. Um novo grupo é criado para cada alteração no valor de $\%R$. Para os valores da Tabela 17 foram criados dois agrupamentos e os índices correspondem à ordem dos eventos. O atributo *índice* é usado para obter, por exemplo, a velocidade inicial e final de cada subplano aplicado.

Tabela 18 – Agrupamento de dados relativos ao plano de condução aplicado durante uma missão.

GRUPO		DADOS DA MISSÃO							
g	ÍNDICE	KM	NL	NW	VELOCIDADE	VELOCIDADE MÁXIMA	%R	PSI	AP
1	1	339414	3	58	9	40	0.28	85	3
1	2	339416	3	58	10	35	0.28	85	3
1	3	339423	3	58	9	40	0.28	85	5
2	1	339404	3	58	9	40	-0.41	85	3
2	2	339409	3	58	10	45	-0.41	85	3
2	3	339429	3	58	11	45	-0.41	85	5

3- Criação de um caso: cada $g \in G$ é convertido em uma ocorrência da base de casos temporária BC_t. De cada grupo g são lidos os valores que compõe o caso. Para tal foi utilizada a seguinte convenção para recuperar dados de um grupo g: seja $f[x]_{g,p}$ uma função genérica, onde x é um atributo, g é um grupo e p é a posição de uma ocorrência dentro de g; os valores a e z para p indicam respectivamente primeira e última ocorrência de um grupo g. A Tabela 19 enumera as funções para obter o conjunto de derivações para um caso completo, quando aplicadas às informações da Tabela 18, resultam nos casos 1 e 2 da Tabela 16.

Tabela 19 – Cálculos dos atributos dos casos

FÓRMULA	DESCRIÇÃO
$KM = f[km]_{g,p=a}$	Quilômetro inicial
$NL = f[nl]_{g,p=a}$	Número de locomotivas
$NW = f[nw]_{g,p=a}$	Número de vagões
$IS = f[speed]_{g,p=a}$	Velocidade inicial
$FS = f[speed]_{g,p=z}$	Velocidade final
$\mathrm{MS} = \max_{1 \le p \le \lvert g \rvert} (f[speed]_{g,p})$	Maior velocidade dentre as ocorrências de um grupo g.
$\%R = f[\%R]_{g,p=a}$	Percentual de rampa
$CL = f[km]_{g,p=l} - f[km]_{g,p=a}$	Deslocamento total do caso
$Perfil = f_{perfil}(\%R)$	Perfil dominante em função do percentual de rampa.
$Ação = \begin{cases} Frear & se \ \frac{\sum_{p=0}^{\lvert g \rvert} f[psi]_{g,p}}{f[psi]_{g,p=l} - f[psi]_{g,p=f}} < (psiPadrao - 6) \\ Acelerar & se \ f[vel]_{g,p=a} < f[vel]_{g,p=z} \\ Reduzir & se \ f[vel]_{g,p=a} > f[vel]_{g,p=z} \\ Manter & cc \end{cases}$	Ação executada. A verificação ocorre nesta ordem: Frear: se o valor médio do *psi* aplicado ao longo de v_i for menor que o valor de referência (*psiMedio*) menos o valor da primeira aplicação; Acelerar: se houve variação positiva na velocidade; Reduzir: se houve variação negativa na velocidade; Manter: se não houve variação na velocidade.

4- Identificação da solução: a solução de cada caso c é também obtida a partir de cada grupo g. O processo consiste em percorrer os grupos g e identificar os locais m_i onde ocorreram mudanças no valor do ponto de aceleração ap_i. São criados então os pares $\langle ap_i : m_i \rangle$ para compor a solução do caso. Os locais são obtidos calculando a distância em relação à posição inicial do g para cada mudança de ponto de aceleração. Por exemplo, a solução derivado do grupo 2 da Tabela 18, segue o seguinte *script*: o primeiro registro deste grupo (índice 1) possui o ponto de aceleração 3, na posição inicial 339404, o segundo registro (índice 2) manteve o ponto de aceleração 3 na posição 339409, e no terceiro registro (índice 3) o ponto de aceleração mudou

para 5. Logo, aufere-se deste grupo o seguinte conjunto de pares $\langle ap_i : m_i \rangle$: $\langle 3:0 \rangle$,$\langle 5:25 \rangle$. O valor da posição do segundo par é obtido pela diferença entre a quilometragem do registro onde houve a mudança do ponto de aceleração e a quilometragem do primeiro registro. Cabe ressaltar que na criação dos casos a quantidade de pares $\langle ap_i : m_i \rangle$ correspondente à solução é indeterminada, podendo variar de 0 até um registro por metro percorrido. Durante a etapa de adaptação, discutida posteriormente, esta quantidade é tratada e limitada a 5 pares $\langle ap_i : m_i \rangle$ por solução de cada caso.

Os casos com valores inválidos são removidas da base BC_t. São inválidos aqueles casos em que não é possível obter o percentual de rampa ($\%R$), a velocidade inicial (IS) ou a velocidade final (FS) por falha nos sensores.

Uma crítica feita na literatura relacionada à etapa de Recuperação é o baixo tempo de resposta quando o número de casos comparados é elevado. Porém, nesta tese o número de casos utilizados não resultou em tempo de resposta baixo devido ao volume de casos ser pequeno e também devido ao uso de um módulo independente fisicamente e logicamente para armazenar, indexar e recuperar casos. Esta segmentação deu-se pela estruturação da Base de Casos de acordo com os comportamentos executados pelos condutores, a saber: *acelerar*, *manter*, *reduzir*, *frear* para obter a base de casos final $BC = BC_{acelerar} \cup BC_{manter} \cup BC_{reduzir} \cup BC_{frear}$. Logo, foi criada uma base de casos para cada comportamento, reduzindo assim a quantidade de casos em cada base de casos.

A reorganização da base de casos ocorre quando um novo plano de viagens é informado. Executa-se então os passos supracitados de preparação sobre o plano informado; esse processo também levam em conta os casos já presentes na base de casos BC.

4.5.2.3 Determinação de pesos

Vários experimentos foram realizados com o objetivo de encontrar o melhor peso para cada atributo utilizado no cálculo de similaridade. Os valores dos pesos dos atributos foram obtidos por meio de um processo de experimentação, o qual consistiu das seguintes etapas: (i) atribuir um valor w_j entre 0 e 1 como peso de cada atributo j, tal que $\sum_{j=1}^{n} w_j$ é 1; (ii) simular uma viagem utilizando os pesos atribuídos e (iii) avaliar a taxa de acerto dos casos recuperados. Durante o processo de experimentação, manteve-se as mesmas configurações de trens e via férrea. Assumindo-se um Δ de 0.1 para a variação do peso de cada atributo em cada experimentação, todas as combinações possíveis de

valores para cada w_i foram testadas. Os melhores valores para cada atributo j se encontram na linha (w) da Tabela 16.

Alguns atributos, apesar de possuírem um peso igual a zero, foram mantidos na assinatura do caso, como velocidade inicial (IS) e velocidade máxima (MS), porque são utilizados para determinar a ação executada de acordo com o Algoritmo 3 e deslocamento total (CL). O valor da velocidade inicial (IS) foi mantido para verificar se a ação executada foi atribuída corretamente. O valor da velocidade máxima (MS) foi mantido com a expectativa de ser usado em experimentos futuros com outros métodos. Por último, CL foi mantido na assinatura do caso pois ele determina, por quantos metros este caso pode ser aplicável ao longo de uma missão.

Durante o processo de experimentação foi utilizado um trem compreendendo de 3 locomotivas e 58 vagões, com peso total de 6278t. Outras configurações de trens poderiam ser utilizadas para este cálculo, contudo escolheu-se tal configuração por apresentar a menor quantidade de registros históricos inválidos. Pode-se dotar o recuperador de caso de um mecanismo que permite o ajuste automático dos pesos em função da configuração de trem. Este mecanismo não foi desenvolvido.

4.5.2.4 Mensagens

O protocolo de comunicação faz uso da performativa *request* para solicitar o caso mais similar ao *problema-alvo* descrito no seu conteúdo e da performativa *inform* para retornar como resposta o caso mais similar ao *problema-alvo* em questão. A quantidade de casos solicitados é descrita na tag *<qteCasos>*.

Exemplos:

```
(request :sender Planejador
       :receiver Memorizador
       :language XML
       :content (
         <?xml version=”1.0”?>
         <qteCasos>1</qteCasos>
         <problem>
           <EAC>Acelerar</EAC>
           <Perfil>Nível</Perfil>
           <KM>339,400</KM>
           <NL>3</NL>
           <NW>58</NW>
           <IS>9</IS>
           <FS>11</FS>
           <FAC>5000</FAC>
           <%R>-0.41</%R>
           <CL>0.400</CL>
         </problem>)
       :reply-with r1)
```

```
(inform  :sender Memorizador
       :receiver Planejador
       :language XML
       :content (
       <?xml version=”1.0”?>
       <listOfCase>
       <case>
         <EAC>Acelerar</EAC>
         <Perfil>Nível</Perfil>
         <KM>339,404</KM>
         <NL>3</NL> <NW>58</NW>
         <IS>10</IS> <FS>11</FS>
         <MS>45</MS> <%R>-0.41</%R>
         <CL>0.403</CL>
         <actions> <pair id=”1”>
              <ap>3</ap> <m>0</m>
           </pair>
           <pair id=”2”> <ap>5</ap>
              <m>25</m> </pair>
         </actions>
       </case>
       </listOfCase>)
       :in-reply-to r1)
```

As mensagens a seguir também utilizam a performativa *request* para solicitar o conjunto R^+ com 50 os casos mais similares ao *problema-alvo*, e a performativa *inform* para retornar como resposta os casos mais similares ao *problema-alvo* em questão. Esse conjunto solicitado por ser útil para formar, por exemplo, a população inicial de indivíduos usados na etapa de adaptação.

Exemplos:

```
(request :sender Planejador
       :receiver Memorizador
       :language XML
       :content (
       <qteCasos>50</qteCasos>
       <?xml version=”1.0”?>
       <problem >
         <EAC>Acelerar</EAC>
         <Perfil>Nível</Perfil>
         <KM>339,400</KM>
         <NL>3</NL> <NW>58</NW>
         <IS>9</IS> <FS>11</FS>
         <Δβ>2</Δβ> <VMP>45</VMP>
         <FAC>5000</FAC>
         <%R>-0.41</%R>
         <CL>0.400</CL>
       </problem>
       )
       :reply-with r1)
```

```
(inform  :sender Memorizador
       :receiver Planejador
       :language XML
       :content( <?xml version=”1.0”?>
       <listOfCases> <case>
         <EAC>Acelerar</EAC>
         <Perfil>Nível</Perfil>
         <KM>339,404</KM> <NL>3</NL>
         <NW>58</NW> <IS>10</IS>
         <FS>11</FS> <MS>45</MS>
         <%R>-0.41</%R> <CL>0.403</CL>
         <actions>
           <pair id=”1”> <ap>3</ap>
             <m>0</m> </pair>
           <pair id=”2”> <ap>5</ap>
             <m>25</m> </pair>
         </actions> </case> <case>…</case>
       </listOfCases>)
       :in-reply-to r1)
```

A seguir é discutido o processo de retenção de novas experiências por parte do Memorizador.

4.5.2.5 Retenção

A retenção de novas experiências inicia com a extração de casos dos históricos de viagens. A origem deste histórico pode ser de uma viagem feita por um maquinista e extraído dos registros do computador de bordo do trem, um simulador de formação de maquinistas ou um autômato. Nesta pesquisa foram utilizados registros históricos e de outros métodos computacionais simulados no laboratório. O processo de indexação para ambas as situações é o mesmo, sendo ele descrito na seção 4.5.2.1 referente à preparação da base de casos. A retenção dos casos segue o mesmo padrão de formato utilizado na geração dos históricos de viagens reais. Este formato foi escolhido para manter uma compatibilidade com os registros reais e possibilitar, futuramente, a utilização dos registros gerados nesta pesquisa por outros profissionais.

4.5.3 *Agente Planejador*

O Planejador é responsável por elaborar o planejamento de um conjunto de ações para rebocar um trem de um ponto A a um ponto B.

Tal planejamento inicia com o recebimento de uma demanda. Essa demanda inclui uma descrição detalhada da missão, a saber: trecho ST_k com início e término, trem T_i, potência e velocidade mínima de T_i sobre ST_k denotadas respectivamente por $paMin$ e $velMin$. Essas duas últimas, são necessárias para garantir a movimentação do trem em todos pontos do trecho ST_k. O cálculo leva em conta a força de tração mínima do conjunto de locomotivas destinadas a rebocar T_i face a maior resistência do trecho ST_k a percorrer. Nota-se que ST_k é formado por um conjunto de pontos, onde cada ponto encerra diferentes medidas extraídas dos projetos planimétricos e altimétricos, conforme ilustrado na Tabela 20; as dez primeiras colunas desta tabela foram copiadas da Tabela 6. De fato, a Tabela 20 acrescenta para cada ponto dois novos atributos: $paMin$ e $velMin$.

Tabela 20 – Pontos do projeto topográfico — incluindo potência e velocidade mínima.

IDEM TABELA 01									paMin	velMin
ID	KM	velMAX	RAMPA	raioCURVA	AC	G20	ALTITUDE	nroSB		
1	349	60	0.95	1150.5	25.48	0.99	526.10	3	2	12
2	349	60	0.95	1150.5	25.48	0.99	526.10	3	2	12
3	349	60	0.95	1150.5	25.48	0.99	526.10	3	2	12
4	349	60	0.95	1150.5	25.48	0.99	526.10	3	2	12
5	349	60	0.95	1150.5	25.48	0.99	526.10	3	2	12
6	350	60	0.95	1150.5	25.48	0.99	526.10	4	2	12
7	350	60	0.95	1150.5	25.48	0.99	526.10	4	2	12
8	350	60	0.95	1150.5	25.48	0.99	526.10	4	2	12
9	350	60	0.95	1150.5	25.48	0.99	526.10	4	2	12
10	350	60	0.95	1150.5	25.48	0.99	526.10	4	2	12
11	351	60	0.95	1150.5	25.48	0.99	526.10	4	2	12
12	351	60	0.95	1150.5	25.48	0.99	526.10	4	2	12
13	351	60	1.20	818.52	66.20	1.40	534.26	4	3	14
14	351	60	1.20	818.52	66.20	1.40	534.26	4	3	14

Os valores de *paMin* e *velMin* são determinados de acordo com o método descrito no Algoritmo 7. Esse método consiste em encontrar a menor potência — ou menor ponto de aceleração — necessária para rebocar um dado trem em uma determinada velocidade mínima, sobre o ponto de medida aonde o trem se encontra. Esse método é repetido para cada ponto de medida — ou segmento de 20m — da via férrea que o trem vai percorrer. Para cada trem diferente, o cálculo é refeito. Os atributos *paMin* e *velMin* da Tabela 20 são preenchidas por meio da aplicação do método ora descrito.

Em termos gerais, o método de planejamento desenvolvido segue a seguinte aproximação. Seja *P* uma política de ações — ou plano — para rebocar um trem T sobre um trecho ST_k. Ela consiste em selecionar e aplicar uma potência adequado para rebocar o trem T sobre cada segmente de mesmo perfil — ou rampa — $st_i \in ST_k$, onde st_i é formato por um conjunto de pontos de medida. A política de ações deve atender os seguintes critérios:

a) respeitar as restrições de segurança;
b) evitar patinagem e superar o somatório das resistências;
c) reduzir o consumo de combustível; e
d) reduzir o tempo de viagem.

Para a consecução desses objetivos recomenda-se: usar velocidades próximas da velocidade cruzeiro (e.g., 5 km/h abaixo da velocidade máxima). Dentro deste critério as ações empreendidas devem buscar um compromisso significativo para os objetivos em oposição, a saber: reduzir o consumo de combustível e o tempo de viagem.

Algoritmo 7 – Determinação da velocidade e do ponto de aceleração mínimo.

```
function calcularVelPaMin(T, ST) : ST
T: é um trem a ser rebocado.
ST: é um trecho onde T será rebocado.
01 pmFinal ← ST[T[pm_destino]]                {recupera o ponto de medida de destino do trem}
02 pmAtual ← ST[T[pm_atual]]                  {recupera o ponto de medida de atual do trem}
03 varVelMax ← 2                              {estabelece uma variação de velocidade de 2 km/h}
04 velocidadeMinima ← 2                       {atribui uma velocidade mínima inicial}
05 while(pmAtual ≠ pmFinal) do                {enquanto o trem não estiver no destino}
06     ap ← 1                                 {atribui o menor valor possível para o ponto de aceleração ap}
07     potencia ← f_pot(ap)                   {obtém a potência associada ao ponto de aceleração}
08     forcaTracao ← f_Ft(potencia, velocidadeMinima)
09     resistenciaTotal ← f_Rt(T, ST, pmAtual)
10     while(forcaTracao ≤ resistenciaTotal ) do
11         if(T[velocidade] > pmAtual[velocidadeMaxima] − varVelMax) then
12             ap ← ap + 1
13             if(ap > max(T[ap])) then
14                 exibirMensagem("Trem muito pesado para o trecho!")
15                 break
16             else
17                 potencia ← f_pot(ap) {obtém a potência do ponto de aceleração}
18                 velocidadeMinima ← velocidadeMinima + varVelMax
19             end if
20         else
21             velocidadeMinima ← velocidadeMinima + varVelMax
22         end if
23         forcaTracao ← f_Ft(potencia, velocidadeMinima)
24         resistenciaTotal ← f_Rt(T, ST, pmAtual)
25     end while
       {atribuição da velocidade mínima na posição pmAtual do trecho ST}
26     ST[pmAtual][velMin] ← velocidadeMinima
       {atribuição do ponto de aceleração mínimo na posição pmAtual do trecho ST}
27     ST[pmAtual][velPaMin] ← ap
28     pmAtual ← ST[T[pm_atual+1]]     {leitura do próximo ponto de medida}
29 end while
30 return ST
31 end function
```

Como já foi dito, cada ação a_k de P pode assumir um dos seguintes comportamentos: ACELERAR, MANTER OU REDUZIR a velocidade do trem. Cada comportamento é ajustado de acordo com uma determinada potência. Assim, de um lado, a elevação de velocidade pode ser feita aplicando uma potência maior que a soma das resistências, e de outro lado, a redução de velocidade pode ser feita aplicando freio ou uma potência menor que a soma das resistências.

Assim, seja T um trem com 4 locomotivas e 80 vagões, ST_k o trecho a ser percorrido, A e B os pontos de início e término, a cabeça do trem se encontra sobre o ponto A. A omissão termina quando a cabeça da locomotiva estiver sobre o ponto B. A sequência dos cálculos é:

a) determinar, para o ponto $pm \in ST_k$ aonde se encontra a cabeça do trem T, qual é o perfil

predominante x de pm — aclive, decline ou em nível;

b) formular, considerando o perfil dominante x, a descrição do problema-alvo pb.
c) recuperar, considerando a descrição do problema-alvo pb, os 50 mais similares casos R^+;
d) reusar o caso mais similar $c_0 \in R^+$ para resolver o problema-alvo pb; essa a tarefa consiste em mapear a solução do caso c_0 para o problema-alvo pb. O resultado é um novo caso *c'*; isto pode envolver a adaptação da solução para adequá-la à nova situação denotada pelo problema-alvo pb.
e) se c' for aplicável, então adicionar c' em P, senão identificar e reparar falha de *c*'.

Para definir os subtrechos $st_i \in ST_k$ de mesmo perfil foi aplicado o método Segmenta e Planeja de via ferra (cf. seção 4.3.2.1). Para cada $st_i \in ST_k$, um problema-alvo pb_i é formulado e enviado ao Memorizador na forma de uma solicitação de casos similares R^+. Esse último é reduzido à um único subplano p_i, adaptado para ser a solução do problema-alvo pb_i. p_i é adicionado a P. Ao final, $P = \{\langle st_1, p_1\rangle, \langle st_2, p_2\rangle, \dots, \langle st_n, p_n\rangle\}$ é o plano completo para rebocar o trem no trecho ST_k. P é repassado ao *Executor* que embarca no trem e o aplica. A função *planejar*, por meio do Algoritmo 8, formaliza tal aproximação.

Algoritmo 8 – Planejamento baseado em casos.

function planejar(ST_k, T, A, B) : $PLANO$
ST_k: um trecho de via férrea na forma de um conjunto de pontos de medidas.
T: um trem com uma ou mais locomotivas e zero ou mais vagões.
A: o ponto de medida aonde a cabeça do trem se encontra no início.
B: o ponto de medida aonde a cabeça do trem se encontrará no final.

```
01  Δl ← 0                                   {deslocamento}
02  Δt ← 0                                   {tempo}
03  p ← makePerception()                     {dicionário para registrar os dados lidos de sensores}
04  p[vel] ← 0                               {velocidade atual}
05  p[km] ← 0                                {posição aonde a cabeça do trem}
06  p[psi] ← 0                               {pressão do encanamento dos freios}
07  p[ap] ← 1                                {ponto de aceleração ou potência atual}
08  pm_atual ← A
09  pm_destino ← B
10  continue ← true
11  p[pm] ← pm_atual                         {ajuste da percepção para o ponto de medida inicial}
12  while(p[pm] ≠ pm_destino and continue) do

        {Percepção}
13      p ← perceber()                       {preenchimento de p por valores lidos dos sensores}

        {Cálculo das resistências}
14      R_t ← f_Rt(T, ST_k, p[pm])           {cálculo da resistência total}
15      perfil ← f_perfil(p[%R])             {determinação do perfil: ACLIVE, DECLINE, EM NIVEL}

        {Seleção do comportamento a empreender}
16      cp ← selecionarComportamento(p[vel], p[velMaxima], p[velMin], p[f_ac], p[%R])
        {ACELERAR, MANTER, REDUZIR, FREAR}
17      pb ← formular(p, T)                  {formular o problema-alvo}
18      R+ ← recuperar(pb, 50)               {recuperar os 50 casos mais simulares}
19      repeat
20          c_0 ← adaptar(R+, p, pb, T, ST_k)   {adaptar um caso de R+ face ao problema-alvo pb}
21          k ← 1
            {Validação do caso c_0}
22          while(k ≤ |c_0[a]| and continue) do   {percorre as ações a da solução adaptada c_0}
23              x = validar(c_0[a_k])        {validar a ação a_k da solução adaptada c_0}
24              if (x = 0) then              {ação a_k é aplicável}
                    {Atuação}
25                  P ← P ∪ c_0[a_k]         {adicionar a ação a_k ao plano P}
26                  aplicar(c_0[a_k])        {modificar variáveis (locais) de ambiente}
27                  k ← k + 1
28              else
29                  ex ← explicar(x, c_0[a_k])   {explicar a razão da falha da ação a_k}
30                  if (ex = 99) then continue ← false end if;
31                  else c_0[a_k] ← reparar(ex, c_0[a_k])   {repara a falha da ação a_k}
32              end if
33          end while
34      until k = |c_0[a]| or not continue
35  end while
36  return P
37  end function
```

O processo de teste e validação encerra-se em duas situações: (i) quando a ação adaptada é

válida ou (ii) quando não é possível obter uma ação que reboque o trem. A segunda situação pode ocorrer em função do peso do trem, mesmo empregando a potência máxima e na velocidade máxima, o trem permanece parado. Neste caso, é necessário adicionar uma locomotiva ao trem para gerar a potência necessária. O cálculo do número de locomotivas necessárias para movimentar um trem é de responsabilidade da central de operações.

4.5.3.1 Percepção e Recuperação

Cada percepção p corresponde a uma leitura de sensores feita a cada em intervalo de tempo t. Esse intervalo é determinado pela locomotiva líder e cálculos de resistência e tração são efetuados de 20m em 20m.

Para cada percepção p um *problema-alvo* pb é formulado. As informações que descrevem o *problema-alvo* pb são enumeradas na Tabela 12. Para relembrar o leitor, tais informações são: quilômetro (KM), número de locomotivas (NL) e vagões rebocados (NW), velocidade inicial (IS) e final (FS), variação da velocidade (Δv), velocidade máxima projetada (VMP), força de aceleração (FA), percentual de rampa (%R), deslocamento total (CL) e ação a executar (EAC). A Figura 33 ilustra a descrição prévia — não normalizada — de *problema-alvo*.

```
<?xml version="1.0"?>
<problem>
  <EAC>Acelerar</EAC>
  <Perfil>Planície</Perfil>
  <KM>339400</KM>
  <NL>3</NL>
  <NW>58</NW>
  <IS>9</IS>
  <FS>11</FS>
  <MS>45</MS>
  <%R>-0.41</%R>
  <CL>0.400</CL>
</problem>
```

Figura 33 – Formulação prévia — não normalizada — de um *problema-alvo*.

Para cada problema pb um conjunto de n casos similares pode ser recuperado a partir de uma base de casos BC. Por exemplo, o caso mais similar $c_0 = recuperar(CB, pb, n = 1)$ é definido pelo parâmetro n igual a um, os 50 casos mais similares $R^+ = recuperar(CB, pb, n = 50)$ é denotado por n igual à 50. A quantidade de casos a recuperar depende da abordagem de adaptação utilizada. Uma abordagem substitutiva que requer apenas um caso (Borges A. P., et al., 2014), enquanto que a

adaptação usando um algoritmo genético requer um número maior de casos para formar a população inicial, conforme descrito na próxima seção.

Um caso recuperado possui um espaço de aplicação (em metros) denotado por CL. Duas situações podem ocorrer. A primeira é quando o espaço a percorrer em $st_i \in ST_k$ é maior que CL, neste caso, um novo caso deve ser recuperado e adaptado para finalizar o deslocamento em st_i. Esta adaptação ocorre, na medida em que o caso recuperado percorreu um trecho menor que o necessário para problema-alvo. Um novo caso também deve ser recuperado quando ocorrer uma mudança no perfil vertical da via.

4.5.3.2 Adaptação e Reuso

O reuso de uma solução para resolver um novo problema, normalmente, requer a aplicação de técnicas de adaptação de casos. Dentre tais técnicas, este trabalho limita-se a duas delas, cujas diferenças estão na quantidade de casos necessários para obter a solução a um problema-alvo: (i) técnica substitutiva — o processo consiste em substituir os valores de um caso para atender as necessidades do *problema-alvo*. Aqui, requer-se apenas um caso — o caso mais similar ao problema-alvo; e (ii) técnica evolucionária — o processo é mais sofisticado à medida que ele requer uma etapa de *otimização* dos valores de um conjunto de n casos para determinar a melhor *solução* para o *problema-alvo* — os n casos mais similares ao *problema-alvo*. Neste trabalho utiliza-se a técnica evolucionária, em particular, o algoritmo genético devido à capacidade de otimização com sucesso quando aplicados em problemas similares. Além disso, este método pode ser inserido nos computadores de bordo das locomotivas, conforme discutido em (Chang & Sim, 1997) e possível de ser aplicado em um problema real (Amdouni, Jeddi, & Amraoui, 2013).

A adaptação genética requer um conjunto de n casos para formar a população inicial. Foi utilizado o valor de 50 para n por representar a quantidade média de casos similares obtidos na etapa de Recuperação. Em tese, a técnica deve determinar o melhor caso aplicável para solucionar o problema-alvo. Em termos de aplicação, ela deve determinar qual ação tomar para otimizar o consumo de combustível sem comprometer as restrições de controle, segurança e tempo.

De modo geral, o algoritmo de adaptação empregado segue a sequência canônica de passos da técnica genética supra destacada (cf. Algoritmo 9). A entrada do algoritmo é composta por uma população inicial — o conjunto de casos mais similares R^+ para um problema-alvo pb; a percepção do ambiente p; a descrição problema-alvo pb; as descrições do trem T e trecho de via férrea ST_k.

Fazendo uma analogia entre o algoritmo em questão e o Algoritmo 2, utiliza-se aqui uma população inicial formada pelo conjunto de casos recuperados mais similares R^+ ao problema-alvo pb e um conjunto de faixas específicas geradas pelo Algoritmo 10 da página 152. A seleção considera o efeito da simulação da aplicação de cada indivíduo ou ação para rebocar o trem T no trecho ST_k, considerando a percepção p e o problema-alvo pb para avaliar o indivíduo.

Algoritmo 9 – Adaptação de um conjunto de casos para solucionar um problema-alvo.

function adaptar(R^+, p, pb, T, ST_k) : SOLUCAO		
R^+: conjunto de casos recuperados; p: percepção; pb: problema-alvo; T: trem; ST_k: trecho.		
01	npop ← |R^+|	{tamanho da população de indivíduos}
02	tamString ← 10	{tamanho da cadeia de representação dos indivíduos}
03	maxGer ← 50	{número máximo de gerações}
04	$prob_m$ ← 50%	{probabilidade de mutação}
05	pop_0 ← gerarPopulacao(R^+, tamString, pb[CL])	
06	ngens ← npop × tamString	{número de genes da população}
07	g ← 1	{número inicial de gerações}
08	**while**(g < maxGer) **do**	{condição de parada cf. número de gerações}
09	solucao ← selecionar(pop_{g-1}, p, pb, $T_i ST_k$)	
10	pop_g ← reproduzir(pop_g)	
11	i ← 0	
12	**while**(i < npop) **do**	
13	pop_g ← fitness(pop_g[i])	{cálculo de fitness do indivíduo da população}
14	i ← i + 1	
15	**end while**	
16	i ← 0	
17	**repeat**	
18	i ← i + 2	
19	pop_g ← cruzar(pop_g, i, tamString)	{cruzamento de dois indivíduos}
20	**until** (i ≥ npop)	
21	i ← 0	
22	**repeat**	
23	i ← i + 1	
24	pop_g[i] ← mutar(pop_g[i], $prob_m$, tamString, minGer, maxGer)	
25	**until** (i ≥ ngens)	
26	g ← g + 1	
27	**end while**	
28	solucao ← selecionar(pop_g)	{seleciona o indivíduo com menor valor de fitness}
29	**return** solucao	{melhor solução}
30	**end function**	

A parte computacional da pesquisa em termos de computação evolucionária baseou-se no framework ECJ (Luke, Panait, Balan, & Et, 2013), com objetivo de auxiliar nas pesquisas em computação evolucionária. ECJ é flexível à medida que permite configurar os parâmetros de execução dos experimentos dinamicamente, via arquivos texto. Dentre outras configurações, ECJ permite configurar, por exemplo, os valores máximos e mínimos dos genes dos indivíduos sempre que a etapa de adaptação de casos ocorrer.

A avaliação de cada indivíduo é feita por uma função de *fitness*. Essa função é usada tanto para avaliação da população da inicial quanto durante a seleção.

— *Função de fitness*

Neste trabalho optou-se por usar a abordagem convencional *fórmula-ponderada* para o cálculo da função de fitness dos indivíduos. A escolha deu-se em função da abordagem de Pareto não combinar vários critérios em uma única fórmula e da complexidade em limitar os limiares utilizados pela abordagem lexicográfica.

Em termos da aplicação, um dos nossos objetivos é reduzir o tempo de uma missão e o consumo de combustível. Tal objetivo pode ser alcançado minimizando o valor da função de *fitness* g(x). Tempo e consumo são os critérios que mais impactam na geração de indivíduos e na sua utilidade na população. Cada critério possui peso, onde $wLTKB$ e wv são respectivamente os pesos para os critérios *consumo* (em LTKB) e *velocidade* (km/h) de acordo com a equação (7).

$$g(x) = \sum_{j=0}^{J} \left(\frac{1}{(wLTKB \times LTKB_j) + (wv \times v_j)} \right) \tag{7}$$

O critério de *velocidade* foi escolhido para compor a função de *fitness* por medida de segurança. Ela evita que o resultado da aplicação dos valores presentes no cromossomo dos indivíduos resulte em uma velocidade máxima maior que a permitida. Este fato poderá ocorrer, pois a velocidade é determinada em função da potência gerada pelo motor da locomotiva que por sua vez é determinada pelos valores dos pontos de aceleração presentes no cromossomo dos indivíduos.

Os valores de consumo e velocidade são obtidos pela simulação da aplicação dos valores que compõem o cromossomo de cada indivíduo. Logo, os seguintes valores são calculados durante a simulação: o deslocamento resultante da ação, resistência normal, esforço trator efetivo, esforço trator aderente, força de frenagem (se necessário), força de aceleração, velocidade no próximo ponto de medida, aceleração, tempo e consumo da ação. Tais valores são calculados de acordo com a Tabela 3 (página 31).

Neste processo é simulada a aplicação de cada ponto de aceleração ap_i na posição m_i indicada no indivíduo. A simulação utiliza a mesma configuração de trem e de trecho. A simulação utiliza também os dados dos sensores percebidos em p para configurar o trem (em termos de posição e ve-

locidade inicial), bem como a distância a percorrer. A Figura 34 ilustra este processo: dada uma percepção p (a), a simulação de um indivíduo é avaliada (b) e a função de fitness é calculada, o trem é reposicionado na posição inicial (c) e é realizada a simulação de um novo indivíduo (d). Esse processo se repete até avaliar todos os indivíduos.

Durante o cálculo de *fitness*, são atribuídas penalidades para as situações que ocasionem falta de segurança, danos a via férrea ou ao trem.

A Tabela 21 apresenta as situações onde são atribuídas penalidades máximas para os indivíduos fora de propósito. Estas penalidades visam evitar que tais indivíduos se propaguem na população. Se algum indivíduo gerado se enquadrar em alguma destas condições, o seu valor de fitness é igual a $100/(tamanhoGene/2)$. O valor 100 é significativo para os valores médios obtidos pela função de *fitness*. Um método de punição semelhante foi utilizado durante a elaboração de planos de condução com uso da técnica de satisfação por restrições (Leite A. R., 2009).

Tabela 21 – Condições de penalidade máxima

CONDIÇÃO	DESCRIÇÃO
$v \geq v_{max}$	Velocidade superior a velocidade máxima.
$F_t > (F_{tm} \times nl)$	Patinagem: F_t é esforço trator, F_{tm} é esforço trator máximo e nl é o número de locomotivas tracionando.
$\Delta\ell \geq \Delta\ell_{max}$	Deslocamento da ação ($\Delta\ell$) é maior que o deslocamento máximo previsto ($\Delta\ell_{max}$).
$F_{ac} < 0$ e $\Delta v > 0$	Aumento de velocidade ($\Delta v > 0$), mas sem força de aceleração suficiente ($F_{ac} < 0$).
$F_{ac} > 0$ e $\Delta v < 0$	Redução de velocidade ($\Delta v < 0$), mas com força de aceleração positiva ($F_{ac} > 0$).
$v_F > v_{max}$ e $v_F > v$	Velocidade final (v_F) superior a velocidade máxima e encontra-se em aceleração ($v_F > v$).

Há também outras situações de punições, porém mais brandas. Elas não são peremptórias por permitirem que o trem continue sendo rebocado com segurança, mas o resultado dos esforços não são ideais: (i) quando o deslocamento resultante da ação ($\Delta\ell$) é menor que 1m; ou (ii) quando o tempo resultante do deslocamento (Δt) é inferior a 1s. Nestas situações, o valor da função de fitness do indivíduo sofre um acréscimo de 10 unidades tal como realizado em (Leite A. R., 2009).

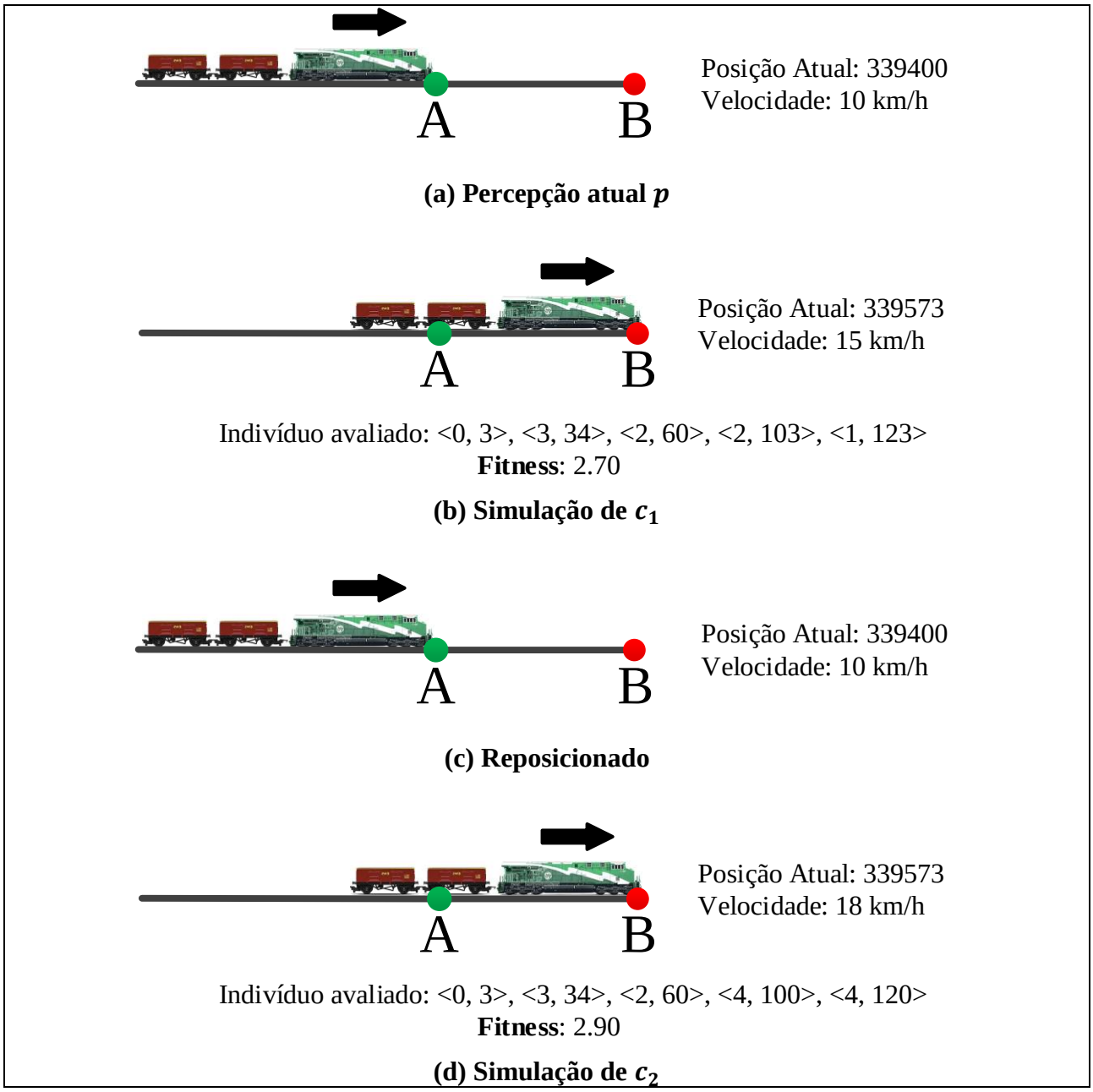


Figura 34 – Exemplo de simulação da função de fitness.

Como já foi dito, na composição da função de fitness são usados os pesos $wLTKB$ e wv, respectivamente para consumo e velocidade. Eles foram determinados usando um algoritmo genético, onde o cromossomo dos indivíduos consistia do par $\langle wLTKB, wv \rangle$. A população foi gerada de forma aleatória, com valores máximos de [0.01; 1.00] para $wLTKB$ e [0.01; 1.00] para wv, totalizando 100 possíveis valores para cada peso. Para cada indivíduo, uma viagem foi gerada e o consumo final atribuído como valor de fitness do indivíduo. Para a geração da viagem, foi utilizado o mesmo trecho ST_k dos experimentos, um trem com 3 locomotivas, 58 vagões e com peso de 6278t. O critério de

parada consistiu de 100 gerações. Os operadores de mutação e cruzamento utilizados para obtenção dos pesos foram os mesmos utilizados na etapa de Reuso. Dentre os indivíduos gerados, o que obteve o melhor desempenho para a função de *fitness*, consequentemente menor consumo, tinha como cromossomo o par $\langle 0.02, 0.01 \rangle$. Sendo assim, assumiu-se que o valor de $wLTKB$ seria igual a 0.02 e o valor de wv igual a 0.01.

— *Geração da população inicial*

Seja R^+ o conjunto dos casos mais similares para um problema-alvo pb. A população inicial é formada pela solução U, onde $U = R^+$ e a estrutura do cromossomo de tamanho fixo. Utilizando a ideia proposta por (Han, et al., 1999), foram determinadas características diferentes para os *cromossomos* pares e ímpares dos indivíduos. Assim, os genes pares representam pontos de aceleração (ap_i) da locomotiva líder e ímpares representam as posições de aplicação dos pontos de aceleração sobre o trecho de via férrea ST_k, cujo valor do alelo corresponde ao local (m_j) de aplicação (em metros) a partir do primeiro gene, de modo que o alelo do primeiro gene ímpar é sempre igual a zero, porque ele representa o ponto inicial definido para o caso a ser adaptado (cf. Figura 35). Quanto ao número de *cromossomos*, optou-se por um tamanho fixo, com 5 pares, i.e., 10 genes.

ap_1	m_1	ap_2	m_2	ap_3	m_3	ap_4	m_4	ap_5	m_5
5	0	3	5	4	25	6	38	8	50

Figura 35 – Exemplo de um indivíduo de entrada do algoritmo genético.

Neste problema foi escolhido a representação dos genes por números inteiros devido à facilidade de interpretação dos valores e por se tratar de um domínio de números inteiros. Além disso, os limites dos alelos superiores e inferiores dos genes pares são determinados dinamicamente em função do comprimento do trecho coberto pelo caso atual ($distancia$) e da quantidade de locais de aplicação de pontos de aceleração (determinado pela divisão do tamanho do indivíduo por 2). Tais alelos indicam a posição mínima e máxima de aplicação de cada ponto de aceleração. O comprimento do trecho coberto representa quantos metros ainda faltam percorrer até uma mudança de perfil da via férrea. Se tal distância for maior que uma janela de horizonte, então é atribuído um valor padrão igual a esta janela. A geração das faixas de valores é formalizada cf. Algoritmo 10.

Algoritmo 10 – Gerador de faixas.

function *gerarFaixas*$(distancia, t)$: FAIXA	
distancia: comprimento do trecho coberto pelo caso atual. *t*: tamanho da cadeia de representação de cada indivíduo.	
01	**for** i ← 0 **to** $(t/2) - 1$ **do**
02	faixas ← faixas ∪ $\left\lfloor \frac{\text{distancia}}{t/2} \times (i + 1) \right\rfloor$
03	**end for**
04	**for** i ← 0 **to** |faixas| **do**
05	**if** faixas[i] ≤ 0 **then**
06	k ← 2
07	**for** j ← 0 **to** $(t / 2) - 1$ **do**
08	faixas[j] ← faixas[j] ∪ k
09	k ← k + 2
10	**end for**
11	**else**
12	break
13	**end if**
14	**end for**
15	**return** faixas
16	**end function**

Por exemplo, se o tamanho do trecho a percorrer for de 173m, é gerado o seguinte conjunto de faixas: $\{34, 69, 103, 138\}$. A porção de código das linhas 6 a 10 evitam que sejam geradas faixas com valores negativos, se o valor da operação $distancia/(t/2)$ da linha 2 for negativo. Isto pode ocorrer para situações onde o deslocamento previsto (*distancia*) é inferior a 10m. Assim, o conjunto de faixas a ser gerado obedece o intervalo de máximo de aplicação do ponto de aceleração de 2m.

A Figura 36 mostra o resultado do processo. O primeiro gene par de cada indivíduo é sempre igual a zero para demarcar o ponto de aceleração inicial do caso. Por outro lado, cada alelo ímpar representa os pontos de aceleração de uma locomotiva a aplicar nas posições dadas pelos alelos pares; tais pontos variam no intervalo $z \in [-1, 8]$; por convenção o valor -1 indica frear.

ap_1	m_1	ap_2	m_2	ap_3	m_3	ap_4	m_4	ap_5	m_5
$[-1,8]$	0	$[-1,8]$	$[1, 34]$	$[-1,8]$	$[35,69]$	$[-1,8]$	$[70,103]$	$[-1,8]$	$[104,138]$

Figura 36 – Intervalos de valores possíveis de cada indivíduo.

A Tabela 22 enumera algumas funções úteis na definição do Algoritmo 11.

Tabela 22 – Funções utilizadas pelo Algoritmo 11.

FUNÇÃO	DESCRIÇÃO
$encode(U)$	Seja $U = \{ap_0: m_0, \dots, ap_n: m_n\} = \{< ap_i: m_i >\}_0^n = \{U_i\}_0^n$ a solução de um problema-alvo, onde $\langle ap_i: m_i \rangle$ é um par atributo/valor. ap_i é um ponto de aceleração ou potência e m_i é a posição de aplicação de ap_i.Esta função tem por objetivo codificar os valores da solução proposta em números inteiros. Por exemplo, o valor 8 é codificado como i8, onde *i* indica um número inteiro.
$keys(S)$	Para um conjunto de pares $S = \{ap_0: m_0, \dots, ap_n: m_n\} = \{\langle ap_i: m_i \rangle\}_0^n = \{S_i\}_0^n$, onde $\langle ap_i: m_i \rangle$ é um par atributo/valor. Ela retorna o conjunto $K = \{m_0, \dots, m_n\} = \{K_i\}_0^n$ com os valores de S.
$lastValue(S)$	Para um conjunto de pares $S = \{ap_0: m_0, \dots, ap_n: m_n\} = \{\langle ap_i: m_i \rangle\}_0^n = \{S_i\}_0^n$, onde $\langle ap_i: m_i \rangle$ é um par atributo/valor. Ela retorna o valor de ap do último par de S.
$Floor\lfloor x \rfloor$	Converte um valor x em seu menor número inteiro. Exemplo: $Floor\lfloor 2.4 \rfloor = 2$.

A entrada do algoritmo é composta por:

- um conjunto de casos recuperados $c_i \in R^+$; Nota-se que cada caso $c_i = \{\langle a_i: V_i \rangle\}_0^n$ é formado por um conjunto de n pares de atributo/valor. Cada caso c_i possui, dentre outros atributos, dois atributos em destaque, são eles: $paInicial$ — ponto de aceleração inicial e U — solução do problema associada ao caso c_i. A solução é expressa por $U = \{ap_0: m_0, \dots, ap_n: m_n\} = \{\langle ap_i: m_i \rangle\}_0^n = \{U_i\}_0^n$, onde $\langle ap_i: m_i \rangle$ é um par atributo/valor. ap_i é um ponto de aceleração ou potência e m_i é a posição de aplicação de ap_i;
- um número de genes t do indivíduo.

As variáveis utilizadas são:

- F: é um conjunto de locais de mudança de pontos de aceleração utilizado na padronização dos casos recuperados. Tal conjunto é gerado pelo Algoritmo 10, onde $F = \{m_0, \dots, m_n\} = \{m_i\}_0^n = \{F_i\}_0^n$, onde m_i é um local. Tal conjunto dado pela função $gerarFaixas$, por exemplo, $gerarFaixas(distancia = 173, t = 10)$, onde a distância CL é igual a 173m, resultará em $F = \{34, 69, 103, 138\}$;
- $UAjustado$: compreende de um conjunto — no mesmo formato de U — responsável por armazenar, ao longo da execução do algoritmo, o resultado do ajuste de U. Por exemplo, seja o caso recuperado for igual a $c_i = \{\langle 2: 60 \rangle, \langle 1: 123 \rangle\}$, ao final da execução do algoritmo, teremos $UAjustado = \{\langle 3: 0 \rangle, \langle 3: 34 \rangle, \langle 2: 60 \rangle, \langle 2: 103 \rangle, \langle 1: 123 \rangle\}$;
- $mTemp$: compreende de um valor de $m_i \in U$ em análise pelo algoritmo. Por exemplo, $mTemp = 60$;

- $deslocamento$: é um conjunto m dos pares de U que representa os locais, do caso recuperado, onde foram executadas as ações em relação ao início da aplicação do caso (posição 0). O formato é $deslocamento = \{m_0, \dots, m_n\} = \{m_i\}_0^n = \{deslocamento_i\}_0^n$. Por exemplo, $deslocamento = \{60, 123\}$;
- $deslocamentoAnalisado$: é um conjunto de valores do conjunto $deslocamento$ do caso recuperado que foram analisados pelo algoritmo durante sua execução. Possui o formato $deslocamentoAnalisado = \{m_0, \dots, m_n\} = \{m_i\}_0^n = \{deslocamentoAnalisado_i\}_0^n$ Por exemplo, para $p = 2$, o $deslocamentoAnalisado$ é igual a $\{60\}$;
- $paMedio$: é um conjunto de valores de ap contidos em c_i, usados para o cálculo do valor médio de ponto de aceleração ($paMedio$), onde $paMedio = \{ap_0, \dots, ap_n\} = \{paMedio\}_0^n$;
- $kmMedio$: é um conjunto de valores de m contidos em c_i, para o cálculo do valor médio de m_i de $U_i(tempKmMedio)$, $kmMedio = \{ap_0, \dots, ap_n\} = \{kmMedio\}_0^n$;
- $C[U][mTemp]$: recupera, da solução U do caso c, o valor de ap do par $\langle ap_i : m_i \rangle$ quando $m_i = mTemp$. Por exemplo, para $c_i = \{\langle 2: 60\rangle, \langle 1: 123\rangle\}$, com $mTemp = 60$ o retorno será igual a '2';
- $c[paInicial]$: recupera o primeiro ponto de aceleração aplicado pelo caso c;
- $c[CL]$: retorna o deslocamento, em metros, coberto pelo caso c;
- $faltaInserir$: compreende de uma variável inteira que determina quantos pares $\langle ap_i : m_i \rangle$ devem ser inseridos de modo que $UAjustado$ contenha 5 pares $\langle ap_i : m_i \rangle$

Algoritmo 11 – Geração de população a partir de um conjunto de casos.

```
function gerarPopulacao(R+,t, distancia) : POPULACAO
R+: conjunto de casos; t: tamanho do indivíduo; distancia: comprimento (em metros) coberto pelo caso.
01  F ← gerarFaixas(distancia, t)
02  for all c ∈ R+ do
03      if c[U] = ∅ then
04          c[U] ← processarCasoVazio(c, F)
05      else
06          UAjustado ← UAjustado ∪ (c[paInicial], 0);
07          deslocamento ← keys(UAjustado);
08          deslocamentoAnalisado ← ∅;
09          for p ← 1, |F| do
10              if F[p] < c[CL] then
11                  deslocamento ← deslocamento − deslocamentoAnalisado;
12                  deslocamentoAnalisado ← ∅;
13                  if p = |F| then
14                      for mTemp ← 1, |deslocamento| do
15                          paMedio ← paMedio ∪ C[U][mTemp];
16                          kmMedio ← kmMedio ∪ mTemp;
17                          deslocamentoAnalisado ← deslocamentoAnalisado ∪ mTemp;
18                      end for
19                  else
20                      for mTemp ← 1, |deslocamento| do
21                          if mTemp ≤ F[p] then
22                              paMedio ← paMedio ∪ c[U][mTemp];
23                              kmMedio ← kmMedio ∪ mTemp;
24                              deslocamentoAnalisado ← deslocamentoAnalisado ∪ mTemp;
25                          else
26                              break;
27                          end if
28                      end for
29                  end if
30                  if paMedio = ∅ then
31                      UAjustado ← UAjustado + (lastValue(UAjustado), F[p]);
32                  else
33                      tempPaMedio ← (1/n) Σ_{i=1}^{n} paMedio_i;
34                      tempKmMedio ← (1/n) Σ_{i=1}^{n} kmMedio_i;
35                      UAjustado ← UAjustado + (Floor⌊tempPaMedio⌋, Floor⌊tempKmMedio⌋);
36                  end if
37              end if
38          end for
39          faltaInserir ← (t/2) − |UAjustado|;
40          if faltaInserir > 0 then
41              UAjustado ← inserirFaltantes(faltaInserir, lastValue(UAjustado), F[|F| − y])
42          end if
43          c[U] ← UAjustado;
44      end if
45  end for
46  fit = funfit(c[U]);
47  content ← content + encode(fit);
48  return content
49  end function
```

Para cada solução $\boldsymbol{U}$ de um caso $\boldsymbol{c_i}'$, indicado por $\boldsymbol{c}[\boldsymbol{U}]$, primeiro é verificado se o conjunto solução $\boldsymbol{U}$ está vazio (linha 3). Se ele estiver vazio, uma tratamento particular é dado (cf. Algoritmo 12). De modo simplificado, gera-se um indivíduo cujos os pares que compõe U são formados da seguinte maneira $U = \{(paInicial, 0), (paInicial, 34), \ \ldots \ , (\ paInicial, 138)\}$, onde $paInicial$ é o mesmo para cada local de mudança de ponto de aceleração de F (e.g., $\{34, 69, 103, 138\}$).

Algoritmo 12 – Procedimento para gerar uma solução padrão quando caso em uso não dispõe de solução associada.

function $processarCasoVazio(c, F)$: CASO	
c: um caso recuperado. *F*: um conjunto de locais de mudança de pontos de aceleração.	
01	x ← (c[paInicial], 0);
02	c[U] ← c[U] ∪ x;
03	**for** i ← 1, |F| **do**
04	x ← (c[paInicial], F_i);
05	c[U] ← c[U] ∪ x;
06	**end for**
07	**return** c
08	**end function**

Se o caso analisado possui um ou mais pares na solução U, gera-se uma variável temporária ($UAjustado$) para armazenar os valores ajustados, contendo inicialmente o ponto de aceleração inicial (linha 06). São armazenados separadamente também os locais onde foram feitas as alterações nos pontos de aceleração, i.e., os valores m_i de U (linha 07). São armazenados os locais de mudança m_i já analisados (linha 08). Em seguida, avalia-se os locais de mudança gerados (linha 09) para adequar cada valor presente na solução do caso recuperado aos limites dos valores dos indivíduos gerados.

É analisado então se o local da mudança gerado é menor, em metros, que o comprimento coberto pelo caso (linha 10). Esta verificação é necessária pois pode ocorrer que o local indicado para mudança de ponto de aceleração possua um valor maior que o comprimento coberto pelo caso recuperado. Se o local gerado ($F[p]$) for menor que o comprimento coberto pelo caso, então os locais de mudança já analisados são removidos do conjunto de pontos de mudança (linha 11).

Em seguida, se o elemento analisado é o último de F (linha 13), então assume-se que foram feitas todas as alterações nos pontos de aceleração para as faixas de F. Os pontos de aceleração utilizados e também dos locais onde foram realizadas as alterações são armazenados (linhas 15-17). Os valores dos conjuntos $paMedio$ e $kmMedio$ são utilizados posteriormente para calcular um valor médio de ponto de aceleração utilizados e local de sua aplicação, respectivamente. Por exemplo, no caso recuperado c_5 da Tabela 23, todos os valores de mudança de local de aplicação de ponto de aceleração são maiores que a última faixa analisada (138). Nesta situação, é calculada a média dos

pontos de aceleração utilizados e também dos locais onde foram realizadas as mudanças. Tais valores médios irão corresponder ao par $\langle 7, 173 \rangle$ do caso ajustado.

Se o local de mudança analisado não for o último (linha 19), as médias de ponto de aceleração e local de aplicação são calculados com base nos valores menores que a faixa analisada (linhas 20-28).

Em seguida, verifica-se, se, para a faixa analisada, houve no caso analisado mais de um ponto de aceleração utilizado (linha 30). Se estiver vazio, armazena-se na temporária que representa a solução do caso ($UAjustado$) um par formado pela faixa analisada e o último ponto de aceleração da solução (linha 31). Se o conjunto de $paMedio$ não for vazio (linha 32), é inserido na solução ($UAjustado$) a média dos valores correspondentes aos locais de aplicação e os pontos de aceleração utilizados (linhas 33-35). Para trabalhar com números inteiros, os valores das médias resultantes são arredondados pela função $Floor\lfloor \cdot \rfloor$ (linha 35).

Para finalizar a padronização dos indivíduos, analisa-se quantos pares a solução temporária possui (linha 39). Caso o número for positivo, indicando que novos pares $\langle ap_i, m_i \rangle$ precisam ser incluídos para formar o indivíduo, são gerados pares cujo ap_i correspondem ao último ponto de aceleração armazenado (linha 41), segundo o procedimento $inserirFaltantes$ (Algoritmo 13). Por exemplo, os casos c_2, c_3, c_4 e c_5 da Tabela 23 fizeram uso desta funcionalidade.

Algoritmo 13 – Procedimento para inserir valores faltantes em *pontoMudancaPA*.

function inserirFaltantes($UAjustado, faltaInserir, ultimoPa, x$) : $UAjustado$	
$UAjustado$: conjunto com os pontos de mudança dos pontos de aceleração *AP*. $faltaInserir$: quantidade de valores a serem inseridos. $ultimoPA$: último ponto de aceleração utilizado. x: última faixa adicionada.	
01	**for** y ← faltaInserir, 0 **step** −1 **do**
02	UAjustado ← UAjustado + (F[x], ultimoPa);
03	**end for**
04	**return** UAjustado
05	**end function**

Finalizado este processo, o caso analisado c tem a solução U sobreposta pela solução gerada (linha 43). O valor do *fitness* do indivíduo é calculado (linha 46) e codificado (linha 47).

Para os casos de c_1 a c_6 da Tabela 23, conjunto de faixas $F = \{34, 69, 103, 138\}$ e $t = 10$, a aplicação do Algoritmo 11 deve gerar as duas últimas colunas da mesma tabela.

Tabela 23 – Exemplos de indivíduos gerados pelo Algoritmo 6

CASO	CL	PAINICIAL	U	VALOR DE U
c_1	176	-1	**Recuperado**	$\emptyset$
			Ajustado	$\{\langle -1,0\rangle, \langle -1,34\rangle, \langle -1,69\rangle, \langle -1,103\rangle, \langle -1,138\rangle\}$
c_2	503	0	**Recuperado**	$\{\langle -1,171\rangle\}$
			Ajustado	$\{\langle 0,0\rangle, \langle 0,34\rangle, \langle 0,69\rangle, \langle 0,103\rangle, \langle -1,171\rangle\}$
c_3	234	3	**Recuperado**	$\{\langle 2,60\rangle, \langle 1,123\rangle\}$
			Ajustado	$\{\langle 3,0\rangle, \langle 3,34\rangle, \langle 2,60\rangle, \langle 2,103\rangle, \langle 1,123\rangle\}$
c_4	66	3	**Recuperado**	$\{\langle 2,27\rangle\}$
			Ajustado	$\{\langle 3,0\rangle, \langle 2,27\rangle, \langle 2,69\rangle, \langle 2,103\rangle, \langle 2,138\rangle\}$
c_5	738	4	**Recuperado**	$\{\langle 6,123\rangle, \langle 7,183\rangle, \langle 8,215\rangle\}$
			Ajustado	$\{\langle 4,0\rangle, \langle 4,34\rangle, \langle 5,52\rangle, \langle 5,103\rangle, \langle 7,173\rangle\}$
c_6	644	8	**Recuperado**	$\{\langle 6,14\rangle, \langle 3,17\rangle, \langle 2,23\rangle, \langle 4,30\rangle, \langle 6,37\rangle, \langle 4,56\rangle, \langle 3,66\rangle, \langle 7,70\rangle, \langle 5,80\rangle, \langle 6,111\rangle, \langle 4,117\rangle\}$
			Ajustado	$\{\langle 8,0\rangle, \langle 3,21\rangle, \langle 4,53\rangle, \langle 6,75\rangle, \langle 2,114\rangle\}$

A partir da população inicial, novos indivíduos, chamados de filhos, são gerados pelo processo de $reproduzir(pop_g)$. Todos os novos indivíduos gerados obedecem os limites superiores e inferiores estabelecidos.

—*Mutação*

A operação de mutação visa trocar os valores de alguns genes da população. Neste trabalho, tal operação é implementada pelo procedimento $mutar$ (cf. Algoritmo 14), onde alguns genes dos indivíduos que compõe a população da sofrem alterações. A escolha de quais genes sofrerão o processo de mutação é aleatória, determinada pela função $prob = rnd()$, onde $prob \in [0,1)$ e, se a probabilidade de mutação $prob_m$ for maior que o valor $prob$, o gene x do indivíduo $iPop$ é alterado. Ele receberá um valor aleatório, gerado pela função $y = rnd(min, max)$, onde $y \in [min, max)$, min e max são os valores mínimos e máximos que o gene x pode utilizar, respectivamente.

Algoritmo 14 – Procedimento de mutação (Luke S. , 2014).

function $mutar(iPop, prob_m, tamString, minGer, maxGer)$: INDIVÍDUO

$iPop$: um individuo da população; $prob_m$: probabilidade de mutação;
$tamString$: tamanho do cromossomo do indivíduo; $minGer$: limite inferior; $maxGer$: limite superior.

```
01  if prob_m > 0 then
02      for x ← 0, tamString do
03          if prob_m > rnd() then
04              iPop[x] ← rnd(minGer, maxGer)
05          end if
06      end for
07  end if
08  return iPop
09  end function
```

Os melhores indivíduos após mutação são selecionados. Como resultado da fase de adaptação, tem-se um caso c. Esse caso é o melhor indivíduo adaptado de acordo com a função objetivo empregada.

— *Cruzamento*

A operação de cruzamento possibilita a recombinação das estruturas genéticas da população. Ela também permite diversificar o espaço de configurações por gerar estruturas diferentes das atuais. A combinação de cromossomos feita de dois em dois indivíduos da população.

Neste trabalho adotou-se o *cruzamento* de 1-ponto, à medida que este modo obteve os melhores resultados quando comparada às demais técnicas de cruzamento no problema do caixeiro viajante, um problema clássico de otimização combinatória (Mendes, 2013) (Picek, Golub, & Jakobovic, 2012).

O cruzamento de 1 ponto foi aplicado da seguinte forma (cf. Figura 37): para uma apenas determinada uma posição p, o primeiro filho (fc_1) recebe os genes do primeiro pai (c_1) até a posição p, e os genes do segundo pai (c_2) a partir da posição p. O segundo filho (fc_2) recebe os genes do segundo pai até o ponto p e os genes do primeiro pai de p em diante. O ponto de corte é determinado por um valor aleatório no intervalo $[0, tamString]$.

c_1	1	0	3	25	4	60	5	75	5	105
c_2	3	0	6	20	5	35	5	85	6	110
fc_1	1	0	3	25	5	35	5	85	6	110
fc_2	3	0	6	20	4	60	5	75	5	105

Figura 37 - Cruzamento de 1 ponto.

O cruzamento de um ponto foi escolhido porque ele repassa uma sequência de ações de um indivíduo para outro, conforme Algoritmo 15. Por exemplo, a sequência de locais e pontos de aceleração do indivíduo c_1 $\langle 4, 60\rangle, \langle 5, 75\rangle, \langle 5, 105\rangle$ é trocada com a sequência $\langle 5, 35\rangle, \langle 5, 85\rangle, \langle 6, 110\rangle$ do indivíduo c_2. Ambos os casos cruzados são avaliados pela função de fitness para determinar a sua aptidão.

Algoritmo 15 – Procedimento de cruzamento.

function $cruzar(pop, i, tamString)$: POPULAÇÃO	
pop_g: uma população. i: um índice que identifica o indivíduo na população. $tamString$: tamanho do cromossomo do indivíduo.	
01	ponto ← rnd(tamString)
02	$iPop_1 \leftarrow pop_g[i-1]$
03	$iPop_2 \leftarrow pop\_g[i]$
04	**for** x ← 0 **to** tamString **do**
05	tmp ← $iPop_1$[x]
06	$iPop_1$[x] ← $iPop_2$[x]
07	$iPop_2$[x] ← tmp
08	**end for**
09	$pop_g[i-1] \leftarrow iPop_1$
10	$pop_g[i] \leftarrow iPop_2$
11	**return** pop_g
12	**end function**

Em seguida a população é submetida ao procedimento de seleção.

— *Seleção*

A operação de seleção é realizada para cada geração. Devido à criação de novos filhos, o tamanho da população cresce; então um mecanismo de seleção controla esse tamanho.

Basicamente, a seleção, indicada pelo procedimento $selecionar(pop_g)$, determina quais indivíduos da população atual passam para a próxima geração. Neste trabalho, a seleção consistiu em, dado uma população pop_g, analisar o valor de *fitness* (indicado por $f(x)$) a cada 3 indivíduos (A, B e C), aquele indivíduo com valor de *fitness* intermediário é o retornado. Por exemplo, se $f(A) < f(B) < f(C)$ então B é selecionado. Este tipo de seleção, embora simples, faz com que seja evitado a seleção do melhor indivíduo e também do pior, garantindo uma homogeneidade da população.

— *Resultado da adaptação*

O processo de adaptação encerra fornecendo o indivíduo melhor avaliado após $ngens$ gerações de evolução; ele é a solução do *problema-alvo*. Por se tratar de um problema de minimização é selecionado o indivíduo que resulte no menor consumo de combustível e no menor tempo de viagem. Este indivíduo será aquele que possuir maior valor de fitness, uma vez que o valor de fitness $f(x)$ é inverso, ou seja, igual a $1/f(x)$. A Figura 38 ilustra o processo de adaptação discutido ao longo desta seção.

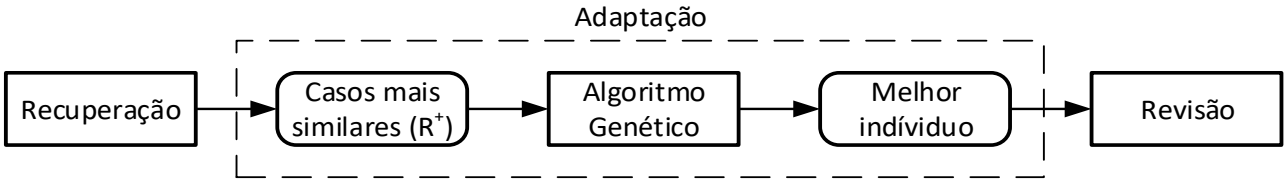


Figura 38 – Processo de adaptação: conjunto de treinamento, algoritmo genético e solução/melhor individuo.

O resultado do processo de adaptação de um conjunto de soluções de casos resulta em uma nova solução U', como por exemplo $\{\langle 3,0\rangle\langle 4{,}25\rangle\langle 4,40\rangle\langle 5,75\rangle\langle 5,110\rangle\}$. Logo, para cada par da solução U' os seguintes valores são calculados, nesta ordem: resistência normal, esforço trator efetivo, esforço trator aderente, força de frenagem (se necessário), força de aceleração, velocidade no próximo ponto de medida, aceleração, e, deslocamento resultante, aceleração, tempo e consumo da ação. Todos os cálculos são realizados para cada ação de acordo com as equações descritas na Tabela 3; cada par de U' pode ser entendido como uma ação a ser aplicada após determinado deslocamento.

O próximo passo é verificar, se a solução adaptada rebocará o trem de maneira segura e eficiente. Para isto, a solução é submetida ao processo de revisão.

4.5.3.3 Revisão

O caso que descreve a solução para o problema-alvo deve ser revisado antes dele ser adicionado ao plano de condução. Este procedimento consiste em avaliar a aplicabilidade de cada ação a_k do par da solução U'. A ordem de aplicação destes critérios é definida no processo de planejamento (cf. Algoritmo 8, da linha 22 a 33). Os critérios de validação estão descritos a seguir (cf. Tabela 24).

Uma reparação deve ser aplicada quando o valor produzido pela função $validar(a_k) \neq SUCESSO$. Tal reparação é sugerida pela função $testar(a_k)$. Essa função verificar primeiramente se há força suficiente para rebocar o trem — função $Força(a_k)$ — ,em seguida se a ação resulta em patinagem — $Patinagem(a_k)$ — , e por último se a ação resulta em redução de consumo de combustível — função $ConsumoExcessivo(a_k)$. Todas estas verificações utilizam os valores de forças calculados ao término da etapa de adaptação.

Tabela 24 – Funções de validações das ações planejadas e executadas.

FUNÇÃO	DESCRIÇÃO
$validar(a_k) = \begin{cases} SUCESSO & if(testar(a_k) = 0) \\ explicar(testar(a_k)) & c.c. \end{cases}$	Uma explicação é gerada se a solução proposta não for considerada aplicável.
$testar(a_k) = força(a_k) + patinagem(a_k) + consumoExcessivo(a_k)$	A soma igual à zero indica sucesso.
$patinagem(a_k) = \begin{cases} 1 & if \quad F_t > (F_{tm} \times nl) \\ 0 & c.c. \end{cases}$	O valor 1 indica patinagem, onde F_t é o esforço trator e F_{tm} é o esforço trator máximo, nl é o número de locomotivas tracionando.
$força(a_k) = \begin{cases} 3 & if \quad F_{ac} \leq 0 \\ 0 & c.c. \end{cases}$	O valor 3 indica falta de força de movimentação, onde F_{ac} é a força de aceleração da ação.
$consumoExcessivo(a_k) = \begin{cases} 7 & if \quad F_{ac}(ap_{i-1}) > 0 \\ 0 & c.c. \end{cases}$	O valor 7 indica excesso de consumo de combustível, onde $F_{ac}(ap_{i-1})$ é à força de aceleração empregada por uma ação uma unidade inferior ao ciclo atual.
$explicar(x) = \begin{cases} \{patinagem\} & if\ x = 1 \\ \{faltaForça\} & if\ x = 3 \\ \{consumoExcessivo\} & if\ x = 7 \\ \{patinagem, faltaForça\} & if\ x = 4 \\ \{patinagem, consumoExcessivo\} & if\ x = 8 \\ \{faltaForça, consumoExcessivo\} & if\ x = 10 \\ todas\ as\ condições & if\ x = 11 \end{cases}$	Indica o tipo de falha, onde *patinagem* indica condição de patinagem, $faltaForça$ indica falta de força para rebocar o trem e $consumoExcessivo$ indica excesso de consumo de combustível.
$reparar(x) = \begin{cases} 1- & if(X \subseteq \{patinagem, consumoExcessivo\}) \\ 1+ & if(X \subseteq \{faltaForça\}) \\ - & caso\ contrário \end{cases}$	Reparação da potência selecionada para uma ação; indica se o ponto de aceleração deve ser decrementado (1-) ou incrementado (1+) de uma unidade.

A solução reparada é novamente testada. Se a solução for considerada aplicável, o caso e todas as informações resultantes da sua aplicação são incluídas no plano, como os dados do trem (e.g., velocidade atual, pressão dos freios, número de locomotivas, número de vagões, origem, destino, quilômetro), da via férrea (e.g., pontos de medida alcançados – cf. Tabela 11) e informações calculadas (cf. Tabela 3).

É importante notar que os dados da última posição da solução validada — ou caso — servirão como os dados da próxima percepção do agente *Planejador*. Se tal posição for diferente da posição final de viagem ($pm_{destino}$), o ciclo de planejamento é repetido. Caso contrário o plano *P* é encerrado. O próximo passo é aplicar o plano P. A aplicação é feita pelo agente $Executor$.

4.5.4 *Agente Executor*

A primeira tarefa do *Executor* é receber o plano de condução *P* definido pelo *Planejador*,

armazená-lo na sua memória (e.g., exemplo de plano Figura 39). O controle da missão está unicamente com o *Executor*, i.e., cabe a ele aplicar cada ação do plano, assim como, proceder as eventuais reparações sobre as mesmas. O *Executor* embarca no computador da locomotiva principal e assume o controle da condução. Ao término na missão ele retorna para a sua estação de embarque e repassa o relatório de viagem ao agente *Memorizador*; esse último incorpora os resultados na forma de um conjunto de casos.

PM	KM	AP	IS	MS	nl	nw	f	R_{Total}	Δt	F_t	F_{ac}
6775	339.495	1	0	45	3	58	0.22	7816.568	6.08	67217.04	59400.47
6775	339.4933	1	6.878233	45	3	58	0.22	8451.187	9.13	8531.994	80.80704
6774	339.4733	1	6.88291	45	3	58	0.22	8449.712	9.13	8526.932	77.22018
6772	339.4533	1	6.887377	40	3	58	0.22	8493.778	9.12	8522.104	28.32537
6772	339.4333	7	6.889014	40	3	58	0.22	8493.938	4.64	369339.8	340930.9

Figura 39 – Exemplo de parte de um plano *P*.

Retornando ao processo básico de operação do *Executor*, ele embarca no trem, e em seguida ele realiza uma leitura inicial dos sensores. Esta leitura visa obter a posição e a operacionalidade dos sensores. Em cada execução da função de percepção do ambiente são lidas as seguintes informações: ponto de medida, quilometragem, velocidade, pressão do encanamento de freios, resistência, coeficiente de atrito (cf. Tabela 11), número de locomotivas e vagões e o ponto de aceleração atual.

A função *executar* (cf. Algoritmo 16) define o script de aplicação de um plano de condução P, definido para um trem T formado por locomotivas e vagões, em um trecho de via ST_i composto de um conjunto de pontos com medidas. Tal script consiste em aplicar, de forma recorrente, uma política de ações definida no plano P. O processo segue as seguintes etapas: recuperar uma ação $a_k \in P$, calcular a resistência total da embarcação, validar/testar a aplicabilidade da ação a_k, se a_k não for aplicável, ela deve ser ajustada — pelo próprio *Executor*. Se a ação a_k não for aplicável, a reparação da ação a_k consiste basicamente em aumentar ou reduzir o ponto de aceleração estabelecido (cf. Tabela 24). Após aplicar a ação a_k, deve-se verificar se a missão foi concluída. A situação consiste em testar se a posição atual pm — após a aplicação de a_k — é igual à posição de destino $pm_{destino}$. Caso negativo, repete-se o script. Caso positivo, finaliza-se o processo calculando: consumo e tempo, e unindo ações efetivamente executadas em um plano *P'*. $P' = P \cup \Delta$, onde Δ são os ajustes feitos localmente. *P'* é armazenado na base de casos como forma de aumentar a capacidade futura de resolver problemas. A atualização da base de casos é competência do agente Memorizador, como dito anteriormente.

Apesar do critério de correção, durante a execução do plano *P*, ser idêntico ao critério de revisão da etapa de planejamento — executada pelo Planejador —, o espaço de busca — executado pelo Executor — tende ser menor visto que a correção de a_k (quando necessária) não está distante, graças ao esforço de planejamento feito pelo *Planejador*.

A Figura 40 ilustra o exemplo das principais variáveis que compõe um plano de ação aplicado.

PM	KM	AP	IS	MS	nl	nw	f	R_{Total}	Δt	F_t	F_{ac}
6775	339.495	1	0	45	3	58	0.22	7816.568	6.087496	67217.04	59400.47
6775	339.4933	1	6.878233	45	3	58	0.22	8451.187	9.139105	8531.994	80.80704
6774	339.4733	1	6.88291	45	3	58	0.22	8449.712	9.133683	8526.932	77.22018
6772	339.4533	1	6.887377	40	3	58	0.22	8493.778	9.128511	8522.104	28.32537
6772	339.4333	8	6.889014	40	3	58	0.22	8493.938	3.74137	250497.8	242003.9

Figura 40 – Exemplo de parte de um plano *P* já aplicado.

De forma resumida, o *Executor* é responsável por aplicar o plano de condução *P* recebido no momento de sua instanciação. O *Executor* embarca na locomotiva líder do trem quando um canal de comunicação entre locomotiva líder do trem e estação é estabelecido. Qualquer canal de comunicação pode ser utilizado, seja por rede ou até mesmo por uma conexão física (e.g., USB). Uma vez embarcado no trem, a primeira atividade do *Executor* é verificar se os requisitos mínimos para executar a missão estão disponíveis, dentre eles: dados do perfil da via contendo informações de todos os pontos de medida que serão percorridos, conjunto de licenças, sensores em funcionamento, computador de bordo estável e demais requisitos de confiabilidade.

Na eventualidade de algum destes requisitos falhar, o *Executor* informa a estação e o maquinista via mensagem, indicando o requisito ausente. Se o requisito não for disponibilizado, a viagem deve ser realizada apenas pelo maquinista, visto que a totalidade dos requisitos mínimos para execução automática não foram atendidos.

Algoritmo 16 – Procedimento de execução de plano.

procedure $executar(P, ST_k, T) : \Delta$		
P: um plano de condução; ST_k: um trecho; T: um trem a ser rebocado.		
01	armazenarPlano(P)	{armazena o plano P na memória do agente}
02	$\Delta l \leftarrow 0$	{deslocamento}
03	$\Delta t \leftarrow 0$	{tempo}
04	p ← makePerception()	{dicionário para registrar os dados lidos de sensores}
05	$pm_{destino} \leftarrow$ p[kmDestino]	{ponto aonde T_i deverá estar no final da missão}
06	p[pm] ← $p[pm_{atual}]$	{ponto aonde T_i se encontra inicialmente}
07	**while**(p[pm] ≠ $pm_{destino}$) **do**	
	{Percepção}	
08	p ← perceber()	{lê dados de sensores—percepção atual p}
	{Cálculo das resistências}	
09	$R_t \leftarrow f_{Rt}(T, ST_k, p[pm])$	{calcula a resistência total}
	{Seleção da ação}	
10	$a_k \leftarrow$ recuperarAcao(p, P)	{recupera ação a_k do plano P}
11	**repeat**	
	{Validação = Cálculo das forças}	
12	**x ← validar(a_k)**	{valida ação a_k}
13	**if** (x = 0) **then**	{testa se ação a_k é aplicável}
	{Atuação}	
14	aplicar(a_k)	{aplica ação a_k}
15	**else**	
	{Validação = Cálculo das forças}	
16	ex ← explicar(x, $c_0[a_k]$)	{explica motivo da falha da ação a_k}
17	$c_0 \leftarrow$ reparar(ex, $c_0[a_k]$)	{repara falha da $a_k$com base na explicação ex}
	{Memória}	
18	$\Delta \leftarrow \Delta + c_0$	
19	**end if**	
20	**until** c_0 é aplicável	
21	**end while**	
22	**end procedure**	

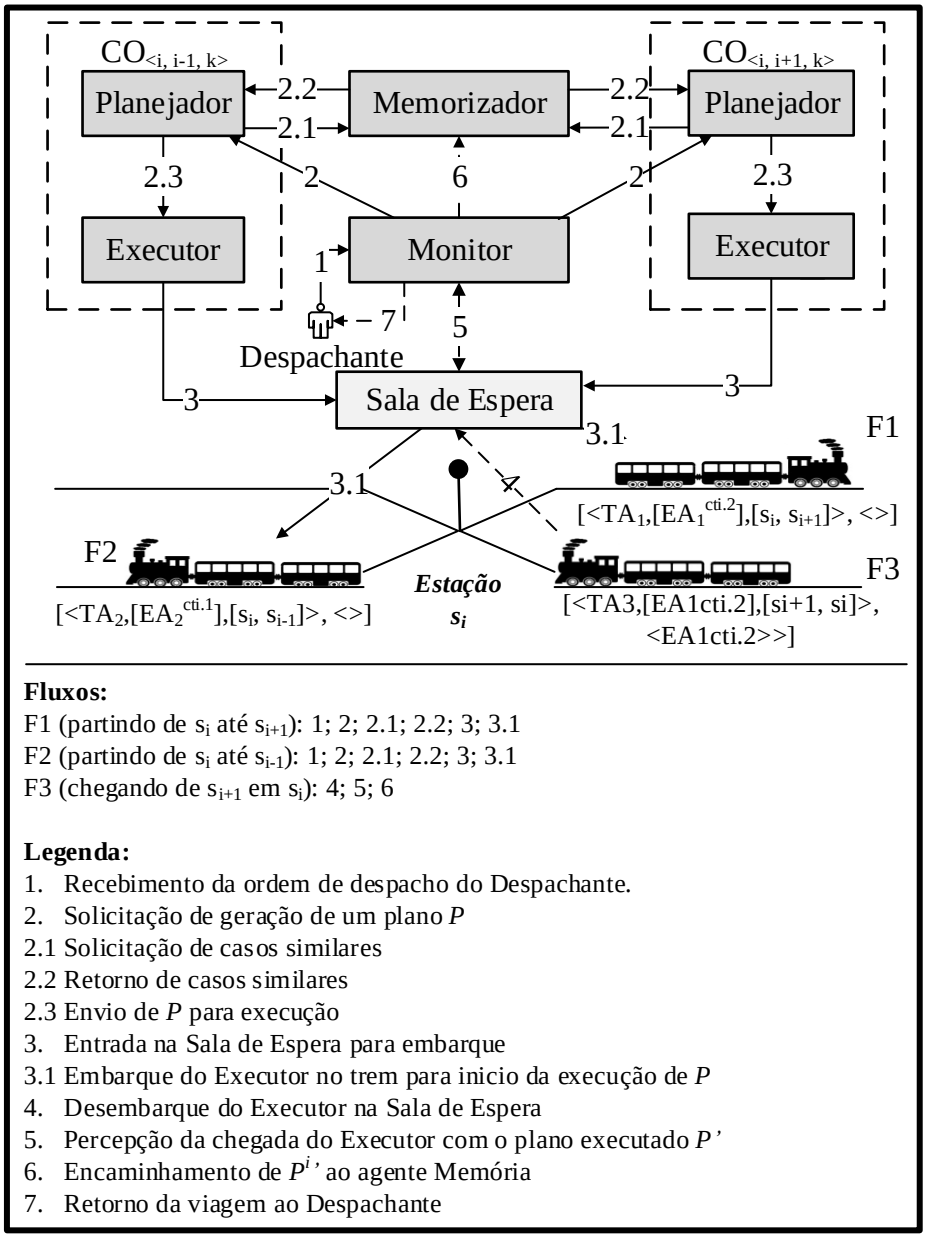
A forma modular como foi tratada a condução trens, nos permitiu distribuir logicamente os tarefas de armazenamento de casos, geração de políticas de ações e aplicação de tais políticas em três tipos diferentes de agentes. Cada um especialista em uma destas tarefas. Pode-se ter ainda várias instâncias de cada tipo de agente distribuídos logicamente e fisicamente em uma rede assíncrona de agentes.

4.6 FLUXO DE DADOS INTER E INTRA AGENTES

A arquitetura proposta para organizar os agentes considera a existência de uma infraestrutura formada por uma rede de unidades computacionais interligadas. As unidades podem estar fisicamente distribuídas nas estações de trens de uma ferrovia. Cada unidade é dotada de contêiner de hospedagem e execução de agentes.

A Figura 41 ilustra uma unidade computacional com contêineres de hospedagem e execução

de agentes. Essa unidade hospeda os seguintes agentes: *Memorizador*, *Planejador*, *Executor*, *Monitor e Sala de Espera.* Os três fluxos básicos F1, F2 e F3 descritos na própria figura dão uma ideia do funcionamento operacional de uma unidade de trabalho. Cada fluxo é associado a um par de *tuplas* $[\langle TA_1,\ [EA_1^{ct_{i.2}}, P],\ \ [s1,\ s2]\rangle, \langle LEA\rangle]$, onde TA_1 representa o trem que será conduzido da estação s_1 até s_2 pelo Executor $EA_1^{ct_{i.2}}$, usando o plano P^1 feito pelo Planejador $PA_1^{ct_{i.2}}$ e LEA[s_i] representa a lista de agentes executores que retornam às suas estações de origem s_i.



Fluxos:
F1 (partindo de s_i até s_{i+1}): 1; 2; 2.1; 2.2; 3; 3.1
F2 (partindo de s_i até s_{i-1}): 1; 2; 2.1; 2.2; 3; 3.1
F3 (chegando de s_{i+1} em s_i): 4; 5; 6

Legenda:
1. Recebimento da ordem de despacho do Despachante.
2. Solicitação de geração de um plano *P*

2.1 Solicitação de casos similares
2.2 Retorno de casos similares
2.3 Envio de *P* para execução

3. Entrada na Sala de Espera para embarque

3.1 Embarque do Executor no trem para inicio da execução de *P*

4. Desembarque do Executor na Sala de Espera
5. Percepção da chegada do Executor com o plano executado *P'*
6. Encaminhamento de P^{i}' ao agente Memória
7. Retorno da viagem ao Despachante

Figura 41 – Fluxo básico das trocas de dados e controle interagentes em uma mesma unidade computacional.

O Fluxo F1 enumera a execução de um conjunto de atividades, que inicia quando o Monitor recebe do *Despachante* uma demanda (1), que consiste conduzir o trem TA_1 da estação s_i até a estação s_{i+1}. Ele termina quando o *Executor* embarga em TA_1 para conduzi-lo (3.1), usando o plano de condução *P* feito pelo *Planejador*. O Fluxo *F*3 representa o retorno do *Executor* do destino s_{i+1} à estação de s_i. Este fluxo termina quando o plano *P'* é efetivamente aplicado pelo *Executor* é repassado ao *Memorizador*, cuja principal finalidade é atualizar a base de casos local da unidade computacional.

A Figura 42, ilustra o processo de planejamento e execução com maiores detalhes, o *Despachante* foi omitido. Assim, o fluxo está centrado na elaboração de um plano *P* para um trem TA_1. O fluxo começa pelo *Planejador*, cuja sua primeira atividade é receber a ordem de despacho gerada pelo *Despachante* e gerar um novo problema. Na sequência, dois ciclos de colaboração se estabelecem: o primeiro envolve iterativamente o *Planejador* e o *Memorizador*, e o segundo envolve sequencialmente *Planejador, Executor e Memorizador*. Doravante, usaremos PA1 para designar um *Planejador*, EA1 para um *Executor* e MA1 para um *Memorizador*.

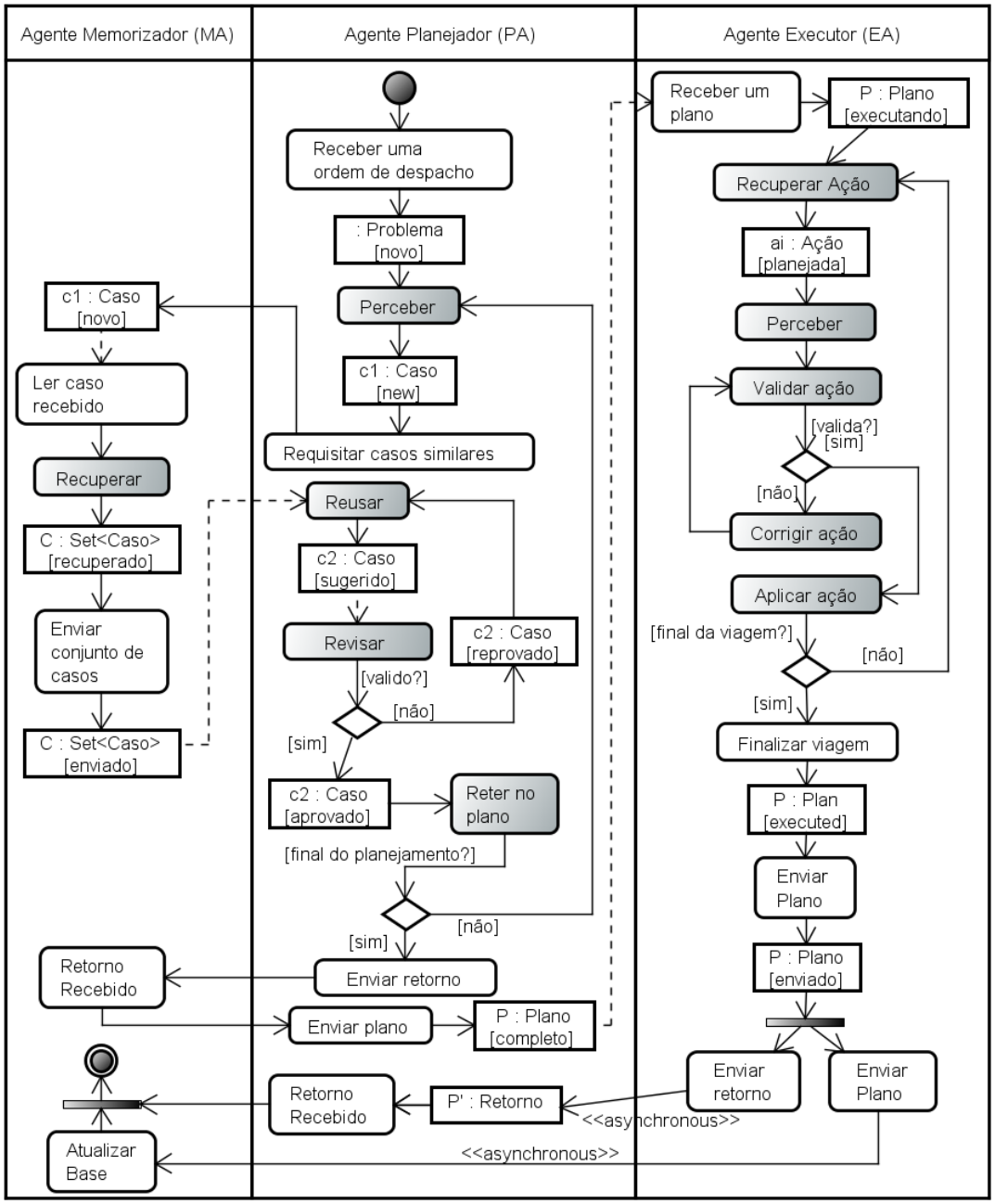


Figura 42 – Fluxo básico das trocas de dados e controle inter & intra-agentes.

A seguir são detalhadas as mensagens de colaboração entre os agentes situados em uma estação (cf. Figura 42).

— *Fluxo de Planejamento: iteração entre Planejador e Memória*

O planejamento inicia quando o PA1 recebe do *Despachante* uma ordem de despacho.

```
(request
    :sender Despachante
    :receiver PA1
    :language XML
    :content (
       <?xml version=”1.0”?>
       <order>
       <origem>Da Luz</origem>
       <destino>Central do Brasil</destino>
       <NL>3</NL>
       <NW>58</NW>
       <data>13/01/2015</data>
       <OperatorID>12312</OperatorID>
       <TrainID>1233</TrainID>
       <Timetable><place><km>268795</km><hour>12:00:00</hour></place></Timetable>
       <MaxSpeed><place><km>300000</km><speed>25</speed></place></MaxSpeed>
       <SectionAllowed><id>1</id></SectionAllowed>
       </order>)
    :reply-with r1)
```

Em seguida, PA1 inicia o planejamento. Ele formula o *problema-alvo* com base na percepção e repassa-o ao MA1 na forma de uma solicitação, esperando receber como resposta o caso similar ao *problema-alvo*. O valor da tag <*qteCasos*> indica a quantidade de casos que PA1 espera receber como resposta da solicitação.

```
(request
    :sender PA1
    :receiver MA1
    :language XML
    :content (
       <qteCasos>50</qteCasos>
       <?xml version=”1.0”?>
       <problem>
          <EAC>Acelerar</EAC> <Perfil>Nível</Perfil> <KM>339,400</KM>
          <NL>3</NL> <NW>58</NW> <IS>9</IS> <FS>11</FS> <FAC>5000</FAC>
          <%R>-0.41</%R>  <CL>0.400</CL>
       </problem>)
    :reply-with r1)
```

Internamente, a mensagem é recebida e tratada pelo *MA1* e o caso recuperado é então enviado ao PA1.

```
(inform
       :sender MA1
       :receiver PA1
       :language XML
       :content(
          <?xml version="1.0"?>
          <listOfCase>
             <case>
             <EAC>Acelerar</EAC> <Perfil>Nível</Perfil>
             <KM>339,404</KM> <NL>3</NL> <NW>58</NW> <IS>10</IS> <FS>11</FS>
             <MS>45</MS> <%R>-0.41</%R> <CL>0.403</CL>
             <actions>
                  <pair id="1"><ap>3</ap><m>0</m></pair>
                  <pair id="2"><ap>5</ap> <m>25</m></pair>
             </actions>
             </case>
          </listOfCase>)
 :in-reply-to r1)
```

Após incluir o caso adaptado em *P*, PA1 verifica se a aplicação do plano resulta na chegada ao destino. Em caso negativo, o ciclo de planejamento é repetido para a próxima parte do trecho ST_i. Caso positivo, *P* é enviado ao EA1.

```
(inform
       :sender PA1
       :receiver EA1
       :content(
          <?xml version="1.0"?>
          <Plan>
          <action>
             <pm>6775</pm> <km>339.495</km> <ap>1</ap> <IS>0</IS>
             <MS>45</MS> <NL>3</NL> <NW>58</NW> <RTotal>7816.568</RTotal>
             <Δt>6.08</ Δt> <Ft>67217.04</Ft> <Fac>59400.47</Fac>
          </action>
          <action>
             ...
          (Demais ações do plano da Figura 39.)
          </action>
          </Plan>)
       :in-reply-to r1)
```

EA1 embarca na locomotiva principal e passa a comandá-la. Se em seguida outro trem passar pela mesma estação, outra instância EA2 do Executor assume o plano para a nova embarcação.

Terminada a tarefa de EA1—conduzir o trem de uma estação s_i até a outra estação s_{i+1}—é executada a atividade *finalizar o plano*. Nela, o plano *P* e os seus ajustes Δ retornam à estação de origem s_i. EA1 retorna a sua estação de origem embarcado em outro trem que faz o trajeto no sentido oposto ao que EA1 fez.

```
(inform
   :sender EA1
   :receiver MA1
   :content(
        <?xml version="1.0"?>
        <Plan>
           <action>
              <pm>6775</pm> <km>339.495</km> <ap>1</ap> <IS>0</IS> <MS>45</MS>
              <NL>3</NL> <NW>58</NW> <RTotal>7816.568</RTotal> <Δt>6.08</ Δt>
              <Ft>67217.04</Ft> <Fac>59400.47</Fac>
           </action>
           <action>
              ...
           (Demais ações do plano da Figura 40.)
           </action>
           </Plan>)
:in-reply-to r1)
```

O transporte do plano modificado à origem por meio de outro trem é assumido porque, durante a viagem, não é usado nenhum canal de comunicação para transmitir os planos do trem para a estação. Caso o trecho possua apenas um sentido, o plano é armazenado na estação e encaminhado para a próxima estação que tiver conexão com à estação de origem do planejamento. A comunicação do trem com a estação limita-se ao envio de informações relativas às condições da via férrea e da posição do trem, devido ao custo de transmissão de dados e limitações de meios de comunicação face, muitas vezes, a condições geográficas.

O Memorizador MA1 ao receber o plano *P'* atualiza a base de casos local seguindo os passos descritos na seção 4.5.2.1.

4.7 COMPARTILHAMENTO DE PLANOS

O compartilhamento de planos busca dotar os resolvedores de problemas de conhecimentos iniciais. Em alguns casos, como por exemplo, na primeira missão em um determinado trecho, o sistema não possui nenhum plano do trecho em questão. Então pode-se usar planos de outros trechos que possuem a mesma estação como ponto de partida. Esses outros planos foram feitos por outros agentes. O compartilhamento dos planos de uma estação é assegurado pelo Memorizador. Ele possui os conhecimentos da estação de todos os planejadores residentes na mesma unidade computacional ou estação. Espera-se com a utilização de conhecimento de outros agentes, que os planos de condução gerados possam ser melhores do que planos formados sem considerar conhecimentos prévios.

4.8 GERAÇÃO DE PLANOS POR SIMULAÇÃO

Em situação de inexistência de experiência compartilhada pode-se gerar casos iniciais por

meio de algum outro método computacional, como por exemplo, desenvolvidos em (Dordal, et al., 2011b) (Sato, et al., 2012), ou até mesmo a inclusão de casos gerados em simuladores de treinamento de maquinistas. A inclusão de casos de outras fontes pode elevar o desempenho do *Planejador*, tendo em vista uma maior diversidade de experiências. Esta característica, inerente a abordagem baseado em casos, é difícil de reproduzir em um sistema baseado em regra, onde este último requer um especialista do domínio da aplicação para a inclusão de novas regras.

5 EXPERIMENTOS E RESULTADOS

O método foi avaliado por diferentes experimentos no contexto do domínio de condução de trens em ambiente de simulação empírico. É importante frisar que o objetivo não foi desenvolver um sistema de gerenciamento do modal férreo e nem mecanismos que aprimorem a capacidade dos sensores instalados nos trens, mas, usar este domínio para ilustrar as contribuições da aprendizagem em torno da abordagem de raciocínio baseado em casos na criação de um sistema agentes capaz de: (i) gerir uma base de casos — aprender com as experiências; (ii) gerar políticas de condução de trens, e (iii) aplicar tais políticas. É importante frisar também que o objetivo principal não é produzir novos algoritmos para o raciocínio baseado em casos, nem para os algoritmos genéticos, mas utilizar tais técnicas para otimizar o desempenho do sistema.

Os experimentos são modelados sob a forma de um sistema de condução, conforme descrito na seção 2, modelado matematicamente de acordo com as equações de (Loumiet, Jungbauer, & Abrams, 2005) (Chandra & Agarwal, 2007). O conjunto implementado de equações foi validado por meio de planilhas específicas, i.e., os resultados dos cálculos do sistema desenvolvido nesta pesquisa, utilizados para movimentar o trem, foram comparados com os cálculos elaborados e validados durante o projeto PAI-L, publicados em (Sato, et al., 2012). Ao longo desta pesquisa, os resultados foram publicados em vários artigos científicos (Borges, et al., 2011), (Borges A. P., et al., 2012) e (Borges A. P., et al., 2014).

Esta seção se divide na apresentação dos cenários de teste, da modelagem dos experimentos e finalmente da descrição e análise dos resultados obtidos.

5.1 VALIDAÇÃO DO MODELO MATEMÁTICO

A validação foi realizada comparando os dados de uma viagem simulada com dados reais obtidos de sensores com históricos de viagens de maquinistas; os dados eram registrados sem a intervenção humana. A comparação foi realizada após a implementação do modelo matemático. Foi confrontada a velocidade praticada na condução real (*velMaquinista*) com a velocidade praticada no simulador (*vel*), cf. Figura 43. Nestes experimentos foi usado um trem com 2 locomotivas, modelo multiponto, modelo C-30, com peso de 169.7t, compreendendo de 52 vagões de 61.88t cada, sendo o peso total da composição igual a 3557t em um trecho de aproximadamente 54 km. Por ter sido utilizado o modelo de condução multiponto, os pontos de aceleração de cada locomotiva (PA – Locomotiva 1 e PA – Locomotiva 2) são mostrados separados no gráfico. A validação foi realizada com trem

na configuração multiponto para uma melhor aproximação dos cálculos. Observa-se no gráfico que os pontos de aceleração empregados para gerar as potências das locomotivas 1 e 2 são coincidentes no gráfico. Desta forma, optou-se em fazer os experimentos usando apenas a abordagem monoponto, cuja complexidade computacional é significativamente menor que a abordagem multiponto.

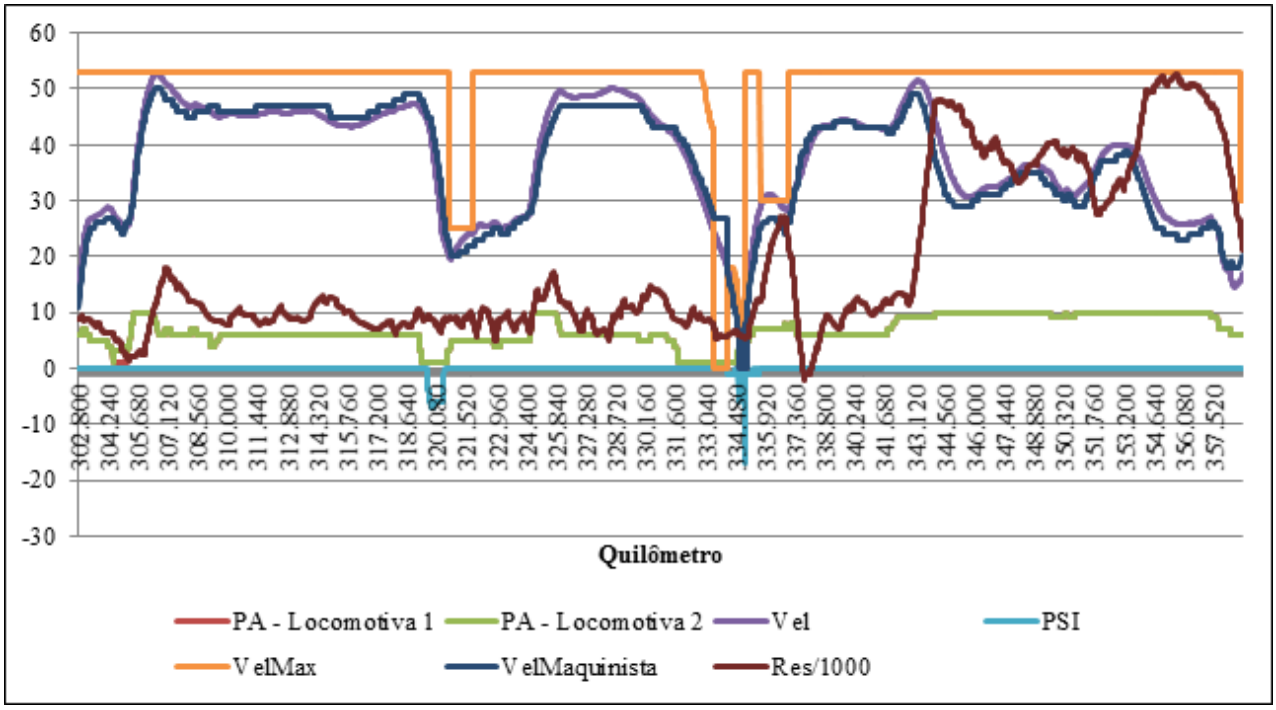


Figura 43 – Comparativo entre os dados reais e simulados.

Foi possível observar que, o uso dos mesmos pontos de aceleração utilizados pelo maquinista, resultou na mesma potência, e a velocidade simulada e real mantiveram-se muito próximas umas das outras. Esta equiparação das velocidades foi possível graças o modelo matemático utilizado para calcular os esforços e resistências de movimentação do trem que gerou resultados muito próximos àqueles praticados em viagens reais.

Valores ainda mais próximos aos reais poderiam ser obtidos durante a simulação se dados de alguns parâmetros utilizados tivessem mais precisão, tais como: o peso de cada vagão, o coeficiente de atrito, área frontal precisa de cada veículo, melhor precisão do trecho, valores da amperagem utilizados no cálculo da potência e comprimento do veículo. Para solucionar esta dificuldade, os valores padrão dos parâmetros foram baseados na literatura existente, principalmente em (Loumiet, Jungbauer, & Abrams, 2005) e (Chandra & Agarwal, 2007).

A seguir são os cenários de testes utilizados nos experimentos para validação da abordagem.

5.2 CENÁRIOS DE TESTE

Os experimentos foram conduzidos em laboratório, porém com características de trens e vias férreas de cenários reais. Foram utilizados dois trechos de vias ST_1 e ST_2 com características diferentes, principalmente no perfil vertical e nas restrições de velocidades máximas. Além disso, em ambos os cenários assume-se um coeficiente de atrito igual a 0.22, equivalente a um dia ensolarado (Loumiet, Jungbauer, & Abrams, 2005). A Figura 44 apresenta o perfil vertical, medido pelo percentual de rampa, dos trechos ST_1 e ST_2.

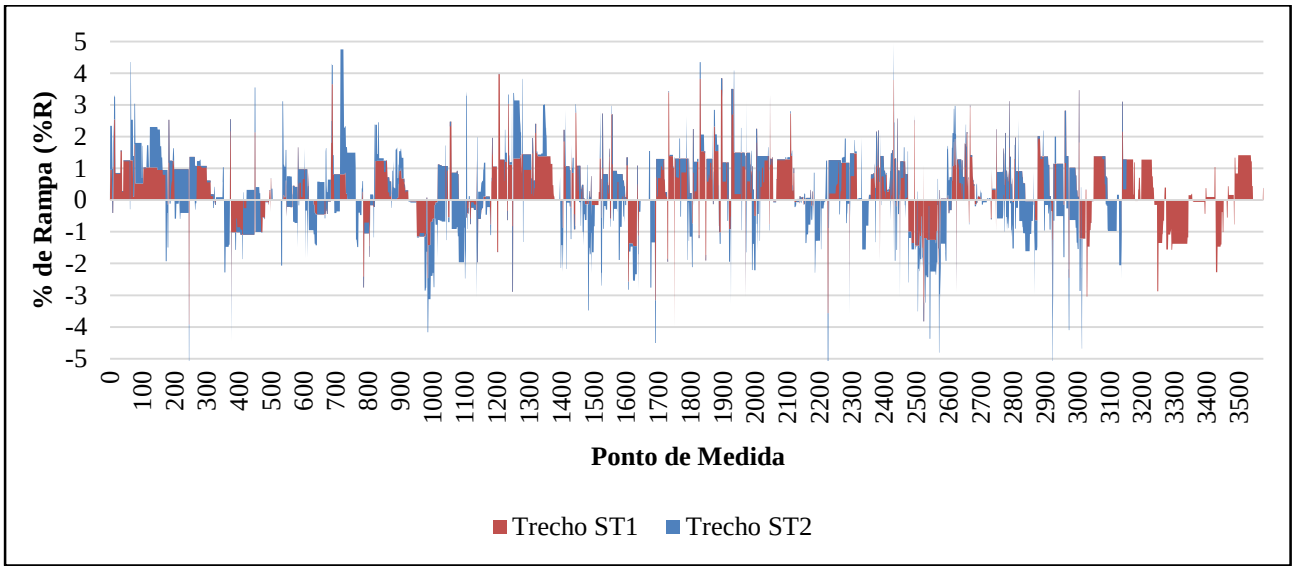


Figura 44 – Perfil vertical dos trechos dos experimentos.

A Figura 45 mostra que os perfis dos trechos são diferentes ao longo do conjunto de pontos de medida analisados. Esta diferença pode ser observada quanto (i) aos valores dos percentuais ou quanto aos (ii) tipos de percurso. Por exemplo, entre os pontos 124 e 171 apresentam valores positivos para o percentual de rampa e superiores à 0.6, o que indica que ambos os trechos estão em subida, mas com inclinações diferentes. Contudo, entre os pontos 172 e 178, o primeiro trecho continua em subida, enquanto o segundo trecho está em apresenta um nivelamento ($-0.6 \leq \%R \leq 0.6$) seguido de descida ($\%R \leq -0.6$).

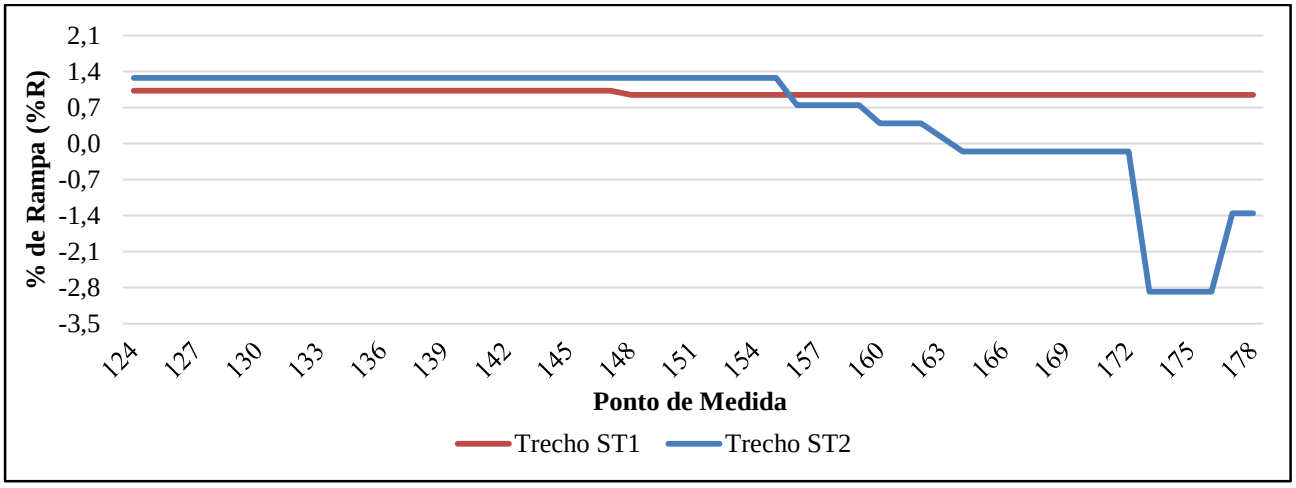


Figura 45 – Detalhamento do percentual de rampa entre o ponto de medida 124 e 178.

A abordagem metodológica desenvolvida levou em consideração, durante a recuperação dos casos, o critério (ii) — tipo de percurso — por este diferenciar claramente os perfis dos trechos. A diferença entre os perfis impacta na necessidade de políticas de ações distintas na condução: em uma subida é necessário a geração de alta potência motor para superar a resistência total; em descida a potência gerada pode ser a mais baixa possível, pois a resistência total é baixa e é necessário um menor esforço trator para vencer o percurso.

Quando observado um trecho qualquer, é possível notar que duas viagens podem ser diferentes devido às restrições de velocidades máximas impostas em momentos distintos. Esta é uma tarefa do agente responsável pelo gerenciamento da via férrea, o qual estabelece limites de velocidades variados dependendo de situações como manutenção e acidentes. Cabe ressaltar que a determinação destas velocidades não foi o objetivo desse trabalho. As velocidades máximas praticadas em cada experimento variaram de acordo com as velocidades máximas das viagens reais usadas como modelos (cf. Figura 46 e Figura 47).

As inúmeras formações de locomotivas e vagões vazios e/ou carregados também tornam a condução de trens uma tarefa única em cada viagem. Tais diferenças ocorrem devido ao peso total do trem, o qual tem influência direta na resistência total. Durante a elaboração do plano de condução, os planejadores têm conhecimentos do peso total do trem, logo, eles elaboram planos distintos de acordo com o peso. Nestes experimentos, foram utilizados trens com as mesmas características daqueles que executaram as viagens reais e também daqueles estudados por (Sato, et al., 2012). A Tabela 25 apresenta as configurações dos trens utilizados nos experimentos.

Tabela 25 – Configuração dos trens usados nos experimentos.

CONFIGURAÇÃO DO TREM	QUANTIDADE		PESO TOTAL (T)
	Locomotivas	Vagões	
1	3	58	6278
2	4	100	6342
3	4	58	6541
4	2	31	3426
5	3	47	5199
6	2	31	3441
7	4	59	6579
8	2	28	3118

Para esclarecer as diferenças na condução, a Figura 46 e a Figura 47 ilustram o resultado de duas viagens executadas por agentes no trecho ST_1. A Figura 46 mostra as ações executadas pelo agente durante a condução do trem com a configuração número 4 da Tabela 25, enquanto a Figura 47 ilustra as ações executadas pelo agente durante a condução do trem com a configuração número 5 da Tabela 25.

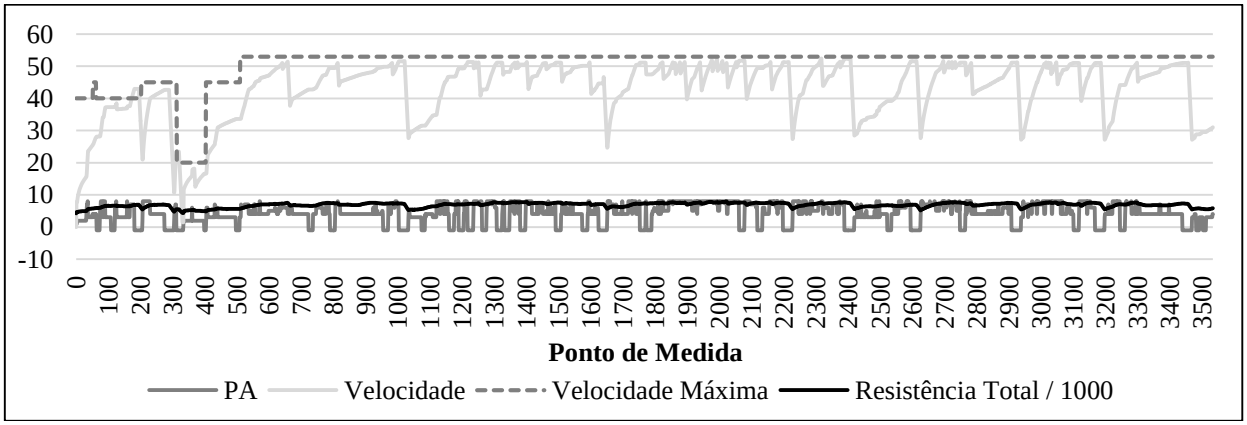


Figura 46 – Viagem realizada com a configuração 4.

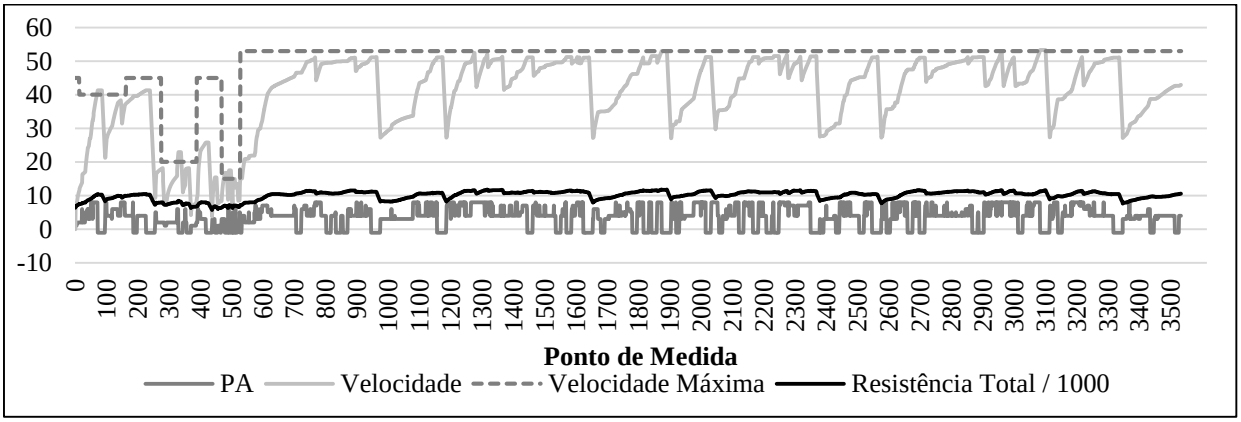


Figura 47 – Viagem realizada com a configuração 5.

Ao comparar o resultado das duas viagens, é possível observar as diferenças nas velocidades máximas permitidas, as quais são diferentes em determinados locais até o ponto de medida 600. As diferenças nas velocidades impedem o agente condutor de andar próximo à velocidade cruzeiro, estabelecida 5 km/h abaixo da velocidade máxima. As diferenças nas características do trem podem ser observadas pelo valor da resistência total, influenciada pelo peso do trem. Na Figura 46 os valores das resistências são inferiores àqueles da Figura 47, devido ao peso do trem ser menor.

As diferenças de velocidades e características dos trens são observadas nos pontos de aceleração utilizados pelos agentes ao longo das viagens, responsáveis por gerar a potência necessária para rebocar o trem. Na configuração 4, as ações foram mais uniformes, com menos alterações nos pontos de aceleração quando comparadas às ações da configuração 5. Quando somadas as trocas de ações ao longo da viagem, a configuração 5 apresentou um número 10% superior à configuração 4 (552 vs. 502). Para um trecho total de aproximadamente 70km, isto representa uma troca a cada 130m aproximadamente. Estes números ocorrem pelo fato do agente visar a velocidade máxima permitida e também o menor consumo de combustível possível, de modo a otimizar ao máximo a condução.

Em determinados momentos, o trem ultrapassou a velocidade máxima permitida em poucos quilômetros por hora, o que implica na ação imediata de frear o trem. Isto ocorreu uma vez na viagem da Figura 46 e na viagem da Figura 47 devido à política de ação do agente ter por objetivo andar sempre muito próximo da velocidade máxima. Nos experimentos, a percepção que o agente possui da velocidade máxima da via férrea afrente da sua posição foi configurada para 5km. Isto significa dizer que o agente, estando no ponto de medida 0, conhece a velocidade máxima da via até o ponto de medida 250. Isto fez com o que o agente "percebesse" que deveria frear o trem um pouco tarde, logo, não conseguiu frear a tempo. O tempo de frenagem (tf) é o tempo necessário para que a pressão do ar diminua em todo o trem. Ele varia de acordo com o tamanho do trem conforme Equação (8) (Loumiet, Jungbauer, & Abrams, 2005).

$$tf = \frac{comprimentoDoTrem \times 1103{,}78}{47{,}17} \tag{8}$$

Por outro lado, ao perceber que ultrapassou a velocidade máxima permitida, o agente tomou as medidas necessárias para reduzir a velocidade, freando o trem e não permitindo que a velocidade se excedesse 3 km/h acima da máxima permitida. Um segundo item a ser esclarecido acerca dos gráficos apresentados na Figura 46 e Figura 47 é sobre a queda brusca da velocidade em determinados locais. Isto ocorreu devido à força da primeira aplicação de freios, convencionada em 6 *psi* por

(Loumiet, Jungbauer, & Abrams, 2005). Uma alternativa para reduzir este problema é a utilização dos freios dinâmicos durante a condução; isto não foi testado neste trabalho, uma vez que os dados reais obtidos não possuem tais informações para comparação.

A seguir são discutidos os resultados obtidos de acordo com as hipóteses levantadas na primeira seção.

5.3 RESULTADOS OBTIDOS

Para avaliar a curva de aprendizagem de cada agente e o desempenho da colaboração em termos de compartilhamento e reuso de planos, quatro cenários foram definidos (cf. Tabela 26).

Tabela 26 – Cenários simulados nos experimentos.

CENÁRIO	CONFIGURAÇÃO DO TREM (CF. TABELA 25)	REUSO DE PLANOS	TRECHO
A	1	No	ST_1
B	1	Yes	ST_1
C	1	Yes	ST_2
D	[1;8]	Yes	ST_1

Os resultados obtidos em cada um dos cenários são descritos nas seções a seguir. A base de casos inicial do *Memorizador*, em todos os cenários testados, é composta de casos obtidos de dados de viagens executados em simuladores desenvolvidos pelo projeto PAI-L e descritos em (Sato, et al., 2012).

Além disso, em todos os cenários testados a estratégia de adaptação do *Planejador* baseou-se em algoritmo genético. A população inicial do algoritmo consistiu de 50 indivíduos, valor atribuído por representar de 5% a 15% do número de casos armazenados na base de casos inicial. Bons resultados foram alcançados com a combinação do algoritmo genético e raciocínio baseado em casos quando utilizados 10% da base de casos formando a população inicial em (Passone, Chung, & Nassehi, 2006). Contudo, em nossa pesquisa o tamanho da base de casos *Memorizador* apresenta variações nos diferentes cenários e como o objetivo principal não era encontrar um valor ideal para o tamanho da população inicial optou-se por utilizar um valor fixo para todos os cenários que fosse próximo ao utilizado por (Passone, Chung, & Nassehi, 2006).

O valor da taxa de mutação utilizada foi igual a 50%, um valor considerado alto na literatura, porém que apresentou bons resultados. Foi utilizado o método de mutação simples, ou canônico. O

método de cruzamento utilizado foi de um ponto devido a sua simplicidade e por ter obtido os melhores resultados quando comparado às demais técnicas de cruzamento em um problema semelhante (Mendes, 2013) (Picek, Golub, & Jakobovic, 2012).

Quanto ao critério de parada, foi utilizado o número de gerações, 10, por apresentar resultados satisfatórios em termos de economia de combustível e uma maior rapidez na adaptação dos casos. Um número maior de gerações foi experimentado, conforme Apêndice A, porém sem apresentar diferenças significativas. Logo, optou-se por este valor dada a maior rapidez na adaptação dos casos. A rapidez da adaptação é influenciada pela simulação que ocorre durante o cálculo de *fitness* de cada indivíduo da população. Um número de gerações elevado requer, consequentemente, um número elevado de simulações, o que retarda o processo de adaptação.

O apêndice A apresenta uma tabela comparativa entre os resultados da adaptação dos casos utilizando algoritmo genético com diferentes configurações. O objetivo foi analisar quais aprimoramentos nas configurações do algoritmo poderiam ser realizados para que fosse aprimorada ainda mais a etapa de adaptação dos casos. De modo geral, este trabalho não teve como objetivo encontrar os parâmetros ideais do algoritmo genético, mas mostrar que este método pode ser efetivo quando utilizado na adaptação de casos em cenários reais, auxiliando também a otimizar os resultados do raciocinador.

Os cenários foram avaliados de acordo com o percentual de eficiência (%) das etapas de recuperação e adaptação de casos. Tais percentuais indicam as taxas de sucesso da etapa de recuperação e adaptação dos casos, i.e., indicam a eficiência do *Memorizador* e do *Planejador*, respectivamente. Para ilustrar o cálculo do percentual de eficiência foi utilizado um exemplo simples: em determinado momento, o *Memorizador* recuperou, para o *Planejador*, um conjunto de casos, sendo o mais similar igual a $A = \{\langle 3, 0\rangle, \langle 4, 25\rangle, \langle 4, 75\rangle, \langle 4, 110\rangle, \langle 4, 125\rangle\}$. Este conjunto foi adaptado, pelo *Planejador* para $A' = \{\langle 4, 0\rangle, \langle 4, 25\rangle, \langle 4, 75\rangle, \langle 4, 110\rangle, \langle 4, 125\rangle\}$. Logo, a eficiência do *Memorizador* foi igual a 80%, visto que dentre 5 ações apenas 1 foi modificada. Continuando, o conjunto A é repassado ao *Executor* como parte de um plano. O *Executor* o aplica sem modificar os pontos de aceleração indicados, logo, a eficiência do *Planejador* em elaborar um plano aplicável foi igual a 100%, dado que o caso A' foi aplicado em sua totalidade.

A seguir são discutidos os resultados obtidos nos cenários avaliados.

5.3.1 *Cenário A*

Neste primeiro cenário, o *Memorizador* utiliza apenas os planos iniciais, i.e., a cada nova

viagem executada pelo *Executor*, os planos não são incorporados à base de casos do *Memorizador*. Este cenário tem por objetivo servir como parâmetro inicial para medir a diversidade da base de casos do *Memorizador*, quando comparado com os demais cenários de experimentação. Um exemplo é mostrado na Tabela 27, onde é possível comparar o tamanho da base de casos ao longo das 10 execuções e verificar que o número total de casos presente em cada uma das bases de casos do *Memorizador* se manteve constante.

Tabela 27 – Tamanhos das bases de casos no cenário A.

		EXECUÇÃO									
		1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
TAMANHO	Acelerar	602	602	602	602	602	602	602	602	602	602
	Frear	104	104	104	104	104	104	104	104	104	104
	Manter	1992	1992	1992	1992	1992	1992	1992	1992	1992	1992
	Reduzir	631	631	631	631	631	631	631	631	631	631
	Total	**3329**	**3329**	**3329**	**3329**	**3329**	**3329**	**3329**	**3329**	**3329**	**3329**

A Figura 48 apresenta a eficiência da etapa de recuperação e adaptação de casos ao longo das 10 viagens. Pode-se observar que a etapa de recuperação é menos eficiente que a etapa de adaptação, o que mostra a contribuição da etapa de adaptação (executada pelo *Planejador*) para tornar um plano de condução aplicável. A média desta diferença foi de 8%, com um desvio padrão de 1%. Além disso, os melhores resultados em termos de recuperação e adaptação ficaram em 46% e 51%, respectivamente.

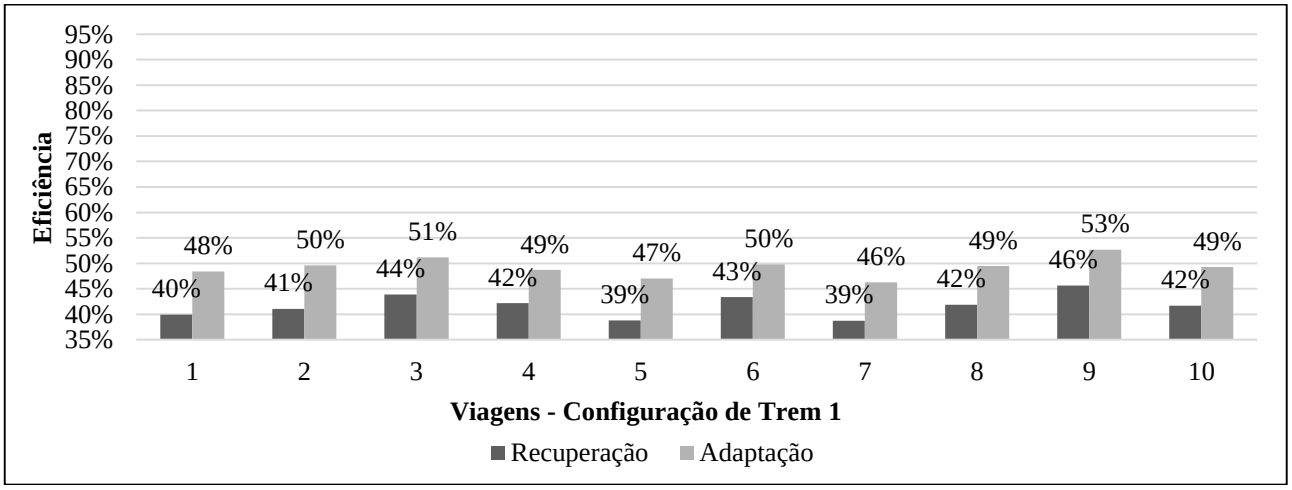


Figura 48 – Resultados do Cenário A: 10 trens com a mesma configuração de viagem (Configuração 1), no trecho ST_1 e sem reuso de planos.

Quando comparado com o método de satisfação de restrições, o tempo médio de ciclo de execução do raciocínio baseado em casos para elaborar cada plano foi, em média de 2s, comparados

aos 4s da abordagem DCOP proposta por (Leite, Giacomet, & Enembreck, 2009). Além disto é comparada a similaridade da condução do *Executor* frente ao planejado neste mesmo cenário ao longo das 10 viagens. A similaridade é calculada comparando a ação executada pelo *Executor* em determinada posição com a ação planejada pelo *Planejador* nesta mesma posição do percurso. A ação referida é o ponto de aceleração empregado, o qual gera a potência para rebocar o trem. Por exemplo, se no ponto de medida 5000, o *Executor* aplicou o ponto de aceleração 5 e o *Planejador* também indicou o ponto de aceleração 5, a ação neste ponto foi 100% similar. Em termos matemáticos, tal métrica foi operacionalizada por meio do *cálculo do cosseno,* dada pela Equação (9), onde o vetor $\vec{u}$ representa os pontos de aceleração usados pelo *Executor* e o vetor e $\vec{v}$ representa os pontos de aceleração sugeridos pelo *Planejador*. O número total de ações é igual a *n*. Os valores resultantes da equação variam entre zero e um, sendo que quanto mais próximo de um, mais similares são os vetores e consequentemente melhor o conhecimento obtido.

$$\cos\theta = \cos(\vec{u},\vec{v}) = \frac{\vec{u}\times\vec{v}}{\sqrt{\sum u_n^2 \times \sum v_n^2}} \tag{9}$$

No Cenário A, a similaridade da condução do *Executor* frente ao planejado foi, em média, de 86% (cf. Figura 49), com desvio padrão de 1%. Tal similaridade ajuda a mostrar a eficiência do planejamento das ações por parte do *Planejador*.

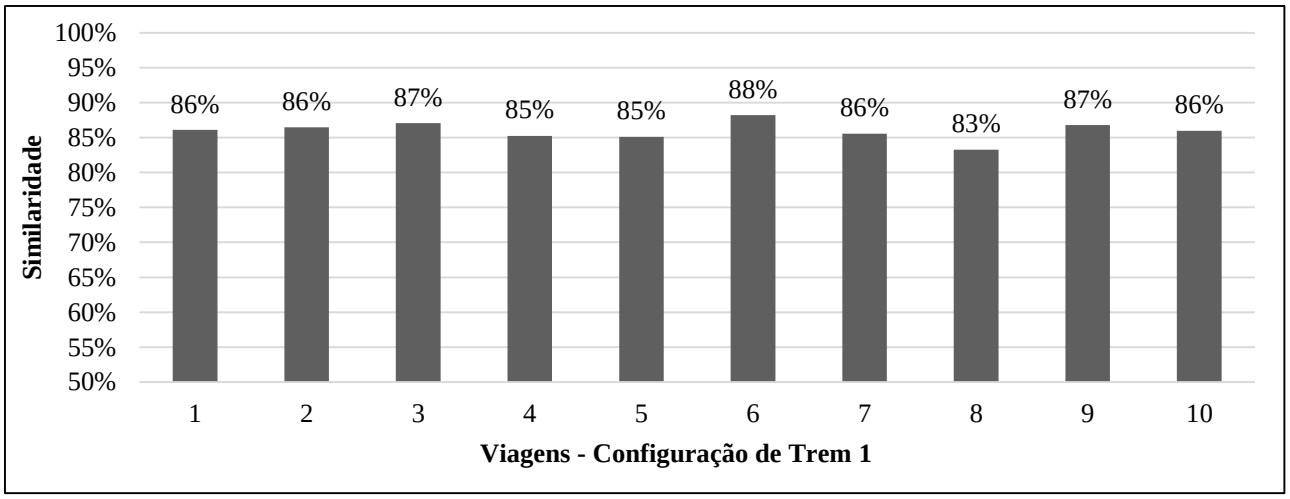


Figura 49 – Similaridade da condução do Executor, no Cenário A, comparado ao plano elaborado.

A terceira análise concerne os consumos de combustível obtidos nos experimentos. Observou-se que, ao longo das viagens executadas, o consumo médio foi de 3.23 litros por tonelada bruta transportada (LTKB), cf. gráfico da Figura 50, valor bem abaixo do consumo real, que foi de 6.19.

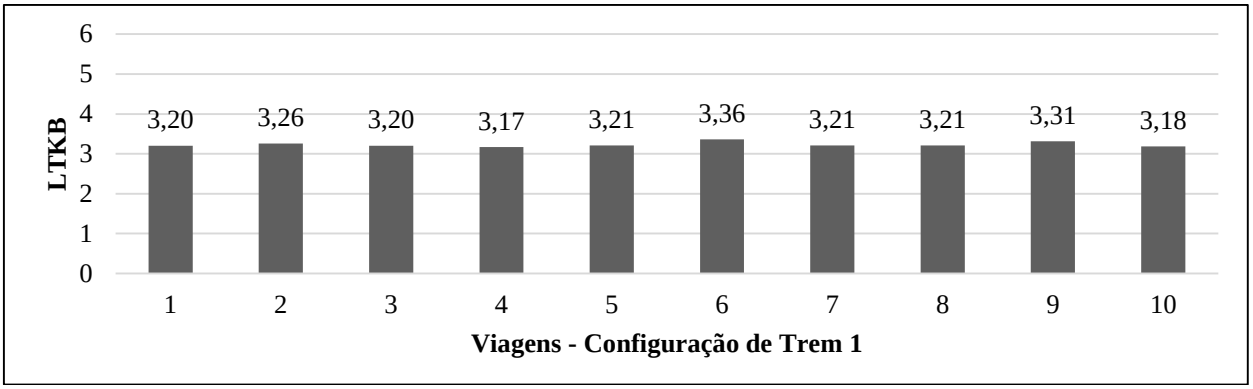


Figura 50 – Consumos obtidos no cenário A.

Finalmente, uma análise é realizada quanto aos tempos de viagens obtidos neste cenário, os quais foram, em média de 118 minutos, novamente abaixo dos 217 minutos da viagem real executada pelo maquinista.

5.3.2 *Cenário B*

Neste cenário foi utilizada a mesma configuração de trem do cenário A, um trem com 3 locomotivas, 58 vagões e aproximadamente 6278t. Contudo, a cada nova viagem realizada pelo *Executor*, o plano aplicado é retornado ao *Memorizador* da estação de origem. Este retorno faz com que os novos planos gerados sejam incorporados à base de casos do *Memorizador* (cf. Tabela 28), onde, a cada nova execução, o tamanho total da base de casos aumentou.

Tabela 28 – Tamanhos das bases de casos no cenário B.

		EXECUÇÃO									
		1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
TAMANHO	Acelerar	602	740	883	991	1107	1247	1393	1507	1654	1762
	Frear	104	155	222	309	376	473	545	650	698	794
	Manter	1992	2283	2590	2864	3193	3462	3713	3925	4189	4390
	Reduzir	631	655	676	705	731	753	774	793	826	856
	Total	**3329**	**3833**	**4371**	**4869**	**5407**	**5935**	**6425**	**6875**	**7367**	**7802**

O gráfico da Figura 51 mostra os resultados do Cenário B. Observa-se que a inclusão de novos planos na base de casos do *Memorizador* melhorou tanto a tarefa de recuperação como a tarefa de adaptação dos casos em aproximadamente 25%. Isto mostra que o tamanho da base de casos impacta na solução final do problema. A tarefa de recuperação teve uma eficiência média de 67%, com desvio padrão de 2% e a tarefa de adaptação uma eficiência média de 74%, também com desvio padrão de 2%. Quando comparadas as tarefas de recuperação e adaptação, o desvio padrão foi de 1%. É possível

observar que entre as viagens 1 e 3, há um incremento linear de 20% na eficiência do *Memorizador* e do *Planejador*. Ademais, a partir da viagem 3, há uma estabilidade ligeiramente crescente, com variação de 4% para ambas as tarefas e desvio padrão de 2%. Esta variação é justificada pois as viagens executaram velocidades similares umas às outras, mas com diferenças no plano de condução em determinados momentos. Esta diferença ocorre devido as adaptações feitas pelo algoritmo genético, que em diferentes locais sugeriu pontos de aceleração diferentes, na tentativa de otimizar as ações tomadas. Este fato é inerente ao comportamento natural do algoritmo genético, como por exemplo, decorrente das mutações e cruzamentos executados.

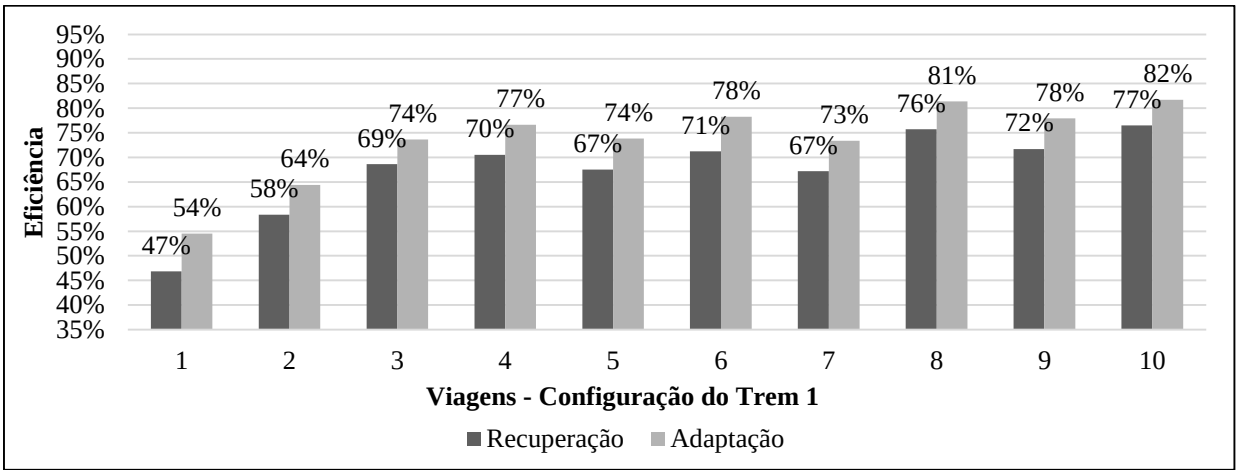


Figura 51 – Resultados do cenário B: 10 viagens com a mesma configuração de trens (Configuração 1), no trecho ST_1 e com reuso de planos.

A similaridade entre o que foi planejado e o que foi executado foi, em média, de 90% (cf. Figura 52). A adição de novos casos na base de experiências fez com que a tarefa de planejamento fosse aprimorada, mostrando assim o reaproveitamento das ações passadas na elaboração de novos planos de condução.

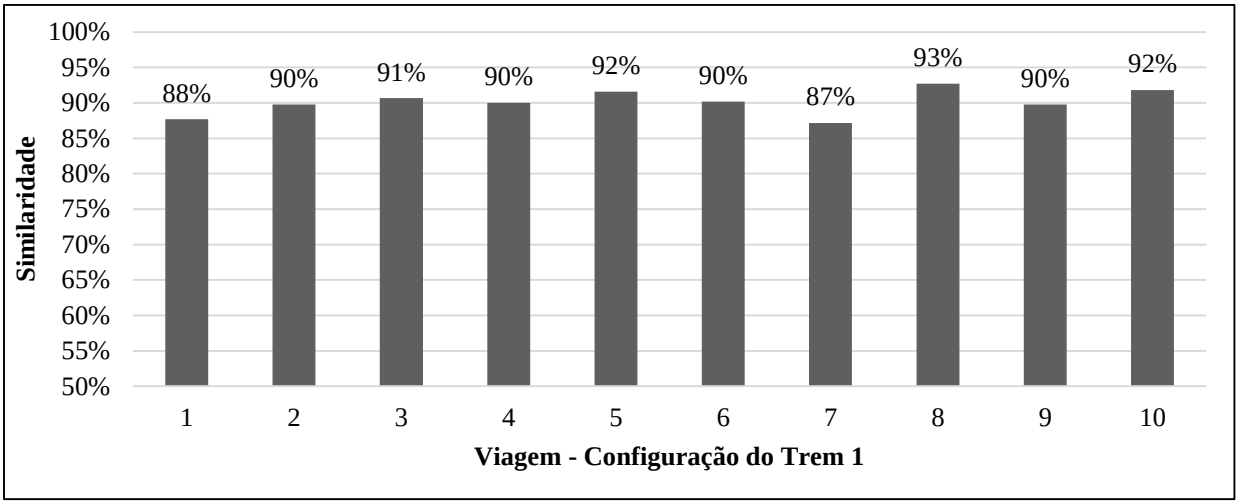


Figura 52 – Similaridades do Cenário B.

A análise dos consumos de combustível resultantes destas viagens mostrou que os valores finais foram muito semelhantes aos obtidos no Cenário A (cf. Figura 53).

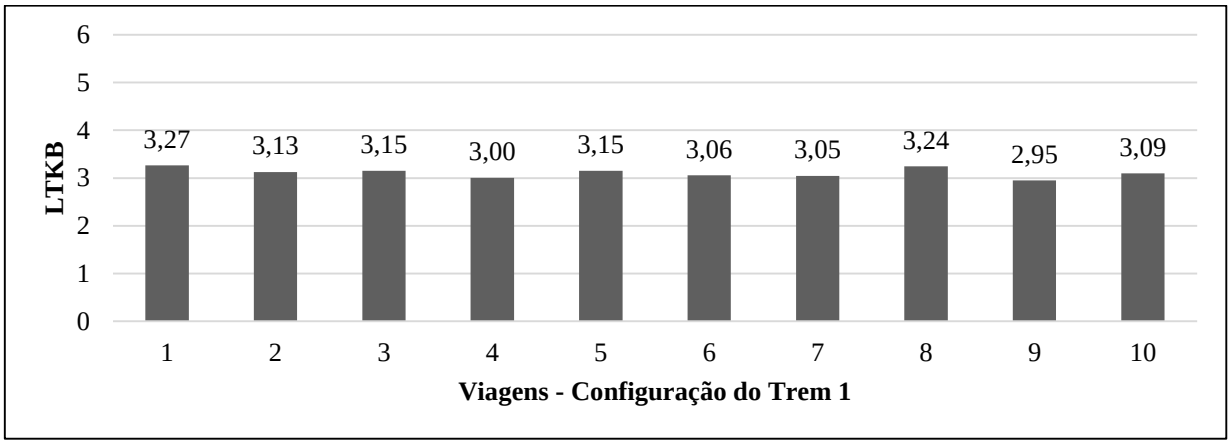


Figura 53 – Consumos obtidos no Cenário B.

Contudo, o reaproveitamento de casos otimizou também o consumo final de combustível, uma vez que ele foi inferior ao obtido no Cenário A (cf. Tabela 29). Isto mostra que o consumo médio das viagens do Cenário B foi 0,121 LTKB, inferior ao consumo médio das viagens realizadas no Cenário A.

Tabela 29 – Diferença entre o consumo do Cenário A e do Cenário B.

VIAGEM	A	B	DIFERENÇA (B-A)
1	3,200	3,266	0,067
2	3,260	3,128	-0,132
3	3,201	3,152	-0,049
4	3,165	3,003	-0,162
5	3,209	3,151	-0,058
6	3,364	3,060	-0,303
7	3,207	3,047	-0,160
8	3,208	3,245	0,037
9	3,311	2,953	-0,358
10	3,183	3,095	-0,088
Média	**3,321**	**3,110**	**-0,121**

O tempo médios das viagens também foi semelhante à abordagem do Cenário A, 117 minutos. Tal resultado mostra que a viagem foi executada no mesmo tempo, mas as ações foram melhores, visto que o consumo de combustível foi reduzido em média 6%.

5.3.3 *Cenário C*

Neste cenário é realizada a avaliação do compartilhamento/reuso de planos de trechos diferentes. Para analisar este critério, a mesma configuração de trem utilizada no cenário anterior foi utilizada, mas, os planos resultantes do Cenário B foram utilizou para conduzir o trem em um trecho diferente, o trecho ST_2, cf. indicado por B10 na Figura 54.

Os resultados da eficiência do *Memorizador* e *Planejador* são mostrados na Figura 54. Neste cenário, a base de casos inicial do *Memorizador* é formada com a base de casos da última viagem realizada no Cenário B, de modo a caracterizar a transferência de conhecimentos. Pôde-se observar que o compartilhamento das experiências entre agentes que atuam em um ambiente com determinadas características pode ser útil na elaboração e execução de planos em um cenário com características diferentes.

As etapas de recuperação e adaptação obtiveram uma eficiência média de e 80% e 86%, respectivamente. Em termos percentuais, a diferença entre a etapa de recuperação e adaptação foi de 9% no início do experimento. Após 2 execuções esta média caiu para 5% em seguida, com desvio padrão de 1%. Apesar dos esforços iniciais devido à não familiaridade dos agentes com o ambiente, os resultados são significantes. Tais resultados mostram a eficiência da abordagem colaborativa entre os agentes, localizados em diferentes estações, para troca de planos.

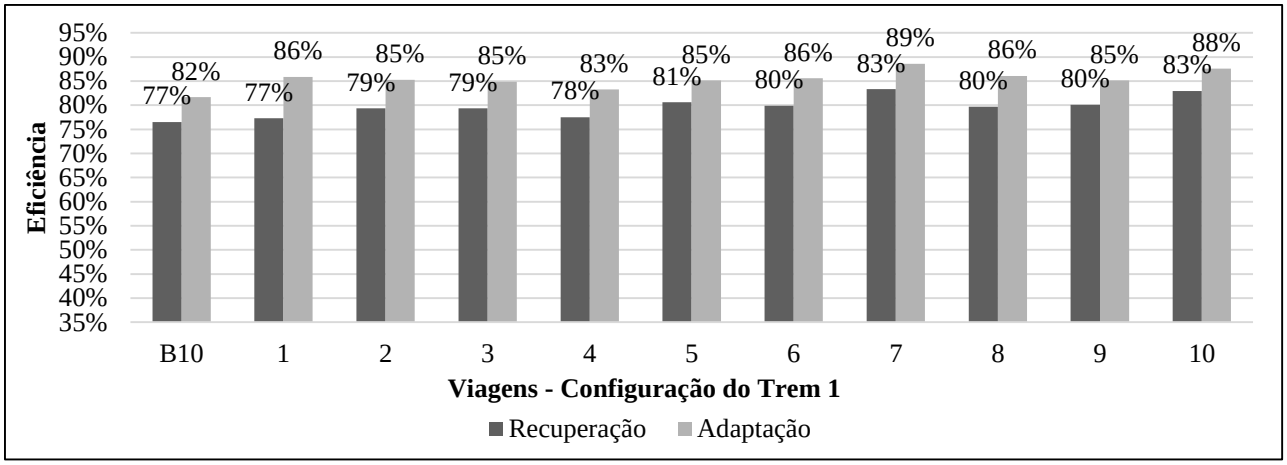


Figura 54 – Resultados do cenário C: 10 viagens com a mesma configuração de trens (Configuração 1), no trecho ST_2 e com reuso de planos.

O cenário foi avaliado também em termos da similaridade da condução planejada vis-à-vis a condução executada. A similaridade média, (cf. Figura 55) foi de 92% o que mostra que o método proposto pode ser eficiente em cenários desconhecidos. Este percentual também mostra que o compartilhamento de planos também pode ser eficaz em um cenário onde os agentes não possuem experiências anteriores.

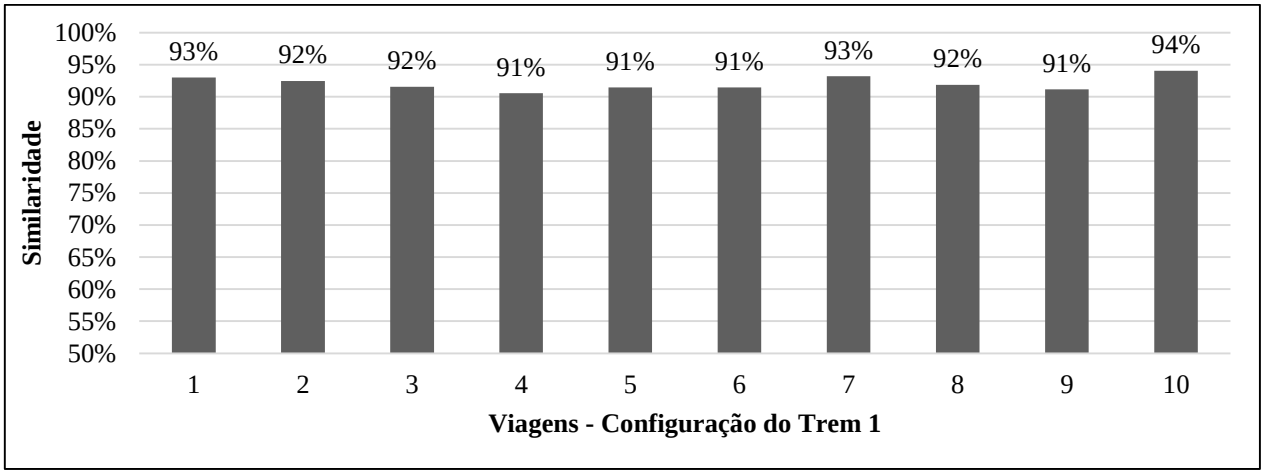


Figura 55 – Similaridades obtidas no Cenário C.

Um comparativo pode ser feito graficamente para mostrar as diferenças entre os trechos ST_1 e ST_2 utilizados nos cenários B e C, respectivamente. A Figura 56 e a Figura 57 mostram as principais informações resultantes da décima viagem executada no cenários B e C, respectivamente.

Algumas considerações importantes acerca de tais informações:

- O valor padrão da pressão do freio automático, em ambas as viagens, foi estabelecido

em 90psi. Logo, sempre que há uma aplicação de freio este valor é reduzido. No Cenário B foram ao todo 8 aplicações de freios ao longo da viagem, enquanto que no Cenário C foram 11 aplicações. Esta diferença ocorreu dada a política de ações utilizada, a qual estabeleceu que a velocidade cruzeiro fosse próxima da velocidade máxima permitida, de modo a otimizar também o tempo de viagem;

- As quedas bruscas nas velocidades são decorrentes da força da primeira aplicação da frenagem automática, estabelecida em 6psi (Loumiet, Jungbauer, & Abrams, 2005). A inserção da frenagem dinâmica deve frear mais suaves e assim reduzir estas quedas de velocidade; isto não foi feito neste trabalho.
- A diferença entre os perfis verticais dos dois trechos pode ser observada pela informação do percentual de rampa. Observa-se que o primeiro trecho é mais sinuoso, com maior número de variações entre aclives, declives e em níveis. Esta mesma informação pode ser visualizada no gráfico da Figura 44.
- As variações nos perfis dos trechos refletem na resistência total enfrentada pelo trem ao longo do percurso, onde o Cenário B apresentou uma resistência total mais uniforme do que o Cenário C.

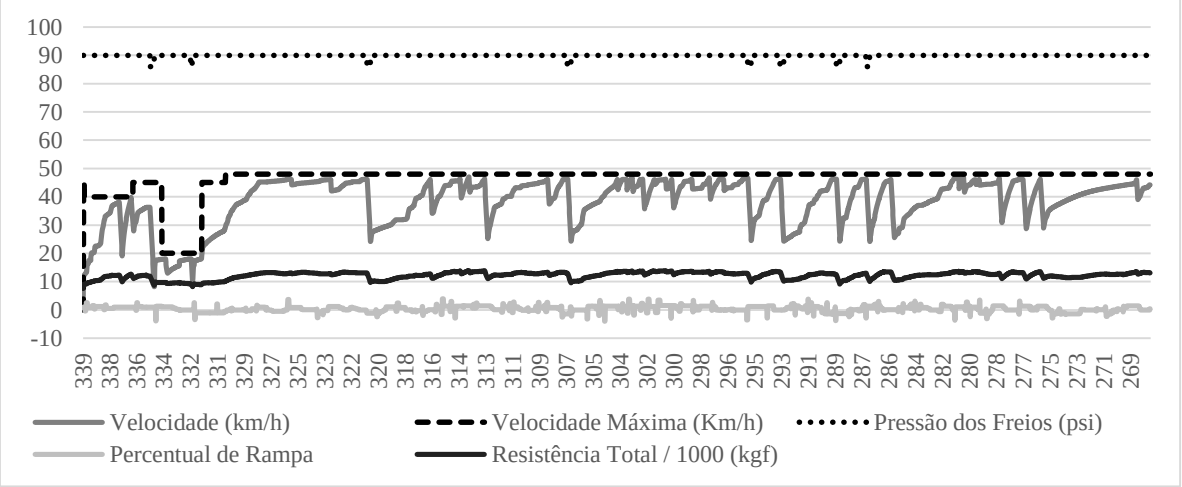


Figura 56 – Dados da 10ª viagem executada no Cenário B.

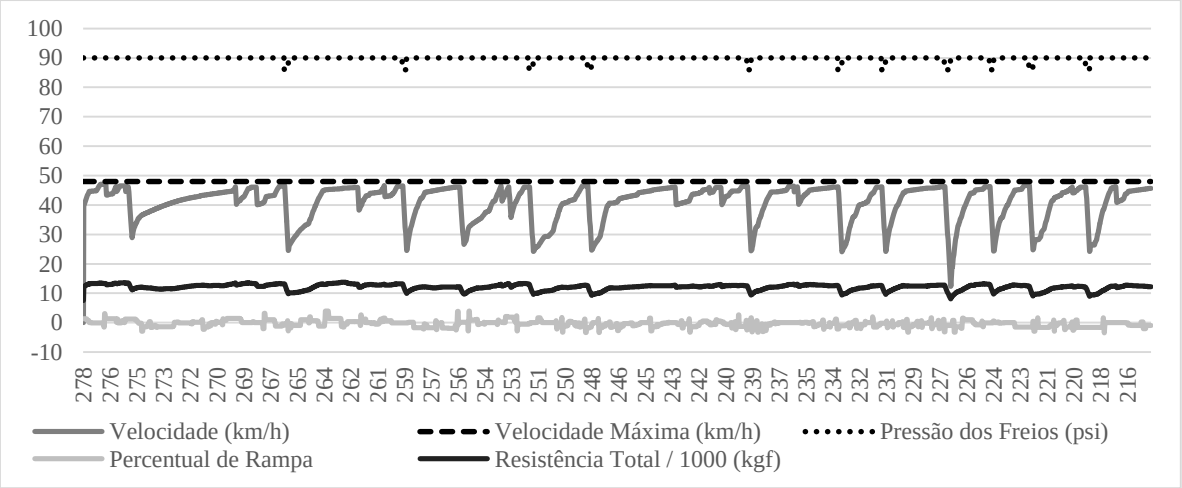


Figura 57 – Dados da 10ª viagem executada no Cenário C.

A seguir discutimos o último cenário de testes onde foram utilizadas diferentes configurações de trens percorrendo o trecho ST_1.

5.3.4 *Cenário D*

O último cenário de testes visa analisar o impacto do reaproveitamento de planos em cenários diferentes daqueles planejados. No cenário anterior, o trecho de planejamento e execução foi alterado, com o mesmo objetivo. Neste cenário, entretanto, a alteração concerne ao perfil dos trens que realizarão as viagens. A Tabela 30 apresenta a quantidade de casos existente na base de casos do *Memorizador,* à medida que novas viagens são executas. Nota-se que a base inicial (utilizada na Configuração 1) apresenta uma quantidade de casos alta, resultantes da execução número 10 do Cenário B.

Tabela 30 – Tamanho das bases de caso no cenário D.

		CONFIGURAÇÃO DO TREM							
		1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
TAMANHO	Acelerar	1970	2046	2180	2312	2452	2588	2763	2809
	Frear	895	1173	1282	1331	1474	1588	1539	2255
	Manter	4804	5001	5262	5542	5774	5991	6349	6192
	Reduzir	922	901	930	965	998	1004	1050	942
	Total	**8591**	**9121**	**9654**	**10150**	**10698**	**11171**	**11701**	**12198**

O gráfico da Figura 58 mostra os resultados do Cenário D, onde as viagens foram realizadas sempre no mesmo trecho (ST_1), mas com 8 diferentes formações de trens (cf. Tabela 25). Neste cenário, o *Memorizador* também é iniciado com as experiências obtidas ao término da simulação do Cenário B.

Em termos de eficiência das tarefas de recuperação e adaptação, a adaptação mostrou-se superior. Esta diferença é esperada visto que as configurações de trens utilizadas nas viagens são diferentes umas das outras. Contudo, mesmo no cenário onde não ouve repetição na configuração do trem é possível observar que a diversidade da base de casos foi importante para a eficiência do processo. Esta importância foi observada pois, à medida que novos casos foram sendo incluídos na base de casos do *Memorizador*, o processo como um todo obteve melhores resultados. Espera-se que, com um número maior de viagens com configurações similares, a taxa de eficiência melhore rapidamente, conforme ocorreu no Cenário B; essa verificação não foi feita.

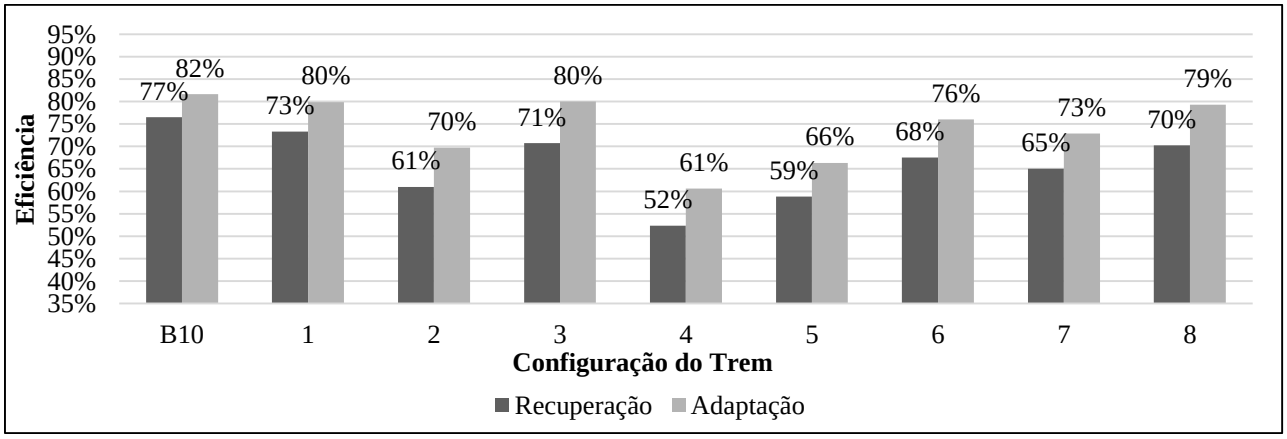


Figura 58 – Resultados do cenário D: diferentes configurações de trens (configurações 1 a 8), no trecho ST_1, planos iniciais do Cenário B e reuso de planos.

Neste cenário, a eficiência média da tarefa de recuperação foi de 65% e a eficiência da tarefa de adaptação foi 73%, ambos com desvio padrão de 7%. É importante ressaltar que a ordem das

viagens executadas teve influência nos resultados das eficiências medidas, visto que a tarefa de planejamento é influenciada diretamente pelos planos na base do *Memorizador*. No que diz respeito às similares da condução, o resultado foi satisfatório. Ao longo das 8 viagens simuladas a similaridade média foi de 89% com desvio padrão de 2%. Este resultado mostra que a abordagem foi eficiente mesmo em cenários com configurações adversas no que diz respeito às configurações de trens.

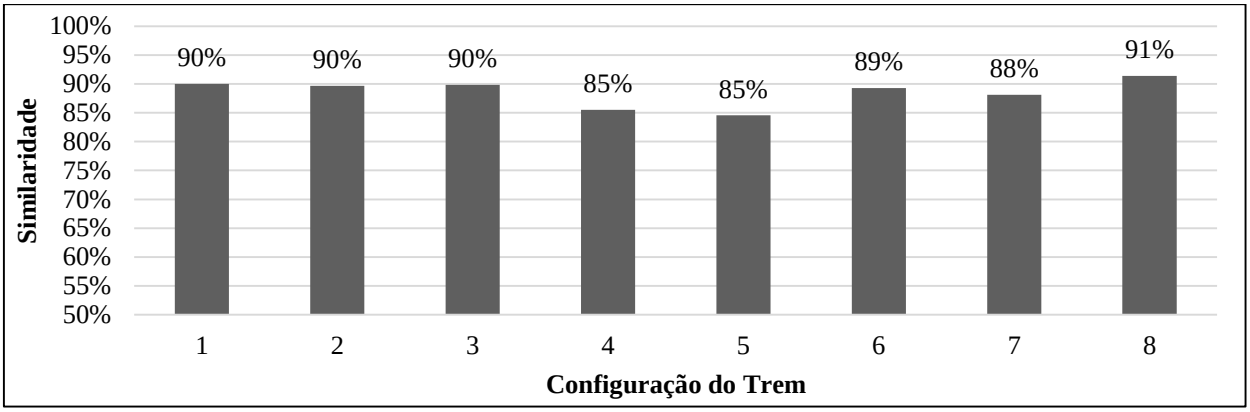


Figura 59 – Similaridades obtidas no Cenário D.

Finalmente uma análise é realizada em termos de consumo de combustível em termos de LTKB (Tabela 31). Ao longo das 8 viagens realizadas, o consumo médio da abordagem desenvolvida nesta pesquisa (CBR) foi de 3.61 LTKB, enquanto o consumo médio da abordagem DCOP desenvolvida por (Leite, Giacomet, & Enembreck, 2009) foi de 4.2 LTKB. Pôde-se observar que a abordagem proposta apresentou um consumo em média 14% menor que a abordagem de condução por satisfação de restrições DCOP. Isto ocorreu graças ao reaproveitamento das ações passadas, que gerou, ao longo da condução, um melhor aproveitamento das potências empregadas.

Tabela 31 – Consumos obtidos no Cenário D.

CONFIGURAÇÃO DO TREM	LTKB			REDUÇÃO	
	Real	DCOP	CBR	(CBR – Real)%	(CBR – DCOP)%
1	6,19	4,16	3,09	-50%	-26%
2	5,68	4,18	3,99	-30%	-5%
3	6,23	4,09	3,67	-41%	-10%
4	6,49	4,51	3,49	-46%	-23%
5	6,29	4,22	3,19	-49%	-24%
6	6,17	3,99	3,67	-40%	-8%
7	6,30	4,07	3,64	-42%	-11%
8	6,26	4,41	4,13	-34%	-6%

A Tabela 31 apresenta também um comparativo entre o consumo real e o consumo obtido

nesta pesquisa. Pôde-se observar uma redução ainda maior no consumo de combustível quando a abordagem usada foi a baseado em casos. Esta redução foi em média de 42%. Os resultados obtidos relativos, ao consumo de combustível, por meio da abordagem desenvolvida em termos de planejamento e execução de condução assistida mostraram-se bastante significativos. Por exemplo, a América Latina Logística, empresa responsável por transporte férreo de cargas no sul do Brasil gastou, somente no 3º trimestre de 2014, 172.9 milhões de reais em combustível (ALL Logística, 2014). Uma redução de 42% equivaleria a uma economia de aproximadamente 72 milhões de reais.

5.4 CONSIDERAÇÕES FINAIS

Este capítulo apresentou os resultados experimentais realizados com o objetivo de avaliar o método proposto. Foram realizados experimentos com 4 cenários para verificar a eficácia do método quando reaproveitadas as experiências passadas na condução de outros trens e em diferentes trechos. Os resultados foram comparados, em termos de consumo de combustível, pela medida do consumo de combustível.

6 CONCLUSÕES E DISCUSSÕES FINAIS

Esta pesquisa apresentou uma arquitetura para o reaproveitamento, elaboração e compartilhamento de planos de ações utilizados na tomada de decisões de agentes em um ambiente dinâmico. O ambiente estudado foi o modal férreo, onde a tomada de decisão é uma tarefa complexa e que exige o reuso da experiência, principalmente quando o grau de dinamismo do ambiente aumenta. A arquitetura desenvolvida teve como foco principal o desenvolvimento de um sistema inteligente capaz de realizar a condução de trens de carga. Para tornar isto possível foi realizado um estudo acerca dos modelos matemáticos que regem a cinemática de trens de carga.

Neste trabalho defende-se a ideia de que mecanismos computacionais são capazes de gerar planos de condução de trens de carga eficientes, considerando experiências passadas como soluções reaproveitáveis. Para tal, foram utilizados mecanismos que permitiram o reaproveitamento de ações passadas na solução de novos problemas e aprendizado ao longo do tempo (raciocínio baseado em casos), combinado com um método de otimização (algoritmo genético) para adaptação e otimização dos casos recuperados.

A iniciativa em planejar e executar ações com base em experiências anteriores mostrou-se útil em um sistema realista e com características únicas, como é o modal férreo, objeto do estudo. Este modal pode ser considerado único devido à complexidade da condução, sinalização e dos cálculos envolvidos no processo. Observou-se que as experiências, quando reaproveitadas em cenários semelhantes, mostraram-se muito eficientes na elaboração e execução de planos de condução, com eficiência média de adaptação de planos na ordem de 74% e uma similaridade, entre o que foi planejado e executado, na ordem de 90%. Esta troca de experiências foi possível graças à relação existente entre o agente *Memorizador* e o agente *Planejador*. O primeiro foi responsável por armazenar e organizar os planos de condução aplicados, transformando-os em conhecimentos aplicáveis pelo *Planejador*. Neste cenário, o procedimento de preparação da base de casos, identificação dos atributos relevantes à condução, a determinação de pesos e a normalização dos atributos mostraram-se necessários e eficazes no processo de recuperação de casos. Esta eficácia foi alcançada também pelo uso da métrica da distância Euclidiana na recuperação dos casos.

Na etapa de organização e identificação de planos, a experiência acerca do domínio do problema foi importante na identificação de quais atributos são relevantes na composição de um caso. A investigação dos atributos mais relevantes foi realizada utilizando o C4.5, conforme publicado em (Borges A. P., et al., 2012). Esta publicação encontra-se no Apêndice B. Contudo, a recuperação dos

casos mostrou-se mais eficiente quando os atributos relevantes foram validados por experimentos.

Um dos principais desafios foi a determinação do método de adaptação dos casos. Segundo estudos realizados, as etapas de recuperação e validação dos conhecimentos utilizam, com frequência, conhecimentos do domínio de aplicação. A etapa de adaptação de casos, entretanto, possui uma variedade de algoritmos e métodos que podem ser utilizados. Um dos nortes usados foi criar um agente capaz de conduzir trens de carga economizando combustível, o que levou a escolha de um método de otimização, o algoritmo genético. Este método foi escolhido por ter sido objeto de estudo neste modal por outros pesquisadores. Contudo, em nenhum trabalho anterior estudado este método foi utilizado juntamente com o raciocínio baseado em casos em um ambiente realista, dinâmico e complexo como o modal férreo. Em laboratório foi possível observar que o algoritmo genético conseguiu adaptar com sucesso os casos recuperados.

O sucesso da etapa de adaptação e recuperação deu-se também em função do aumento do número de casos/diversidade presentes na base de casos do agente *Memorizador*. A diversidade foi alcançada com variações nos cenários de teste — mudanças de trechos de via férrea e características dos trens utilizados. Entretanto, os maiores benefícios da diversidade puderam ser notados rapidamente quando houve uma variação nos trechos de vias férreas.

Outra hipótese levantada no início deste trabalho foi verificar se o compartilhamento de experiências entre agentes que atuam num mesmo ambiente poderia aprimorar as ações que o agente deve executar. Esta hipótese foi comprovada pelo compartilhamento, ao longo do tempo, dos planos de ações executados por um agente com os outros agentes que operam no mesmo trecho. Observou-se que, à medida que novos planos foram incorporados à base de conhecimento disponível na estação, os planos elaborados obtiveram uma maior eficiência e uma satisfatória similaridade entre o que foi planejado e o executado. Neste caso, a técnica de raciocínio baseado em casos mostrou-se eficaz.

Quanto à capacidade dos agentes em planejarem e executarem a condução de trens de carga, os resultados empíricos mostraram que a abordagem adotada pode ser generalizada e empregada em várias estações de uma ferrovia. Foi mostrado que a eficiência das tarefas dos agentes *Memorizador* e *Planejador* melhoraram, à medida que novos casos foram obtidos. Tal eficiência gera uma tendência de reduzir esforços em planejamento e replanejamento dos planos de condução. Obviamente, se as condições mudarem significativamente, os esforços de planejamento aumentam, pelo menos inicialmente.

Finalmente, em termos de domínio de aplicação as principais contribuições são: em termos monetários, os planos de condução gerados podem produzir ganhos significativos; e em termos de

reuso de experiências, a abordagem sugere que bons condutores podem ser utilizados para conduzir trens em diferentes trechos da via férrea, por determinado tempo, de modo a gerar experiências. Tais experiências podem ser utilizadas para gerar bons planos de condução para condutores menos experientes. Isto ajudará os especialistas na tarefa de conduzir trens de modo eficiente. Neste caso, o sistema inteligente seria um assistente que recomenda ações, ajudando novatos em sua tarefa.

6.1 SUGESTÕES PARA TRABALHOS FUTUROS

Os resultados observados com a arquitetura desenvolvida para o reaproveitamento, elaboração e compartilhamento de planos de ações utilizados na tomada de decisões de agentes indicam que novas pesquisas podem ser realizadas nos ambientes estudados.

No que concerne ao agente *Memorizador* pesquisas podem ser realizadas no sentido de identificar uma combinação ideal dos atributos do problema a serem utilizados na etapa de recuperação de casos. Sugere-se diversificar os atributos que compõe um caso, incluindo novos atributos (e.g., perfil do trecho seguinte) e também diversificar os valores que compõe alguns perfis (e.g., ampliar o conjunto que determina o perfil de ‘Aclive, Declive e Em Nível’ para ‘Aclive, Declive, Em Nível e Crista’). Acredita-se que, com a maior precisão do perfil atual e uma indicação do próximo perfil a ser percorrido, a eficiência da recuperação dos casos seja ainda maior, uma vez que são fornecidos maiores detalhes sobre o *problema-alvo*. Abre-se também um horizonte de pesquisa quanto à aplicação de métodos de reorganização da base de casos em cenários reais, como por exemplo, a utilização de técnicas de *clusters* para identificação e remoção de casos obsoletos, com o objetivo de otimizar a base de casos do agente *Memorizador*.

Apesar dos resultados encorajadores da etapa de adaptação com algoritmo genético, é possível que novas técnicas de adaptação sejam utilizadas, como por exemplo, a própria técnica de satisfação por restrições, pois a etapa de adaptação pode ser implementada de diversas formas. Ao implementar a técnica de satisfação por restrições na adaptação, pode-se comparar o desempenho da técnica com e sem o reuso de casos. Outras alternativas é a utilização de modelos bayesianos, os quais apresentaram boa capacidade de generalização.

Quanto à própria utilização do algoritmo genético como método de adaptação de casos, este é passível de vários estudos, como por exemplo, variações no tamanho da população inicial, nas taxas de *cruzamento* e *mutação*. Pode-se analisar, por exemplo, o impacto que uma maior diversidade da população tem sobre à eficácia do planejamento, alterando as taxas de mutação e cruzamento. Pode-se analisar o impacto de uma população maior de casos na geração dos planos, e também estudar

outras formas de avaliar os indivíduos em termos de múltiplos objetivos, pela inclusão de novos fatores à função de *fitness* ou até mesmo pela aplicação de outras técnicas de cruzamento e mutação.

Quanto à aplicação foram utilizados dois trechos nos cenários de experimentos. Novas pesquisas podem analisar o impacto da arquitetura desenvolvida em novos trechos e também com uma diversidade maior nas configurações de trens. Ainda em termos de aplicação, um horizonte abre-se quanto a utilização de mecanismos de certificação das informações trocadas pelos agentes *Memorizador*, *Planejador* e *Executor*.

6.2 PUBLICAÇÕES RELACIONADAS

Neste trabalho procurou-se desenvolver uma arquitetura para o reaproveitamento, a elaboração e o compartilhamento de planos de ações em um ambiente dinâmico e também estudar o impacto desta arquitetura.

Os principais resultados deste trabalho foram objetos das seguintes publicações:

1. BORGES, ANDRE PINZ; DORDAL, OSMAR BETAZZI; SATO, DENISE MARIA VECINO; AVILA, BRAULIO COELHO; ENEMBRECK, FABRICIO; SCALABRIN, EDSON EMILIO. **Efficient Approach for Reusing and Sharing Train Driving Plans Using Case-Based Reasoning**. In: 30th ACM/SIGAPP Symposium On Applied Computing ACM SAC, Salamanca, Spain. 2015. (*To appear* – Artigo aceito para publicação)
2. BORGES, ANDRE PINZ; DORDAL, OSMAR BETAZZI; RIBEIRO, RICHARDSON; AVILA, BRAULIO COELHO; SCALABRIN, EDSON EMILIO. **An Economic Approach for Generation of Train Driving Plans Using Continuous Case-Based Planning**. In: Proceedings of 17th International Conference on Enterprise Information Systems (ICEIS), Barcelona, Spain. 2015. (To appear – Artigo aceito para publicação)
3. BORGES, ANDRE PINZ; BÔTELHO, VANDERSON; DORDAL, OSMAR BETAZZI; AVILA, BRAULIO COELHO; SCALABRIN, EDSON EMILIO. **Safety in Multi-Agent Systems: Reputation based on Dossier**. In: Proceedings of 28th International Florida Artificial Intelligence Research Society Conference, Hollywood, USA. 2015. (To appear – Artigo aceito para publicação)
4. BORGES, ANDRE PINZ; DORDAL, OSMAR BETAZZI; SATO, DENISE MARIA VECINO; AVILA, BRAULIO COELHO; ENEMBRECK, FABRICIO; SCALABRIN, EDSON EMILIO. **An intelligent system for driving trains using Case-Based Reasoning.** In: 2012 IEEE International Conference on Systems, Man and Cybernetics SMC, Seoul.

2012 IEEE International Conference on Systems, Man, and Cybernetics (SMC). IEEE, 2012. p.1694;

5. BORGES, ANDRE PINZ; GRANATYR, JONES; DORDAL, OSMAR BETAZZI; RIBEIRO, RICHARDSON; AVILA, BRAULIO COELHO; ENEMBRECK, FABRICIO; SCALABRIN, EDSON EMILIO. **Knowledge discovery applied in modal rail**. In: 2011 15th International Conference on Computer Supported Cooperative Work in Design (CSCWD), Laussane. Proceedings of the 2011 15th International Conference on Computer Supported Cooperative Work in Design (CSCWD). IEEE, 2011. v.1. p.253;

7 REFERÊNCIAS BIBLIOGRÁFICAS

Aamodt, A., & Plaza, E. (Mar de 1994). Case-based reasoning: foundational issues, methodological variations, and system approaches. *AI Commun., 7*(1), 39-59.

Abbink, E., Mobach, D., Fioole, P., Kroon, L., van der Heijden, E., & Wijngaards, N. (2010). Real-time train driver rescheduling by actor-agent techniques. *Public Transport, 2*(3), 249-268.

Abelleyro, R. (2009). *Wireless PTC TechnologyOutline.* STI-CO Industries, Inc.

Abuiziah, I., & Shakarneh, N. (2013). A Review of Genetic Algorithm Optimization: Operations and Applications to Water Pipeline Systems. *International Journal of Physical, Natural Science and Engineering, 7*(12), 341-347.

Acikbas, S., & Soylemez, M. (May de 2008). Coasting point optimisation for mass rail transit lines using artificial neural networks and genetic algorithms. *Journal of Electric Power Applications, IET, 2*(3), 172-182.

Aha, D. W., Breslow, L., & Muñoz-Avila, H. (2001). Conversational Case-Based Reasoning. *Applied Intelligence, 14*(1), 9-32.

Aha, D. W., Kibler, D., & Albert, M. K. (January de 1991). Instance-Based Learning Algorithms. *Machine Learning, 6*(1), 37-66.

Albrecht, T., Luddecke, K., & Zimmermann, J. (2013). A precise and reliable train positioning system and its use for automation of train operation. *Intelligent Rail Transportation (ICIRT), 2013 IEEE International Conference* (pp. 134-139). Beijing: IEEE.

ALL Logística. (2014). *Release de resultados trimestrais.* ALL Logística, Curitiba.

Allotta, B., Chisci, L., D'Adamio, P., Papini, S., & Pugi, L. (6-7 de June de 2013). Design of an Automatic Train Operation (ATO) system based on CBTC for the management of driverless suburban railways. *12th IMEKO TC10 Workshop on Techinical Diagnostics*, pp. 1-6.

Althoff, K.-D., & Wess, S. (1991). Case-Based Reasoning and Expert System Development. *Contemporary Knowledge Engineering and Cognition*, (pp. 146-158).

Alves, F., & Pires, C. (April de 2010). Energy saving strategy in São Paulo Metro. *Railway Traction Systems (RTS 2010), IET Conference on*, pp. 1-4.

Amdouni, I., Jeddi, N., & Amraoui, L. (2013). Optimal control approach developed to Four-Wheel Active Steering Vehicles. *Modeling, Simulation and Applied Optimization (ICMSAO), 2013 5th International Conference on, 1*(1), 1-6.

Andreotti, A., Martinis, V., & Torrieri, V. (2014). An Approach to Energy Efficient Speed Profiles Optimization in Railway Systems. *International Journal of Applied Engineering Research, 9*(18), 4489-4502.

Back, T., Fogel, D. B., & Michalewicz, Z. (2000). *Evolutionary Computation 1: Basic Algorithms and Operators* (1 ed., Vol. 1). (T. Back, D. B. Fogel, & Z. Michalewicz, Eds.) New York: Taylor & Francis Group.

Bain, W. M. (1986). *Case-based reasoning: a computer model of subjective assessment.* Ph.D. dissertation, Yale University.

Bajpai, A., Karad, D., Pawar, M., & Pilakar, M. (2007). *An Introduction to the Train Management System Of Western Railway.* Mumbai: Shah and Anchor Kutchhi Engineering College, W. T. Patil Marg, Chembur, Mumbai-400088.

Barletta, R. (1991). An introduction to case-based reasoning. *AI Expert, 6*, 43-49.

Barletta, R., & Mark, W. (1988). Explanation-Based Indexing of Cases. Em H. E. Shrobe, T. M. Mitchell, & R. G. Smith (Ed.), *AAAI* (pp. 541-546). AAAI Press / The MIT Press.

Behbood, V. (2011). Intelligent financial warning model using Fuzzy Neural Network and case-based reasoning. *Computational Intelligence for Financial Engineering and Economics (CIFEr), 2011 IEEE Symposium on*, 1-6.

Bellifemine, F. L., Caire, G., & Greenwood, D. (2007). *Developing Multi-Agent Systems with JADE* (1 ed., Vol. I). Wiley.

Bergmann, R., & Wilke, W. (1996). PARIS: Flexible Plan Adaptation by Abstraction and Refinement. *Proceedings of the Workshop on Adaptation in Case-Based Reasoning, th European Conference on Artificial Intelligence*, 1-5.

Bergmann, R., Kolodner, J., & Plaza, E. (2005). Representation in case-based reasoning. *The Knowledge Engineering Review, 20*(3), 209-213 .

Bhardwaj, A., Ghosh, S., & Dutta, A. (2013). Modeling of Multiagent Based Railway System. *Proceedings of Asia Conference on Future Trends in Computing and Communication*

(ACFTCC).

Biddle, B. J. (1 de August de 1986). Recent Developments in Role Theory. *Annual Review of Sociology, 12*, pp. 67-92.

Bocharnikov, Y., Tobias, A., & Roberts, C. (April de 2010). Reduction of train and net energy consumption using genetic algorithms for Trajectory Optimisation. *Railway Traction Systems (RTS 2010), IET Conference on*, pp. 1-5.

Borges, A. P. (2009). *Descoberta de Regras de Condução de Trens de Carga.* Master's thesis, Pontifícia Universidade Católica do Paraná.

Borges, A. P., Bôtelho, V., Dordal, O. B., Ávila, B. C., & Scalabrin, E. E. (18-20 de May de 2015). Safety in Multi-Agent Systems: Reputation based on Dossier. *Proceedings of 28th International Florida Artificial Intelligence Research Society Conference*, p. to appear.

Borges, A. P., Dordal, O. B., Ribeiro, R., Ávila, B. C., & Scalabrin, E. E. (27-30 de April de 2015). An Approach for Generate Train Driving Plans Using Case-Based Reasoning. *Proceedings of 17th International Conference on Enterprise Information Systems (ICEIS)*, p. to appear.

Borges, A. P., Dordal, O. B., Sato, D. M., Ávila, B. C., Enembreck, F., & Scalabrin, E. E. (14-17 de Oct de 2012). An intelligent system for driving trains using Case-Based Reasoning. *Systems, Man, and Cybernetics (SMC), 2012 IEEE International Conference on*, pp. 1694-1699.

Borges, A. P., Dordal, O. B., Sato, D. M., Ávila, B. C., Enembreck, F., Scalabrin, E. E., & Ribeiro, R. (May de 2014). A Multi-Layer Architecture Proposal for Conducting Trains Employing CBR. *Proceedings of 2014 IEEE 18th International Conference on Computer Supported Cooperative Work in Design (CSCWD 2014)*, pp. 23-28.

Borges, A. P., Ribeiro, R., Ávila, B. C., Enembreck, F., & Scalabrin, E. E. (22-24 de April de 2009). A learning agent to help drive vehicles. *Proceedings of 13th International Conference on Computer Supported Cooperative Work in Design, 0*, pp. 282-287.

Borges, A., Dordal, O., Sato, D., Enembreck, F., Ávila, B., & Scalabrin, E. (13-17 de April de 2015). Efficient Approach for Reusing and Sharing Train Driving Plans Using Case-Based Reasoning. *Proceedings of 30th ACM/SIGAPP Symposium On Applied Computing*, p. to appear.

Borges, A., Granatyr, J., Dordal, O., Ribeiro, R., Avila, B., Enembreck, F., & Scalabrin, E. (8-10 de June de 2011). Knowledge discovery applied in modal rail. *Proceedings of 15th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*,

pp. 253-260.

BRASIL. (2007). *Plano Nacional de Logística e Transportes – PNLT: Relatório Executivo.* Ministério dos Transportes, Brasília.

Butdee, S. (2011). Adaptive Aluminum Extrusion Die Design Using Case-Based Reasoning and Artificial Neural Networks. *Advanced Materials Research, 383 - 390*, 6747-6754.

Carbonell, J. (1985). *Derivational Analogy: A Theory of Reconstructive Problem Solving and Expertise Acquisition.* Carnegie-Mellon University.

Chakraborty, B., Ghosh, D., Ranjan, R., Garnaik, S., & Debnath, N. (2010). Knowledge Management with Case-Based Reasoning applied on Fire Emergency Handling. *Industrial Informatics (INDIN), 2010 8th IEEE International Conference on*, (pp. 708-713).

Chandra, S., & Agarwal, M. M. (2007). *Railway Engineering.* Oxford University Press.

Chang, C. S., & Sim, S. S. (1997). Optimising train movements through coast control using genetic algorithms. *IEEE Proceedings of Electric Power Applications, , 144*(1), pp. 65-73.

Chang, C., & Xu, D. (May de 2000). Differential evolution based tuning of fuzzy automatic train operation for mass rapid transit system. *147*(3), 206-212.

Charniak, E., & McDermott, D. (1985). *Introduction to Artificial Intelligence.* Addison-Wesley Pub.Company.

Chen, B., & Cheng, H. (2010). A Review of the Applications of Agent Technology in Traffic and Transportation Systems. *Intelligent Transportation Systems, IEEE Transactions on, 11*(2), 485-497.

Chen, Y., & Jiang, C. (2013). Research of the Case Retrieval Model Based on CBR. *Intelligent System Design and Engineering Applications (ISDEA), 2013 Third International Conference on*, (pp. 1132-1136--).

Chou, C. H., Kuo, B. H., & Chang, F. (2006). The generalized condensed nearest neighbor rule as a data reduction method. *Pattern Recognition, International Conference on, 2*, 556-559.

Coello, C. A., Lamont, G. B., & Veldhuizen, D. A. (2006). *Evolutionary Algorithms for Solving Multi-Objective Problems (Genetic and Evolutionary Computation).* Secaucus, NJ, USA:

Spring-Verlag New York.

Corchado, J. M., & Laza, R. (2003). Constructing deliberative agents with case-based reasoning technology. *International Journal of Intelligent Systems, 18*(12), 1227-1241.

Coutand, O. (2009). *A Framework for Contextual Personalised Applications.* Thesis, University of Kassel, kassel university press.

Craw, S., Wiratunga, N., & Rowe, R. (2006). Learning adaptation knowledge to improve case-based reasoning. *Artificial Intelligence, 170*(16-17), 1175–1192.

Cunningham, P. (2009). A Taxonomy of Similarity Mechanisms for Case-Based Reasoning. *IEEE Transactions on Knowledge and Data Engineering, 21*(11), 1532-1543.

da Silva, M., Borges, A., Dordal, O., Sato, D., Avila, B., Enembreck, F., & Scalabrin, E. (23-25 de May de 2012). An architecture of BDI agent for autonomous locomotives controller. *Proceedings of 16th IEEE International Conference on Computer Supported Cooperative Work in Design (CSCWD), 2012* , pp. 22-29.

Daddario, E. Q., & de Simone, D. V. (1976). *Automatic Train Control in Rail Rapid Transit.* Washington D.C.: Technical Report of Congress of the United States.

Darwin, C. (1859). *On the Origin of Species by Means of Natural Selection.* London: John Murray.

Davies, S. D. (2000). *Automatic Train Protection for the Railway Network in Britain - A Study.* London: The Royal Academy of Engineering.

de Mántaras, R. L., Bridge, D., & Mcsherry, D. (1997). Case-based reasoning: an overview. *AI Communications, 10*(1), 21-29.

de Mántaras, R., Mcsherry, D., Bridge, D., Leake, D., Smyth, B., Craw, S., . . . Watson, I. (2005). Retrieval, reuse, revision and retention in case-based reasoning. *The Knowledge Engineering Review, 20*(03), 215-240.

Deterline, W. A. (2004). *An Introduction to Programmed Instruction.* Upper Saddle River, NJ: Prentice Hall.

Dias, T. G., de Sousa, J. P., & Cunha, J. F. (2002). Genetic Algorithms for the Bus Driver Scheduling Problem: A Case Study. *The Journal of the Operational Research Society,*

53(3), 324-335.

DNIT. (2014). Glossário dos Termos Ferroviários. pp. 1-70. Acesso em 01 de Dezembro de 2014, disponível em http://www.transportes.gov.br/images/GLOSSARIO/glossarioTermosFerroviarios.pdf

DNIT. (2014). ISF 209: Projeto Geométrico. Acesso em 20 de Novembro de 2014, disponível em http://www.dnit.gov.br/sala-de-imprensa/isf-209-projeto-geometrico.pdf/view

DNIT. (2014). ISF: 203 Estudos topográficos. Acesso em 20 de Novembro de 2014, disponível em http://www.dnit.gov.br/sala-de-imprensa/isf-203-estudos-topograficos-basico.pdf/view

Dong Hairong, L. L. (2010). Fuzzy Tuning of ATO System in Train Speed Control with MultipleWorking Conditions. *Proceedings of the 29th Chinese Control Conference.* Beijing.

Dordal, O. B., Borges, A. P., Ribeiro, R., Enembreck, F., Scalabrin, E. E., & Ávila, B. C. (2011a). Towards an optimal driving trains in single line using crossing loops. Em W. Shen, J.-P. A. Barthès, J. Luo, P. G. Kropf, M. Pouly, J. Yong, . . . M. P. Ramos (Ed.), *CSCWD* (pp. 772-779). IEEE.

Dordal, O. B., Borges, A. P., Ribeiro, R., Enembreck, F., Scalabrin, E. E., & Avila, B. C. (oct. de 2011b). Strong reduction in fuel consumption driving trains in bi-directional single line using crossing loops. *Systems, Man, and Cybernetics (SMC), 2011 IEEE International Conference on*, (pp. 1597-1602).

Duda, R. O., Hart, P. E., & Stork, D. G. (2000). *Pattern Classification* (2 ed.). New Jersey, NY: Wiley-Interscience.

Durfee, E., Lesser, V., & Corkill, D. D. (1989). Trends in cooperative distributed problem solving. *Knowledge and Data Engineering, IEEE Transactions on, 1*(1), 63-83.

Eiben, A. E., & Smith, J. E. (2003). *Introduction to Evolutionary Computing* (1ª ed.). SpringerVerlag.

Einer, S., Slovak, R., & Schnieder, E. (2000). Modeling train control systems with Petrinets-an operational specification. *Systems, Man, and Cybernetics, 2000 IEEE International Conference on, 5*, pp. 3207-3211 vol.5.

Eldredge, D., & Houpt, P. (2011). *Trip optimizer for railroads.* IEEE Control Systems Society.

Ferber, J. (1999). *Multi-agent systems: an introduction to distributed artificial intelligence* (1a. ed., Vol. I). (J. Rouchier, Ed.) Harlow: Addison Wesley Longman.

Finin, T., Weber, J., Wiederhold, G., Genesereth, M., Fritzson, R., McKay, D., . . . Beck, C. (1994). *Specification of the KQML Agent-Communication Language.* University of Toronto, Toronto.

Franklin, S., & Graesser, A. (1997). Is it an Agent, or Just a Program?: A Taxonomy for Autonomous Agents. *Proceedings of the Workshop on Intelligent Agents III, Agent Theories, Architectures, and Languages* (pp. 21-35). London, UK, UK: Springer-Verlag.

Freitas, A. (01 de December de 2004). A Critical Review of Multi-objective Optimization in Data Mining: A Position Paper. *SIGKDD Explor. Newsl., 6*(2), 77-86.

Freitas, A. A. (2008). A Review of evolutionary Algorithms for Data Mining. Em O. Maimon, & L. Rokach (Eds.), *Soft Computing for Knowledge Discovery and Data Mining* (pp. 79-111). Springer US.

Fuchs, B., Lieber, J., Mille, A., & Napoli, A. (2014). Differential adaptation: An operational approach to adaptation for solving numerical problems with CBR. *Knowledge-Based Systems*, *68*(1), 103–114.

Ghosh, S., Dutta, A., & Alam, M. A. (2013). Multi-agent based railway track management System. *Advance Computing Conference (IACC)*, 1408-1413.

Ginty, L. M., & Smyth, B. (2001). Collaborative Case-Based Reasoning: Applications in Personalised Route Planning. *Case-Based Reasoning Research and Development, 2080*, pp. 362-376.

Glover, F., & Laguna, M. (1997). *Tabu Search.* Norwell, MA, USA: Kluwer Academic Publishers.

Goel, P. P. (2010). *Indian Railway Signal Engineering.* Astha Publications.

Göker, M. H., & Roth-Berghofer, T. (December de 1999). The development and utilization of the case-based help-desk support system HOMER. *Engineering Applications of Artificial Intelligence, 12*(1), 665-680.

Goldberg, D. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning* (1st ed.). Boston, MA, USA: Addison-Wesley Longman Publishing Co.

Goldstein, E. (2009). *Sensation and Perception.* Stamford, USA: Cengage Learning.

Gómez, M., & Plaza, E. (2012). Case-Based Project Scheduling. *Case-Based Reasoning Research and Development, 7466*, 122-136.

Gouttaya, N., & Begdouri, A. (2012). Integrating data mining with case based reasoning (CBR) to improve the proactivity of pervasive applications. *Information Science and Technology (CIST), 2012 Colloquium in*, (pp. 136-141--).

Gu, Q., Cao, F., & Tang, T. (Sep de 2012). Energy efficient driving strategy for trains in MRT systems. *Intelligent Transportation Systems (ITSC)*, 427-432.

Hammond, K. J. (11-15 de August de 1986). CHEF: A Model of Case-Based Planning. *AAAI-86 Proceedings*, pp. 267-271.

Han, S., Byen, Y. S., Baek, J. H., An, T. K., Lee, S., & Park, H. J. (1999). An optimal automatic train operation (ATO) control using genetic algorithms (GA). *TENCON 99. Proceedings of the IEEE Region 10 Conference*, *1*, pp. 360-362 vol.1.

Harré, R., Clarke, D. D., & Carlo, N. D. (1985). *Motives and Mechanisms: An Introduction to the Psychology of Action.* London, UK: Methuen.

Hartong, M., Goel, R., & Wijesekera, D. (2006). Communications Based Positive Train Control Systems Architecture in the USA. Em IEEE (Ed.), *Vehicular Technology Conference. 6*, pp. 1550-2252. Melbourne: IEEE.

Hartong, M., Goel, R., & Wijesekera, D. (2012). Transportation. *Critical Infrastructure Protection, 7130*, pp. 330-355. doi:10.1007/978-3-642-28920-0_14

Hay, W. W. (1982). *Railroad Engineering* (2ª ed., Vol. 1). Wiley.

Hengyu, L., & Hongze, X. (July de 2012). An Integrated Intelligent Control Algorithm for High-Speed Train ATO Systems Based on Running Conditions. *Digital Manufacturing and Automation (ICDMA), 2012 Third International Conference on*, 202-205.

Henriet, J., P-E., L., Laurent, R., & Salomon, M. (2014). Case-Based Reasoning adaptation of numerical representations of human organs by interpolation. *Expert Systems with Applications, 41*(2), 260 - 266.

Hinrichs, T. (May de 1989). Strategies for Adaptation and Recovery in a Design Problem Solver. *Proceedings of the 1989 DARPA Workshop on Case-Based Reasoning*, pp. 115-118.

Howlett, P. G., & Cheng, J. (January de 1997). Optimal driving strategies for a train on a track with continuously varying gradient. *Journal of the Australian Mathematical Society. Series B. Applied, 38*(3), 388-410.

Iacob, S. M., Nieuwenhuis, C. H., Wijngaards, N. J., Pavlin, G., & van Veelen, J. B. (2009). Actor-Agent Communities: Design Approaches. *Intelligent Distributed Computing III, 237*, 237-242.

International Energy Agency. (2012). *Railway Handbook 2012: Energy Consumption and CO2 Emissions.* International Energy Agency, Paris.

Jalali, V., & Leake, D. (2013). Extending case adaptation with automatically-generated ensembles of adaptation rules. *Case-Based Reasoning Research and Development, 7969*(1), 188-202.

Jong, A. D. (1975). *An Analysis of the Behavior of a Class of Genetic Adaptive Systems.* Ann Arbor, MI, USA: University of Michigan.

Jung, S., Lim, T., & Kim, D. (2009). Integrating radial basis function networks with case-based reasoning for product design. *Expert Systems with Applications, 36*(3, Part 1), 5695 - 5701.

Junior, I., Hino, C., Gonçalves, P., Andrade, L. d., Moreira, C., Costa, G., . . . Magalhães, D. J. (24-28 de July de 2011). Reducing CO2 Emissions due to a shift from Road to Cabotage Transport of Cargo in Brazil. *Proceedings of the 29th International Conference of the System Dynamics Society*, pp. 1-27.

Jurisica, I., & Glasgow, J. (1995). Applying Case-Based Reasoning to Control in Robotics. *In 3 rd Robotics and Knowledge-Based Systems Workshop*, (pp. 189-196).

Kass, A., Leake, D., & Owens, C. (1986). SWALE: A program that explains. *Explanation Patterns: Understanding Mechanically and Creatively, 1*(1), 232-254.

Kavuluri, B. R., & Kumar, R. A. (2011). CBL- Case Based Planning for Learning. *In Procedings of AAAI.*

Ke-Ping, L., Zi-You, G., & Bao-Hua, M. (2007). Energy-optimal control model for train movements. *Chinese Physics, 16*, pp. 1-6.

Khattak, A., & Kanafani, A. (October de 1996). Case-based reasoning: a planning tool for intelligent transportation systems. *Transportation Research Part C: Emerging Technologies, 4*(5), 267-288.

Klenk, M., Aha, D. W., & Molineaux, M. (2011). The Case for Case-Based Transfer Learning. *AI Magazine, 32*(1), pp. 54-69.

Ko, H., Koseki, T., & Miyatake, M. (2004). Application Of Dynamic Programming To The Optimization Of The Running Profile Of A Train. *Computers in Railways IX*, 1-10.

Kolodner, J. (1993). *Case-based reasoning.* Morgan Kaufmann Publishers.

Koton, P. (1989). *Using experience in learning and problem solving.* Ph.D. dissertation, Massachusetts Institute of Technology, Laboratory of Computer Science.

Kumar, B. N., & Basha, A. M. (May de 2013). QoS Enhancements using IEEE 802.11p WLANs for Communication Based Train Control Systems. *International Journal of Advanced Research in Computer and Communication Engineering, 2*.

Kumar, M., Husian, M., Upreti, N., & Gupta, D. (2010). Genetic Algorithm: Review and Application. *International Journal of Information Technology and Knowledge Management, 2*(2), 451-454.

Kurzweil, R. (1999). *The Age of Spiritual Machines: When Computers Exceed Human Intelligence.* Phoenix: Viking Press.

Kutz, M. (2011). *Handbook of Transportation Engineering.* McGraw Hill Professional.

Laurence, S., & Margolis, E. (1999). Concepts and Cognitive Science. *Concepts: Core Readings: Massachusetts Institute of Technology*, 3-83.

Leake, D. (1992). *Evaluating Explanations: A Content Theory* (1 ed.). Taylor & Francis.

Leake, D. (1996). *Case-based reasoning: experiences, lessons, and future directions.* (D. Leake, Ed.) AAAI Press.

Leake, D. B., & Sooriamurthi, R. (2002). Managing Multiple Case Bases: Dimensions and Issues. *Proceedings of the 15th International Florida Artificial Intelligence Research Society Conference*, 106-110.

Leake, D. B., & Sooriamurthi, R. (2003). Dispatching Cases versus Merging Case-Bases: When MCBR Matters. *FLAIRS Conference*, pp. 129-133.

Lechelle, S. A., & Mouneimne, Z. (2010). OptiDrive: A practical approach for the calculation of energy-optimised operating speed profiles. *Railway Traction Systems (RTS 2010), IET Conference on*, (pp. 1-8).

Leite, A. R. (2009). *Um esquema para redução do consumo de combustível em sistemas de condução férrea baseado em otimização distribuída de restrição.* Master's thesis, Pontifícia Universidade Católica do Paraná, Curitiba, Brazil.

Leite, A. R., Giacomet, B., & Enembreck, F. (2009). Railroad Driving Model Based on Distributed Constraint Optimization. *Web Intelligence and Intelligent Agent Technologies, 2009. WI-IAT '09. IEEE/WIC/ACM International Joint Conferences on, 2*, pp. 474-481.

Liao, Z., Mao, X., Hannam, P., & Zhao, T. (2012). Adaptation methodology of CBR for environmental emergency preparedness system based on an Improved Genetic Algorithm. *Expert Systems with Applications, 39*(8), 7029 - 7040.

Lin, J., Deng, Q., & Peng, Y. (2000). Intelligent Reasoning System Of Die & Mold Structure Cases Based On Multiple Concept Learning. *Forging & Stamping Technology, 4*, 52-54.

Liu, J., Cai, B.-g., & Wang, J. (2014). A GNSS/Trackmap Cooperative Train Positioning Method for Satellite-based Train Control. Em IEEE (Ed.), *17th International Conference on Intelligent Transportation Systems* (pp. 8-11). Qingdao: IEEE.

Liu, T., Ma, J., Guan, W., Song, Y., & Fu, P. (2012). Design and Implementation of Bus Crew Scheduling System Using Integrated Case-based and Rule-based Reasoning. *Computational Sciences and Optimization (CSO), 2012 Fifth International Joint Conference on*, (pp. 475-479).

Lodhi, I., Hasan, K., Hasan, U., Mahmood, N., Yoshida, T., & Anwar, M. (2003). Optimizing retrieval process and using neural networks for adaptation process in case based reasoning systems. *Multi Topic Conference, 2003. INMIC 2003. 7th International*, 354-360.

Loumiet, J., Jungbauer, W., & Abrams, B. (2005). *Train Accident Reconstruction and FELA and Railroad Litigation.* Lawyers & Judges Publishing Company.

Luke, S. (2014). *Essentials of Metaheuristics* (second ed.). Lulu.

Luke, S., Panait, L., Balan, G., & Et. (2013). ECJ 21: A Java-based Evolutionary Computation Research System. Acesso em 05 de Jun de 2013, disponível em http://cs.gmu.edu/~eclab/projects/ecj/

Maes, P. (November de 1995). Artificial life meets entertainment: lifelike autonomous agents. *Communications of the ACM, 38*(11), 108-114.

Mahdi, W., Soui, M., & Abed, M. (2014). A new personalization approach by case-based reasoning and fuzzy logic. *Advanced Logistics and Transport (ICALT), 2014 International Conference on*, 103-108.

Manry, M. T., & Wilson, D. R. (2005). Prototype classifier design with prunning. *International Journal on Artificial Intelligence Tools*, 261-280.

Manzano, S., Ontañón, S., & Plaza, E. (2011). Amalgam-Based Reuse for Multiagent Case-Based Reasoning. *Case-Based Reasoning Research and Development, 6880*, 122-136.

Mckenna, E., & Smyth, B. (Feb de 2001). An Interactive Visualisation Tool for Case-Based Reasoners. *Applied Intelligence, 14*(1), 95-114.

Mendes, J. M. (2013). A Comparative Study of Crossover Operators for Genetic Algorithms to Solve the Job Shop Scheduling Problem. *WSEAS Transactions on Computers, 12*(4), 164-173.

Min, H., Huang, L., & Gan, X. (2012). Research and design of multi-agent model structure for embedded robot dog. *Intelligent Control and Automation (WCICA), 2012 10th World Congress on*, (pp. 3629-3633).

Minor, M., Bergmann, R., & Görg, S. (2014). Case-based adaptation of workflows. *Information Systems, 40*(1), 142–152.

Mitchell, M. (1996). *An Introduction to Genetic Algorithms.* Cambridge, MA, USA: MIT Press.

Mitchell, T. (1997). *Machine Learning* (1 ed.). New York: McGraw-Hill.

Mitra, R., & Basak, J. (Jun de 2005). Methods of case adaptation: A survey: Research Articles. *International Journal of Intelligent Systems, 20*(6), 627-645.

Nakanishi, Y., Tsuji, T., & Hakozaki, K. (1999). Development and Evaluation of Context Aware

Messaging Service using Location Information and Schedule Information. Em U. o. Chofu-si (Ed.)., (pp. 182-185--).

Neagu, N., & Faltings, B. (2001). Exploiting Interchangeabilities for Case Adaptation. Em D. Aha, & I. Watson (Eds.), *Lecture Notes in Computer Science* (Vol. 2080, pp. 422-436). Springer Berlin Heidelberg.

Nguyen, K., Beugin, J., & Marais, J. (2014). RAMS analysis of GNSS based localisation system for the train control application. *2nd International Conference on Computing, Management and Telecommunications.*

Nielsen, S. S., Danoy, G., & Bouvry, P. (2013). Vehicular Mobility Model Optimization Using Cooperative Coevolutionary Genetic Algorithms. *Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation*, 1349-1356.

NSW, T. T. (2013). *ESG 100 - Engineering Standard Signals - Signalling Design Principles.* RailCorp Engineering Standard.

OECD. (2005). *Oslo Manual: Guidelines for Collecting and Interpreting Innovation Data* (1 ed., Vol. 1). (N. Tanaka, M. Glaude, & F. Gault, Eds.) Organization for Economic Co-operation and Development Publishing.

Ontañón, S., Mishra, K., Sugandh, N., & Ram, A. (2010). On-line Case-Based Planning. *Computational Intelligence Journal, 26*(1), 84-119.

Oshima, H., Yasunobu, S., & Sekino, S.-i. (1988). Automatic train operation system based on predictive fuzzy control. *Artificial Intelligence for Industrial Applications, 1988. IEEE AI '88., Proceedings of the International Workshop on*, (pp. 485-489).

Otman, A., & Jaafar, A. (2011). A Comparative Study of Adaptive Crossover Operators for Genetic Algorithms to Resolve the Traveling Salesman Problem. *International Journal of Computer Applications, 31*(11), 49-57.

Pal, S., & Shiu, S. (2004). *Foundations of Soft Case-Based Reasoning.* Wiley.

Palumbo, M. (2013). *Railway Signalling since the birth to ERTMS.* railwaysignalling.eu.

Panayiotopoulos, T., & Zacharis, N. (September de 2001). Machine Learning and Intelligent Agents. *Machine Learning and Its Applications, 2049*(1), 281-285.

Pant, S., & Joshi, S. (2012). Case-based reasoning in Neurological Domain. *Internet (AH-ICI), 2012 Third Asian Himalayas International Conference on*, (pp. 1-5).

Pareto, V. (1896). *Cours D'Economie Politique* (1ª ed., Vol. 1). F. Rouge.

Parkinson, T., & Fisher, I. (1996). *Rail Transit Capacity TCRP Report 13.* Washington, D.C.: National Academy Press.

Passone, S., Chung, P., & Nassehi, V. (2006). Incorporating domain-specific knowledge into a genetic algorithm to implement case-based reasoning adaptation. *Knowledge-Based Systems, 19*(3), 192 - 201.

Perreira, A. L. (1958). *Estradas - Rodovias e Ferrovias* (1ª ed., Vol. Único). Ao Livro Técnico.

Peters, J. C., & Frittelli, J. (2012). *Positive Train Control (PTC): Overview and Policy Issues.* Congressional Research Service. CRS Report for Congress.

Petit, W. A. (2009). Interoperable Positive Train Control (PTC). *Annual Conference Communications and Signals Track.*

Picek, S., Golub, M., & Jakobovic, D. (2012). Evaluation of Crossover Operator Performance in Genetic Algorithms with Binary Representation. *Bio-Inspired Computing and Applications, 6840*(1), 223-230.

Plowman, L., Rogers, Y., & Ramage, M. (1995). What are Workplace Studies for? *Proceedings of the Fourth Conference on Computer-Supported Cooperative Work*, pp. 309-324.

Poole, D., Mackworth, A., & Goebel, R. (1998). *Computational Intelligence: A Logical Approach.* Oxford University Press on Demand.

Prased, M. V., & Plaza, E. (1996). Corporate Memories as Distributed Case Libraries. *Proceedings of the 10th Banff Knowledge Acquisition for Knowledge-based Systems Workshop, 2*, pp. 1-19.

Profillidis, V. A. (2006). *Railway Management and Engineering.* Ashgate Publishing, Ltd.

Purvis, L., & Pu, P. (1995). Adaptation Using Constraint Satisfaction Techniques. Em M. M. Veloso, & A. Aamodt (Ed.), *ICCBR. 1010*, pp. 289-300. Springer.

Qi, J., Hu, J., & Peng, Y. (2012). A new adaptation method based on adaptability under k-nearest neighbors for case adaptation in case-based design. *Expert Systems with Applications, 39*(7), 6485 - 6502.

Quinlan, R. (1993). *C4.5: Programs for Machine Learning.* San Mateo, CA: Morgan Kaufmann Publishers.

Redmond, M. (1990). Distributed cases for case-based reasoning; facilitating use of multiple cases. *Proceedings of the eighth National conference on Artificial intelligence - Volume 1* (pp. 304-309). AAAI Press.

Richter, M. M. (1992). *Classification and Learning of Similarity Measures.* Tech. rep., Fachbereich Informatik (Technische Universit\"at Darmstadt).

Riesbeck, R. S. (1989). *Inside Case-Based Reasoning.* Lawrence Erlbaum.

Roldán, E., Neágny, S., Lann, J. M., & Cortés, G. (2010). Constraint Satisfaction Problem for Case-Based Reasoning Adaptation: Application in Process Design. Em S. Pierucci, & G. B. Ferraris (Eds.), *20th European Symposium on Computer Aided Process Engineering* (Vol. 28, pp. 397-402). Elsevier.

Rosen, K. H. (2012). *Discrete Mathematics and Its Applications* (1st ed.). (M. Lange, Ed.) McGraw-Hill.

Rubin, J., & Watson, I. (2007). Investigating the Effectiveness of Applying Case-Based Reasoning to the Game of Texas Hold'em. Em D. Wilson, & G. Sutcliffe (Ed.), *FLAIRS Conference* (pp. 417-422). AAAI Press.

Russell, S., & Norvig, P. (1995). *Artificial Intelligence: A Modern Approach* (1ª ed.). New Jersey: Pearson Education/Prentice Hall. Fonte: http://books.google.com.br/books?id=8jZBksh-bUMC

Sato, D., Borges, A., Leite, A., Dordal, O., Avila, B., Enembreck, F., & Scalabrin, E. (2012). Lessons learned from a simulated environment for trains conduction. *Industrial Technology (ICIT), 2012 IEEE International Conference on*, (pp. 533-538).

Schank, R. (1975). *Conceptual Information Processing.* North-Holland.

Schank, R. C. (1983). *Dynamic Memory: A Theory of Reminding and Learning in Computers and People* (1 ed.). New York, NY, USA: Cambridge University Press.

Schank, R. C. (1986). *Explanation Patterns: Understanding Mechanically and Creatively.* (R. Schank, Ed.) Psychology Press.

Schank, R. C. (1999). *Dynamic Memory Revisited* (2 ed.). New York, NY, USA: Cambridge University Press.

Schank, R. C., & Abelson, R. P. (1977). *Scripts, plans, goals, and understanding: an inquiry into human knowledge structures.* (R. C. Schank, & R. P. Abelson, Eds.) L. Erlbaum Associates.

Schank, R. C., Osgood, R., Brand, M., Burke, R., Domeshek, E., Edelson, D., . . . Pryor, L. (1990). *A Content Theory of Memory Indexing.* Northwestern University. Evanston, IL: Institute for the Learning Sciences.

Shabana, A. A., Zaazaa, K. E., & S. H. (2007). *Railroad Vehicle Dynamics: A Computational Approach.* CRC Press.

Sharifi, M., Naghibzadeh, M., & Rouhani, M. (2013). Adaptive case-based reasoning using support vector regression. *Advance Computing Conference (IACC), 2013 IEEE 3rd International, 1*(1), 1006-1010.

shik Shin, K., & Han, I. (1999). Case-based reasoning supported by genetic algorithms for corporate bond rating. *Expert Systems with Applications, 16*(2), 85-95.

Shmeil, M. A. (1999). *Sistemas Multiagente na Modelação da Estrutura e Relações de Contratação de Organizações.* Porto: Universidade do Porto.

Silva, A. d. (2008). *Uma Introdução a Engenharia Ferroviária.* Clube de Autores, 2012.

Smiti, A., & Elouedi, Z. (October de 2011). Overview of Maintenance for Case based Systems. *International Journal of Computer Applications, 32*(2), 49-56.

Smiti, A., & Elouedi, Z. (2012). Competence and Performance-Improving approach for Maintaining Case-Based Reasoning Systems. *The 2nd International Conference on Computational Intelligence and Information Technology, CIIT*, 37-42.

Smiti, A., & Elouedi, Z. (2013). Using clustering for maintaining case based reasoning systems. *The 5th International Conference on Modelling, Simulation, and Applied Optimization, ICMSAO'13*, 1-6.

Smyth, B., & Cunningham, P. (1992). Dejá Vu: A Hierarchical Case-Based Reasoning System for Software Design. *ECAI*, pp. 587-589.

Smyth, B., & McKenna, E. (1999). Footprint-Based Retrieval. *In Proceedings of the Third International Conference on Case-Based Reasoning*, pp. 343-357.

Solomon, B. (2003). *Railroad Signaling.* MBI Publishing Company.

Spalzzi, L. (September de 2001). A Survey on Case-Based Planning. *Artificial Intelligence Review, 16*(1), 3-36.

Sycara, K. P., & Navinchandra, D. (1991). Influences: A Thematic Abstraction for Creative Use of Multiple Cases. *Proceedings of Workshop on Case-Based Reasoning*, (pp. --).

Tiako, R., Jayaweera, D., & Islam, S. (2011). A case-based reasoning approach for dynamic security assessment of power systems with large penetration of wind power. *Proceedings of Universities Power Engineering Conference (AUPEC), 2011 21st Australasian*, pp. 1-6.

Tiako, R., Jayaweera, D., & Islam, S. (2012). Real-time dynamic security assessment of power systems with large amount of wind power using Case-Based Reasoning methodology. *Proceedings of IEEE Power and Energy Society General Meeting, 2012* , pp. 1-7.

Tinghuai, M., Yong-Deak, K., Qiang, M., Meili, T., & Weican, Z. (2005). Context-aware implementation based on CBR for smart home. *Wireless And Mobile Computing, Networking And Communications, 2005. (WiMob'2005), IEEE International Conference on, 4*, pp. 112-115 Vol. 4.

Vacek, S., Gindele, T., Zollner, J., & Dillmann, R. (2007). Using case-based reasoning for autonomous vehicle guidance. *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, (pp. 4271-4276).

van der Krogt, R., & de Weerdt, M. (2005). Plan Repair as an Extension of Planning. Em S. Biundo, K. L. Myers, & K. Rajan (Ed.), *ICAPS* (pp. 161-170). AAAI.

Viana, G. V. (1998). *Meta-heurísticas e programação paralela em otimização combinatória* (1ª ed.). (G. V. Viana, Ed.) Fortaleza: UFC Edições.

von Wangenheim, C., Wangenheim, A., & Rateke, T. (2013). *Raciocínio baseado em casos com software livre e aplicativos móveis* (2ª ed.). Florianópolis: Bookess.

Vong, C., & Wong, P. (2010). Case-based adaptation for automotive engine electronic control unit calibration. *Expert Systems with Applications, 37*(4), 3184–3194.

Voyiatzis, A. (2012). A Survey of Delay- and Disruption-Tolerant Networking Applications. *Journal of Internet Engineering, 5*(1), 331-344.

Wang, Y., De Schutter, B., Ning, B., Groot, N., & Van den Boom, T. J. (2011). Optimal trajectory planning for trains using mixed integer linear programming. *Intelligent Transportation Systems (ITSC), 2011 14th International IEEE Conference on*, (pp. 1598-1604).

Wang, Y., Ning, B., Cao, F., & De Schutter, B. (2011). A survey on optimal trajectory planning for train operations. *Service Operations, Logistics, and Informatics (SOLI), 2011 IEEE International Conference* (pp. 589-594). Beijing: IEEE.

Watkins, C. J. (1989). *Learning from Delayed Rewards.* Ph.D. dissertation, King's College, Cambridge, UK.

Watson, I. (1997). *Applying case-based reasoning: techniques for enterprise systems.* (I. Watson, Ed.) San Francisco, CA, USA: Morgan Kaufmann.

Watson, I., & Rubin, J. (2008). CASPER: A Case-Based Poker-Bot. *Proceedings of the 21st Australasian Joint Conference on Artificial Intelligence: Advances in Artificial Intelligence* (pp. 594-600). Berlin, Heidelberg: Springer-Verlag.

White, D. R. (2012). Software review: the ECJ toolkit. *Genetic Programming and Evolvable Machines, 13*(1), 65-67.

Wilson, D. C., & Leake, D. B. (May de 2001). Maintaining Case-Based Reasoners: Dimensions and Directions. *Computational Intelligence*, pp. 196-213.

Wilson, D. L. (1972). Asymptotic properties of nearest neighbor rules using edited data. *IEEE Transactions on Systems, Man and Cybernetics, 3*(2), 408-421.

Wooldridge, M. (2002). An Introduction to MultiAgent Systems. John wiley & sons Ltd.

Xi-Shi, W., & Yong, Z. (September de 1999). Safety Control Techiniques Concerning the Speed Raising on Trunk Lines of Chinese Railway. *Journal of the Eastern Asia Strciety for Trausportation Studies, 3*.

Zadeh, L. (1975). Fuzzy logic and approximate reasoning. *Synthese, 30*(3-4), 407-428.

Zia, S. S., Akhtar, P., Mughal, T. J., & Mala, I. (2014). Case Retrieval Phase of Case-Based Reasoning Technique for Medical Diagnosis. *World Applied Sciences Journal, 32*(3), 451-458.

8 APÊNDICE A

Este apêndice apresenta um comparativo da execução de vários experimentos com o objetivo de analisar o comportamento do algoritmo genético frente no problema desta pesquisa, dadas variações em alguns de seus parâmetros.

O experimento consistiu em configurar o algoritmo genético de acordo com os parâmetros estabelecidos na Tabela 1, e simular uma viagem sobre o trecho ST_1 utilizado nesta pesquisa. A configuração do trem utilizado obedeceu a Configuração 1 da Tabela 25. A base de casos inicial do agente *Memorizador* consistiu dos casos resultantes de cada viagem executada; não foram reutilizados, i.e., não houve reaproveitamento de casos. O reaproveitamento não ocorreu para o que fosse possível analisar apenas o algoritmo genético, sem a influência dos casos gerados. Em suma, estes experimentos seguiram a mesma configuração de trens, casos e via do Cenário A.

Em todos os experimentos a população consistiu de 50 indivíduos e taxa de mutação de 0,5. O número de pais utilizados para geração do próximo filho foi 2. Para compreensão da tabela deve-se utilizar a seguinte legenda:

A. Experimento;
B. Viagem executada;
C. Taxa de mutação;
D. Número de filhos;
E. Número de gerações;
F. Iterações;
G. Acertos;
H. Similaridade;
I. Eficiência da adaptação;
J. Média de eficiência da adaptação.

Tabela 1 – Comparativo entre os resultados de simulações da aplicação de vários parâmetros do Algoritmo Genético.

A	B	C	D	E	F	G	H	I	J
1	1	50%	300	10	3577	1851	86,6%	51,7%	52,4%
	2	50%	300	10	3578	1927	86,1%	53,9%	
	3	50%	300	10	3576	1959	87,4%	54,8%	
	4	50%	300	10	3576	1717	84,5%	48,0%	
	5	50%	300	10	3576	1847	87,9%	51,6%	

	6	50%	300	10	3577	1873	87,2%	52,4%	
	7	50%	300	10	3576	1851	87,5%	51,8%	
	8	50%	300	10	3576	1898	86,9%	53,1%	
	9	50%	300	10	3576	1929	88,3%	53,9%	
2	1	100%	500	10	3576	1805	85,6%	50,5%	52,8%
	2	100%	500	10	3580	1956	88,1%	54,6%	
	3	100%	500	10	3576	2023	86,8%	56,6%	
	4	100%	500	10	3576	1934	86,7%	54,1%	
	5	100%	500	10	3579	1885	88,8%	52,7%	
	6	100%	500	10	3580	1782	84,2%	49,8%	
	7	100%	500	10	3576	1748	85,6%	48,9%	
	8	100%	500	10	3583	1971	88,0%	55,0%	
	9	100%	500	10	3576	1911	88,0%	53,4%	
3	1	50%	500	10	3580	1769	84,4%	49,4%	51,3%
	2	50%	500	10	3577	1860	87,5%	52,0%	
	3	50%	500	10	3580	1822	84,2%	50,9%	
	4	50%	500	10	3576	1865	87,2%	52,2%	
	5	50%	500	10	3576	1785	86,1%	49,9%	
	6	50%	500	10	3576	1830	84,7%	51,2%	
	7	50%	500	10	3580	1944	88,8%	54,3%	
	8	50%	500	10	3578	1765	84,5%	49,3%	
	9	50%	500	10	3580	1875	86,6%	52,4%	
4	1	50%	500	25	3576	1750	83,6%	48,9%	52,1%
	2	50%	500	25	3576	1764	88,6%	49,3%	
	3	50%	500	25	3576	1823	85,6%	51,0%	
	4	50%	500	25	3609	1907	87,9%	52,8%	
	5	50%	500	25	3580	1777	86,2%	49,6%	
	6	50%	500	25	3577	1938	89,0%	54,2%	
	7	50%	500	25	3578	1907	90,0%	53,3%	
	8	50%	500	25	3577	1913	86,9%	53,5%	
	9	50%	500	25	3580	2013	88,0%	56,2%	
5	1	50%	500	10	3576	1896	85,8%	53,0%	51,9%
	2	50%	500	10	3577	2005	89,2%	56,1%	
	3	50%	500	10	3576	1827	87,0%	51,1%	
	4	50%	500	10	3581	1859	84,9%	51,9%	
	5	50%	500	10	3576	1916	88,3%	53,6%	
	6	50%	500	10	3580	1804	84,1%	50,4%	
	7	50%	500	10	3576	1671	81,3%	46,7%	
	8	50%	500	10	3576	1836	86,9%	51,3%	
	9	50%	500	10	3576	1894	85,5%	53,0%	
6	1	50%	500	50	3577	1907	86,9%	53,3%	52,2%
	2	50%	500	50	3576	1900	87,1%	53,1%	
	3	50%	500	50	3576	1863	87,7%	52,1%	

4	50%	500	50	3601	1873	83,7%	52,0%	
5	50%	500	50	3576	1852	87,4%	51,8%	
6	50%	500	50	3579	1830	86,1%	51,1%	
7	50%	500	50	3577	1887	88,0%	52,8%	
8	50%	500	50	3580	1963	85,3%	54,8%	
9	50%	500	50	3576	1757	85,8%	49,1%	

Foi possível observar uma leve vantagem da utilização de um cenário com uma grande quantidade de indivíduos na população (500), visto que este foi o experimento que obteve uma taxa de acerto média superior aos demais (52.8%). Contudo, esta configuração demandava um tempo computacional superior às demais, dada a quantidade de indivíduos. Por este motivo, não optou-se por ela na configuração do algoritmo genético utilizado nos experimentos.

9 APÊNDICE B

Este apêndice apresenta uma cópia do artigo intitulado “An Intelligent System for Driving Trains using Case-Based Reasoning” publicado em 2012 na IEEE International Conference on Systems, Man, and Cybernetics (IEEE SMC 2012) (Borges A. P., et al., An intelligent system for driving trains using Case-Based Reasoning, 2012).

2012 IEEE International Conference on Systems, Man, and Cybernetics
October 14-17, 2012, COEX, Seoul, Korea

An Intelligent System for Driving Trains Using Case-Based Reasoning*

André Pinz Borges, Osmar Betazzi Dordal, Denise Maria Vecino Sato,
Bráulio Coelho Ávila, Fabrício Enembreck, Edson Emílio Scalabrin
Graduate Program in Computer Science
Pontifícia Universidade Católica do Paraná
Curitiba, Brazil
{andre.borges, osmar.dordal, denise, avila, fabricio, scalabrin@ppgia.pucpr.br}

***Abstract*—This paper presents a planning approach using Case-Based Reasoning (CBR) to generate plans for driving trains. The main idea of a planning strategy is to generate a sequence of actions for an agent, which can use these actions to change its environment. CBR allows using prior experiences in the situation assessment task. In the proposed approach, each previous experience (if not applicable) is adjusted resulting in cases specializations. Our interest is reducing the number of corrections triggered when a case retrieved is not applicable, based on these specializations. Experiments showed that the plans generated using this proposed method had a significant increase in the number of cases recovered satisfactorily, also reducing the need of adaptations for the cases recovered.**

Index Terms-Artificial Intelligence, Case-Based Reasoning, Planning, Driving of Trains

I. INTRODUCTION

Many approaches for planning based on artificial intelligence have been proposed. One of them, called case-based planning, uses previous experiences, or cases, to aid in establishing new plans [10]. The idea is that, to solve a particular problem, a person does not plan step by step. Instead, he remembers past similar situations, and adapts them to the new situation [11]. This way, a case-based planner uses past cases to establish new plans. Previous succeeding experiences are used to construct the new plans. One way of implementing this approach is using case-based reasoning [4], where a conductor agent remembers previous situations, similar to the current one, and uses them to help solving the new problem. The case-based reasoning allows the generation of new plans for areas where repeatedly situations occurred, reusing previously elaborated situations and reducing the computational effort to construct new plans.

In modal rail, the train conductor needs to quickly react based on the environment conditions changes in a dynamic road network. To aim such requirement, we propose to reuse previous information of an experienced freight train conductor to develop driving plans. Each plan can be applied in a similar context to the one where the initial experiments were generated. Therefore, it is expected that each plan will drive a train from a start point to an end point of a stretch of the road. The suggested actions should meet the goals of a good conduction policy, which can be expressed in terms of economy (e.g., low fuel consumption), security (e.g., driving respecting the maximum speed allowed) and speed (e.g., in the shortest time possible). It should be emphasized that the previous experiences considered in the experiments contain the best action policies carried out by experienced conductors in a business competition, where the best practices are rewarded monetarily. Each action requires a high knowledge and experience, because a wrong action can damage both the train and the railroad and increase fuel consumption, causing multi-million losses. These costs affect the internal market and the exportations. In this context, an intelligent system for driving trains can generate plans with optimal action policies [6].

Operationally, these experiences are first recorded in a cases base ($BC_{Initial}$) named initial, and later, explored by a case-based reasoned agent [4][8], responsible for generating new driving policies. Here, the experiences recorded in $BC_{Initial}$ represent success stories. Each case includes a context and a decision taken by the conductor in a specific situation. The context is defined by a set of data, such as: kilometer, speed, maximum speed allowed, the current acceleration point, profile of the track for the stretch, train data, among others. Each decision-making can include the implementation of one of the following basic actions: maintaining, increasing, reducing a acceleration point or breaking the train.

The operational basic model of the system follows the standard flow of a case-based reasoner [4]. Given the description of a situation *p*, the effort is concentrated on recovering a similar case from a cases database. In these terms, to improve the recovery process, the internal organization of BC includes two levels of structure:

- Level I: uses a decision tree as an abstracted form of the set of cases for $BC_{Initial}$ and for basic structure selection and navigation in such cases;
- Level II: uses grouping and decision trees to distinguish the specializations of a case and as a way of basic selection structure for one specialization of a case.

Canonically, a decision tree can be obtained by an induction algorithm (e.g., C4.5 [9]). Analogously, a cluster can be obtained from a measure of similarity between cases (e.g., *K-nn*

* This research is supported by Brazilian Federal Agency for Support and Evaluation of Graduate Education. (Capes)

978-1-4673-1714-6/12/$31.00 ©2012 IEEE

[12]). Some algorithms that generate both structures are studied, particularly, in the Machine Learning area ([9][12][16]).

After recovering a case the next steps are: (i) adapt the retrieved case to meet the description of the current problem and (ii) test the adapted case in order to verify its applicability, for example, verify if the implementation of the case will generate the necessary tractor effort to move the train to a new position.

If the adapted case is not applicable, then it will be repaired and the process will return to the adaptation stage. Otherwise, such case will be applied and it becomes a specialization of the original case. It is expected that over time, the combination of adaptation activities, repairs and specializations of a case, will result in a significant improvement in the performance of the cases recovery procedure, also reducing the number of repairs.

In our laboratory, three other policies for driving trains have already been implemented, using respectively Machine Learning, Distributed Constraint Optimization and Specialized Rules [6]. In [2] and [1], the interest was to apply the inductive learning to create a policy similar to the driving policy based on the knowledge of experienced drivers, but without taking future advantages from the experience of the any adaptations applied in any case. In this paper, our interest is to reuse these adaptations. Consequently, it is expected to reduce the number of adaptations needed to generate good policy actions.

This paper contains five main sections. Section 2 starts with an overview of some related works. In Section 3, we present the system proposal. The Section 4 is dedicated to the experimental cases and simulation results, containing some analysis. The conclusion and future works are proposed in Section 5.

II. Related Works

In this section, we present a brief review of related works about planning algorithms on driving systems. In [3], a fast path algorithm for finding the best shortest paths in the road network was compared with the Dijkstra algorithm in order to find the best and shortest paths using Tehran city road network sample. The authors of [7] addressed mission planning for autonomous underwater gliders based on predictions of an uncertain, time-varying current yield. Glider submersibles are highly sensitive to prevailing currents so mission planners must account for ocean tides and eddies. Like us, the authors evaluated plan fragility using empirical tests.

CBR was successfully used in [13], where the authors developed a distributed diagnostic agent system, which detected faults of a device based on signal analysis and machine learning. The CBR techniques presented are used to rind root the cause of vehicle faults based on the information provided by the signal agents. In [15], an approach based on autonomous vehicle guidance using CBR was showed. The CBR was used in order to predict the implication of the current situation and to select the appropriate behavior.

The train driving was addressed in [6], where the authors discuss the results of a software agent development, named *SDriver*, which is able to drive an intercity freight train in a secure, economic and fast way. The *SDriver* performance was evaluated comparing fuel consumption and actions similarity with a real conduction, using a simulated environment.

III. Architecture of the System

In this section, we describe the architecture of the developed intelligent system of trains' driving. One of the requirements of this system is the need of knowing real and current information about the travel team for the plan, which can be acquired by the sensors installed on the locomotive. Other necessary information is the train's profile and the railway details. Such information is necessary because its influence on the way the driving plan is elaborated.

As stated earlier, the driving schedule is prepared using the structure of CBR, which helps reusing previous cases, since the driving plans executed to travel in the same stretch are similar and suitable for reuse. As the cases are retrieved, adapted and evaluated, the driving plans generated, containing a number of cases for drive the train from a source to a destination.

A case is defined as a set of attributes and values of the train and rail. The initial case base ($BC_{Initial}$) is organized using decision trees generated from machine learning techniques. When a new problem is received, it is submitted to the decision tree, which returns a subset of cases with a single acceleration point (*AP*) in common. An *AP* can be compared based on the gear of a car, being responsible for generating the acceleration force required to move the train. The case returned will then be adapted and evaluated according to some metrics. If not applicable, it will be repaired. The corrections made will result in specializations related to the former case and can be used in attempt to increase the accuracy of the process.

A. Definition of Case

A case *C* is a representation of a real world object or episode in a particular representation scheme. A case is represented as a finite set of *n* attribute/value pairs:

$$C = \{a_0 : V_0, ..., < a_n : V_n\} = \{< a_i : V_i >\}_0^n = \{A_i\}_0^n \quad (1)$$

where $<a_i:V_i>$ is an attribute/value pair (A_i). A finite set of cases is called a case base $BC = \{C_1, C_2, ..., C_n\}$. A case represents a snapshot of the situation executed by a train conductor during one travel. The main components of a case are the description and the behavior of the train, and also the profile of the track, both represented by a set of attributes.

A context (Ω) is defined as a finite set of attributes associated with constraints on the attribute values:

$$\Omega = \{a_0 : CV_0, ..., < a_n : CV_k\} = \{< a_i : CV_i >\}_0^k = \{A_i\}_0^k \quad (2)$$

where a_i is an attribute name and the constraint CV_i specifies the set of "allowable" values for a_i.

Figure 1 illustrates, through four levels, the internal organization of the cases base. The first level, level (a), defines a decision tree. Such tree contains a subset of features that sign a case. This subset is determined by calculating the average gain of information, described in [9]. The second level, level (b), represents the subset of cases for which their signatures are the branches of the tree defined on the first level. Besides its signature, each subset is identified by a given label AP_i and have the same acceleration point in common.

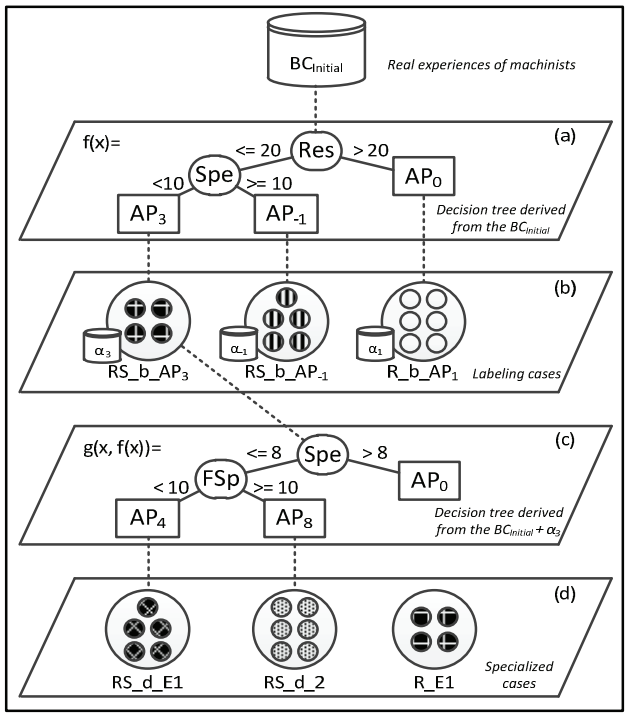


Figure 1. Internal organization of the case base.

The elements of level (a) and level (b) are automatically obtained from the cases base. The decision tree of level (a) is generated by applying a classical algorithm of induction for decision trees (e.g., C4.5 [9]) on the set of cases of $BC_{Initial}$. The subsets of level (b) are obtained from the mapping of each case of $BC_{Initial}$, through the decision tree of level (a), for a label AP_i. In a simplified manner, the mapping is defined by the function *f(x)*, where *x* is any case and *f(x)* is the reference of the subset of the cases for where *x* was mapped. Such grouping is important with higher number of cases.

$$f(\{A_i\}_0^n): AP \in \mathbb{Z} \mid \mathbb{Z} = \{-1, 2, \dots, 8\} \quad (3)$$

The levels (c) and (d) correspond to the structure of the specialization (if any) of each subset of cases of the level (b). Each subset of level (b) can be associated to a set of specializations α_i. In an analogous manner to the levels (a) and (b), the level (c) defines a decision structure for each subset of level (b). However, this structure is simpler because it only maps each specialized case, to a maximum of distinct *j-labels*, where *j* is parameterized. Also, in a simplified manner, it is possible to represent such mapping by the function *g(x, f(x))*, where *x* is a case, *f(x)* is a reference to a subset of cases of the level (b) and *g(x, f(x))* is a reference of subset of cases of the level (d).

$$g(f(\{A_i\}_0^n)): AP \in \mathbb{Z} \mid \mathbb{Z} = \{-1, 2, \dots, 8\} \quad (4)$$

The process of recovering one case involves a multi-level navigation. Taking as an example Figure 1, where the subsets of cases were labeled to simplify the understanding, the steps to take to a new situation is the following: if the nodes of the branch *Res* and *Spe*, representing the attributes Resistance and Speed have been activated, the subset of cases selected is *RS_b_AP3*, as *Res* e *Spe* meets the criteria established by the edges (*Res <= 20 e Spe < 10*). However, as there are specializations for the subset $RS\_b\_AP_3$, the navigation process should continue, considering also the subset of specializations α_3. If the nodes of the branch *Res* e *Spe*, of the level (c), have been activated, so the subset of cases selected is *RS_d_E1*. In summary, the information $RS\_b\_Ap_3$ is obtained by the application of the function *f(x)* and the information *RS_d_E1* is obtained by the application of function *g(x,f(x))*.

The number of cases of the subsets of the level (b) can be defined based on: (i) the similarity with the actions taken by the conductor, as the record of every action is linked to a given interval of time parameterized on the on-board computer of each locomotive; and (ii) the profile of the rail where the train is traveling, as it directly affects the time of use of each point acceleration.

During the execution of a travel, the suggestions made by *f(x),* which are not applicable, are corrected and stored in subset α_i. Such procedure allows determining the specializations shown in the levels (c) and (d) of Figure 1. We believe that the use of specializations of the retrieved case increases the performance of the procedure of recovery cases, reducing the number of repairs. Such an increase may occur as specializations of one case is retrieved are first evaluated. If no specialization is applicable, the case returned for *f(x)* is considered.

The discriminant function *g (x, f (x))*, illustrated in the level (c) of Figure 1, may take different behaviors, for example: (i) inform the most selected cases (*Best-first*), (ii) inform the most similar by grouping (eg, *K-nn*) or decision tree (e.. g., C4.5). The choice of which method to use may be done dynamically, observing the minimum quantity of each discriminant method.

The function *g(x,f(x))* operates as a recovery procedure for a similar case to a new situation described in *x*. Such procedure corresponds to the first task of a CBR system.

B. The Process

The process of execution of the system shown in Figure 2, is based operating cycle of CBR [4][8]. The flow begins with the perception of the environment. In the sequence, the indices are assigned based on information from the train and the track. A similar case or equal to the current situation is restored and adapted to meet the new situation, resulting in a proposed solution. The proposed solution will be evaluated to test its applicability, if applicable, the action will be executed and the case stored, if not applicable, the solution will be repaired.

The general operating cycle of the system, described as Figure 2, begins by executing the process of *Assign Indices*. The input of this process involves three sources of data, which are: profile of the train (number of locomotives, number of wagons, weight train, etc.), profile of the railway (ramps, curves, maximum speeds, etc.) and perception (current position, speed in mi/h, acceleration point (*AP*), etc.). The perception corresponds to a module dedicated to reading speed sensors, position, adhesion, etc. Other values are derived from these sources, which are: resistance (*R*), total tractive effort of all working locomotives in pounds (*TE*), acceleration force (*AF*), adhesion tractive effort (*ATE*), etc. This set of data (read and derivatives) defines, for a new problem situation, the rates required to recover similar cases.

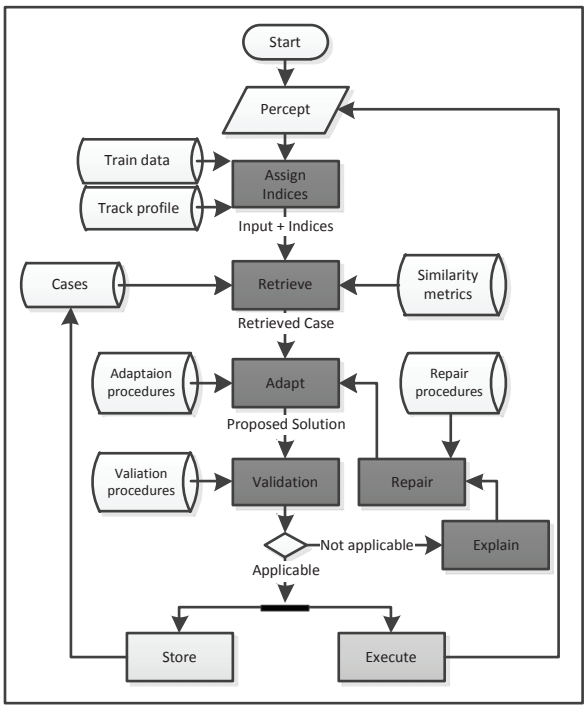


Figure 2. Detailed system flow process.

The *recovery* process cases consists in filtering, from a cases base, a subset of cases that meet the indices determined in the previous process. More formally, filtering cases requires similarity metrics to evaluate how close the current case (new problem situation) is from the recovered case (possible solution). A case retrieved is that which satisfies a context, *sat(C, Ω)*, iff for all pair $<a_i : CV_i> \in \Omega$, exists a pair $<a_i : V_i> \in C$ such that V_i is in CV_i. A case C_1 is similar to a case C_2 for a given context Ω, denoted: $C_1 \sim \Omega\ C_2$, iff C_1 and C_2 satisfy Ω:

$$C_1 \sim \Omega C_2 \leftrightarrow sat(C_1, \Omega) \wedge sat(C_2, \Omega) \quad (5)$$

The function *f(x)* defined in the previous section meets such condition of satisfaction *sat(C, Ω)*.

Figure 3 illustrates part of a recovered case. The pairs *<attribute, value>* the pairs have showed information that will be used as such or adjusted in Adaptation step.

```
C1 = <Speed,0>,<AP,8>,<FinalSpeed,2>,
   <Resistence,978.335>,<MaximumSpeed,60>,
   <E1,6>,<E2,7>,< AF,461021.665>,<TE,462000>, …
```

Figure 3. Example of a recovered case.

The recovered case *C* is adapted (if necessary) by the procedure *Adapt(C)*. In simplified manner, such adaptation involves calculating different values, such as: train resistance (Eq. 7), total tractive effort (Eq. 8), the adhesion tractive effort (Eq. 9), and acceleration force (Eq. 10). The variables used in the equations are presented in Table 1. These different calculations for the definition of a political of moving trains are detailed in [14].

$$Adapt(C,P) = R \wedge TE \wedge ATE \wedge AF \quad (6)$$

$$R = 1.3 \times \frac{29}{w} + b \times SP + \frac{0.0024 \times A \times SP^2}{w} \quad (7)$$

$$TE = \frac{308 \times HP}{SP} \quad (8) \qquad ATE = \frac{W \times f}{1 + (0.01 \times SP_{medium})} \quad (9)$$

$$AF = TE - R \quad (10)$$

Table 1 shows also part of the profile data from a data train and a railway, as well as some data from the perception system. Such data are used to illustrate the adaptation shown in Figure 4.

```
C1' = <Speed,0>,<AP,8>,<FinalSpeed,2>,
     <Resistence,987.481>,<MaximumSpeed,60>,
     <E1,6>, <E2,7>,<AF,230021.665>,<TE,231000>, …
```

Figure 4. Example of adaptation to case C1.

The resistance *R* (Eq. 7) is composed just by normal resistance, because the track profile, described in Table 1, is in level and without curves. Throughout the experiments the resistance will comprise of the sum of all the resistances involved in the displacement of the train, such as curve resistance, ramp resistance and inertial resistance, described in [14].

TABLE I. CONTEXT OF A TRAIN

Attribute	Symbol	Value
Weight (in tons)	W	600
Weight per axle(in tons)	w	27.5 (for locomotives) 13.75 (for wagons)
Coefficient of Adhesion	f	0.22
Experimental coefficient based on flange friction, shock, sway and concussion	b	0.03 (for locomotives) 0.045 (for wagons)
Cross Sectional Area	A	110 (for locomotives) 85 (for wagons)
Speed (in mi/h)	SP	2
Medium Speed (in mi/h)	SP_{medium}	3
Horse Power of all working locomotives	HP	1957.5 for AP = 8

Analyzing the current situation, the case adapted *C1'* is a candidate solution as it needs to be validated before application. Such validation is done by a particular procedure (Eq. 15). This procedure is very simple as it checks if the proposed solution by *C1'* will not result in at least one of the following situations, where *l* is the number of locomotives and AP_{i-1} is the previous acceleration point available: stacking (Eq. 12), lack of strength (Eq. 13), overconsumption (Eq. 14).

$$Test(C) = Stacking(C) + LackOfStrength(C) + Overconsumption(C) \quad (11)$$

$$Stacking\ (C) = \begin{cases} 1, & if \quad TE > (ATE \times l) \\ 0, & \text{otherwise.} \end{cases} \quad (12)$$

$$LackOfStre\ nght\ (C) = \begin{cases} 3, & if \quad AF \le 0 \\ 0, & \text{otherwise.} \end{cases} \quad (13)$$

$$Overconsumption(C) = \begin{cases} 7, & if \quad AF(AP_{i-1}) > 0 \\ 0 & \text{otherwise.} \end{cases} \quad (14)$$

$$Validation\ (C) = \begin{cases} Sucess, & if \quad Test(C) = 0 \\ Explain\ \ (Test(C)) & if \quad Test(C) > 0 \end{cases} \quad (15)$$

An explanation should be provided when the proposed solution is not considered appropriate. This explanation is generated by a particular procedure (Eq. 16).

$$Explain\ (x) = \begin{cases} \{stacking\}, & if \quad x = 1 \\ \{lack\ of\ strength\}, & if \quad x = 3 \\ \{overconsumption\}, & if \quad x = 7 \\ \{stacking, lack\ of\ strength\}, & if \quad x = 4 \\ \{stacking, overconsumption\}, & if \quad x = 8 \\ \{lack\ of\ strength, overconsumption\}, & if \quad x = 10 \\ \{stacking, lack\ of\ strength, overconsumption\}, & if \quad x = 11 \end{cases} \tag{16}$$

A repair must be provided when the value produced by the validation procedure is different of *Success*. Such repair is suggested by a particular procedure (Eq. 17). For the example, this procedure is very simple, as it indicate if the current acceleration point must be decremented (1-) or increased (1+) of a unit. The following lists of explanations: {*stacking, lack of strength*}, {*lack of strength, overconsumption*}, {*stacking, lack of strength, overconsumption*}, are not eligible for reparation as they have contradictions.

$$Repair\ (X) = \begin{cases} 1-, & if \quad X \subseteq \{stacking, overconsumption\} \\ 1+, & if \quad X \subseteq \{lack\ of\ strength\} \\ contradiction, & otherwise \end{cases} \tag{17}$$

If the proposed solution is applicable, two procedures are executed: the execution of the application of the solution and the storage of this solution as a new case applied in the base case. Following, a new perception is accomplished and the flow just described is repeated.

It is important to note that, as new cases are stored, the functions *f(x)* and *g(x, f(x))* can be modified to consider the new experiences. These changes should result in restructuring the $BC_{Initial}$ and impact both in the recovery task and in the task of indexing cases. The expectation is that in both tasks, the performance will be increased. However, despite obvious, such restructuring of $BC_{Initial}$ has not been developed here.

IV. Experimental Results

In this section, simulation experiments have been carried out. The results shown here refer to experiments in the laboratory. The experiments were made possible by the developed computational models that allow a similar travel with different characteristics. We analyzed eight different configurations of travel. These settings were made by experienced conductors and for our driving simulator, with different train profiles: for trains consisting of 2, 3 or 4 locomotives, and train weight varying from 3393 to 6579 tons. Each locomotive has 8 acceleration points (*AP*) represented by a set of discrete values *AP* = {-1, 0, 1, 2, 3, 4, 5, 6, 7, 8}. The vehicle acceleration is made employing an *AP*. The greater the *AP*, greater will be the power generated by the locomotive. The *AP* -1 indicates application of brake. For us, the *AP* applied will be ever the same for all the locomotives.

The $BC_{Initial}$ was generated from real conductions made by experienced train conductors. The number of cases was approximately 14.000. Each case has 113 attributes. They represent the configuration data of the train, the profile of the railway and the data read from the sensors. Since it is too large, the base case was initially organized in the form of decision trees. Due to lack of computational resources to generate the decision tree with all the cases, the decision trees were generated under the following conditions:

- *Sample Size (SS)* of 10, 20 and 30% of initial set of cases.
- *Use of Cross Validation (CV)* to generate samples used during the stages of obtaining and testing of the decision tree.
- *Travels with and without stops:* over a travel, have been observed sections where the speed of the train was equal to 0. As the on-board computer registers an instance every 5 seconds, this results in similar records and maybe useless. Thus, we decided to retrieve cases with and without intermediate stops to consider whether this information influenced the recovery process of the cases.

The experiments were analyzed considering the following aspects: run time, hit rate in the selection of the case to be applied facing a new situation.

The first aspect analyzed was the runtime system. The simulation time of the travels, when executed to retrieve the best specialization for each case, was greater than the run time to retrieve a case without regard to their specializations. This was accomplished due to the fact that amount of bug fixes (new cases) increased significantly with each new simulated travel. The increased volume of data was the reason for the increase in recovery time.

Different methods were used to evaluate the recovery process of specializations of cases. Each cell in Table 2 shows the results from four different methods: without specializations and with specializations (*Best-First*, *K-nn* e C4.5), according to the scheme of Figure 5. In the experiments we considered only two specializations with more fixes and *k* equals 3 to *K-nn*.

Without Specilization	*Best-First*
K-nn	C4.5

Figure 5. Behaviors of the function *g(x, f(x))*.

Analyzing the number of adaptations made in the selection of the case to be applied against a new situation, all the approaches that made use of the procedure specialization obtained better results. The average hit rate of the cases recovered without specialization was 12.19%. Moreover, the specialization using the *Best-First* obtained a hit rate of average around 51.78%, also be performed more quickly. The approach using *K-nn* obtained an average hit rate around 37.49% and C4.5 obtained an average hit rate around 35.25%. The highest hit ratio occurs as the method *Best-First* returns the two most used cases and test the application of both. On the other hand, the approaches *K-nn* and C4.5 suggest only one case. This reduces the chance of success of a recovered case.

We expected that, as new travel have been made and the base of cases have been enhanced, the number of adjustments involving repairs increased for some time and then remain stable or even or decrease. However, experiment showed that this did not occur. With the increase of the number of cases, the rates of success in recovery of cases varied from travel to travel. It is believed that this did not occur because the decision trees used in the recovery of cases have been generated only

TABLE II. RESULTS

		Percentile of hit of recovered cases															
SS	**Stops?**	**Travel 01**		**Travel 02**		**Travel 03**		**Travel 04**		**Travel 05**		**Travel 06**		**Travel 07**		**Travel 08**	
10	Y	16.11	55.21	13.13	54.02	15.15	50.37	12.88	50.87	14.92	43.09	14.18	48.36	15.90	55.18	12.08	49.16
		35.46	31.51	39.19	34.95	34.7	27.62	37.44	36.24	28.85	23.38	30.33	33.09	33.72	31.81	33.26	30.83
10	N	9.47	55.68	9.11	58.26	11.98	49.63	6.33	58.07	7.88	38.02	5.45	54.18	8.23	51.98	6.87	49.37
		49.75	45.02	52.11	48.51	42.96	39.11	50.23	49.12	32.39	34.08	41.57	47.27	42.30	44.60	48.08	41.04
20	Y	9.23	56.81	11.93	52.27	11.3	51.75	12.66	59.82	11.97	42.89	11.63	51.85	10.42	52.95	13.79	47.87
		45.32	42.95	43.38	33.18	40.55	37.68	40.93	38.64	31.84	29.52	31.83	36.36	38.52	40.56	41.97	36.51
20	N	12.08	54.73	13.34	54.87	11.73	45.72	15.50	57.07	11.83	45.48	14.54	49.63	17.61	56.53	12.08	53.15
		34.69	36.49	38.13	35.59	28.15	24.93	36.71	35.58	26.42	27.68	31.22	35.63	36.54	35.79	33.80	34.79
30	Y	5.81	50.93	6.94	51.84	5.28	49.51	5.96	57.39	5.84	42.89	25.81	50.90	7.32	49.01	4.66	49.08
		47.3	39.81	44.46	34.92	40.25	34.37	42.99	40.17	36.28	30.64	40.29	40.87	37.98	36.05	42.02	35.90
30	N	16.58	58.05	17.79	50.63	18.09	55.16	20.52	56.22	18.87	48.16	6.18	50.56	19.88	57.81	18.12	52.54
		37.64	33.64	31.35	29.87	36.7	28.85	30.76	31.95	26.47	25.07	29.73	29.58	30.11	26.84	32.91	33.40

with driving data of conductors. In other words, the decision trees were not updated at the end of each travel to take as into consideration the new experiences. In future experiments, new travels will be implemented and new decision trees will be generated to include the new experiences in the recovery process, to verify if the number of adaptations followed by repairs will reduce linearly.

V. CONCLUSION AND FUTURE WORK

This paper presented the use of CBR in the process of conducting intercity freight trains. The system allowed that cases of travel made by good conductors have been recovered from, adapted and tested on simulated travels. The main objective was to evaluate the real contribution of the use of machine learning techniques in the retrieval of cases. We analyzed also if the specialization of cases provided improvements in the recovery process of cases and in the preparation of plans. Experimental results showed a significant increase in performance of the recovery procedure of cases when used techniques that allowed the selection of specialized cases. Thus, the driving plans have been generated with a smaller number of cases repairs.

Currently, the planner is able to recover only one case at a time. In future study, will be considered the recovery of sets of cases, involving different techniques of decision making, allowing them to be retrieved multiple actions of specific stretches of track. Thus, it is expected reduce further the computational efforts employed in the preparation of travel plans.

In future, the specialized cases may also be considered during the generation of decision trees used in the recovery of the cases - the level (a) of Figure 1. As the cases have been specialized, decision trees used in the recovery of the cases can also be updated based on information from specializations. Thus, new specializations can be generated for cases even better. We believe that such way can allow that recovery process of cases can obtain even better performance.

REFERENCES

[1] A. P. Borges; J. Granatyr, O. B. Dordal, R. Ribeiro; B. C. Ávila; F. Enembreck; E. E. Scalabrin, "Knowledge Discovery Applied in Modal Rail". Proceedings of the 2011 15th International Conference on Computer Supported Cooperative Work in Design, pp. 253-260, 2011.

[2] A. P. Borges; R. Ribeiro; B. C. Ávila; F. Enembreck; E. E. Scalabrin, "A Learning Agent to Help Drive Vehicles". Proceedings of the 2009 13th International Conference on Computer Supported Cooperative Work in Design, pp. 282-287, 2009.

[3] A. Selemat, M. Z-Arokhlo, S. Z. Hashim and M. H. Selemat. "A Fast Path Planning Algorithm for Route Guidance System". Proceedings of 2011 IEEE International Conference on Systems, Man, and Cybernetics (SMC), pp. 2773-2778, November 2011.

[4] C. Riesbeck and R. C. Schank. "Inside Case-Based Reasoning". Lawrence Erlbaum Associates Publishers, New Jersey, 1989.

[5] D. M. Gaines, T. Estlin, F. Fisher, C. Chouinard, R. Castaño and R. C. Anderson. "Planning for Rover Opportunistic Science". International Workshop on Planning and Schedule for Space, Darmstadt, Germany, pp. 1-10, June, 2004.

[6] D. M. Sato, A. P. Borges, A. R. Leite, O. B. Dordal, B. C. Ávila, F. Enembreck and E. E. Scalabrin. "Lessons Learned from a Simulated Environment for Trains Conduction". Proceedings of IEEE International Conference on Industrial Technology, March, 2012. *(to appear)*

[7] D. R. Thompson, S. Chien, A. Balasurayia, S. Petillo, Y. Chao, P. Li, B. Cahill, J. Levin, M. Meisinger, M. Arrott, and O. Schofield. "Spatiotemporal Path Planning in Strong, Dynamic, Uncertain Currents". Proceedings of the IEEE International Conference on Robotics and Automation, 2010.

[8] J. L. Kolodner. "Case-Based Reasoning". Morgan Kaufmann, San Mateo, CA, 1993.

[9] J. R. Quinlan. "C4.5. Programs for machine learning". Morgan Kaufman, San Francisco, 1993.

[10] J.K. Hammond. "Case-Based Planning: Viewing Planning as a Memory Task". Academic Press, Cambridge, Massachusetts, 1989.

[11] L. Spalazzi. "A Survey on Case-Based Planning". Artificial Intelligence Review, Vol 16, Issue 1, pp. 3-36, September, 2001.

[12] P-N. Tan, M. Steinbach, V. Kumar. "Introduction to data mining". Pearson Addison-Wesley, 2006.

[13] R. McCartney. "Case-based planning meets de frame problem". International Conference on Artificial Intelligence Planning Systems, San Mateo, Morgan Kaufmann, pp. 172-178, 1992.

[14] S. Iwnicki, Handbook of Railway Vehicle Dynamics., CRC Press, Taylor & Francis Group, 2006.

[15] S. Vacek, T. Gindele, J. M. Zollner and E. Dillmann. "Using case-based reasoning for autonomous vehicle guidance," Proceedings of International Conference on Intelligent Robots and Systems, pp.4271-4276, November 2007.

[16] W. Cohen. "Fast Effective Rule Induction". In Proceedings of Twelfth International Conference on Machine Learning, pp. 115-123, 1995

[17] W. Ziyan, J. Crossman, J. Cardillo and Y. L. Murphey. "Case-base reasoning in vehicle fault diagnostics," Neural Networks, 2003. Proceedings of the International Joint Conference on , vol.4, pp. 2679-2684, July 2003.