ALLAN RODRIGO LEITE

UM ESQUEMA PARA REDUÇÃO DO CONSUMO DE COMBUSTÍVEL EM SISTEMAS DE CONDUÇÃO FÉRREA BASEADO EM OTIMIZAÇÃO DISTRIBUÍDA DE RESTRIÇÃO

Dissertação apresentada ao Programa de Pós-Graduação em Informática da Pontifícia Universidade Católica do Paraná como requisito para obtenção do título de mestre em Informática.

Curitiba

Julho/2009

ALLAN RODRIGO LEITE

UM ESQUEMA PARA REDUÇÃO DO CONSUMO DE COMBUSTÍVEL EM SISTEMAS DE CONDUÇÃO FÉRREA BASEADO EM OTIMIZAÇÃO DISTRIBUÍDA DE RESTRIÇÃO

Dissertação apresentada ao Programa de Pós-Graduação em Informática da Pontifícia Universidade Católica do Paraná como requisito para obtenção do título de mestre em Informática.

Área de Concentração: Agentes de Software

Orientador: Dr. Fabrício Enembreck

Curitiba

Julho/2009

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

Leite, Allan Rodrigo
L533e Um esquema para redução do consumo de combustível em sistemas de
2009 condução férrea baseado em otimização distribuída de restrição / Allan Rodrigo Leite ; orientador, Fabrício Enembreck. – 2009.
79 f. : il. ; 30 cm

Dissertação (mestrado) – Pontifícia Universidade Católica do Paraná, Curitiba, 2009
Bibliografia: f. 72-74

1. Inteligência artificial distribuída. 2. Restrições (Inteligência artificial). 3. Transporte ferroviário. I. Enembreck, Fabrício. II. Pontifícia Universidade Católica do Paraná. Programa de Pós-Graduação em Informática. III. Título.

CDD 20. ed. – 006.3

Pontifícia Universidade Católica do Paraná
Centro de Ciências Exatas e de Tecnologia
Programa de Pós-Graduação em Informática

ATA DE DEFESA DE DISSERTAÇÃO DE MESTRADO
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

DEFESA DE DISSERTAÇÃO Nº 19/2009

Aos 10 dias do mês de julho de 2009 realizou-se a sessão pública de Defesa da Dissertação **"Um Esquema para Redução do Consumo de Combustível em Sistemas de Condução Baseado em Otimização Distribuída de Restrição"**, apresentada pelo aluno **Allan Rodrigo Leite** como requisito parcial para a obtenção do título de Mestre em Informática, perante uma Banca Examinadora composta pelos seguintes membros:

Prof. Dr. Fabrício Enembreck PUCPR (Orientador)	_____ *(assinatura)*	APROVADO *(aprov/reprov.)*
Prof. Dr. Edson Emílio Scalabrin PUCPR	_____	Aprovado
Prof. Dr. Felix Francisco Ramos Corchado CINVESTAV – IPN	_____	APROVADO

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. Dr. Mauro Sérgio Pereira Fonseca
Diretor do Programa de Pós-Graduação em Informática

Programa de Pós-Graduação em Informática PPGIa PUCPR

Rua Imaculada Conceição, 1155 Prado Velho CEP 80215 901 Curitiba Paraná Brasil
Tel.: (41) 3271 1669 Fax: (41) 3271 2121 www.ppgia.pucpr.br

Dedico este trabalho à minha família, minha fonte de inspiração e determinação.

AGRADECIMENTOS

Agradeço ao meu pai, mãe e irmão (Luiz, Silvia e Ricardo), pelo amor incondicional, carinho e dedicação. Sou afortunado por pertencer a esta família.

Ao meu amor, Ingrid, pela paciência e incentivo para conclusão desta etapa em minha vida. Sem seu apoio, este sonho não seria concretizado.

Ao meu orientador, professor Dr. Fabrício Enembreck, pela excepcional paciência, dedicação e por acreditar em meu potencial. Um pesquisador ímpar, seu conhecimento e suas idéias foram fundamentais para o desenvolvimento deste trabalho.

Aos professores Dr. Edson Emílio Scalabrin e Dr. Bráulio Coelho de Ávila pelas inúmeras contribuições valiosas a este trabalho.

Aos amigos do Laboratório de Agentes de Software, Bruno, Osmar, Richardson e, em especial, ao André, pelo companherismo e pelos momentos de descontração.

Ao Henrique e à Rosangela, pessoas que amo e que tenho como parte de minha família.

À CAPES, pelo subsídio financeiro para realização deste trabalho.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SÍMBOLOS

ADOPT	ASYNCRONOUS DISTRIBUTED OPTIMIZATION
ALL	AMÉRICA LATINA LOGÍSTICA
BB	BRANCH AND BOUND
CP	CONSTRAINT PROGRAMMING
CSP	CONSTRAINT SATISFACTION PROBLEM
DCSP	DISTRIBUTED CONSTRAINT SATISFACTION PROBLEM
DCOP	DISTRIBUTED OPTIMIZATION PROBLEM
DPOP	DYNAMIC PROGRAMMING OPTIMITATION
IAD	INTELIGÊNCIA ARTIFICIAL DISTRIBUÍDA
LTKB	LITROS POR TONELADA BRUTA TRANSPORTADA
NCBB	NO-COMMITMENT BRANCH AND BOUND
OPTAPO	OPTIMAL ASYNCHRONOUS PARTIAL OVERLAY
SMA	SISTEMA MULTI-AGENTE

LISTA DE ALGORITMOS

RESUMO

Sistemas multi-agente representam um novo paradigma de análise, projeto e implementação de sistemas complexos. Uma das dificuldades encontradas durante a concepção de um sistema multi-agente é a definição do modelo de coordenação, devido à característica de autonomia dos agentes. A otimização distribuída de restrição tem emergido como uma das principais técnicas de coordenação em sistemas multi-agente, devido a capacidade de otimizar a função objetivo global do problema modelada como um conjunto de restrições. Este trabalho propõe a utilização da otimização distribuída de restrição para alcançar a eficiência energética no transporte ferroviário. O objetivo deste trabalho é demonstrar que técnicas de otimização distribuída de restrição podem ser aplicadas para obter políticas ótimas de condução de composições férreas.

Palavras-chave: Inteligência Artificial; Sistemas Multi-agente; Otimização Distribuída de Restrição

ABSTRACT

Multi-agent systems represent a new paradigm of analysis, design and development of complex systems. Due to characteristic of autonomy of the agents, one difficulty found during the construction of a multi-agent system is the coordination model. The distributed constraint optimization has emerged as one of key coordinate techniques in multi-agent systems, due to ability to optimize the global objective function of the problem modeled as a constraint set. This work proposes to employ techniques of distributed constraint optimization to reach the energetic efficiency on railroad transport. The main objective of this work is to demonstrate that techniques of distributed constraint optimization can be applied to achieve optimal driving strategies on railroad transport.

Keywords: Artificial Intelligence; Multi-agent Systems; Distributed Constraint Optimization.

Capítulo 1

Introdução

Atualmente temas sobre a eficiência energética no transporte são objetos de estudo de diversos pesquisadores. Contudo, grande parte destas pesquisas objetiva a produção de veículos com consumo eficiente de combustível, de modo a se adequar às normas das agências de proteção ambiental e apresentar diferencial competitivo de mercado. No entanto, a eficiência energética também está relacionada com a perícia do operador em conduzir o veículo. Em termos gerais, o operador deve tomar suas ações considerando um grande número de variáveis e restrições, tais como tempo de viagem, segurança, clima, desgaste dos componentes do veículo e da via permanente, dentre outras. A eficiência energética no transporte ferroviário, tema que constitui o propósito deste trabalho, também permanece proporcional à habilidade do maquinista em conduzir a composição férrea, apesar dos avanços tecnológicos do setor.

De uma maneira geral, as locomotivas são equipadas com computadores de bordo para auxiliar o maquinista na condução da composição férrea. Um dos objetivos dos computadores de bordo é coletar dados em tempo real sobre uma viagem e apresentá-los ao maquinista para tomar suas ações a partir destas informações. Entretanto, estes sistemas embarcados poderiam ser dotados de inteligência de modo a auxiliar o maquinista durante as tomadas de decisão, ao invés de apenas fornecer dados sobre a viagem. Tal aspecto amplia a contribuição destes sistemas embarcados, elevando-os da condição de software de nível operacional para estratégico.

1.1 Motivação

Nos últimos anos o preço dos combustíveis derivados do petróleo sofreu um aumento vertiginoso no Brasil e no exterior. Segundo a Agência Nacional do Petróleo, Gás Natural e Biocombustíveis (ANP), entre os anos de 2001 e 2007 houve um reajuste de aproximadamente 123% no preço final do combustível diesel (ANP, 2008). Este crescimento afeta diretamente o transporte ferroviário, uma vez que o consumo de diesel combustível constitui sua principal fonte do custo operacional.

A América Latina Logística (ALL), grupo que detém a concessão das malhas ferroviárias da região sul e sudoeste do Brasil, apresentou um custo de 423,9 milhões de reais com combustível no ano de 2007 (ALL, 2007). Como conseqüência, o custo operacional do transporte inflaciona o preço dos fretes, que por sua vez impacta no preço final dos produtos. Tal evento prejudica os mercados interno e externo que dependem do transporte, tornando-os menos competitivos. Além do aumento dos preços dos combustíveis, o aquecimento global e a escassez das reservas de petróleo são realidades que não podem mais ser ignoradas.

Apesar da existência de inúmeros trabalhos propondo estratégias para alcançar a eficiência energética em sistemas ferroviários, eles permanecem incompletos ou insuficientes devido às restrições impostas pelo desenvolvimento de tais modelos. Em face a estes fatos, novas soluções para aumentar a eficiência energética do transporte ferroviário são imprescindíveis.

1.2 Hipótese

O presente trabalho propõe a concepção de um sistema de condução férrea eficiente no que concerne o consumo de combustível. Acredita-se que a aplicação de métodos da Inteligência Artificial Distribuída, sobretudo técnicas de otimização distribuída de restrição, aliado aos modelos de redução do consumo de combustível existentes proporcione soluções satisfatórias para o problema. A adoção de técnicas de otimização distribuída de restrição como estratégia para reduzir o consumo de combustível se deve ao fato de que este formalismo tem se mostrado um excelente método para resolução de problemas complexos de otimização. Inúmeros problemas de otimização em sistemas multi-agente podem ser modelados através deste formalismo, tais como planejamento, escalonamento e alocação de recursos. Portanto, a otimização distribuída de restrição investiga estratégias de como os agentes podem coordenar suas decisões, de modo que os agentes alcancem um estado estável por meio da otimização da função objetivo global definida para o problema.

1.3 Objetivos

O propósito deste trabalho é demonstrar que técnicas de otimização de restrição podem ser utilizadas para obter políticas ótimas de conduções férreas. A proposta deste trabalho compreende em formular o problema de redução do consumo de combustível

em sistemas de condução férrea como um problema de otimização distribuída de restrição (DCOP), o qual implementa um modelo de consumo de energia inspirado em Chang e Howllet (1995) a ser apresentado no capítulo 2.

Deste modo, o objetivo principal deste trabalho consiste em modelar o problema de condução de composições férreas como um DCOP, comparar o desempenho dos principais algoritmos para DCOP existentes e avaliar as soluções obtidas a partir da formulação proposta. Fatores como qualidade da solução, tempo de execução e recursos requeridos (processamento e de comunicação) são considerados na avaliação da formulação proposta. Portanto, os objetivos específicos deste trabalho são:

- Modelar o problema condução de composição férrea como um DCOP;
- Desenvolver um framework para DCOP contendo os principais algoritmos deste formalismo e comparar seus desempenhos para o cenário abordado;
- Propor um módulo de controle inteligente para sistemas de condução férrea que priorize a eficiência energética e que seja compatível com as exigências dos computadores de bordo de locomotivas; e
- Avaliar a qualidade das soluções obtidas a partir do módulo de controle inteligente proposto.

1.4 Contribuições

As contribuições científicas do presente trabalho são: (i) propor uma modelagem DCOP para um problema complexo do mundo real; e (ii) a concepção de um módulo de controle inteligente para sistemas de condução férrea. O módulo de controle inteligente é formado por um sistema multi-agente cuja estratégia adotada é baseada em otimização distribuída de restrição. Esta estratégia foi capaz de obter uma substancial redução do consumo de combustível em relação às viagens operadas por maquinistas que não dispõem de recursos para auxiliá-los durante a condução.

1.5 Estrutura do Documento

O presente trabalho é composto por seis capítulos, no qual o primeiro fornece uma introdução onde são contextualizados o problema abordado, as motivações, a hipótese e objetivos para este projeto. Os capítulos subseqüentes estão organizados conforme descrito a seguir:

- **Capítulo 2 – Fundamentos sobre a Eficiência Energética em Sistemas de Condução Férrea**: este capítulo apresenta uma introdução sobre os sistemas de condução férrea e as dificuldades existentes para alcançar a eficiência energética. Também são apresentados alguns trabalhos previamente realizados sobre o tema;
- **Capítulo 3 – Fundamentos sobre Problemas de Otimização Distribuída de Restrições**: o objetivo deste capítulo é fornecer um embasamento teórico quanto ao estado da arte sobre problemas de otimização distribuída de restrição, bem como apresentar uma discussão sobre os principais algoritmos deste formalismo;
- **Capítulo 4 – Um esquema para Redução do Consumo de Combustível em Sistemas de Condução Férrea Baseado em Otimização Distribuída de Restrição**: o capítulo quatro descreve o modelo proposto por este trabalho para alcançar a eficiência energética em sistemas de condução férrea. Para tal, foi desenvolvido um sistema multi-agente cuja estratégia adotada para reduzir o consumo de combustível faz uso de técnicas de otimização distribuída de restrição;
- **Capítulo 5 – Avaliação e Discussão sobre os Resultados Obtidos**: por sua vez, este capítulo descreve a metodologia de avaliação aplicada e os resultados obtidos pelo modelo proposto. Além da avaliação dos resultados obtidos, foi realizado um comparativo em termos de desempenho entre os principais algoritmos para DCOP; e
- **Capítulo 6 – Conclusões**: por fim, o último capítulo apresenta as considerações finais e sugestões para trabalhos futuros.

Capítulo 2

Fundamentos sobre a Eficiência Energética em Sistemas de Condução Férrea

Este capítulo apresenta uma contextualização sobre sistemas de condução férrea, bem como as dificuldades existentes para alcançar a eficiência energética neste meio de transporte. Os sistemas ferroviários aqui abordados são voltados para transportes de carga baseados em locomotivas diesel-elétricas.

2.1 Sistemas de Condução Férrea

Sistemas ferroviários são meios de transporte guiados, isto é, a composição férrea tem sua mobilidade limitada à direção do trilho. Devido ao volume de carga transportada, tanto as rodas da composição quanto os trilhos são compostos de materiais metálicos, em geral feitos de aço, a fim de minimizar o desgaste destes elementos. As rodas das locomotivas e dos vagões são apoiadas sobre os trilhos, que possuem uma distância definida. Esta distância entre os trilhos é chamada de bitola, sendo passível de pequenas variações ao decorrer da via. Para a composição não descarrilar, as rodas contém pequenos frisos em suas extremidades que permitem uma suave movimentação lateral do veículo sobre os trilhos.

Devido às características dos materiais das rodas da composição e dos trilhos, o atrito produzido pelo contato destes elementos é menor que os demais meios de transporte terrestre. Como conseqüência, o transporte ferroviário é mais econômico, pois necessita menos esforço trator para percorrer um determinado trecho plano e em reta. Entretanto, os sistemas férreos são mais sensíveis à geometria da via, sendo necessária a adoção de sistemas de curvas e rampas suaves. Outra vantagem do transporte ferroviário em relação aos demais meios de transporte terrestre é a flexibilidade para ajustar a composição férrea de acordo com as peculiaridades do trajeto ou com o volume de carga a ser transportada. Tal característica é conhecida na literatura como lotação do trem, que consiste em dimensionar a quantidade de esforço trator necessário para um determinado trecho do trajeto.

A força necessária para uma composição férrea com uma dada lotação percorrer um determinado trecho da via deve ser maior que sua resistência total ao movimento. Em locomotivas diesel-elétricas, um motor movido a combustão atua como um gerador de energia alimentando os motores elétricos de tração, produzindo um esforço trator contínuo e potência contínua. Por sua vez, o maquinista controla a força tratora transferida aos eixos da locomotiva através dos pontos de aceleração (análogo às marchas de automóveis). Conduzir uma composição férrea exige um significativo grau de conhecimento e experiência do maquinista, pois a seleção indevida de um ponto de aceleração em relação ao trecho da via pode danificar tanto a locomotiva quanto os trilhos da ferrovia. A Figura 1 ilustra o esforço trator (F) em função dos pontos de aceleração (P_1, P_2 e P_3) e da velocidade (V). É possível observar que o esforço trator produzido é inversamente proporcional à velocidade. Outro aspecto relevante demonstrado pela Figura 1 é a velocidade crítica e a velocidade limite, que corresponde respectivamente às situações onde a locomotiva possui baixa velocidade com um elevado torque e quando a locomotiva atinge a velocidade máxima.

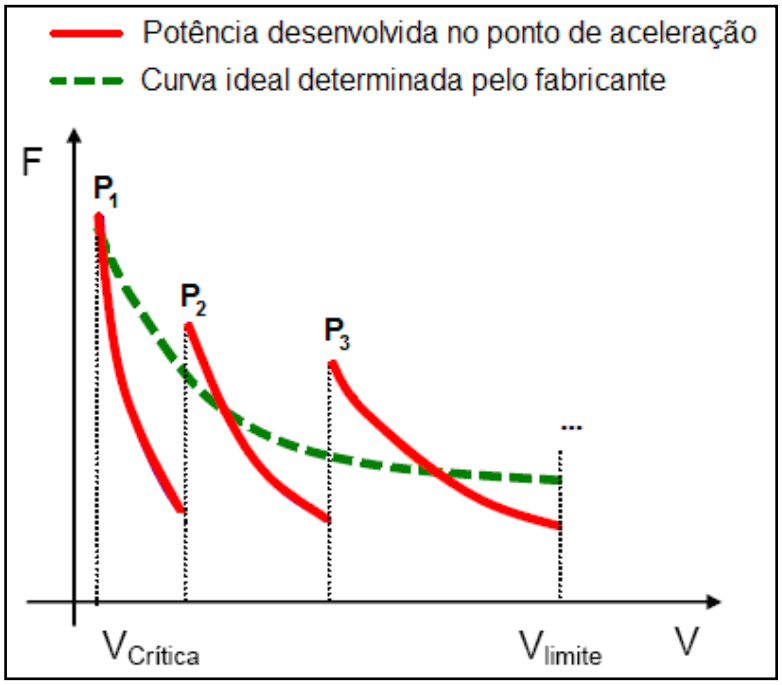

Figura 1. Esforço trator em função do ponto de aceleração e velocidade.

O esforço trator é calculado a partir da potência efetiva da locomotiva produzida em um determinado ponto de aceleração e da velocidade atual da composição. A Equação 1 descreve uma das maneiras para obter do esforço trator (BRINA, 1983).

$$F_e = \frac{273{,}24\ W_{HPe}}{V} \quad (1)$$

Onde:

W_{HPe} = Potência efetiva da locomotiva em HP

V = Velocidade em km/h

F_e = Esforço trator efetivo em kgf

Cada ponto de aceleração possui uma respectiva potência associada a um consumo, conforme apresentado pela Tabela 1. Desta forma, uma redução no consumo de combustível pode ser obtida ao selecionar o ponto de aceleração mais adequado para o trecho atual da ferrovia. Entretanto, para promover uma estratégia ótima de seleção dos pontos de aceleração das locomotivas, é necessário conhecer diversas informações referentes ao movimento da composição, tais como esforço necessário para percorrer um dado trecho da ferrovia e o limite de aderência entre os trilhos e a composição. Estas informações são obtidas a partir de modelos matemáticos fornecidos pela dinâmica do movimento de composições férreas.

Tabela 1. Consumo em função do ponto de aceleração para locomotivas modelo C30.

Ponto de aceleração	**Potência (HP)**	**Consumo (l/min)**
Lenta baixa	0	0,3168
Marcha lenta	0	0,3168
Freio dinâmico	0	1,767
1	100	0,567
2	275	1,0668
3	575	1,95
4	960	3,033
5	1440	4,533
6	1930	6,183
7	2500	7,6998
8	2940	9,4002

2.2 Dinâmica do Movimento

A dinâmica do movimento pode ser dividida em três grandes sessões: os esforços resistentes, a força de aderência e as variáveis do movimento referentes ao sistema de tração (PIRES, 2005). Estas três sessões, quando combinadas, permitem realizar simulações matemáticas do movimento de composições férreas. De uma maneira geral, sobretudo para cálculos dos esforços resistentes, as expressões

matemáticas compreendidas pela dinâmica do movimento são de cunho empírico, em virtude da complexidade envolvida em tais fenômenos físicos.

2.2.1 Esforços resistentes

São chamados de esforços resistentes todas as forças que se opõe ao movimento da composição férrea (PIRES, 2005). Os esforços resistentes agem sobre o ponto de contato entre as rodas da composição e os trilhos, podendo ser classificados em duas categorias: resistências normais e resistências acidentais. As resistências normais são permanentes, isto é, esta força sempre atuará sobre cada veículo da composição. Em contrapartida, as resistências acidentais são ocasionais, atuando em função da geometria da via (curvas e rampas). A inércia também é considerada como resistência acidental, agindo em virtude da alteração do estado de repouso ou durante a aceleração da composição férrea.

A resistência normal é gerada através da resistência do rolamento das rodas sobre os trilhos e o atrito nos mancais do eixo (BRINA, 1983). Existem diversas equações experimentais para cálculo da resistência normal, porém, serão apresentadas a seguir as fórmulas propostas por W. L. Davis Jr., por serem amplamente aceitas pela comunidade científica da área (BRINA, 1983). Como cada fórmula elaborada por Davis se aplica a uma classe específica de veículo ferroviário, serão apresentadas apenas as fórmulas destinadas a veículos de carga.

Locomotiva com peso por eixo superior a 5 toneladas (Equação 2):

$$R_n = 1{,}3 + \frac{29}{w} + 0{,}03\,V + \frac{0{,}0024\,A\,V^2}{w\,n} \tag{2}$$

Locomotiva com peso por eixo inferior a 5 toneladas (Equação 3):

$$R_n = \frac{9{,}4}{\sqrt{w}} + \frac{12{,}5}{w} + 0{,}03\,V + \frac{0{,}0024\,A\,V^2}{w\,n} \tag{3}$$

Vagão com peso por eixo superior a 5 toneladas (Equação 4):

$$R_n = 1{,}3 + \frac{29}{w} + 0{,}045\,V + \frac{0{,}0005\,A\,V^2}{w\,n} \tag{4}$$

Vagão com peso por eixo inferior a 5 toneladas (Equação 5):

$$R_n = \frac{9,4}{\sqrt{w}} + \frac{12,5}{w} + 0,045\,V + \frac{0,0005\,A\,V^2}{w\,n} \quad (5)$$

Onde:

R_n = Resistência do veículo em lb/short-ton

A = Área frontal em sq-ft

V = Velocidade em mi/h

n = Número de eixos do veículo

w = Peso médio por eixo em short-ton

Analisando as Equações 2, 3, 4 e 5 é possível observar o emprego de quatro constantes em cada uma. As duas primeiras constantes são equivalentes ao atrito gerado nos mancais. A terceira e a quarta constante representam respectivamente o atrito nos frisos das rodas e a resistência do ar. Portanto, a resistência obtida através destas fórmulas considera o veículo situado em um trecho da via cuja geometria é plana e reta.

Por sua vez, a resistência acidental é provocada pelas curvas e/ou rampas da ferrovia ou também pela inércia (PIRES, 2005). Deste modo, a resistência acidental é a soma das resistências de rampa, curva e inércia. As Equações 6, 7 e 8 descritas abaixo permitem o cálculo da resistência acidental. Com exceção da equação de resistência de curva, proposta por C. W. Stevenson, todas as demais fórmulas possuem embasamento teórico (BRINA, 1983).

Resistência de rampa (Equação 6):

$$R_r = 10\,i \quad (6)$$

Onde:

R_r = Resistência de rampa em kg/t

i = Inclinação da rampa em %

Resistência de curva para locomotivas (Equação 7):

$$R_c = 0,2 + \frac{100}{R}\,(p + b + 3,8) \quad (7)$$

Onde:

R_c = Resistência de curva em kg/t

R = Raio da curva em m

p = Base rígida do veículo em m

b = Bitola em m

Resistência de curva para vagões (Equação 8):

$$R_c = \frac{500\ b}{R} \tag{8}$$

Onde:

R_c = Resistência de curva em kg/t

R = Raio da curva em m

b = Bitola em m

A partir das Equações 6, 7 e 8 é possível obter a resistência total ao movimento da composição férrea com uma determinada lotação em um determinado segmento da via. Isto permite calcular o esforço trator útil da locomotiva (Equação 9), que representa a força necessária para deslocar a composição férrea considerando sua resistência total ao movimento (BRINA, 1983).

$$F_u = F_e - R_t \tag{9}$$

Onde:

F_u = Esforço trator útil em kgf

F_e = Esforço trator efetivo em kgf

R_t = Resistência total ao movimento da composição em kgf

Em outras palavras, o esforço trator útil corresponde à força aceleradora da composição férrea. Portanto, quando a força aceleradora for positiva, a composição férrea aumentará sua velocidade. Do contrário, quando a força aceleradora for negativa, a composição férrea reduzirá a velocidade, devido ao esforço trator produzido pelas locomotivas ser inferior à resistência total dos veículos. Não obstante, o esforço trator útil pode ser influenciado por outra grandeza, denominada força de aderência.

2.2.2 Força de Aderência

Conforme dito anteriormente, o contato entre as rodas da composição e os trilhos da ferrovia produz uma força de atrito, que se opõe ao movimento da roda. Tal fenômeno, chamado força de aderência, é resultante do produto do coeficiente de atrito pelo peso da locomotiva. Em outras palavras, a força de aderência neutraliza o esforço trator aplicado aos trilhos através das rodas da locomotiva.

O coeficiente de atrito é um índice que afere a intensidade de aderência entre as superfícies em contato, variando de acordo com os tipos de materiais, condições atmosféricas, velocidade, dentre outros fatores. De maneira geral, é adotado o valor 0,20 como coeficiente de atrito em repouso para o contato entre as rodas da composição e os trilhos da ferrovia. Contudo, foi observado experimentalmente que, conforme a locomotiva aumenta a velocidade, o coeficiente de atrito diminui. São inúmeras as causas de tal evento, geralmente associadas às irregularidades existentes nas rodas da locomotiva e nos trilhos. A variação do coeficiente de atrito pode ser expressa empiricamente a partir da Equação 10 (BRINA, 1983).

$$f = \frac{f_0}{1 + 0{,}01\,V} \tag{10}$$

Onde:

f = Coeficiente de atrito

f_0 = Coeficiente de atrito em repouso

V = Velocidade em km/h

A força de aderência também atua como limitador do esforço trator exercido pelas rodas da locomotiva. Caso o esforço trator aplicado pelos motores de tração for maior que a força de aderência, as rodas da locomotiva irão patinar. Este fenômeno é descrito pela Equação 11 proposta por Coulomb, que representa a base da tração por aderência (BRINA, 1983).

$$F \leq P\,f \tag{11}$$

Onde:

F = Esforço trator em kgf

P = Peso da locomotiva em kg

f = Coeficiente de atrito

A partir da Equação 11, pode-se concluir que a aceleração máxima permitida depende essencialmente do peso da locomotiva, uma vez que o coeficiente de atrito não apresenta consideráveis variações. Tal característica evidencia a influência da força de aderência na dinâmica do movimento, visto que em determinadas situações o esforço trator será restrito devido ao limite de aderência. Além da força de aderência, as variáveis referentes ao sistema de tração são outros fatores a serem considerados pela dinâmica do movimento.

2.2.3 Sistema de Tração

Em locomotivas diesel-életricas, os componentes básicos contidos na mecânica do sistema de tração são: motor principal movido à combustão, gerador principal, motores elétricos de tração e engrenagens. Basicamente o motor à combustão produz uma determinada potência a fim de alimentar o gerador principal, que por sua vez fornece energia aos motores elétricos de tração. Toda esta sistemática possui uma razão de eficiência, isto é, nem toda a energia produzida será aproveitada. Deste modo, cada um destes componentes possui um coeficiente de rendimento em função do volume de energia dissipada.

Para identificar a potência produzida em cada componente do sistema de tração são utilizadas diferentes classificações de potência. Denomina-se potência indicada a potência gerada pelo motor de combustão, normalmente fornecida pelo fabricante. Por sua vez, a potência fornecida ao gerador compreende a potência a ser absorvida pelo gerador principal. Esta potência deve considerar a energia dissipada pelo sistema de engrenagens, que é aproximadamente de 8%. A Equação 12 apresenta uma maneira de obter a potência entregue ao gerador (BRINA, 1983).

$$W_{HPg} = 0{,}92\ W_{HPi} \tag{12}$$

Onde:

W_{HPg} = Potência fornecida ao gerador em HP

W_{HPi} = Potência indicada em HP

Por fim, a potência efetiva representa a potência entregue às rodas da locomotiva. Contudo, também há perda de energia ao transferir a potência dos motores elétricos de tração até as rodas. O rendimento da transmissão elétrica tem um valor de aproximadamente 82%. Deste modo, a potência efetiva pode ser expressa conforme a Equação 13 (BRINA, 1983):

$$W_{HPe} = 0{,}82\ W_{HPg} \tag{13}$$

Onde:

W_{HPe} = Potência efetiva em HP

W_{HPg} = Potência no gerador em HP

2.3 Eficiência Energética

Conforme visto nos tópicos anteriores, a realização de simulações matemáticas do movimento de veículos ferroviários é uma atividade de alto grau de complexidade, devido aos inúmeros aspectos a serem considerados. De fato, a concepção de estratégias ótimas para condução férrea, em especial a eficiência energética, eleva ainda mais a dificuldade do problema. Atualmente este é um tema de intenso estudo por diversos pesquisadores, como Kraay et al. (1989), Howlett e Cheng (1995), Khmelnitsky (2000), Medanic (2002), Juraska e Magyla (2004) e Gianelli et al. (2005). Estes trabalhos, em sua essência, investigam os principais fatores envolvidos no consumo de combustível e propõem estratégias para redução do consumo combinando as equações da dinâmica do movimento a métodos numéricos de resolução. A seguir serão apresentadas alguns trabalhos previamente realizados para alcançar a redução de consumo em sistemas de condução férrea.

2.3.1 Modelo de Consumo de Energia

O modelo de consumo de energia, introduzido por Benjamin et al. em 1989, tem sua estratégia baseada no consumo energético em relação aos pontos de aceleração da locomotiva (Howlett e Cheng, 1995). Em outras palavras, a variável de controle deste modelo é a configuração do ponto de aceleração e o custo da estratégia é o consumo total de combustível. Este modelo considera o fato da potência desenvolvida pela locomotiva ser diretamente proporcional à taxa de consumo de combustível. Em termos

gerais, uma estratégia ótima é composta por segmentos de potência máxima, manutenção da velocidade, ponto morto e frenagem respectivamente. Entretanto, em função da geometria da via, estes segmentos podem não ocorrer nesta ordem.

A variável de controle do modelo possui um conjunto finito de valores que representa o controle de tração da locomotiva. O valor negativo representa o controle de frenagem, o valor zero representa o ponto morto e os demais valores referem-se aos pontos de aceleração da locomotiva. Cada valor da variável de controle possui uma taxa de consumo de combustível correspondente. Por sua vez, o trajeto a ser percorrido pela composição pode ser dividido em diversos pontos de troca, os quais representam as ocasiões em que podem ser efetuadas as mudanças de ponto de aceleração em cada locomotiva.

A combinação entre os valores da variável de controle e os pontos de troca do trajeto resulta em todas as possíveis configurações de controles do problema. Para encontrar uma estratégia que minimize o consumo de combustível, portanto, é necessário localizar os pontos de troca ótimos para o trajeto em questão. Tal aspecto caracteriza este modelo como uma relação entre a força tratora desempenhada pela locomotiva e a interação entre a composição férrea e a ferrovia. Howlett e Cheng (1995) introduziram um algoritmo adotando este modelo para encontrar estratégias ótimas em trajetos cuja geometria é constante em cada seção da via.

2.3.2 Escalonamento Ferroviário

Além do traçado da ferrovia, outro aspecto que influencia diretamente no consumo de combustível são as paradas efetuadas pela composição férrea ao longo do percurso. O impacto de tal evento foi estudado por diversos pesquisadores, entre eles Rakha e Ding (2003) e Juraska e Magyla (2004). Em média 15% do combustível total consumido em uma viagem estão relacionados às paradas geralmente não previstas no planejamento da rota. Segundo Juraska e Magyla (2004), as paradas não previstas são freqüentemente conseqüências de uma má organização no escalonamento das composições férreas.

Devido ao fato da malha ferroviária ser compartilhada, a coordenação entre as composições férreas deve, além de prezar pela segurança, manter o maior número possível de composições em movimento. Contudo, o escalonamento ferroviário é considerado um problema complexo de alocação de recursos, pois envolve uma grande

quantidade de variáveis e de restrições dinâmicas (MEDANIC, 2002). Tormos et al. (2006) apresentam uma modelagem para resolução de tal problema baseada em programação por restrição. O modelo proposto permite o uso de computação paralela, onde o problema global é dividido em diversos subproblemas, reduzindo o esforço computacional necessário para resolvê-los. Este modelo emprega com três grupos de regras para o escalonamento das composições: regras de tráfego, regras de requisições do usuário e regras topológicas.

Em sua essência, as regras de tráfego referem-se ao compartilhamento da malha ferroviária. Em outras palavras, estas restrições consideram situações onde duas composições se cruzam em sentidos opostos ou quando duas composições viajam no mesmo sentido, porém, em velocidades diferentes. As regras de requisições do usuário compreendem as restrições informadas pelo usuário do sistema. Estas restrições podem ser: o tipo e a quantidade de composições na malha ferroviária, a rota a ser realizada por cada composição, o intervalo de partida, dentre outras. Por fim, as regras topológicas englobam as restrições pertinentes à malha ferroviária. Informações como a quantidade de vias em cada estação, o número de estações e a distância entre as estações são convertidas em regras topológicas.

2.3.3 Impacto da Aceleração e Desaceleração no Consumo de Combustível

Em Rakha e Ding (2003), dentre outras contribuições, foi investigado o impacto da aceleração e da desaceleração no consumo energético. Esta análise foi realizada a partir de experimentos sistematicamente quantificados considerando diferentes estilos de condução, tais como moderada, normal e agressiva. Através dos resultados dos experimentos, ficou evidente a redução do consumo de combustível ao manter uma velocidade constante próxima à velocidade limite da rota. Entretanto, foi observado também que o consumo também tende a aumentar em função da velocidade.

Contudo, apesar da manutenção constante da velocidade se capaz de reduzir o consumo de energia, sabe-se que isto é raramente aplicável, uma vez que a geometria da ferrovia freqüentemente não é plana. Em virtude disto, Chang e Morlok (2005) propuseram uma alternativa para este problema através da normalização da geometria da ferrovia. Em outras palavras, foram propostas derivações combinando tangentes, curvas, gradientes, dentre outros fatores da geometria da via que atuam diretamente sobre o efeito de aceleração e desaceleração da velocidade. Estas derivações têm o

objetivo de eliminar a necessidade de acelerações e desacelerações ao longo da ferrovia, permitindo assim manter uma velocidade constante.

Existem outras abordagens para a normalização da geometria da ferrovia. Uma abordagem bastante conhecida é o comprimento virtual de um traçado, que corresponde a um traçado equivalente em nível e em tangente, sob o ponto de vista da resistência oferecida à tração (BRINA, 1983). Para calcular o comprimento virtual, é necessário dividir o traçado em vários segmentos e calcular a resistência total da composição em cada segmento. Em outras palavras, o comprimento virtual de um traçado deve possuir aproximadamente a soma das resistências para todos os segmentos semelhante ao do traçado original.

2.4 Considerações Finais

Prover políticas ótimas para condução de composições férreas tem sido objeto de estudo de diversos pesquisadores nas últimas décadas. Inúmeras estratégias bem sucedidas foram propostas, no entanto, elas permanecem incompletas ou insuficientes devido às restrições impostas pelo desenvolvimento de tais modelos. Contudo, este trabalho tem como objetivo comprovar que aplicação destas estratégias aliadas à técnicas da Inteligência Artificial Distribuída resulte em soluções adequadas para o problema. Uma técnica capaz de modelar naturalmente problemas otimização do mundo real é a otimização distribuída de restrição, descrita no capítulo a seguir.

Capítulo 3

Fundamentos sobre Problemas de Otimização Distribuída de Restrição

O presente capítulo fornece uma introdução à problemas de satisfação distribuída de restrição, Distributed Constraint Satisfaction Problem (DCSP), enfatizando conceitos sobre problemas de otimização distribuída de restrição, Distributed Constraint Optimization Problem (DCOP). Nesta seção, também são apresentados os principais algoritmos do formalismo DCOP.

3.1 Programação por Restrição

A programação por restrição, Contraint Programming (CP), é o estudo de sistemas computacionais baseado em restrições, sendo um dos inúmeros métodos de resolução de problemas da Inteligência Artificial. Em termos gerais, a programação por restrição consiste em resolver problemas através de restrições definidas sobre a área do problema. Nos últimos anos, este paradigma tem ganhado muita atenção devido ao seu potencial de modelar naturalmente diversas categorias de problemas do mundo real.

> Constraint Programming represents one of the closest approaches computer science has yet made to the Holy Grail of programming: the user states the problem, the computer solves it (apud BARTAK, 1999, p. 1).

Em sua essência, uma restrição é uma relação lógica entre uma ou mais variáveis, onde cada variável possui um determinado domínio. Deste modo, o objetivo destas relações lógicas é restringir possíveis valores que as variáveis podem assumir. A s restrições podem especificar parte das informações de um dado problema, podem ser heterogêneas (restrições entre variáveis de domínios diferentes), são declarativas, aditivas e raramente são independentes (BARTAK, 1999).

Por sua vez, CP consiste de um conjunto de técnicas para resolução de diversos tipos de problemas, em especial problemas de escalonamento de recursos, planejamento e otimização combinatória. Uma das técnicas mais populares do CP é a satisfação de

restrição (BARTAK, 2001). Nesta técnica, o objetivo do CP é encontrar atribuições para as variáveis cujos domínios são finitos, de modo que nenhuma restrição seja violada.

3.2 Problema de Satisfação de Restrição

O problema de satisfação de restrição, Constraint Satisfaction Problem (CSP), consiste em encontrar uma seqüência de ações a ser realizada para alcançar um estado alvo, que a priori não é conhecido. Contudo, métodos inteligentes podem ser aplicados para resolução destes problemas de modo a otimizar a exploração das alternativas (WEISS, 1999). Um CSP pode ser descrito como um problema cuja finalidade é encontrar uma combinação de valores para todas as variáveis do problema, de modo que sejam satisfeitas todas as restrições (BARTAK, 2003). Segundo Russel e Norvig (2004), um CSP é composto por: um conjunto de variáveis, um domínio discreto e finito para cada variável e um conjunto de restrições aplicadas sobre os possíveis valores. Deste modo, o objetivo consiste em encontrar valores consistentes a todas as variáveis, ao passo que nenhuma restrição seja violada.

Com o advento de avanços tecnológicos, sobretudo na área de redes de computadores, novos campos de pesquisa da Inteligência Artificial Distribuída surgiram. O CSP é um exemplo de formalismo que pode ser estendido para atuar de forma distribuída (YOKOO, 1998). Vários problemas da Inteligência Artificial Distribuída podem ser formalizados como DCSP (Distributed Constraint Satisfaction Problem). Nesta modalidade, as variáveis do problema são distribuídas entre agentes, onde o objetivo é alcançar a coerência entre os agentes (WEISS, 1999).

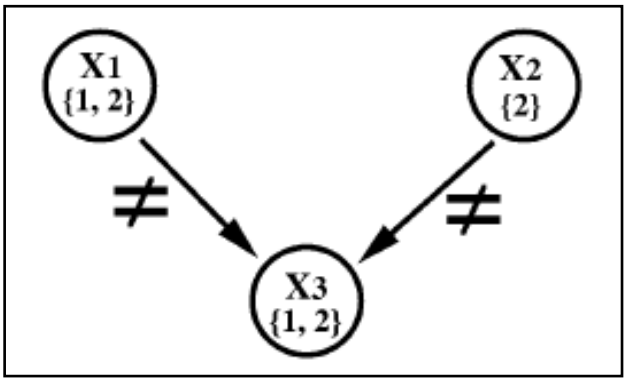

Figura 2. Representação de um DCSP como um grafo de restrições (YOKOO, 1998).

Um DCSP pode ser representado através de um grafo (Figura 2). Cada agente corresponde a um nó e as arestas representam as restrições em comum aos dois nós

conectados. O problema é então resolvido pelo acúmulo de computações locais realizadas por cada nó do grafo. A ordem de execução destas computações locais pode ser arbitrária ou altamente flexível e síncrona ou assíncrona (WEISS, 1999). Entretanto, como uma das principais motivações do DCSP é a eficiência através do processamento paralelo e distribuído (YOKOO, 1998), este capítulo irá retratar com maior ênfase os métodos de busca assíncronos.

3.3 Problema de Otimização Distribuída de Restrição

Conforme visto anteriormente, um DCSP é capaz de encontrar soluções satisfatórias quando existentes. Porém, este formalismo não é capaz de modelar problemas onde as soluções podem ter graus de qualidade ou custo. Um DCOP (Distributed Constraint Optimization Problem), apesar de ser uma extensão do DCSP, oferece técnicas que vão além da busca por soluções satisfatórias ou de simples métodos de otimização (LESSER, 2003).

Em um DCOP, cada restrição do problema é caracterizada como uma função de otimização, ou também chamada de função de custo. Desta forma, o mecanismo de busca para um DCOP consiste em encontrar valores para as variáveis de modo a otimizar as funções de custo, proporcionando garantia de qualidade para as soluções encontradas (LESSER, 2003). Em um sentido oposto, um DCSP é limitado a problemas de satisfação, designando as soluções apenas como satisfatórias ou insatisfatórias.

Outra particularidade de um DCOP é a eficiência em busca de soluções cuja qualidade esteja dentro de uma distância pré-definida. Este recurso é útil quando não há recursos computacionais disponíveis para encontrar soluções ótimas, contudo, a qualidade de solução estará dentro de um intervalo previamente estipulado.

3.4 Definição de um DCOP

De modo geral, a definição de um DCOP é semelhante a um DCSP. Um DCOP é composto por n variáveis $V = \{v_1, v_2, ..., v_n\}$, no qual cada variável está associada a um agente x_i. Por sua vez, uma variável contém um domínio finito e discreto, $D_1, D_2, ... D_n$, respectivamente. Apenas o agente x_i é capaz de atribuir valores para a variável v_i e conhecer o domínio D_i. Cada agente deve escolher um valor d_i para sua variável, tal que $d_i \in D_i$. Portanto, os agentes deverão se coordenar para escolherem os valores para suas

variáveis, de modo que o custo da função objetivo global definida para o problema seja minizado (MODI, 2003).

De fato, a principal diferença entre a definição do DCOP e do DCSP é o conceito de restrição, onde no DCOP é denominado como função de custo. Uma função de custo para um par de variáveis x_i e x_j é dada por $f_{ij} : D_i \times D_j \rightarrow N$. Dois agentes x_i e x_j serão vizinhos quando existir alguma restrição entre eles. Deste modo, um DCOP deve então encontrar um conjunto $A^* = \{d_1,d_2,...,d_n \mid d_1 \in D_1 , d_2 \in D_2, ..., d_n \in D_n\}$ de atribuições para as variáveis, de modo que o custo F acumulado seja minimizado (MAILLER, 2004). A função objetivo global F é definida conforme a Equação 14.

$$F(A) = \sum_{x_i,x_j \in V} f_{ij}(d_i,d_j), \text{ onde } x_i \leftarrow d_i, x_j \leftarrow d_j \text{ em } A \tag{14}$$

3.5 Algoritmos para DCOP

Os métodos para resolução de um DCOP podem ser divididos em duas categorias: síncronos e assíncronos. Entretanto, os métodos síncronos são dispendiosos, no sentido que os agentes devem aguardar até o recebimento de uma mensagem particular para desempenharem suas ações (MODI, 2003). Tal característica onera o desempenho da busca, devido ao fato de não ser possível explorar as vantagens do processamento paralelo ao distribuir o problema. Em contrapartida, em métodos assíncronos os agentes devem ser capazes de tomar suas ações com base em suas visões locais do problema, o que aumenta a complexidade do mecanismo de busca. Um dos principais obstáculos para resolução de um DCOP com métodos assíncronos é a garantia de qualidade. A razão disto está na dificuldade em promover uma busca sistemática enquanto os agentes alteram assincronamente seus valores de variáveis. A seguir são apresentados os principais algoritmos para resolução de DCOP. Os pseudo-códigos dos algoritmos apresentados neste capítulo constam no anexo do trabalho.

3.5.1 ADOPT

O Asynchronous Distributed OPTimization (ADOPT), proposto por Modi et al. (2003), foi o primeiro algoritmo assíncrono completo a oferecer garantia de qualidade. Desde modo, o ADOPT é capaz de encontrar soluções ótimas usando comunicação assíncrona e localizada entre os agentes, isto é, a troca de mensagens é restrita entre os agentes vizinhos.

No ADOPT os agentes devem ser priorizados em uma estrutura de pseudo-árvore. Por meio desta ordem de prioridade, o ADOPT executa uma busca em profundidade por backtracking distribuído usando uma estratégia oportunista. Em outras palavras, cada agente mantém a escolha do melhor valor baseado em sua visão local. Entretanto, uma rotina de pré-processamento é requerida para transformar o grafo de restrições do problema em uma estrutura de pseudo-árvore.

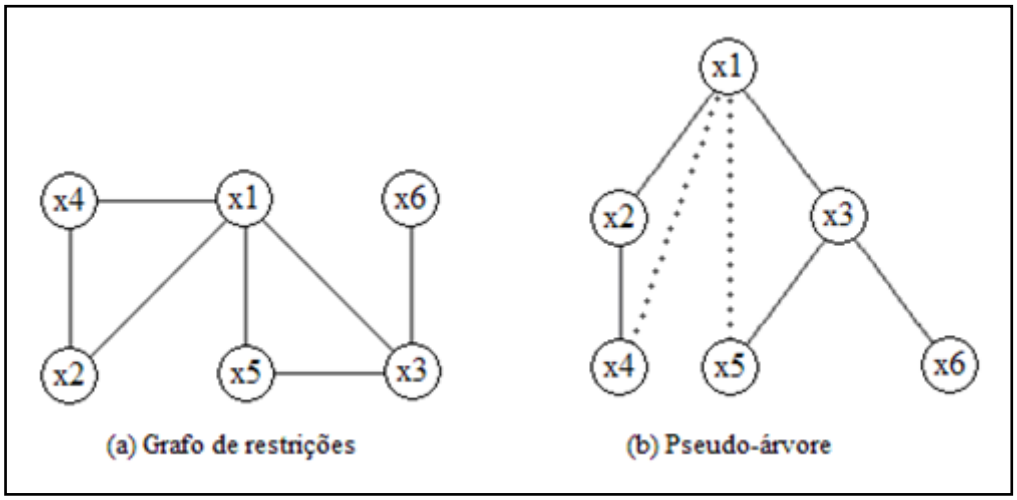

Figura 3. Pseudo-árvore gerada a partir de um grafo de restrições.

A Figura 3 ilustra um exemplo de uma pseudo-árvore (b) gerada a partir de um grafo de restrições (a). Ao analisar este exemplo, é possível concluir que o grafo representado é cíclico. Uma das alternativas para eliminar os ciclos do grafo e conseqüentemente facilitar o processo de busca é transformá-lo em uma pseudo-árvore. Por definição, uma pseudo-árvore é semelhante a uma árvore tradicional, porém, cada nó pode estar conectado a múltiplos nós de maior hierarquia. Contudo, apenas um dos nós de maior hierarquia é definido como pai, enquanto os demais nós de maior hierarquia são denominados pseudo-pais. A Figura 3b ilustra o exemplo de uma pseudo-árvore, onde as linhas contínuas representam as ligações entre pai e filho e as linhas pontilhadas representam as ligações entre pseudo-pai e pseudo-filho. A seguir é apresentado em detalhes o mecanismo de busca do ADOPT.

3.5.1.1 Definição do ADOPT

O mecanismo de busca do ADOPT utiliza três idéias chave para resolução de um DCOP: busca assíncrona, reconstrução eficiente de soluções abandonadas e detecção de término integrada (MODI, 2003). A busca assíncrona do ADOPT

possibilita aos agentes alterarem os valores de suas variáveis a partir de visões locais, utilizando uma técnica de propagação de limites de custo. Esta técnica de propagação de limites de custo empregada no ADOPT é uma variante da busca em profundidade conhecida na literatura como Branch and Bound (BB). O BB tem como objetivo guiar a busca em uma árvore através de uma heurística aplicada sobre a função de otimização. Ao aplicar uma heurística na sistemática de busca é possível realizar podas no espaço de estados ao detectar que não há possibilidade da solução atual ser melhor que a solução anterior. Quando identificada esta situação, é efetuado um backtracking na busca em profundidade (TSANG, 1993).

Contudo, um dos problemas do BB é a necessidade de uma variável (chamada *bound*) para armazenar o custo global acumulado até o momento (TSANG, 1993). Em vista disto, o BB requer que os agentes tenham acesso ao limite superior (*upper bound*) global, impossibilitando a utilização desta estratégia em modelos assíncronos. Em contrapartida, o ADOPT propõe uma estratégia alternativa para calcular o custo de uma solução, através do limite inferior (*lower bound*). O *lower bound* pode ser calculado sem a necessidade de um custo global acumulado. Assim, esta informação é iterativamente refinada por um agente ao receber novas informações de custo de seus filhos (MODI, 2003).

A estratégia do ADOPT permite que uma solução parcial seja abandonada antes mesmo de ser constatada como insatisfatória. Em alguns casos, é inviável armazenar as soluções abandonadas devido ao uso excessivo de memória e da busca seqüencial necessária para revisá-las. Portanto, um sistema eficiente para reconstrução de soluções abandonadas é de fato necessário para revisar estas soluções anteriormente consideradas. Para isto, o ADOPT utiliza o *lower bound* a fim de reduzir o custo de reconstrução da solução. Esta informação é armazenada em uma variável chamada *threshold* antes de ser efetuado um backtracking. Caso seja necessário restaurar a solução parcial abandonada, o *threshold* irá conduzir a busca para uma solução de menor ou igual qualidade à que foi abandonada. Neste caso, um agente somente poderá alterar o valor de sua variável se o novo custo da solução for menor ou igual ao *threshold* (MODI, 2003).

Por fim, o mecanismo de detecção de término deve oferecer garantia de qualidade para as soluções encontradas. O ADOPT usa os intervalos do limite (*lower bound* e *upper bound*) para delinear o progresso da busca em direção à solução ótima. O custo da solução ótima é determinado quando o intervalo dos limites é igual a zero. Ao

ser encontrada uma solução com este custo, a busca pode ser seguramente encerrada. Em outras palavras, quando o nó raiz da árvore de busca em profundidade detectar que os intervalos de limite são iguais, a solução para o problema foi encontrada.

O mecanismo de detecção de término do ADOPT também é capaz de utilizar os intervalos de limite para fornecer um critério de parada limitado a erro. Isto torna possível estabelecer graus mínimos de qualidade para as soluções encontradas. Assim, o algoritmo poderá encerrar a busca quando for encontrada uma solução cuja qualidade está dentro do limite especificado (no pior caso) ou quando a solução ótima for encontrada (no melhor caso).

3.5.1.2 Modelo de Comunicação Assíncrona

Uma vez que os agentes podem alterar assincronamente o valor de suas variáveis apenas com base em sua visão local, um modelo de comunicação é necessário para atualizar e conseqüentemente refinar estas informações. No modelo de comunicação do ADOPT, as mensagens somente podem ser emitidas entre agentes vizinhos, isto é, agentes que apresentam restrições em comum. As mensagens são divididas em três categorias: atribuição de valor, custo e *threshold* (Figura 4).

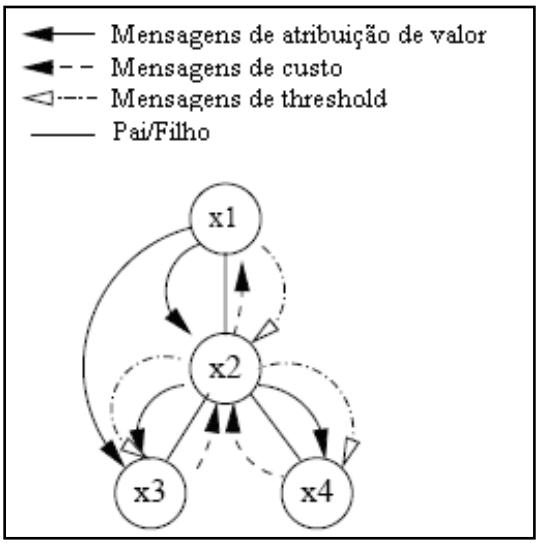

Figura 4. Modelo de comunicação do ADOPT (MODI, 2003)

Quando um agente altera o valor de uma variável, são emitidas mensagens de atribuição de valor para seus vizinhos de menor prioridade. Este tipo de mensagem tem como objetivo informar aos agentes de menor prioridade que possuem restrição em comum sobre o novo valor assumido pelo agente de maior prioridade. Ao receber esta

mensagem, o agente de menor prioridade irá avaliar se o novo valor assumido pelo agente de maior prioridade é compatível com sua visão local. Caso seja incompatível, o agente de menor prioridade escolhe outro valor para sua variável de modo a minimizar o custo da solução. Os valores de variáveis atribuídos por um agente de maior prioridade são armazenados em uma estrutura chamada *ContextoCorrente* seguindo o formato $\{(x_i,d_i), ..., (x_k,d_k)\}$.

As mensagens de custo são emitidas dos agentes filhos para os agentes pais, sendo análogas às mensagens de *nogood* do DCSP. Um *nogood* especifica um conjunto de atribuições de variáveis que possuem conflitos. Em outras palavras, um *nogood* contém uma solução parcial não satisfatória do problema. No entanto, um problema de otimização requer informações sobre o custo de um determinado conjunto de atribuições. Deste modo, as mensagens de custo contêm o custo local do agente, representado por δ, mais a soma de todos os custos reportados de seus agentes filhos.

As mensagens de custo são compostas por três informações: contexto, *LB* e *UB*. O contexto é uma solução parcial do problema sendo composta pelas atribuições dos vizinhos de maior prioridade. Esta informação é utilizada para cálculo do δ conforme as atribuições de valores dos agentes de maior prioridade. Quando o *ContextoCorrente* é incompatível com o valor escolhido pelo agente corrente, o *LB* e o *UB* são atualizados com 0 e ∞ respectivamente. O *LB* representa o menor *lower bound* de todos os possíveis valores para o agente corrente. Em outras palavras, o *LB* contém o custo mínimo para δ mais a soma dos *LB* de seus vizinhos de menor prioridade. Semelhantemente, o *UB* representa o menor *upper bound* de todos os possíveis valores para o agente corrente. Isto é, o *UB* corresponde ao custo mínimo para δ mais a soma dos *UB* de seus vizinhos de prioridade menor.

Por conseguinte, as mensagens de *threshold* são emitidas dos agentes pais para os agentes filhos atualizarem seus respectivos *threshold*. O *threshold* pode ser atualizado de três formas distintas. O agente irá incrementar seu *threshold* sempre que o custo de uma solução ótima em sua sub-árvore for maior que o valor corrente do *threshold*. O agente irá decrementar seu *threshold* sempre que o custo de uma solução ótima em sua sub-árvore for menor que o valor corrente do *threshold*. Ao receber uma mensagem de *threshold*, o agente também atualiza o valor de seu *threshold*. O motivo disto é que em algumas situações o agente pai deve ser capaz de determinar o limite de um custo ótimo para as sub-árvores iniciadas em seus agentes filhos.

3.5.2 Variantes do ADOPT

O ADOPT tem se mostrado um poderoso algoritmo distribuído e assíncrono para problemas de otimização distribuída de restrição. Em decorrência disto, inúmeras versões do ADOPT foram propostas para aprimorá-lo em diferentes aspectos, visto que em algumas situações seu mecanismo de busca apresenta consideráveis ineficiências. A seguir, são apresentadas as principais variantes do ADOPT.

3.5.2.1 Técnicas de Pré-processamento

Devido ao fato do ADOPT utilizar um método de busca não informada para resolução de um DCOP, técnicas de pré-processamento podem aumentar significativamente seu desempenho pelo fornecimento de uma heurística para conduzir a busca mais rapidamente a uma solução. É comum a estratégia oportunista do ADOPT fazer com que os agentes troquem desnecessariamente seus valores de variáveis. Como conseqüência, isto gera um alto custo no processo de busca, pois obriga os sucessores do agente que alterou indevidamente seu valor a reconstruírem suas soluções. Em função destes problemas, Ali et al. (2005) apresentam um framework para melhorar o desempenho do ADOPT.

Ao iniciar o processo de busca, o ADOPT atribui um custo zero para cada agente. De fato, se os agentes iniciassem com um custo estimado diferente de zero, o desempenho do mecanismo de busca iria melhorar devido à redução do espaço de estados candidatos. Estes custos estimados são chamados de valores de heurística e são obtidos através de um pré-processamento do DCOP em uma versão relaxada, dividindo a execução do ADOPT em duas fases: pré-processamento e busca principal.

Contudo, a fase de pré-processamento também demanda custo, o que em algumas condições pode onerar o processo de busca, ocasionando um desempenho pior do que a execução do problema em uma versão do ADOPT sem a fase de pré-processamento. Em decorrência disto, são propostas três técnicas de pré-processamento, chamadas: DP0, DP1 e DP2. Estas técnicas implementam algoritmos distribuídos de programação dinâmica onde os nós enviam mensagens com o custo estimado para seus pais. Os algoritmos de pré-processamento propostos consideram duas métricas distintas para obter os valores de heurística: custo de comunicação e de computação.

Normalmente o custo de comunicação em ambientes distribuídos é mensurado em ciclos, onde cada ciclo representa o processamento de todas as mensagens de um

determinado agente. A técnica de pré-processamento DP0 requer apenas um ciclo, pois os agentes não propagam seu custo estimado, apenas o custo local. De forma contrária, as demais técnicas requerem um número de ciclos igual à profundidade da árvore de restrição mais um, pois os custos estimados dos agentes são propagados para os agentes de maior prioridade. Enquanto o DP0 obtém mais rapidamente os valores de heurística, o DP1 e o DP2 conseguem estimar melhores valores para a heurística, com relação ao custo estimado dos agentes.

O custo de computação é afetado pela quantidade de restrições que um determinado nó deverá avaliar para estimar seu valor de heurística. A técnica de pré-processamento DP1 necessita avaliar apenas os custos das restrições que um agente tem com seu pai, enquanto que as demais técnicas avaliam as restrições com os demais agentes que o antecedem na árvore de restrição. A técnica de pré-processamento DP1 obtém os valores de heurística em um tempo menor ou igual às técnicas DP0 e a DP2. Entretanto, as técnicas DP0 e DP2 conseguem estimar melhores valores para a heurística, com relação ao custo local dos agentes.

Por fim, a escolha da técnica de pré-processamento deve ser orientada às características do problema de otimização em questão. As técnicas DP0 e DP1 conseguem obter os valores de heurística a um custo reduzido através de um relaxamento de restrições do DCOP em sua versão original. Por outro lado, a técnica DP2 requer mais recursos durante o pré-processamento, mas estima melhores valores de heurística. Deste modo, questões como tamanho do espaço de estados, custo de comunicação e de computação e tempo devem ser considerados durante a seleção da técnica de pré-processamento.

3.5.2.2 ADOPT-ng

Apesar do ADOPT ser considerado pela comunidade de pesquisadores um elegante método distribuído e assíncrono para resolução de DCOP, existem situações onde a complexidade da árvore de restrições do problema aumenta consideravelmente o custo de computação. De fato, o modelo de comunicação do ADOPT, sobretudo as mensagens de custo, pode ser criticado por não prover informações necessárias para os agentes tomarem melhores decisões. Com isto, os agentes necessitam de um número maior de ciclos e de trocas de mensagens para alcançar um estado estável em seu subproblema. Outro aspecto negativo do ADOPT é a exigência de uma ordenação dos

agentes em forma de uma pseudo-árvore, o que demanda custo para construí-la a partir do grafo de restrições do problema.

Para resolver tais problemas no ADOPT, Silagui e Yokoo (2006) propuseram uma nova versão do algoritmo, chamada ADOPT-ng. A principal diferença entre a versão do ADOPT-ng e a original está no modelo de mensagens, que é baseada em um novo tipo de *nogood*. A implementação de *nogoods* na sistemática do ADOPT, além de eliminar a necessidade da etapa de pré-processamento por não exigir uma ordenação dos agentes, oferece uma significativa melhoria do desempenho do algoritmo. Tal motivação deu origem a um novo tipo de *nogood*, chamado *nogood* valorado, no qual a informação sobre o custo de uma atribuição é acoplada ao *nogood*.

O ADOPT-ng substitui a mensagem de custo do ADOPT original por uma mensagem de *nogood* no formato [*c*,*N*], onde *c* é o custo mínino em um dado conjunto *N* de atribuições de variáveis distintas. Com estas informações, um agente pode inferir um custo de atribuição *v* no formato (*v*,*c*,*N*) a partir dos *nogoods* recebidos, sendo possível deduzir um custo mínimo para as atribuições conflitantes. Através desta inferência, é possível maximizar a eficiência do raciocínio de busca acelerando o progresso da exploração.

Além da mensagem de *nogood*, o ADOPT-ng também contém outro tipo de mensagem, chamada *add-link*. Uma mensagem de *add-link* anuncia o interesse em uma variável contida no *nogood*, cujo agente remetente não é vizinho do agente destinatário. O objetivo desta mensagem é minimizar de maneira mais eficiente o custo total das atribuições contidas no *nogood*. O processo de reconstrução de uma solução parcial contida no *nogood* é mais eficiente que o ADOPT tradicional, pois somente as variáveis contidas no *nogood* irão alterar seus valores.

3.5.2.3 ADOPT Multi-critério

O ADOPT provou ser capaz de resolver inúmeros problemas de otimização, sejam problemas simples ou altamente complexos, envolvendo pequenos ou amplos espaços de estados ou com um número considerável de restrições. Não obstante, existem categorias de problemas do mundo real onde grande parte dos principais métodos para resolução de DCOP falha, em especial, problemas cujos objetivos são múltiplos. Um exemplo de problema do mundo real que se enquadra nesta situação seria a eficiência energética em conduções férreas. Além do objetivo evidente do problema,

reduzir o consumo de combustível, poderiam existir outros objetivos, tais como tempo mínimo de duração de uma determinada viagem.

Um dos principais desafios em problemas de otimização com múltiplos objetivos há restrições recursos. Em função deste fator e outros, Bowring et al. (2006) propuseram um novo algoritmo, chamado Multiply-Constrained ADOPT (MCA) para resolver DCOP com múltiplas restrições. Uma das inovações que o MCA apresentou também é a busca com restrições dinâmicas.

Ao se tratar de DCOP com múltiplas restrições, é necessária uma nova função de custo g_i no subconjunto das restrições de x_i. Um limite para o custo do recurso também deve existir, denotado como *g-budget*. Portanto, o custo do recurso G_i deve ser menor ou igual ao *g-budget* deste recurso. As funções de custo utilizadas para limitar o custo de um dado recurso são caracterizadas como *g-function*, enquanto que as demais funções de custo denominam-se *f-function*. Uma *g-function* aliada ao *g-budget* constitui uma restrição chamada *g-constraint*. A definição de DCOP com múltiplas restrições parte do princípio que as *g-functions* não podem ser tomadas como *f-functions* e, cada atribuição de valor deve considerar ambas estas funções. Assim, uma notável característica de um DCOP com múltiplas restrições é a interação entre as *f-functions* e as *g-functions* para preempção do espaço de estados do problema. Formalmente, um DCOP com múltiplas restrições pode ser definido conforme a Equação 15.

$$\begin{aligned} &F(A) = \sum_{x_i,x_j \in V} f_{ij}(d_i,d_j), \text{ onde } x_i \leftarrow d_i,\ x_j \leftarrow d_j \text{ em } A \\ &\text{e} \\ &\forall x_i \in V\ \ g_i(d_i,\{d_j \mid x_j \in \text{vizinhos}(x_i)\}) \leq G_i \end{aligned} \tag{15}$$

Para realizar a busca com restrições dinâmicas, o MCA reestrutura o grafo de restrições do problema e insere variáveis virtuais que detém as *g-constraints* n-ária. Estas variáveis virtuais indicam as violações de restrições de recursos, realizando podas no espaço de busca sempre que esta situação ocorrer. Desta forma, um agente descendente recebe de seu antecessor um limite de custo, chamado *g-threshold*, cuja premissa é informar o custo máximo que é permitido para seu subproblema. Caso uma atribuição exceda o *g-threshold* de seu antecessor, este valor não será explorado para otimização das *f-functions*. No entanto, se uma atribuição não puder satisfazer nenhuma das *g-constraints*, será enviado para seu antecessor um custo igual a ∞, forçando seus antecessores a mudarem seus valores. Em outras palavras, as variáveis virtuais irão comunicar assincronamente custos infinitos ao detectarem violações de *g-constraints*.

Esta estratégia de busca permite ao MCA resolver problemas de otimização com restrição de recursos. O MCA também pode ser aplicado onde os problemas exigem que as restrições de recursos sejam privadas. Neste cenário, a variável virtual é conectada sob os agentes que deterão a restrição privada como um nó folha na árvore de busca em profundidade, protegendo a privacidade das *g-functions* e do *g-budget*. Porém, a privacidade de restrições de recursos demanda custo, uma vez que somente os agentes ligados a variável virtual conhece sua *g-function* e o *g-budget*. Ao serem encontradas violações de *g-constraint*s, o custo para reconstruir uma solução tende a ser maior em relação a uma restrição de recurso compartilhado, pois não é possível avaliar previamente o impacto de uma dada atribuição sobre a *g-constraint* privada.

3.5.3 OptAPO

Inúmeros algoritmos distribuídos foram propostos até então para resolver problemas de otimização de restrição. Contudo, em sua grande maioria, os agentes mantêm uma separação total de seus conhecimentos durante o processo de resolução do problema. Em muitos problemas de otimização distribuída de restrição, estes conhecimentos poderiam ser compartilhados, permitindo que os agentes encontrem soluções parciais estáveis demandando menos esforço. Em virtude desta carência, Mailler e Lesser (2005) desenvolveram um algoritmo para DCOP baseado em mediação cooperativa, chamado Optimal Asynchronous Partial Overlay (OptAPO).

No OptAPO, os agentes são capazes de compartilhar seus conhecimentos a fim de melhorar suas decisões locais. Quando um agente atua como um mediador, é computada uma solução para uma fração do problema global, no qual são recomendadas novas atribuições aos agentes envolvidos na sessão de mediação. Tal artifício reduz consideravelmente o custo associado à comunicação para resolução local do problema exigido em algoritmos completamente distribuídos como o ADOPT. Na seqüência, será introduzido em maiores detalhes o mecanismo de busca do OptAPO.

3.5.3.1 Definição do OptAPO

O modelo de busca do OptAPO opera através de duas estruturas: *agent-view* e *good-list*. A *agent-view* armazena os nomes, valores, domínios e restrições dos agentes que estão conectados ao seu dono. A *good-list* mantém os nomes dos agentes em que o seu dono identificou ter restrições direta ou indiretamente no grafo de restrições. Cada

agente tenta melhorar o valor de seu subproblema contido à *good-list* ou justificar seu custo pela identificação de áreas super-restritas no grafo de restrição. Os agentes desempenham esta tarefa ao tomar o papel de mediador, computando um valor ótimo de seu subproblema e recomendando mudanças de atribuições das variáveis contidas na sessão de mediação. Quando estas mudanças influenciam no custo de agentes externos à sessão, estes agentes são adicionados na *good-list* do mediador para fazer parte da justificativa de custo de seu subproblema. Uma sessão de mediação se mantém ativa até que cada um dos agentes seja capaz de justificar o custo de seus subproblemas centralizados, garantindo que a fração do problema centralizada na sessão de mediação contenha um custo ótimo. Para estabelecer qual agente é apto a mediar, é atribuída aos agentes uma prioridade dinâmica baseada no tamanho de sua *good-list*. Isto faz com que os agentes com mais conhecimento sobre o problema tomem as decisões.

Ao iniciar o processo, os agentes atribuem um valor a suas variáveis e emitem uma mensagem *init* para seus vizinhos. Nesta mensagem, está contido o nome da variável, a prioridade do agente, o valor corrente da variável, o domínio da variável e as restrições do agente. Quando uma mensagem *init* é recebida, o agente receptor armazena as informações desta mensagem em sua *agent-view* e adiciona a variável na *good-list* quando esta possuir uma restrição direta ou indireta com outras variáveis contidas em sua *agent-view*.

Após todas as mensagens *init* serem recebidas, os agentes executam o processo de avaliação de sua *agent-view*, calculando o custo corrente F_i das restrições do sub-grafo formado pelos agentes contidos na *good-list*. Se um agente constatar que F_i é menor que F_i^* (melhor custo encontrado no sub-grafo) é conduzida uma sessão de mediação passiva ou ativa. Isto garante o término do processo de busca e a condição de algoritmo ótimo. Inicialmente é atribuído 0 ao F_i^*, supondo que a melhor solução não tenha custo. Ao ocorrer uma sessão de mediação, um novo valor para F_i^* é calculado através de uma busca centralizada sobre as variáveis contidas na *good-list*.

3.5.3.2 Sessões de Mediação

Uma sessão de mediação pode ser passiva ou ativa. A decisão de como será a mediação é baseada no agente de menor prioridade contida em sua *good-list* ao ser detectado a necessidade de uma sessão de mediação. Caso a menor prioridade

encontrada em sua *good-list* seja menor ou igual a sua própria prioridade, será conduzida uma sessão de mediação ativa, do contrário será passiva.

Uma mediação passiva objetiva compreender os resultados que os agentes de maior prioridade obtiveram sem alterar suas soluções correntes. De maneira inversa, em uma medição ativa, os agentes de maior prioridade podem sugerir alterações nas soluções correntes dos agentes de menor prioridade, visto que estes agentes possuem mais conhecimento sobre esta parcela do problema. Enquanto a mediação passiva visa mudar o valor de F_i^*, a mediação ativa objetiva mudar F_i^* e F_i. Em outras palavras, o propósito de uma mediação passiva é expandir as visões dos agentes de menor prioridade, enquanto que a mediação ativa tenta mudar as atribuições dos agentes de menor prioridade.

Quando um mediador ativo encontrar atribuições que tornem F_i igual à F_i^*, são emitidas mensagens do tipo *value?* para os agentes de menor prioridade que necessitam mudar seus valores de variáveis. Se não for possível encontrar atribuições que satisfaçam esta condição, o mediador inicia o processo de mediação emitindo mensagens de *evaluate?* para cada um de seus agentes em sua *good-list*. Ao receber uma mensagem de *evaluate?*, o agente irá responder com informações sobre as variáveis e as restrições não contidas na visão local do agente emissor. Estas informações são retornadas em uma mensagem do tipo *evaluate!*. Contudo, caso o agente já estiver envolvido em outra sessão de mediação ao receber uma mensagem de *evaluate?*, esta solicitação será respondida com uma mensagem de *wait!*. Quando um mediador recebe uma mensagem de *wait!*, o agente emissor desta mensagem é excluído da sessão de mediação.

Após obter resposta de todos os agentes de sua *good-list*, o mediador realiza uma busca por BB centralizada, objetivando a minimização dos custos do subproblema contido na *good-list* e para os agentes fora da sessão. O resultado dos custos do subproblema é armazenado em F_i^*. Caso a mediação seja ativa, mensagens de *value?* são emitidas para os agentes que necessitam rever suas soluções locais.

3.5.4 DPOP

Apesar da maioria dos algoritmos para resolução de problemas de otimização estar baseada em técnicas de backtracking, esta pode não ser a melhor estratégia, uma vez que os agentes podem alterar assincronamente seus valores. Deste modo, outras

estratégias de busca, tais como a programação dinâmica, poderiam ser exploradas de modo a oferecer melhor desempenho em relação aos custos de computação e de comunicação. Em virtude disto, Petcu e Faltings (2005) propuseram um novo algoritmo para DCOP, chamado Dynamic Programming OPtimization (DPOP), que resolve problemas de otimização a um custo linear de mensagens utilizando técnicas de programação dinâmica. A seguir é descrito o funcionamento do algoritmo DPOP.

3.5.4.1 Definição do DPOP

O processo de busca do DPOP constitui-se de três fases. Primeiramente os agentes realizam um pré-processamento para transformar o grafo de restrições em uma pseudo-árvore. Em seguida, os custos de todas as atribuições (para cada agente vizinho) são calculados a partir dos nós-folha da pseudo-árvore e propagados através de uma matriz de utilidade para os nós adjacentes de maior prioridade. Por fim, a última fase tem como objetivo a atribuição dos valores com base nos custos calculados na fase anterior.

Assim como em outros métodos para resolução de um DCOP, o DPOP também necessita que o grafo de restrição seja estruturado em forma de uma pseudo-árvore. Isto é necessário para a estrutura de representação do problema não se tornar cíclica. Outro motivo para explorar a utilização de árvores é a possibilidade de executar buscas em paralelo em estruturas que estão localizadas em ramos independentes.

Durante a fase de cálculo de custos, mensagens do tipo *UTIL* são emitidas dos nós pais para os filhos. Um agente somente poderá emitir uma mensagem de *UTIL* para seu pai quando todas as mensagens de *UTIL* de seus filhos sejam recebidas. Desta forma, o processo de propagação destas mensagens inicia-se a partir dos nós-folhas da pseudo-árvore do problema. Uma mensagem de *UTIL* contém o custo de cada conjunto de atribuições para o subproblema iniciado no agente emissor da mensagem. Desta forma, quando um agente recebe uma mensagem de *UTIL*, é possível calcular seu custo ótimo para cada conjunto de atribuições do subproblema iniciado no agente corrente.

Quando o nó raiz da pseudo-árvore do problema receber todas as mensagens de *UTIL*, o processo de atribuição é iniciado. O nó raiz irá escolher a atribuição que minimize o custo global do problema e transmitirá sua decisão para seus nós-filho através de mensagens do tipo *VALUE*. Os demais agentes somente poderão realizar a atribuição de suas variáveis ao receber uma mensagem do tipo *VALUE* de seu pai. Ao

receber esta mensagem de seu pai, o agente corrente deve escolher uma atribuição que minimize o custo global do problema e propagar esta mensagem aos seus filhos, acoplando sua atribuição ao corpo da mensagem que será retransmitida.

3.5.4.2 Variantes do DPOP

Apesar do DPOP ser capaz de resolver problemas de otimização em tempo polinomial com um número linear de mensagens (em relação ao número de nós do grafo), o tamanho das mensagens também é linear (em relação ao número de arestas do grafo). Neste sentido, em situações onde os problemas são super-restritos, o tamanho das mensagens pode ser consideravelmente grande. Tal aspecto poderia inviabilizar a utilização do DPOP em cenários onde o consumo de memória é limitado. Em função desta deficiência, Petcu e Faltings (2007a) apresentam uma versão híbrida do DPOP, denominada Memory Bound DPOP (MB-DPOP), que permite limitar o consumo de memória exigido. Petcu et al. (2007b) também propuseram uma versão parcialmente centralizada do DPOP para aumentar a eficiência do processo de busca. Esta versão do DPOP, chamada Partial Centralization DPOP (PC-DPOP), une características dos algoritmos DPOP original e do OptAPO descrito na sessão 3.5.3.

3.5.5 NCBB

Embora existam vários algoritmos que resolvem DCOP, muita investigação ainda é necessária para tornar estes algoritmos mais eficientes, ao se tratar de problemas do mundo real. Em vista disto, Chechetka e Sycara (2006) introduziram um novo algoritmo, denominado No-Commitment Branch and Bound (NCBB), cujo desempenho é melhor do que os principais algoritmos propostos até então. O NCBB, assim como diversos outros algoritmos de DCOP, também é baseado na estratégia de busca BB.

No NCBB, os agentes devem ser priorizados em uma estrutura de árvore de busca em profundidade. As restrições são permitidas entre agentes que possuem relacionamento pai e filho na pseudo-árvore do problema. As informações dos agentes são privadas, isto é, um agente somente conhece as restrições entre seus vizinhos, seja seu pai ou seus filhos. Esta etapa de pré-processamento da pseudo-árvore de busca em profundidade permite decompor o problema, visando a exploração do paralelismo em sub-árvores independentes.

3.5.5.1 Definição do NCBB

Ao iniciar o processo de busca, os agentes computam *upper bounds* e *lower bounds* sobre o custo da solução. Os agentes escolhem oportunistamente suas atribuições em função dos valores de seus antecessores, de modo a minimizar seu *AgentCost*, que corresponde ao custo da atribuição para o agente corrente. Estes custos são propagados para os agentes de maior prioridade, os quais são utilizados como heurística para guiar o processo de busca posteriormente.

Após a etapa de pré-processamento, é iniciado o processo de busca. Contudo, os agentes irão iniciar efetivamente o processo de busca apenas quando for recebida uma mensagem de *SEARCH* de seu nó pai. Uma mensagem de *SEARCH* é emitida quando todos os agentes de maior prioridade selecionam seus valores e os comunicam para seus descendentes. Isto assegura que, ao receber uma mensagem do tipo *SEARCH*, o agente terá um conhecimento consistente sobre os valores de seus agentes antecessores.

A principal característica do mecanismo de busca do NCBB está na aprendizagem sobre as atribuições de seus antecessores. Sempre que um agente recebe uma mensagem com o valor atribuído de um dado antecessor, esta mensagem é imediatamente respondida com o valor do *lower bound* do agente descendente recalculado com a nova atribuição baseada em sua *AgentCost*. Assim, um agente x pode computar seu $LB(x, context_x, k)$ para cada atribuição k. Tal estratégia permite aos agentes criarem mapas de custo para cada um de seus valores. Também é possível reconsiderar soluções anteriormente computadas e consideradas como insatisfatórias, limitando a atribuição dos valores que ainda não foram explorados em sua sub-árvore.

3.6 Considerações Finais

Problemas de otimização distribuída de restrição têm emergido como um poderoso formalismo para resolução de uma vasta classe de problemas combinatórios, em especial problemas de escalonamento, planejamento e otimização. Devido às características de um DCOP, inúmeros problemas complexos do mundo real podem ser naturalmente modelados neste paradigma, instigando muitos pesquisadores ao tema.

Diversos algoritmos para DCOP têm sido propostos até então. De uma maneira geral, cada algoritmo apresenta particularidades em sua sistemática de busca. A escolha de um algoritmo deve, portanto, ser orientado às características do problema e do ambiente em questão. Duas métricas podem ser empregadas para avaliar a eficiência de

um algoritmo sobre um dado problema: o custo de comunicação e o custo de processamento. Problemas super-restritos podem ter um grande volume de comunicação, favorecendo o uso de algoritmos que minimizam a quantidade de mensagens transmitidas. Em contrapartida, em ambientes cuja capacidade de processamento é estritamente limitada, recomenda-se o uso de algoritmos que minimizam a computação local.

A redução do consumo de combustível em conduções de composições férreas, objeto de estudo deste trabalho, é um exemplo de problema do mundo real que pode ser modelado como um DCOP. No capítulo a seguir é apresentada a formulação para problema de redução do consumo de combustível como um DCOP.

Capítulo 4

Um Esquema para Redução do Consumo de Combustível em Sistemas de Condução Férrea Baseado em Otimização Distribuída de Restrição

Conforme visto no capítulo 2, prover mecanismos para a redução no consumo de combustível em sistemas de condução férrea é uma tarefa de elevada complexidade devido à dinâmica dos fenômenos envolvidos. Entretanto, o propósito deste trabalho é comprovar que a aplicação de métodos da Inteligência Artificial Distribuída, sobretudo técnicas de otimização distribuída de restrição, aliada aos modelos matemáticos para redução do consumo de combustível proporciona soluções adequadas para o problema. Deste modo, o presente capítulo tem como objetivo apresentar uma arquitetura baseada em otimização distribuída de restrição capaz de alcançar uma eficiência energética satisfatória em sistemas de condução férrea.

4.1 Modelagem do Problema de Condução Férrea como um DCOP

O DCOP é um poderoso formalismo capaz de modelar naturalmente problemas de otimização do mundo real. Entretanto, em algumas situações isto não é uma tarefa trivial, pois este formalismo requer que sejam abstraídas do problema definições como: ambiente, variáveis do problema, domínio das variáveis, dependências entre as variáveis e funções de custo. Portanto, a modelagem de um problema do mundo real como um DCOP pode não ser algo muito óbvio.

A condução de composições férreas pode ser classificada como um problema complexo de coordenação. Em termos gerais, o maquinista deve tomar suas ações considerando um grande número de variáveis e restrições, tais como tempo de viagem, segurança, clima, desgaste dos componentes do veículo e da via permanente, consumo energético, dentre outras. Além do grande número de variáveis envolvidas, cada ação tomada influencia diretamente as ações futuras, o que aumenta vertiginosamente a complexidade do problema. A seguir será apresentada a estratégia proposta para modelar o problema de condução férrea como um DCOP.

4.1.1 Variáveis e Domínio

A estratégia adotada para modelar o problema de condução de composições férreas como um DCOP consiste em definir lacunas de tempo como variáveis (MAHESWARAN, 2004). Esta estratégia tem como objetivo segmentar um plano de viagem em sub-planos, de modo a reduzir a complexidade do problema (Figura 5). Neste caso, para cada sub-plano, é determinado um intervalo de tempo T definido por $[T_{inicial}, T_{final}]$, onde eventos $E = \{E_1, E_2, ..., E_k\}$ serão escalonados. Por sua vez, para cada intervalo de tempo T, existe $t \in \mathcal{T} = \{1, 2, ..., \Delta T\}$, onde $\Delta T = T_{final} - T_{inicial}$. Cada agente t_n poderá atribuir para sua variável um único evento. Um evento $E_k \in E$ é definido pela tupla (A_k, L_k, V_k), onde A_k corresponde aos recursos requeridos, L_k é o número de lacunas de tempo contíguas requeridas e V_k representa um vetor com a importância de cada recurso.

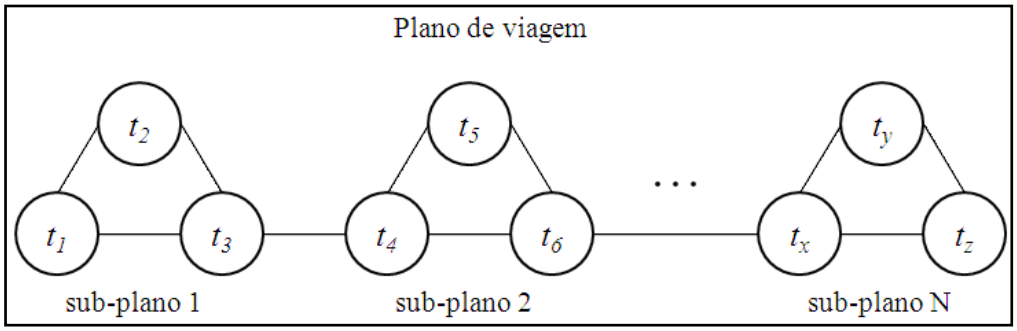

Figura 5. Decomposição do plano de viagem em sub-planos.

Para o contexto do problema de condução férrea, A_k representa as locomotivas que necessitam alterar seus pontos de aceleração, L_k corresponde ao número de janelas contíguas de tempo nas quais a composição férrea irá manter a configuração atual dos pontos de aceleração e V_k é um vetor que contém os respectivos pontos de aceleração de cada locomotiva em A_k. Por sua vez, um evento E_k pode ser caracterizado como aumento, redução ou manutenção dos pontos de aceleração ou frenagem das locomotivas.

Portanto, para a estratégia adotada, as variáveis representam os períodos em que cada ação será tomada durante a condução férrea. Deste modo, o domínio das variáveis é dado por $d_i = \{0, 1, 2, 3\}$, onde estes valores representam as ações de aumentar, reduzir, manter potência ou frear as locomotivas respectivamente. Neste sentido, um evento E_k consiste na atribuição de um valor $v_i \in d_i$ para o intervalo de tempo t_k. Além

disso, cada evento E_k possui um custo que representa a utilidade da ação realizada por tal evento. Com isto, esta estratégia permite otimizar o escalonamento dos eventos sobre uma janela de planejamento em função da utilidade de cada evento alocado.

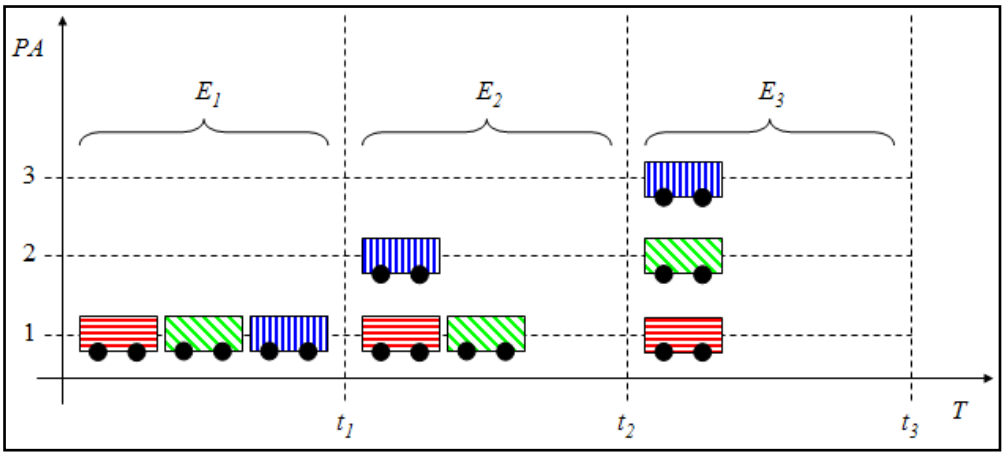

Figura 6. Eventos escalonados sobre uma janela de planejamento.

A Figura 6 ilustra um exemplo de uma janela de planejamento utilizando a estratégia proposta, composta por três locomotivas e por três lacunas de tempo. Na primeira lacuna de tempo, foi escalonado o evento E_1 onde todas as locomotivas mantiveram-se no primeiro ponto de aceleração (PA) durante o intervalo de tempo t_1. Na seqüência, o evento E_2 foi alocado na lacuna de tempo t_2 e uma das locomotivas alterou para o segundo ponto de aceleração. Por fim, no evento E_3 as locomotivas estão no primeiro, segundo e terceiro ponto de aceleração respectivamente, permanecendo nesta configuração durante o intervalo de tempo t_3.

4.1.2 Dependência entre as Variáveis

Não obstante, a utilidade de um determinado evento pode ser influenciada pelos demais eventos contidos no mesmo sub-plano. Isto ocorre devido à dependência existente entre as variáveis de um mesmo sub-plano, cujo objetivo é avaliar o impacto de uma determinada ação sobre as ações futuras. A dependência entre as variáveis pode ser considerada como um dos aspectos chave da estratégia adotada, pois este recurso evita situações onde são atribuídas ações localmente ótimas, mas que possuem um impacto negativo sobre a qualidade global da solução. Em outras palavras, a dependência entre as variáveis permite que os eventos escalonados sejam avaliados globalmente sobre o problema.

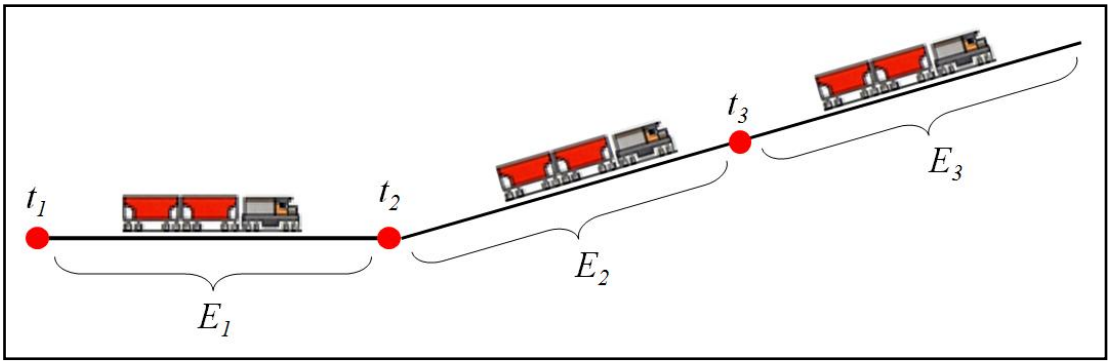

Figura 7. Exemplo de dependência entre os eventos escalonados.

A Figura 7 ilustra a importância da dependência entre os eventos escalonados para a modelagem DCOP proposta. No intervalo de tempo t_1, a composição férrea está localizada em uma região plana e reta da ferrovia. Adiante, no intervalo de tempo t_2, a mesma composição férrea efetuou um deslocamento durante o intervalo de tempo t_1 e neste momento se encontra no início de uma rampa ascendente. E finalmente, no intervalo de tempo t_3, novamente houve um deslocamento durante o evento t_2 e a composição está situada ao final da rampa ascendente. Caso a ação atribuída no intervalo de tempo t_1 seja localmente ótima, por exemplo, manter a velocidade constante de modo a consumir menos combustível, as ações t_2 e t_3 irão demandar um consumo muito maior em função do aumento da resistência ao movimento ocasionado pela rampa ascendente, onerando a qualidade global da solução. Nesta situação, uma solução globalmente ótima consiste em aumentar gradualmente a potência das locomotivas a partir do intervalo de tempo t_1, de modo que a composição férrea produza potência suficiente para percorrer o trecho em aclive nos intervalos de tempo t_2 e t_3.

4.1.3 Ambiente e Funções de Custo

Para a estratégia proposta, as funções de custo são binárias e avaliam a utilidade dos eventos escalonados. Todavia, para calcular o impacto de um determinado evento sobre o problema é necessário conhecer informações referentes à situação atual da composição férrea. Estas informações compõem o ambiente do problema e são caracterizadas como: velocidade e localização atual da composição férrea sobre a ferrovia, resistência total ao movimento da composição férrea, consumo de combustível e outros aspectos relacionados à dinâmica do movimento de trens. Em adição, estas informações são temporais, no sentido que após cada evento ser escalonado é necessário atualizar o ambiente do problema. Em outras palavras, cada evento escalonado irá

implicar em um deslocamento, variação da velocidade e consumo de combustível, acarretando na atualização das informações dinâmicas contidas no ambiente. Este artifício é necessário para que o próximo evento a ser escalonado disponha de informações atualizadas sobre a situação atual da composição férrea.

Portanto, a utilidade de cada evento escalonado é calculada através do impacto de tal evento sobre o problema. Para isto, o cálculo da utilidade de cada evento é realizado ponderando três atributos: *velocidade da composição férrea ao final do evento*, *deslocamento efetuado* e o *consumo previsto pelo evento*. Cada atributo está associado a um peso, P_v, P_d e P_c respectivamente. Deste modo, a utilidade do evento é calculada através da soma ponderada de cada atributo.

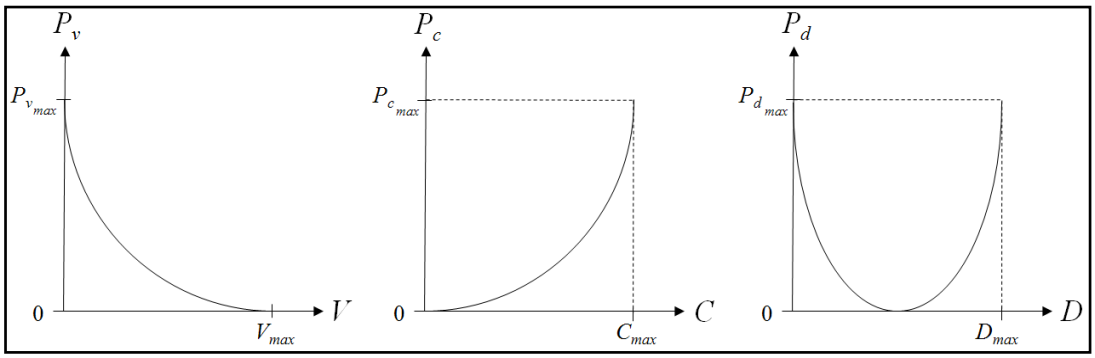

Figura 8. Representação do cálculo dos pesos dos atributos da função de custo.

A Figura 8 descreve o modelo utilizado para obter os pesos sobre cada atributo, onde *V*, *C* e *D* representam respectivamente a velocidade da composição férrea ao final da ação, o consumo de combustível previsto e o deslocamento a ser efetuado pela ação. Todavia, para calcular os pesos é necessário conhecer a velocidade máxima permitida da ferrovia (V_{max}) e estabelecer um consumo (C_{max}) e deslocamento máximo (D_{max}) por ação. Desta forma, ao analisar a Figura 8, é possível concluir que um evento ótimo irá resultar em uma velocidade próxima à velocidade máxima da ferrovia, um baixo consumo de combustível e um deslocamento próximo ao deslocamento médio por ação.

Uma das vantagens deste modelo para cálculo da utilidade dos eventos escalonados é a possibilidade de alterar o objetivo do problema apenas ajustando os pesos das funções de custo. Por exemplo, para obter eventos que priorizem um baixo consumo de combustível, basta tornar proeminente o peso referente ao consumo de combustível em relação aos demais. Entretanto, as funções de custo requerem diversas informações contidas na dinâmica do movimento de composições férreas, incluindo informações relacionadas à geografia da ferrovia e às características dos veículos que

compõe o trem. Deste modo, para calcular a utilidade dos eventos escalonados é fundamental o prévio mapeamento do perfil da ferrovia e dispor de informações pertinentes à composição férrea em questão.

4.2 Framework DCOP

Conforme dito anteriormente, um dos objetivos deste trabalho é o desenvolvimento de um módulo de controle inteligente para sistemas de condução férrea no que concerne o consumo de combustível, permitindo uma integração com os demais sistemas residentes nos computadores de bordo das locomotivas. Entretanto, os computadores de bordo das locomotivas são equipamentos severamente restritos com relação à capacidade de processamento e de armazenamento de dados. Em função disto, os algoritmos para DCOP devem priorizar o uso eficiente dos recursos computacionais disponíveis nos computadores de bordo.

Neste contexto, para a resolução do DCOP proposto para o problema de condução férrea foram escolhidos os algoritmos ADOPT e o DPOP. Esta escolha foi realizada baseada nas características do ambiente, visto que tais algoritmos priorizam a redução da computação local e do uso de memória (no caso do ADOPT) e a quantidade de ciclos e mensagens trocadas entre os agentes (no caso do DPOP). Outro fator que contribuiu para esta escolha foram as diferentes estratégias de busca implementadas por cada algoritmo. A grande maioria dos algoritmos para DCOP atuais são baseados em estratégias de *backtracking*, produzindo desempenhos similares na maioria dos casos. Por outro lado, o DPOP implementa uma busca baseada em programação dinâmica, a qual apresenta desempenhos e características peculiares.

Apesar da existência de diversos frameworks genéricos para resolução de DCOP, tais como DiMES (MAHESWARAN, 2004), FRODO (PETCU, 2006) e DCOPolis (SULTANIK, 2007), foi necessário o desenvolvimento de um novo framework devido ao fato das ferramentas atuais serem incapazes de modelar problemas do mundo real com tal nível de complexidade. Os frameworks para DCOP propostos até então apresentam diversas limitações no que tange à especificação do problema, as quais comumente estão associadas à definição das funções de custo. De maneira geral, a definição das funções de custo por tais frameworks é dada pela tupla $\{(x_i,v_i), (x_j,v_j), c\}$, onde os agentes x_i e x_j possuem restrições em comum e c é o custo quando x_i atribui o valor v_i e x_j atribui o valor v_j. Para o problema de condução de composições férreas, as

funções de custo envolvem cálculos complexos da dinâmica do movimento de trens, sendo de fato indispensável o desenvolvimento de um novo framework para DCOP que permita integrá-lo a rotinas mais sofisticadas.

Neste sentido, foi concebido um novo framework para DCOP capaz de integrar rotinas complexas às funções de custo definidas para o problema. Este framework foi desenvolvido utilizando a linguagem de programação Java, onde as funções de custo são classes incorporadas à ferramenta. Deste modo, com o framework proposto é possível modelar um DCOP cujo cálculo das funções de custo requer o uso de rotinas externas ou a integração com outra aplicação. Esta integração pode ser realizada via *socket*, *WebServices*, chamadas remota de procedimentos (RPC) ou qualquer outro modelo de integração suportado pela linguagem de programação Java.

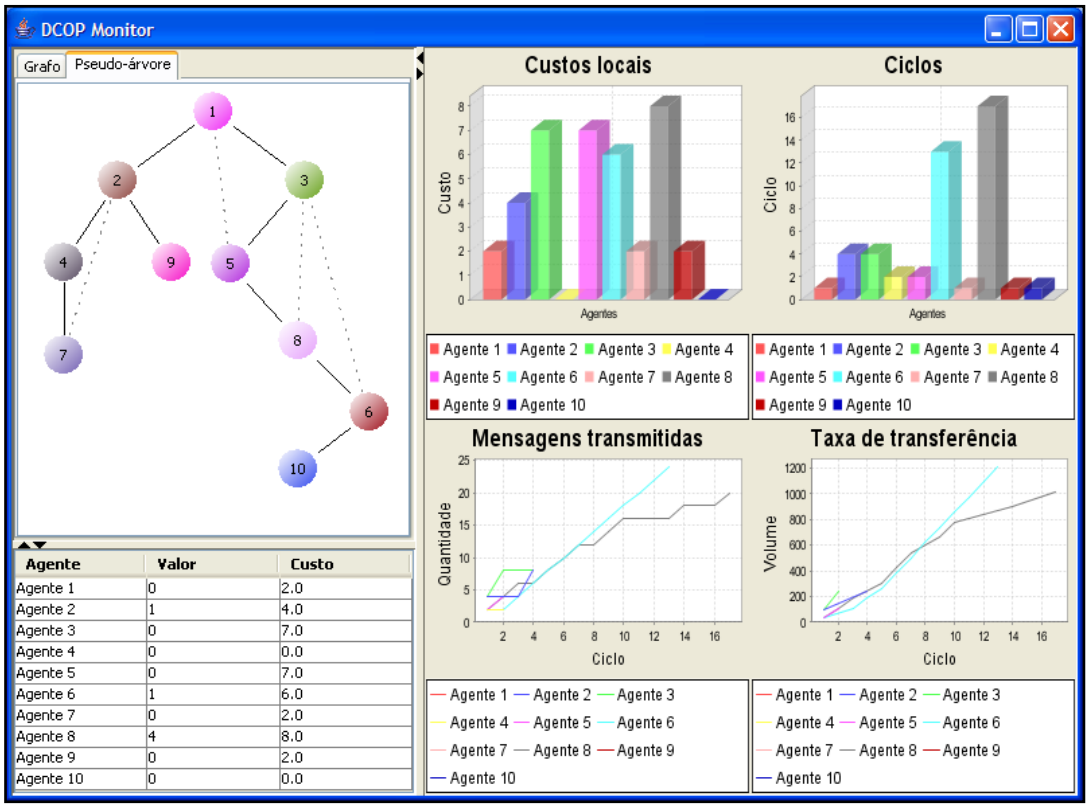

Figura 9. Interface gráfica do framework para DCOP desenvolvido.

De fato, uma das principais características do framework proposto é a possibilidade de modelar um problema do mundo real como um DCOP e avaliar a qualidade das soluções e o desempenho de múltiplos algoritmos. Inicialmente este framework dispõe apenas dos algoritmos ADOPT e DPOP. Contudo, sua arquitetura foi projetada para suportar diferentes algoritmos, independente das particularidades

existentes em cada um. Outro importante recurso que este framework provê é uma interface gráfica (Figura 9) que fornece diversas informações sobre o problema modelado nesta ferramenta. Este recurso provê uma representação gráfica do grafo de restrições do problema e informações em tempo real referentes ao desempenho dos algoritmos, tais como o número de ciclos e mensagens trocadas e o volume de dados transmitidos entre os agentes. Com isto, é possível acompanhar a evolução do processo de busca em termos de desempenho durante a execução dos algoritmos.

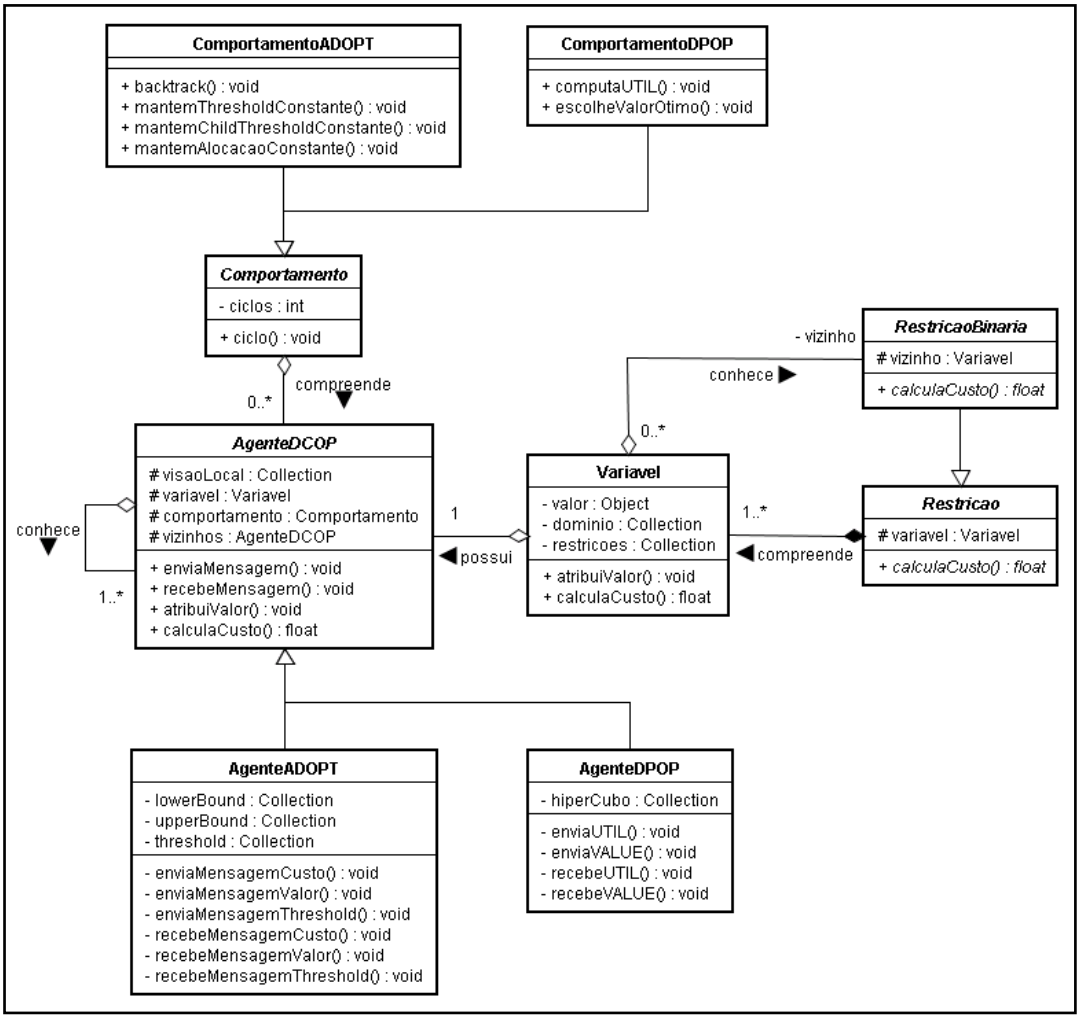

Figura 10. Diagrama de classes do framework para DCOP proposto.

A Figura 10 descreve o diagrama de classes do framework para DCOP proposto. Para modelar um DCOP neste framework, devem ser instanciadas as variáveis do problema (classe *Variavel*) e suas restrições (classes *Restricao* ou *RestricaoBinaria*). Para cada instância de *Variavel*, é necessário definir um domínio discreto e finito, cujos valores podem ser qualquer objeto válido para a linguagem de programação Java. Já as funções de custo devem ser especificadas através de subclasses de *Restricao* para

restrições unárias ou *RestricaoBinaria* para restrições binárias, onde as rotinas para cálculo do custo devem ser implementadas pelo método *calculoCusto*. Por sua vez, cada algoritmo implementado no framework deve possuir uma sub-classe de *AgenteDCOP* e uma sub-classe de *Comportamento*. Neste caso, as sub-classes de *AgenteDCOP* devem conter as estruturas necessárias para cada algoritmo. Finalmente, as sub-classes de *Comportamento* definem as ações que os agentes irão realizar para encontrar uma solução ótima para um problema. Em outras palavras, cada sub-classe *Comportamento* deve implementar a estratégia de busca do algoritmo para DCOP em questão.

4.3 Sistema Multi-agente para o Problema de Condução Férrea

Para a concepção do módulo de controle inteligente para condução férrea, foi necessário o desenvolvimento de um sistema multi-agente (SMA) cujo modelo de coordenação é baseado em técnicas de DCOP. Contudo, o desenvolvimento de um SMA não se restringe apenas ao modelo de coordenação das ações realizadas pelos agentes. Aspectos como o ciclo de vida, relacionamento e comportamento dos agentes, modelo de comunicação e de interação também devem ser considerados durante a concepção de um SMA. Em adição, o problema de condução férrea também requer a implementação dos modelos matemáticos para simulação do movimento de trens.

4.3.1 Arquitetura do SMA Proposto

A arquitetura do SMA desenvolvido pode ser segmentada em quatro módulos (Figura 11): gerenciamento dos agentes, infra-estrutura de cálculos, framework para DCOP e estratégia do controlador. O módulo de gerenciamento dos agentes provê recursos para gerenciar o ciclo de vida dos agentes e toda a infra-estrutura necessária para o modelo de comunicação. Para tal, foi utilizado um middleware para desenvolvimento de SMAs chamado JADE (JADE, 2008). Esta ferramenta oferece um ambiente completo para execução, desenvolvimento e depuração de SMAs. Além disso, o JADE permite o desenvolvimento de SMAs totalmente distribuídos de maneira transparente, além de fornecer um modelo de comunicação sofisticado seguindo as especificações FIPA[1]. Outros recursos úteis oferecidos pela ferramenta são a localização de um agente contido na plataforma do SMA e o gerenciamento dos serviços providos pelos agentes.

[1] FIPA: Foundations for Intelligent Physical Agents – www.fipa.org

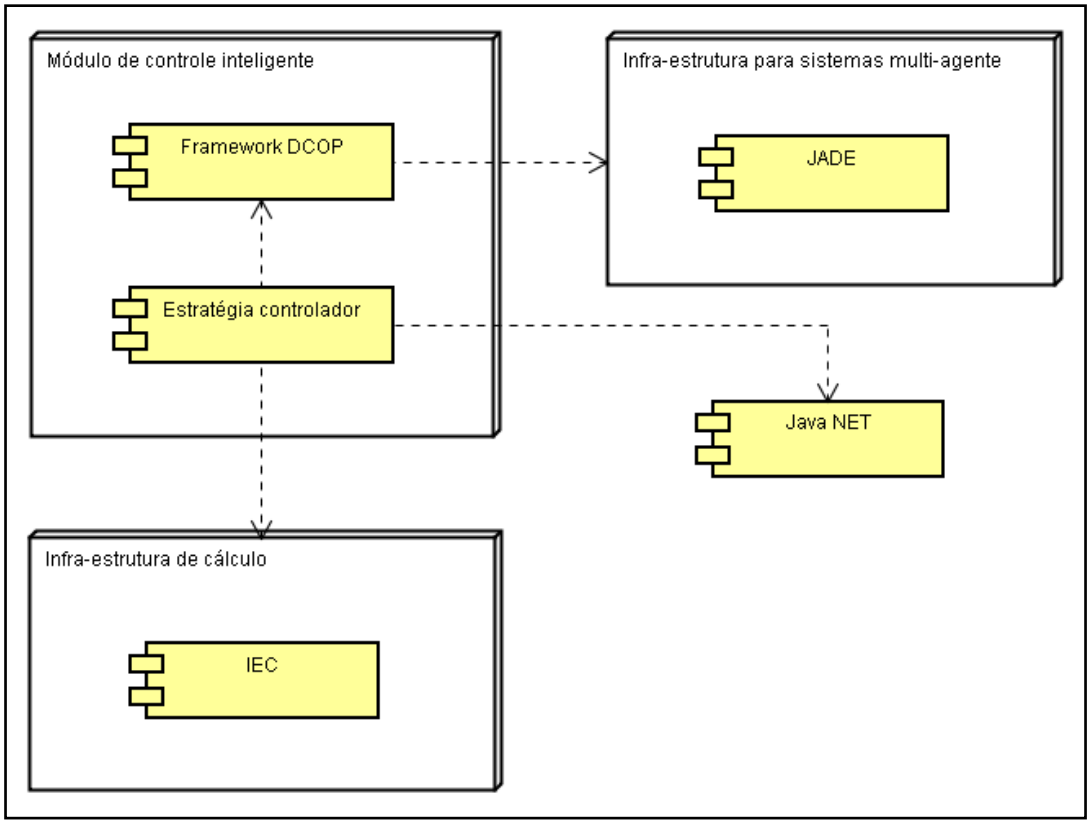

Figura 11. Diagrama de arquitetura do SMA para condução férrea.

4.3.2 Infra-estrutura de Cálculos para Simulação do Movimento de Trens

O módulo de infra-estrutura de cálculos compreende toda a dinâmica envolvida para a simulação do movimento dos trens. Este módulo de infra-estrutura de cálculos, nomeado IEC, é um componente do projeto PAI-L (Piloto Automático Inteligente para Locomotivas) financiado pela FINEP convênio 3560/06, o qual permanece em desenvolvimento pelos integrantes do Laboratório de Agentes de Software da Pontifícia Universidade Católica do Paraná.

De fato, para realizar os cálculos matemáticos envolvidos na dinâmica do movimento de trens, a IEC requer um conjunto de informações referentes à geometria da ferrovia e às características da composição férrea em questão. Estas informações são fornecidas à ferramenta por meio de arquivos estruturados no formato XML[2]. A Figura 12 ilustra o diagrama de classe adotado para a representação de uma ferrovia. Para a IEC, uma via férrea é dividida em vários trechos denominados pontos de medida, onde cada trecho possui aproximadamente 20 metros de comprimento. Cada ponto de medida possui informações sobre sua localização, tais como latitude, longitude e altitude. Além

[2] XML: Extensible Markup Language – www.w3.org/XML

disto, um ponto de medida também detém informações sobre a geometria do trecho em questão, tais como percentual de rampa ou raio da curva. Através destas informações, a IEC é capaz de calcular a resistência acidental de uma composição férrea a partir de sua localização sobre a ferrovia.

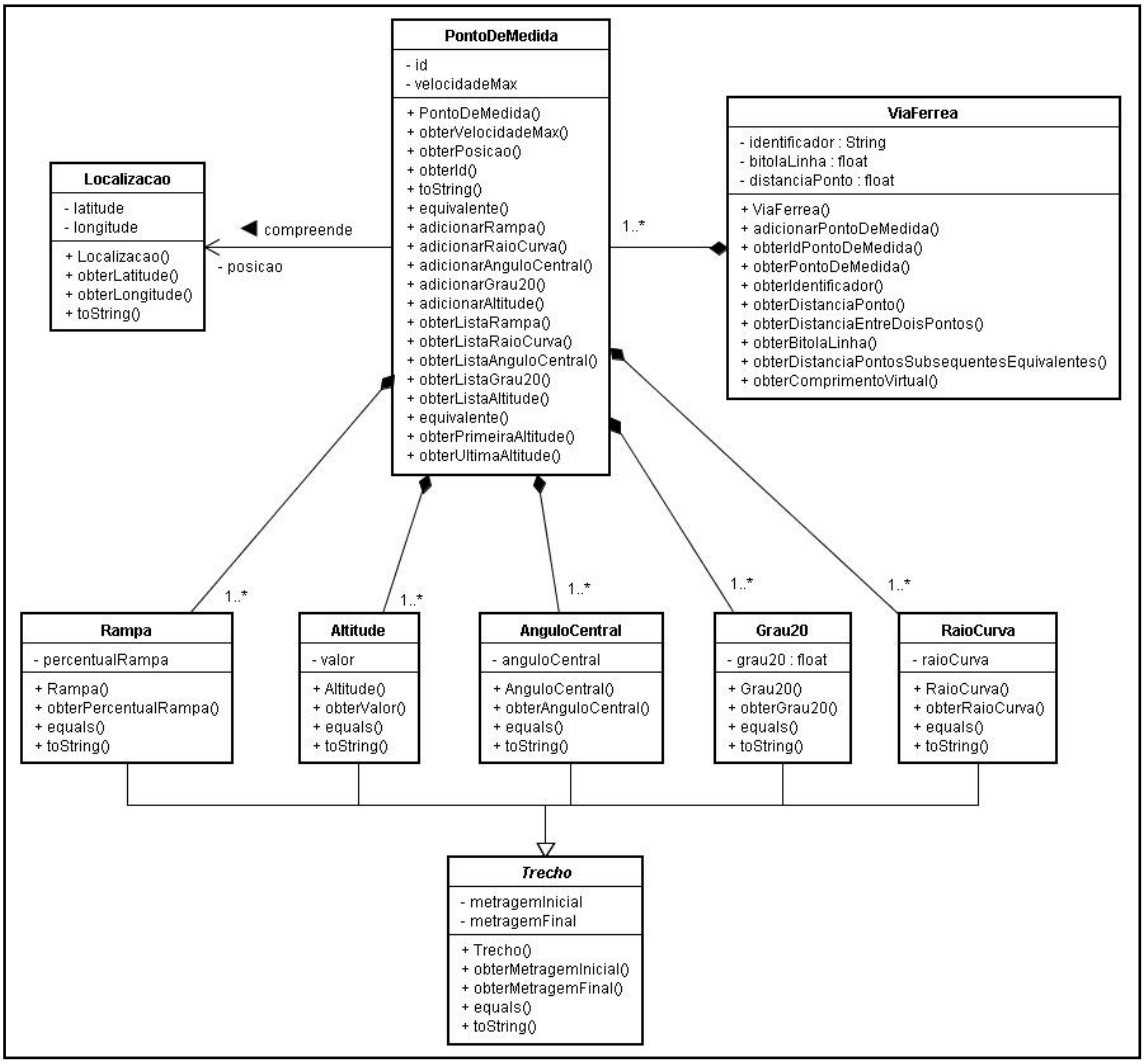

Figura 12. Diagrama de classe para representação da ferrovia.

Na seqüência, a Figura 13 apresenta o diagrama de classe adotado para representação de uma composição férrea. Uma composição férrea é constituída por vários veículos, os quais podem ser classificados como vagões de carga ou locomotivas. Cada veículo possui informações específicas de acordo com as características de seu modelo. Neste sentido, cada vagão pode conter cargas diferentes e cada locomotiva pode gerar potências diferentes. Por sua vez, uma locomotiva possui um conjunto de equipamentos que oferecem informações sobre o estado atual da viagem, tais como distância percorrida, latitude, longitude e velocidade da composição férrea.

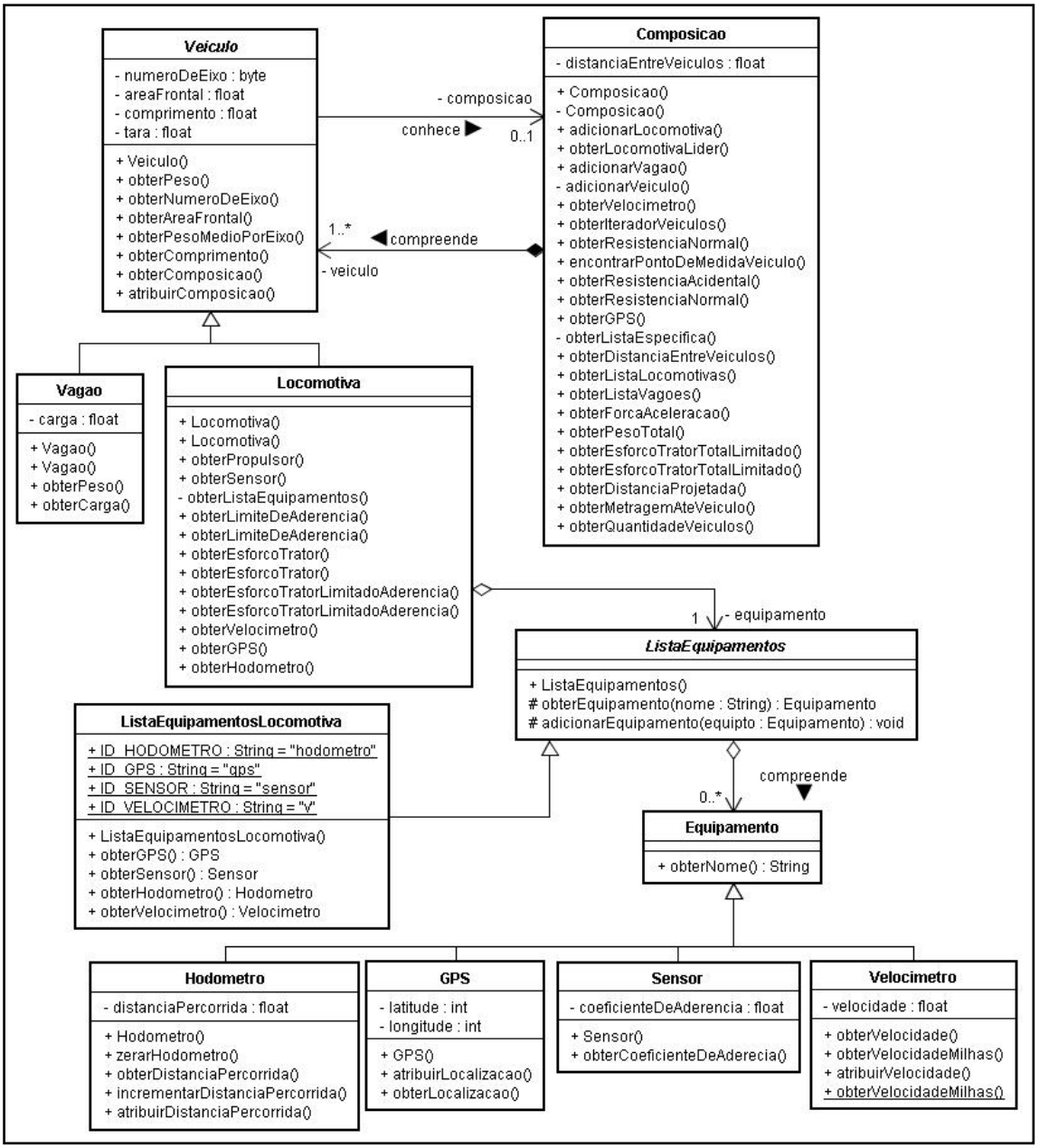

Figura 13. Diagrama de classe para representação da composição férrea.

Em termos gerais, as informações referentes à composição férrea aliadas às informações sobre o perfil da ferrovia permitem realizar os cálculos matemáticos da dinâmica do movimento dos trens. Para tal, a IEC dispõe de uma interface de requisição bem definida, onde é descriminada a situação atual e a ação a ser desempenhada pela composição férrea, retornando o impacto de tal ação. Especificamente, as informações requeridas pela IEC são: a localização da composição férrea sobre a ferrovia, velocidade atual, combinação de pontos de aceleração e a variação de velocidade a ser realizada. Como resultado, a IEC retorna o deslocamento, tempo e o consumo de combustível previsto para realizar tal ação.

De fato, a interface de requisição da IEC é um dos recursos mais importantes para o módulo de controle inteligente desenvolvido. Sua importância é devido ao fato da IEC ser desenvolvida utilizando a linguagem de programação C++, enquanto que o módulo de controle inteligente foi desenvolvido na linguagem de programação Java. Desta maneira, as requisições remotas entre o módulo de controle inteligente e a IEC ocorrem através de uma conexão via *socket*. Em outras palavras, a IEC atua como um servidor, fornecendo toda a infra-estrutura de cálculos necessária para a simulação da dinâmica do movimento de composições férreas.

4.3.3 Implementação do SMA para Condução Férrea

Conforme já mencionado, o framework para DCOP desenvolvido apenas fornece um modelo para coordenação das ações entre os agentes. Entretanto, o framework requer a definição das variáveis envolvidas e seus respectivos domínios, bem como o relacionamento entre as variáveis e a especificação das funções de custo envolvidas na modelagem DCOP. No contexto da condução de composições férreas, as funções de custo estão associadas à dinâmica de movimento de trens e ao cálculo da utilidade das ações avaliadas. Não obstante, as informações relacionadas à simulação do movimento de trens são obtidas com a IEC. Por sua vez, as funções de custo têm como objetivo calcular a utilidade das ações requisitadas à IEC e escolher a melhor ação baseando-se no modelo de avaliação descrito na sessão 4.1.3 deste capítulo.

O Algoritmo 1 descreve sucintamente as etapas realizadas pelo módulo de controle para gerar e avaliar as ações para o problema da condução férrea. A primeira etapa do processo consiste em gerar um DCOP para cada sub-plano. O processo de geração do DCOP envolve a criação das variáveis e a definição de seus domínios e suas restrições no framework, representados pelas funções *CriaVariavel*, *DefineDominio* e *DefineRestricao*. Há também uma variável chamada *TamanhoSubPlano*, que define o número de lacunas de tempo para cada sub-plano. Por fim, a execução do DCOP no framework é repesentada pela função *ExecutaDCOP*. É possível notar que a geração de sub-planos ocorre enquanto o deslocamento acumulado, representado pela função *RetornaDeslocamentoAmbiente*, for menor que a distância prevista para a viagem, representada pela variável *FinalViagem*. Portanto, para cada sub-plano, será executado um DCOP para obter o resultado para uma fração da viagem, de modo que cada ação contida no sub-plano será avaliada pela função *CalculaCusto*.

Algoritmo 1. Módulo de controle inteligente para condução férrea.

```
1.    enquanto RetornaDeslocamentoAmbiente() < FinalViagem faça
2.       para i <= TamanhoSubPlano faça
3.          variaveis[i] ← CriaVariavel();
4.          DefineDominio(variaveis[i], {0, 1 , 2, 3});
5.          se i > 1 então
6.             DefineRestricao(variaveis[i], variaveis[i – 1]);
7.          fim se
8.       fim para
9.       ExecutaDCOP();
10.   fim enquanto
fim
```

```
função CalculaCusto(d)
1.    utilidade ← ∞;
2.    custo ← ∞;
3.    PAs ← {};
4.    melhorVelocidade ← 0;
5.    melhorDeslocamento ← 0;
6.    melhorPA ← ⊥;
7.    posicao ← RetornaDeslocamentoAmbiente();
8.    paAnterior ← RetornaPAAmbiente();
9.    velocidadeAnterior ← RetornaVelocidadeAmbiente();
10.   velocidade ← velocidadeAnterior – 1;
11.   velocidadeLimite ← velocidadeAnterior + 1;
12.   se velocidadeLimite > VelocidadeMaximaViaFerrea() então
13.      velocidadeLimite ← VelocidadeMaximaViaFerrea();
14.   fim se
15.   para velocidade < velocidadeLimite faça
16.      PAs ← RetornaCombinacaoPA(paAnterior,d);
17.      para pa ∈ PAs faça
18.         deslocamento,consumo ← IEC(posicao,velocidade,pa);
19.         se velocidade e pa implicar em uma ação inválida então
20.            utilidade ← ∞;
21.         senão
22.            utilidade ← CalculaUtilidade(velocidade,deslocamento,consumo);
23.         fim se
24.         se utilidade < custo então
25.            custo ← utilidade;
26.            melhorVelocidade ← velocidade;
27.            melhorDeslocamento ← deslocamento;
28.            melhorPA ← pa;
29.         fim se
30.      fim para
31.      velocidade ← velocidade  + 0.1;
32.   fim para
33.   AtualizaAmbiente(melhorVelocidade,melhorDeslocamento,melhorPA);
34.   retorna custo;
fim função
```

A função *CalculaCusto* recebe como argumento a ação que será realizada pela composição férrea, isto é, aumentar, reduzir, manter potência ou frear as locomotivas. Logo no início da execução, são obtidas algumas informações contidas no ambiente, tais como a velocidade, combinação de pontos de aceleração e posição atual da composição férrea sobre a via, retratadas respectivamente pelas funções *RetornaDeslocamentoAmbiente*, *RetornaVelocidadeAmbiente* e *RetornaPAAmbiente*. Em seguida, é definido um intervalo para a variação da velocidade limitado à velocidade máxima permitida para o trecho da ferrovia, retratado por *VelocidadeMaximaViaFerrea*. Após isto, é obtido o conjunto de combinações de pontos de aceleração baseado na ação *d* recebida como argumento. Por exemplo, dada uma composição férrea com duas locomotivas, caso a combinação de pontos de aceleração da última ação realizada for {1, 1} e ação *d* for aumentar potência, o conjunto de combinações de pontos de aceleração obtido será {{1,2},{2,1},{2,2}}. Esta regra é implementada pela função *RetornaCombinacaoPA*, que recebe como argumento a combinação dos pontos de aceleração da ação anterior e a ação que será realizada, retornando um conjunto de combinações de pontos de aceleração para a ação em questão (aumentar, reduzir ou manter potência ou frear).

Na seqüência, será realizada uma iteração para cada combinação de pontos de aceleração e avaliada a utilidade desta ação. Para isto é realizada uma chamada remota à IEC informando a localização atual da composição férrea, a velocidade desejada e a combinação de pontos de aceleração. Como resposta, a IEC retorna o deslocamento e o consumo previsto para a ação em questão. Contudo, nem todas as ações são válidas, como por exemplo, aumentar a velocidade sem que haja potência suficiente. Além disto, há ações que são consideradas inválidas devido à violação de regras de segurança, como por exemplo, a geração de potência em excesso, causando a patinagem das rodas da locomotiva. Estas consistências sobre as ações são representadas no algoritmo pela condição localizada na linha 19. Neste caso, para as ações inválidas, independente da natureza, é atribuída a utilidade ∞. Do contrário, será obtida a utilidade da ação através do modelo para avaliação das ações proposto, representado no algoritmo pela função *CalculaUtilidade*. Ao final da função *CalculaCusto*, será escolhida a ação que resulta na melhor utilidade e retornado o custo desta ação para o algoritmo DCOP realizar o processo de busca. Todavia, antes de encerrar a execução desta função, é necessário atualizar as informações contidas no ambiente para a próxima ação a ser avaliada. A Figura 14 descreve de maneira resumida as etapas envolvidas na função *CalculaCusto*.

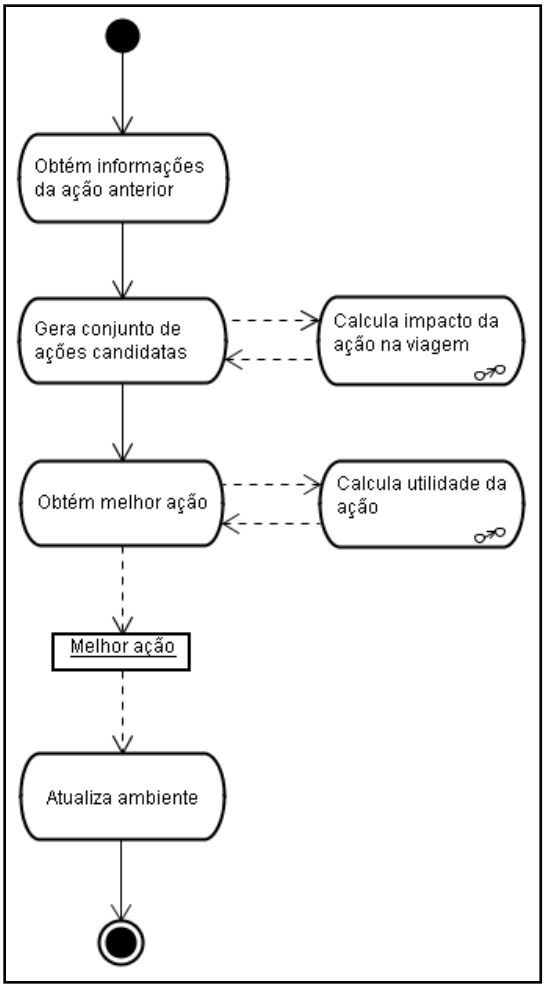

Figura 14. Etapas da função de custo para a avaliação da utilidade das ações.

4.4 Considerações Finais

De fato, o desenvolvimento de um módulo de controle inteligente para condução férrea é uma atividade de alta complexidade, em função do grande número de variáveis e restrições envolvidas no problema. Não obstante, acredita-se que a aplicação de métodos da IAD em conjunto aos modelos matemáticos para redução do consumo de combustível proporcione soluções satisfatórias para o problema, caracterizando o propósito deste trabalho. Para alcançar este objetivo, o módulo de controle inteligente proposto dispõe de uma estratégia eficiente para o planejamento da condução férrea, cuja premissa consiste em reduzir o consumo de combustível. Esta estratégia compõe o núcleo deste trabalho, o qual implementa técnicas de DCOP para obter a eficiência energética em sistemas de transporte ferroviário. Deste modo, o capítulo a seguir irá apresentar os resultados obtidos a partir do módulo de controle inteligente para condução férrea proposto.

Capítulo 5

RESULTADOS OBTIDOS

O presente capítulo tem como objetivo apresentar a metodologia aplicada para a avaliação do módulo de controle inteligente para condução férrea proposto, bem como os resultados obtidos. Além disso, também é apresentada uma comparação entre os algoritmos para DCOP utilizados em termos de desempenho. Por fim, é encerrado este capítulo com uma discussão final sobre o esquema proposto para redução do consumo de combustível em sistemas de condução férrea.

5.1 Metodologia de Avaliação

O módulo de controle inteligente para condução férrea proposto por este trabalho utiliza técnicas de DCOP como principal estratégia para alcançar a eficiência energética no transporte ferroviário. Para avaliar a modelagem DCOP proposta, foram efetuados diversos experimentos em oito cenários diferentes. Estes cenários reproduzem fielmente viagens reais de uma composição férrea realizada, considerando o mesmo perfil da ferrovia e as mesmas características dos veículos. Deste modo, para cada cenário, foram executadas quatro diferentes simulações, onde foram utilizados sub-planos de tamanhos diferentes, contendo respectivamente 3, 5, 7 e 9 lacunas de tempo.

Por sua vez, a avaliação da qualidade das soluções obtidas foi realizada comparando os resultados encontrados com os dados de viagens reais conduzidas por maquinistas que não dispõem de recursos auxiliares. Para tal, foi utilizado como métrica o consumo de combustível previsto para cada viagem. A medida utilizada para comparar os resultados foi o consumo em litros por tonelada bruta transportada, ou LTKB, que corresponde à quantidade de combustível consumido para transportar uma determinada carga em um determinado percurso, conforme a Equação 16.

$$LTKB = \frac{C}{\frac{D}{1000} \text{ x } P} \text{ x } 1000 \tag{16}$$

Onde:

C = Consumo em litros

P = Peso em toneladas

D = Deslocamento em metros

Além da avaliação da modelagem DCOP proposta, também foi comparado o desempenho entre os algoritmos DCOP utilizados. Para a comparação de desempenho dos algoritmos, foram utilizadas como métricas a quantidade de ciclos e de mensagens transmitidas, o volume de dados transmitidos e o tamanho de cada mensagem por agente, o volume de dados armazenados localmente em cada agente e o tempo de execução dos algoritmos.

5.2 Avaliação dos Algoritmos

Conforme dito anteriormente, para avaliar os resultados obtidos foram utilizadas simulações com sub-planos de tamanhos diferentes, contendo 3, 5, 7 e 9 lacunas de tempo. O tamanho dos sub-planos implica na quantidade de agentes envolvidos no problema, visto que cada lacuna de tempo corresponde a uma variável onde será alocado um evento. A Tabela 2 apresenta os resultados médios encontrados para cada simulação realizada. Por conveniência, os resultados apresentados representam a média de todos os cenários avaliados sobre cada simulação.

Tabela 2. Comparação entre os resultados obtidos para cada algoritmo em um sub-plano.

Tamanho sub-plano	Algoritmo	Ciclos	Mensagens	Tamanho mensagens (byte)	Volume de dados transmitidos (byte)	Memória (byte)	Tempo (segundo)
3 lacunas	ADOPT	14,18	13,18	50,22	662,21	67,93	3,72
	DPOP	2,33	1,33	137,42	182,69	539,63	3,06
5 lacunas	ADOPT	181,77	180,78	85,86	15538,97	106,64	4,79
	DPOP	2,59	1,59	2184,38	867,73	1871,94	4,34
7 lacunas	ADOPT	1897,52	1896,51	123,27	233996,36	145,31	9,69
	DPOP	2,71	1,71	2184,38	3744,65	8684,32	9,05
9 lacunas	ADOPT	52211,05	52210,05	162,31	8492212,04	183,95	50,34
	DPOP	2,78	1,78	8840,36	15716,19	35457,54	27,45

Analisando os resultados obtidos, é possível observar que, para o DPOP, a quantidade de ciclos e mensagens transmitidas é linear em relação ao número de agentes envolvidos no problema. Outra particularidade presente no DPOP é que, ao repetir a execução do algoritmo, a quantidade de ciclos e de mensagens transmitidas não é

alterada, desde que as características do problema e a pseudo-árvore gerada permaneçam iguais. Em um sentido oposto, para o ADOPT, a quantidade de ciclos e de mensagens pode variar ao repetir a execução em função de sua busca assíncrona.

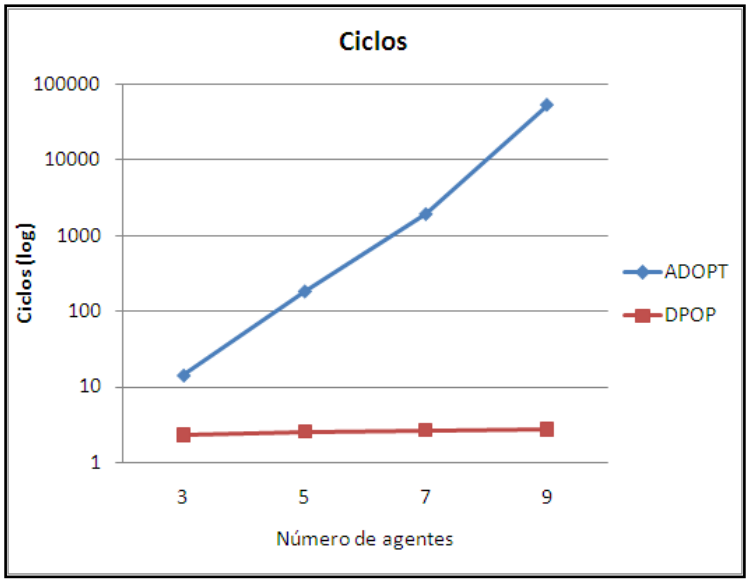

Figura 15. Quantidade média de ciclos por agente em um sub-plano.

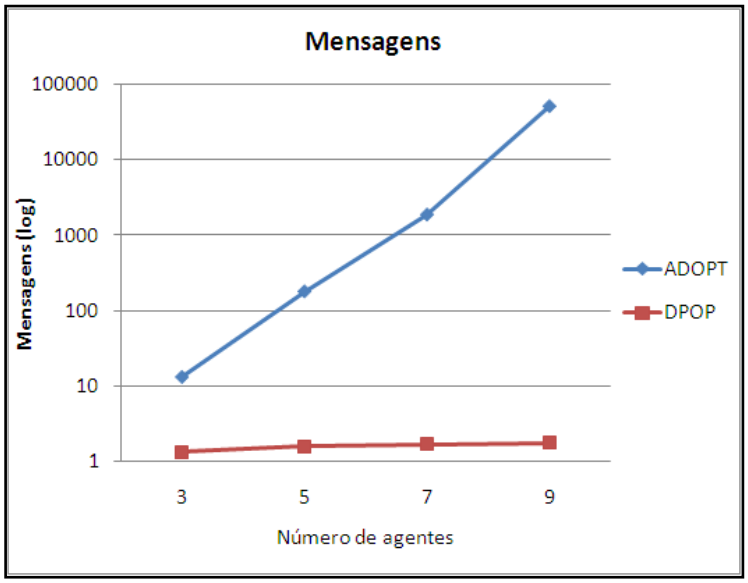

Figura 16. Quantidade média de mensagens trocadas por agente em um sub-plano.

As Figura 15 e Figura 16 apresentam a evolução do número de ciclos e de mensagens transmitidas por agente em função do tamanho do sub-plano. Deste modo, é possível notar um crescimento exponencial em ambos os critérios para o ADOPT. Tal característica está associada à estratégia de busca do algoritmo, que mantém nos agentes pouca informação local sobre seus vizinhos, sendo necessária uma grande quantidade de ciclos e troca de mensagens para refinar as informações armazenadas localmente.

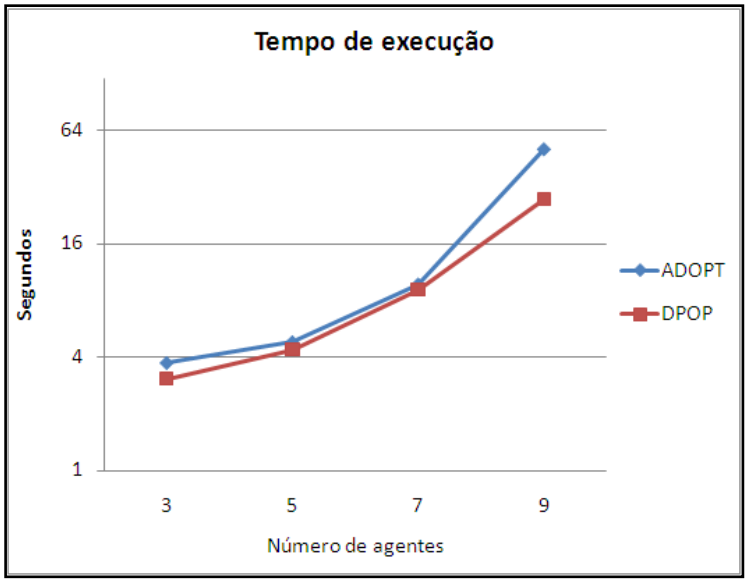

Figura 17. Tempo médio de execução de um sub-plano.

Apesar de o DPOP implementar uma busca parcialmente assíncrona, seu desempenho é melhor do que o ADOPT, conforme ilustrado pela Figura 17. O principal motivo para o ADOPT requerer um tempo maior de execução é devido à quantidade de ciclos e mensagens trocadas durante o processo de busca. Por outro lado, para o DPOP, cada agente mantém uma matriz de utilidade contendo os custos para cada possível atribuição das variáveis contidas na sub-árvore iniciada no agente em questão. Esta estratégia faz com que o processo de busca necessite de poucas trocas de mensagens para que os agentes encontrem uma solução ótima.

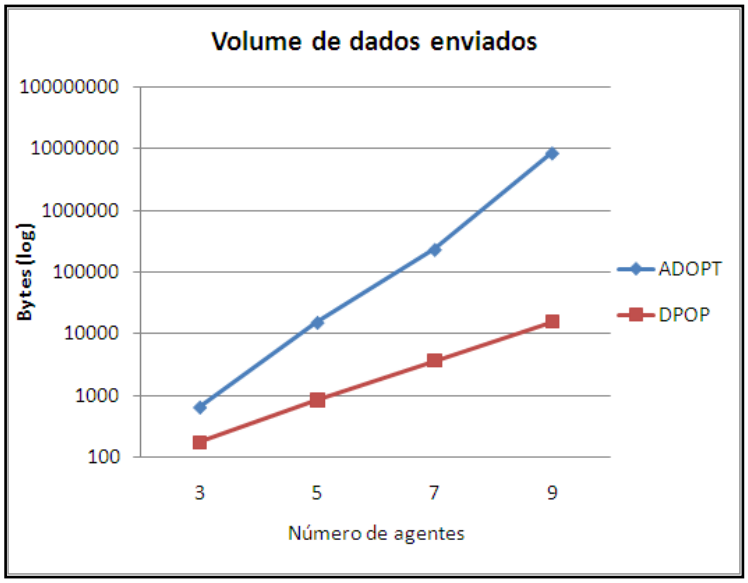

Figura 18. Volume de dados médio enviados por agente em um sub-plano.

De fato, uma das principais características do ADOPT é a elevada quantidade de ciclos e de mensagens necessárias para os agentes alcançarem uma solução. Este aspecto influencia tanto o tempo de execução necessário para encontrar a solução, quanto o volume total de dados transmitidos entre os agentes, apesar das mensagens do ADOPT apresentar em média um tamanho consideravelmente pequeno. Por sua vez, embora o DPOP apresente uma quantidade baixa de mensagens transmitidas, o volume total de informações trocadas entre os agentes também é notável devido ao tamanho de cada mensagem. A Figura 18 apresenta um comparativo entre o volume médio de dados transmitidos em cada algoritmo para um sub-plano.

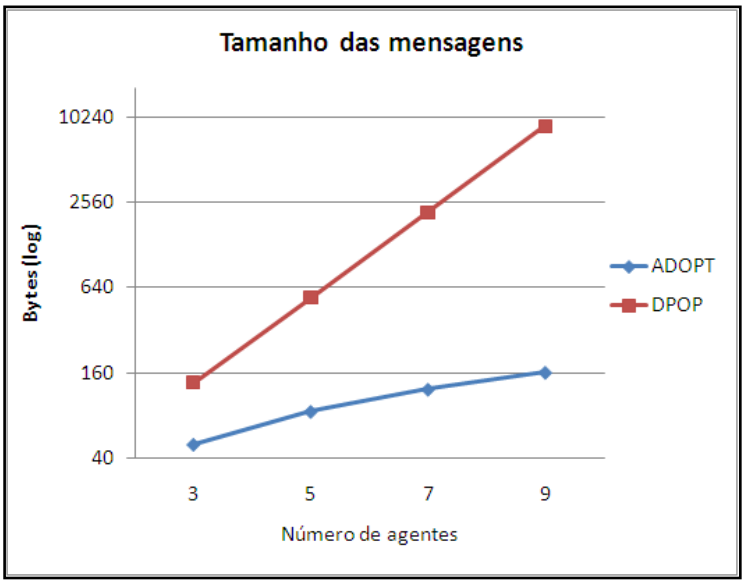

Figura 19. Tamanho médio de mensagens por agente em um sub-plano.

Um dos motivos para o DPOP requerer poucas trocas de mensagens para encontrar uma solução ótima é a quantidade de informação contida nas mensagens. Para o DPOP, cada agente encaminha aos seus vizinhos de maior prioridade uma mensagem de custo, a qual contém uma matriz de utilidade dos custos para cada combinação de atribuição das variáveis de menor prioridade. Portanto, para o DPOP, o tamanho das mensagens é drasticamente influenciado pela quantidade de agentes envolvidos no problema. Por sua vez, o ADOPT apenas transmite sua visão local sobre o problema e uma estimativa de custo das atribuições de seus vizinhos de menor prioridade. Em outras palavras, o ADOPT propaga os valores atuais de seus vizinhos de menor prioridade e o impacto destas atribuições. A Figura 19 ilustra o tamanho médio das mensagens em função da quantidade de agentes envolvidos no sub-plano.

Pode-se dizer que o aspecto chave da estratégia de busca do DPOP é a matriz de utilidade. Não obstante, a geração e o armazenamento da matriz de utilidade exigem uma quantidade razoável de memória disponível para cada agente. A partir dos experimentos realizados, foi constatado que a quantidade de memória requerida cresce exponencialmente em função da quantidade de agentes no problema. Em contrapartida, o ADOPT apenas armazena localmente em cada agente o valor atual de seus vizinhos e uma estimativa sobre o impacto desta atribuição. A Figura 20 apresenta uma comparação entre os algoritmos em termos de memória requerida para execução do processo de busca de um sub-plano.

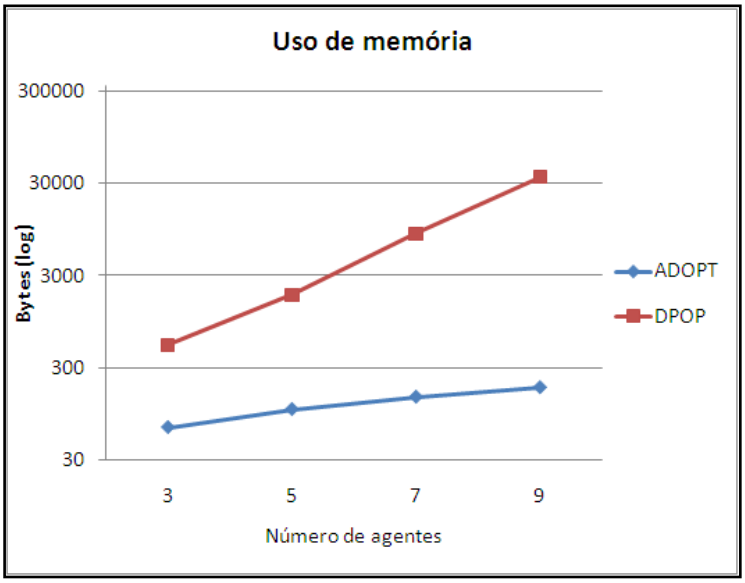

Figura 20. Memória requerida para execução de um sub-plano.

De maneira geral, através dos experimentos realizados foi possível observar que, com o aumento da quantidade de agentes envolvidos no problema, ambos os algoritmos sofreram severamente em diferentes aspectos. No caso do ADOPT, houve um grande crescimento na quantidade de ciclos e mensagens necessárias para a resolução do problema, afetando também o tempo de execução do processo de busca do algoritmo. Já o DPOP apresentou um elevado uso de memória e mensagens contendo um grande volume de dados. Entretanto, apesar das particularidades de cada algoritmo, ambos apresentaram resultados satisfatórios para o problema de condução férrea, com relação às restrições de hardware e de tempo de execução para resolução do problema. A seguir será apresentada uma avaliação da qualidade das soluções obtidas a partir do módulo de controle inteligente proposto.

5.3 Avaliação da Modelagem DCOP Proposta

Conforme já mencionado, para avaliação da qualidade das soluções encontradas, foi utilizado como métrica o consumo de combustível realizado em cada viagem. Os experimentos foram realizados em oito viagens diferentes, onde os trechos percorridos e as características da composição, tais como quantidade de locomotivas e peso dos vagões, são diferentes para cada viagem. A Tabela 3 apresenta os resultados obtidos pela modelagem DCOP para cada cenário. Por sua vez, a Figura 21 fornece um comparativo entre o consumo de combustível previsto pela modelagem DCOP e o consumo realizado pelo especialista humano em cada cenário.

Tabela 3. Resultados obtidos a partir da modelagem DCOP para cada cenário.

Experimento	Modelagem DCOP		
	Tamanho sub-plano	Quantidade sub-planos	Consumo (LTKB)
Cenário 1	3	245	4,2243
	5	138	4,2031
	7	104	4,1751
	9	88	4,163
Cenário 2	3	234	4,2252
	5	142	4,2147
	7	98	4,201
	9	77	4,1887
Cenário 3	3	215	4,1736
	5	132	4,1344
	7	91	4,1234
	9	71	4,0954
Cenário 4	3	256	4,6504
	5	167	4,5918
	7	127	4,5461
	9	102	4,5114
Cenário 5	3	257	4,256
	5	157	4,25
	7	109	4,2435
	9	86	4,2252
Cenário 6	3	315	3,942
	5	193	3,8841
	7	141	3,8611
	9	101	3,9982
Cenário 7	3	248	4,1272
	5	150	4,1038
	7	102	4,0787
	9	80	4,0756
Cenário 8	3	310	4,4801
	5	192	4,44
	7	133	4,4172
	9	107	4,4129

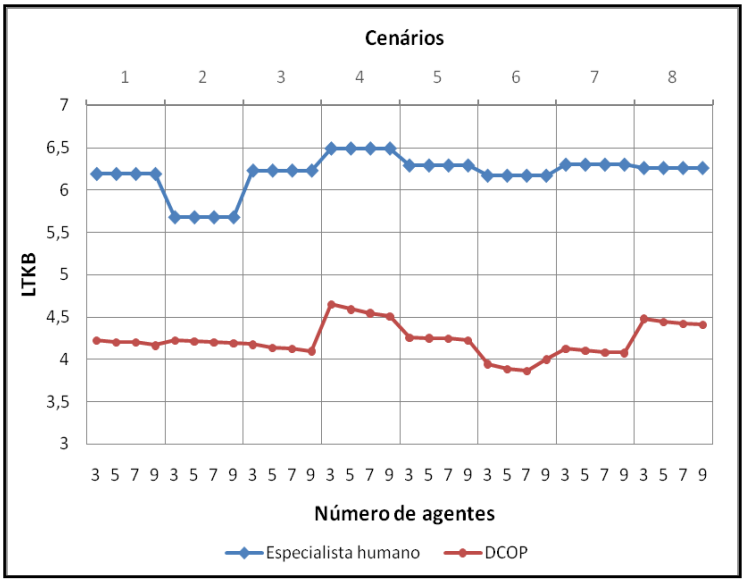

Figura 21. Comparação entre o especialista humano e a modelagem DCOP (LTKB)

Ao analisar os resultados, fica evidente a redução no consumo energético nos experimentos realizados a partir da modelagem DCOP proposta, sobretudo em situações em que foram utilizadas nove lacunas de tempo durante a simulação. Não obstante, para alcançar tais resultados, os pesos das funções de custo da modelagem DCOP proposta foram ajustados empiricamente para objetivar a redução do consumo energético em cada viagem (Tabela 4). Em termos gerais, os critérios identificados como mais relevantes para alcançar a eficiência energética foram a velocidade e o consumo previsto para cada ação. Todavia, estes pesos podem sofrer variações para cada viagem, em virtude das particularidades dos trechos percorridos, veículos utilizados e carga transportada. Em outras palavras, foi observado que o perfil da ferrovia e as características da composição férrea, tais como o modelo dos veículos e carga transportada, influenciam diretamente na estratégia de condução.

Tabela 4. Pesos das funções de custo por cenário.

Experimento	Peso Velocidade (P_v)	Peso Consumo (P_c)	Peso Deslocamento (P_d)
Cenário 1	0,38	0,32	0,30
Cenário 2	0,35	0,35	0,30
Cenário 3	0,20	0,30	0,50
Cenário 4	0,40	0,30	0,30
Cenário 5	0,25	0,45	0,30
Cenário 6	0,30	0,40	0,30
Cenário 7	0,25	0,45	0,30
Cenário 8	0,25	0,45	0,30

Tabela 5. Comparativo de resultados entre diferentes objetivos de otimização.

Otimização	Peso Velocidade (P_v)	Peso Consumo (P_c)	Peso Deslocamento (P_d)	Consumo (LTKB)	Tempo viagem (minutos)
Consumo de combustível	0,38	0,32	0,30	4,1630	262,18
Tempo de viagem	0,60	0,20	0,20	4,3454	167,63

Uma importante característica da modelagem DCOP proposta é a flexibilidade para a definição da função objetivo do problema. A modelagem DCOP proposta permite alterar o objetivo do problema, como por exemplo, otimizar o tempo de viagem, apenas ajustando os pesos das funções de custo. Para avaliar esta característica da modelagem DCOP proposta, foi realizado um comparativo entre os resultados obtidos pelo primeiro cenário utilizando pesos que priorizassem respectivamente a redução do consumo de combustível e a redução no tempo de viagem. A Tabela 5 apresenta os pesos para as funções de custo e os resultados obtidos em cada objetivo de otimização. Para estes experimentos foram utilizados sub-planos contendo nove lacunas de tempo.

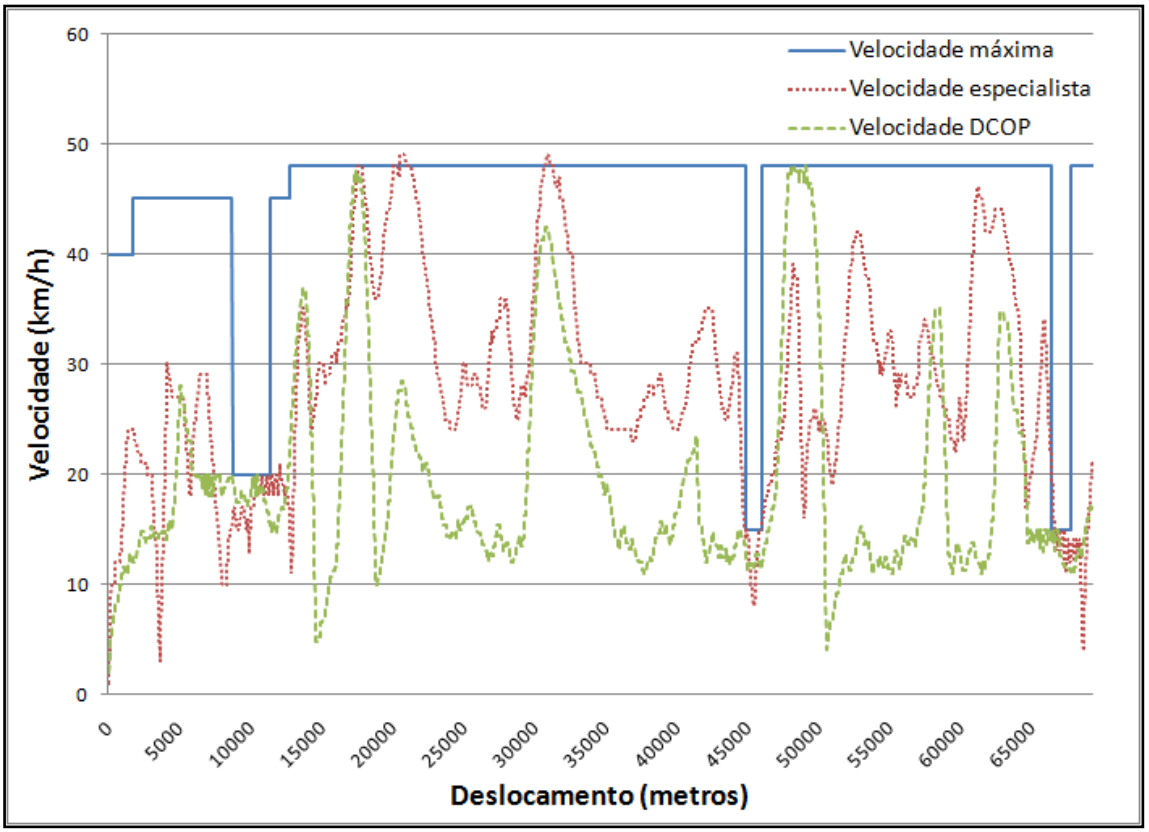

Figura 22. Comparativo da variação de velocidade entre o maquinista e o modelo DCOP.

Durante a realização dos experimentos, foi constatado que a variação brusca de velocidade ocasiona um alto consumo energético, conforme mencionado no capítulo 2.

Desta maneira, espera-se que uma boa estratégia tenha como objetivo variar suavemente a velocidade, mantendo-a próxima à velocidade limite do trecho atual da ferrovia, exceto para as situações cujo consumo é demasiadamente alto para tal ação. A Figura 22 apresenta um comparativo entre a velocidade prevista no primeiro cenário pelo especialista humano e a modelagem DCOP proposta, utilizando nove lacunas de tempo.

De fato, a idéia chave da modelagem DCOP proposta é a decomposição de uma viagem em sub-planos. Esta decomposição do problema se faz necessária devido à grande quantidade de ações envolvidas em uma única viagem. A Figura 23 apresenta um comparativo entre a quantidade de ações realizadas pelo especialista humano e a quantidade de ações previstas pelo modelo DCOP para cada cenário. Ao analisar estes dados, é possível concluir que é inviável encontrar uma solução ótima para o problema de condução férrea sem a decomposição da viagem em sub-planos.

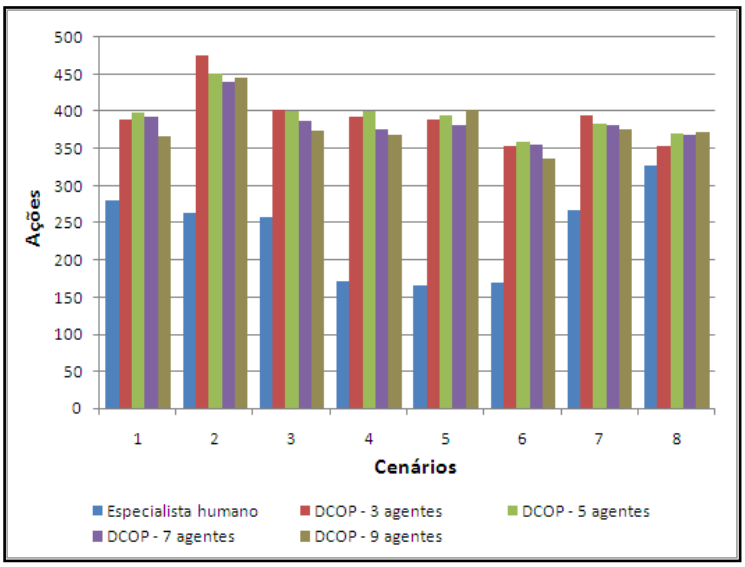

Figura 23. Comparativo do número de ações realizadas pelo maquinista e o modelo DCOP.

Deste modo, os sub-planos têm uma grande importância para a modelagem DCOP proposta. Além disto, o tamanho dos sub-planos também influencia na qualidade das soluções encontradas. Foi observado experimentalmente que os sub-planos que possuem poucos agentes envolvidos apresentam um desempenho melhor do que os sub-planos contendo muitos agentes, em relação ao tempo de execução. Contudo, quanto menor um sub-plano, menos informações os agentes dispõem para tomar suas ações. Em outras palavras, o tamanho dos sub-planos limita a visão dos agentes sobre o problema, reduzindo a capacidade dos agentes planejarem o impacto de suas ações sobre as ações futuras. Em contrapartida, os sub-planos que envolvem muitos agentes

apresentam um alto custo computacional para encontrar uma solução ótima. Deste modo, a modelagem DCOP proposta permite ajustar-se às particularidades do ambiente, em virtude das restrições de hardware ou do grau de qualidade das soluções encontradas, através da quantidade de agentes envolvidos no problema.

5.4 Considerações Finais

Um dos principais desafios que o problema de condução férrea aborda é o planejamento sobre o impacto das ações realizadas sobre as ações futuras. Em geral, o esquema proposto para redução do consumo de combustível foi capaz de obter uma qualidade de solução satisfatória, sobretudo nas simulações onde os sub-planos eram compostos por nove lacunas de tempo. A partir dos resultados obtidos, é possível concluir que o DCOP é um formalismo eficiente para problemas de coordenação em sistemas multi-agente. Em especial, para o cenário abordado, foi possível obter resultados apropriados, apesar da complexidade do problema e do grande número de variáveis e restrições envolvidas.

Capítulo 6

CONCLUSÕES

Este trabalho teve como objetivo demonstrar que métodos da IAD podem ser utilizados para obtenção de políticas ótimas de condução férrea no que concerne o consumo de combustível. Para alcançar este objetivo, foi desenvolvido um módulo de controle inteligente para condução de composições férreas, cuja principal estratégia utilizada para alcançar a eficiência energética faz uso de técnicas de otimização de restrições. A condução de composições férreas pode ser classificada como um problema complexo de coordenação, onde o desafio consiste em encontrar uma combinação de ações que resulte na melhor condução considerando aspectos como: tempo de viagem, segurança, desgaste dos componentes mecânicos e consumo energético.

Deste modo, para a concepção do módulo de controle inteligente proposto, foi necessário modelar o problema de condução férrea como um DCOP. Apesar do DCOP ser um formalismo capaz de modelar naturalmente uma vasta classe de problemas de otimização do mundo real, esta atividade apresentou um elevado grau de complexidade devido ao grande número de variáveis e restrições envolvidas no problema. Além disto, o problema de condução férrea também apresenta características dinâmicas, o que dificultou a modelagem do problema como um DCOP. Além da modelagem do problema de condução férrea como um DCOP, foi realizada a implementação dos algoritmos ADOPT e DPOP. A escolha dos algoritmos foi realizada considerando as características do problema e os recursos requeridos, tais como comunicação e processamento.

Para avaliar o módulo de controle inteligente proposto, foram realizados diversos experimentos reproduzindo viagens reais de uma composição férrea, considerando o mesmo perfil da ferrovia e as mesmas características dos veículos. Os resultados obtidos pelos experimentos foram comparados com dados de viagens operadas por maquinistas que não dispõem de recursos para auxiliá-los durante a condução. Também foi avaliado o desempenho entre os algoritmos implementados para DCOP. Para esta avaliação, foram consideradas métricas como: quantidade de ciclos e

de mensagens transmitidas, o volume de dados transmitidos e o tamanho de cada mensagem por agente, o volume de dados armazenados localmente em cada agente e o tempo de execução dos algoritmos.

A partir dos resultados obtidos, foi possível concluir que o DCOP é um eficiente formalismo para problemas de coordenação em sistemas multi-agente. Em especial, para o cenário abordado, foi possível obter resultados apropriados, apesar da complexidade do problema e do número de variáveis e restrições envolvidas. Através do módulo de controle inteligente proposto, foi possível obter uma substancial redução no consumo de combustível, sobretudo nos experimentos envolvendo nove agentes.

Com relação à eficiência dos algoritmos comparados, ambos apresentaram um desempenho aceitável. Contudo, os algoritmos possuem características distintas que devem ser avaliadas previamente. O ADOPT apresentou uma elevada quantidade de ciclos e de trocas de mensagens, afetando o tempo de execução da busca em simulações envolvendo um número maior de agentes. Por sua vez, o tamanho das mensagens e a quantidade de memória requerida pelo DPOP crescem exponencialmente em função da quantidade de agentes envolvidos. Estes fatores podem ser decisivos em outros cenários do mundo real, sobretudo em situações onde há severas restrições de hardware ou de tempo de execução.

Por fim, este trabalho poderia ser estendido para tornar mais dinâmica a estratégia do módulo de controle inteligente proposto, onde os agentes poderiam se beneficiar das informações contidas no sub-plano anterior a fim de convergirem mais rapidamente para uma solução. Em outras palavras, poderiam ser melhor exploradas questões relacionadas à adaptabilidade dos agentes, ao invés de recomeçar o processo de busca a cada sub-plano executado. Outro aspecto que poderia ser abordado é a obtenção dinâmica das informações sobre o perfil da ferrovia. Atualmente o módulo de controle inteligente proposto requer um mapeamento prévio do perfil da via, entretanto, estas informações poderiam ser obtidas dinamicamente ao longo da viagem por meio de sensores. Isto implicaria na capacidade dos agentes se adaptarem às mudanças no ambiente.

REFERÊNCIAS BIBLIOGRÁFICAS

ALI, S.; et al.. Preprocessing Techniques for Accelerating the DCOP Algorithm ADOPT. *Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multiagent Systems* (*AAMAS*), Holanda. New York: ACM, p. 1041-1048, 2005.

ALL. América Latina Logística - Resultados de 2007. Acesso em 10 de março de 2008, disponível em http://www.mzweb.com.br/all/web/arquivos/ALL_RA_2007_20080425_port.pdf, 2007.

ANP. Agência Nacional do Petróleo, Gás Natural e Biocombustíveis – Relatórios Mensais de Preços. Acesso em 10 de março de 2008, disponível em http://www.anp.gov.br/petro/relatorios_precos.asp, 2008.

BARTAK, R.. Constraint Programming: In Pursuit of the Holy Grail. *Proceedings of Week of Doctoral Students* (*WDS*), República Tcheca. Prague: Universidade Charles, 10 p., 1999.

BARTAK, R.. Theory and Practice of Constraint Propagation. *Proceedings of the 3rd Workshop on Constraint Programming in Decision and Control*, Polônia. Gliwice: Wydavnictvo Pracovni Komputerowej, p. 7-14, 2001.

BOWRING, E.; et al.. Multiply Constrained Distributed Constraint Optimization. *Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems* (*AAMAS*), Japão. New York: ACM, p. 1413-1420, 2006.

BRINA, H. L.. *Estradas de Ferro*. 2 ed. Rio de Janeiro, 1983.

CHECHETKA, A.; SYCARA, K.. No-Commitment Branch and Bound Search for Distributed Constraint Optimization. *Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems* (*AAMAS*), Japão. New York: ACM, p. 1427-1429, 2006.

CHANG, D. J.; MORLOK, E. K.. Alternative Approach to Deriving the Optimality of Uniform Speed to Minimize Vehicle Work and Fuel Consumption. *Journal of American Transportation Engineering*, v. 131, p. 173-182, 2005.

GIANELLI, R. A.; et al.. Heavy-Duty Diesel Vehicle Fuel Consumption Modeling Based on Road Load and Power Train Parameters. *Society of Automobile Engineers* (*SAE*) *International Congress*, 2005.

HOWLETT, P. G.; CHENG J.. Optimal Driving Strategies for a Train on a Track with Continuously Varying Gradient. *Journal of the Australian Mathematical Society*, v. 38, p. 388-410, 1995.

JADE. JADE Programmer's Guide. Acesso em 13 de outubro de 2008, disponível em http://jade.tilab.com/doc/programmersguide.pdf, 2008.

JURASKA, M.; MAGYLA, T.. Investigation of Fuel Consumption of Non-Scheduled Trains Thermal Traction. *Journal of Vilnius Gediminas Technical University and Lithuanian Academy of Sciences of Transport*, v. 19, n. 5, p. 230-235, 2004.

KHMELNITSKY, E.. On an Optimal Control Problem of Train Operation. *IEEE Transactions on Automatic Control*, v. 45, n. 7, p. 1257-1266, 2000.

KRAAY, D.; et at.. Optimal Pacing of Trains in Freight Railroads: Model Formulation and Solution. *Institute of Operation Research and Management Sciences* (*INFORMS*), v. 39, p. 82-99, 1989.

LESSER, V.; ORTIZ, C. L.; TAMBE, M.. *Distributed Sensor Networks: a Multiagent Perspective*. Massachusetts, New York: Kluwer Academic Publishers, v. 9, 386 p., 2003.

MAILLER, R.; LESSER, V.. Solving Distributed Constraint Optimization Problems Using Cooperative Mediation. *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent* (*AAMAS*), EUA, Washington DC: IEEE Transactions, p. 438-445, 2004.

MAHESWARAN, R. T.; et al.. Taking DCOP to the Real World: Efficient Complete Solutions for Distributed Multi-Event Scheduling. *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent* (*AAMAS*), EUA. Washington DC: IEEE Transactions, p. 310-317, 2004.

MEDANIC, J.; DORFMAN, M. J.. Energy Efficient Strategies for Scheduling Trains on a Line. *Fifteenth Triennial World Congress of Optimization Theory and Applications*, Barcelona: Spain, v. 115, p. 587-602, 2002.

MEISELS, A.. *Distributed Search by Constrained Agents: Algorithms, Performance, Communication*. England, London: Springer-Verlag, 216 p., 2008.

MODI, P. J.; et al.. ADOPT: Asynchronous Distributed Constraint Optimization with Quality Guarantees. *Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems* (*AAMAS*), Melbourne. New York: ACM, p. 149-180, 2003.

PETCU, A.; FALTINGS, B.. A Scalable Method for Multiagent Constraint Optimization. *Proceedings of the International Joint Conference on Artificial Intelligence* (*IJCAI*), Edinburgh. San Fransisco: Morgan Kaufmann Publishers, p. 266-271, 2005.

PETCU, A.; FALTINGS, B.. MB-DPOP: A New Memory-Bounded Algorithm for Distributed Optimization. *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence* (*IJCAI*), Hyderabad. Menlo Park: AAAI Press, p. 1452-1457, 2007a.

PETCU, A.. FRODO: A Framework for Open/Distributed constraint Optimization. Technical Report n. 2006/001, *Swiss Federal Institute of Technology* (*EPFL*), 2006.

PETCU, A.; et al.. PC-DPOP: A New Partial Centralization Algorithm for Distributed Optimization. *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence* (*IJCAI*), Hyderabad. Menlo Park: AAAI Press, p. 167-172, 2007b.

PIRES, L. C.; et al.. Simulação de Composição Ferroviária Acionada por Motores de Indução e Inversores de Tensão. *Revista Controle & Automação*. v. 16, n. 1, p. 1-12, 2005.

RAKHA, H.; DING, Y.. Impact of Stops on Vehicle Fuel Consumption and Emissions. *Journal of Transportation Engineering*, v. 129, p. 23-32, 2003.

RUSSEL, S. ; NORVIG, P. *Inteligência Artificial*. 3ed. Rio de Janeiro: Elsevier, 2004.

SILAGHI, M. C.; YOKOO, M.. Nogood based Asynchronous Distributed Optimization (ADOPT ng). *Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems* (*AAMAS*), Japão. New York: ACM, p. 1389-1396, 2006.

SULTANIK, E. A.; et al.. DCOPolis: a Framework for Simulating and Deploying Distributed Constraint Optimization Algorithms. *Proceedings of the ninth International Workshop on Distributed Constraint Reasoning*, 2007.

TORMOS, P.; et al.. Distributed Constraint Satisfaction Problems to Model Railway Scheduling Problems. *Computers in Railway X*. Southampton: WIT Press. v. 10, p. 289-297, 2006.

TSANG, E.. *Foundations of Constraint Satisfaction*. Essex, UK: Department of Computer Science, University of Essex, 1993.

YOKOO, M.; et al.. The Distributed Constraint Satisfaction Problem: Formalization and Algorithms. *IEEE Transactions on Knowledge and Data Engineering*, v. 10, p. 673-685, 1998.

WEISS, G.. *Multiagent Systems - A Modern Approach to Distributed Modern Approach to Artificial Intelligence*. Cambridge, Massachusetts: MIT Press, 1999.

ANEXOS

Pseudo-código do ADOPT

```
     Início
1    threshold ← 0; CurrentContext ← {};
2    ∀d ∈ Di, x1 ∈ Children faça
3      lb(d,x1) ← 0; ub(d,x1) ← Inf;
4      t(d,x1) ← 0; context(d,x1) ← {};
5    fim faça;
6    di ← d que minimize LB(d);
7    backtrack;
```

```
     Ao receber (THRESHOLD,t,context)
8    se context é compatível com
        CurrentContext
9      threshold ← t;
10     mantem_threshold_constante;
11     backtrack;
12   fim se;
```

```
     Ao receber (TERMINATE,context)
13   armazena TERMINATE recebida;
14   CurrentContext ← context;
15   backtrack;
```

```
     Ao receber (COST,xk,context,lb,ub)
16   d ← valor de xi em context;
17   remove (xi,d) em context;
18   se TERMINATE não foi recebida
19     ∀(xj,dj) ∈ context e xj não é
       vizinho de xi faça
20       adiciona (xj,dj) em CurrentContext;
21     fim faça;
22     ∀d' ∈ Di, x1 ∈ Children faça
23       se context(d',x1) é incompatível
            com CurrentContext
24         lb(d,x1) ← 0; ub(d,x1) ← Inf;
25         t(d,x1) ← 0; context(d,x1) ← {};
26       fim se; fim faça; fim se;
27   se context é compatível com
        CurrentContext
28     lb(d,x1) ← lb; ub(d,x1) ← ub;
29     context(d,xk) ← context;
30     mantem_child_threshold_constante;
31     mantem_threshold_constante;
32   fim se;
33   backtrack;
```

```
     Ao receber (VALUE,(xj,dj))
34   se TERMINATE não foi recebida
35     adiciona (xj,dj) em CurrentContext;
36     ∀d ∈ Di, x1 ∈ Children faça
37       se context(d,x1) é incompatível
            com CurrentContext
38         lb(d,x1) ← 0; ub(d,x1) ← Inf;
39         t(d,x1) ← 0; context(d,x1) ← {};
40       fim se; fim faça;
41     mantem_threshold_constante;
42     backtrack; fim se;
```

```
     backtrack
43   se threshold = UB
44     di← d que minimize UB(d);
45   senão
46     di← d que minimize LB(d); fim se;
47   envia (VALUE,(xi,di)) para vizinhos de
     menor prioridade;
48   mantem_alocacao_constante;
49   se threshold = UB e (TERMINATE foi
        recebida ou x1 é o agente raiz)
50       envia (TERMINATE,CurrentContext ∪
         {(xi,di)}) para cada agente filho;
51       termina execução; fim se;
52   envia (COST,xi,CurrentContext,LB,UB) ao
     agente pai;
```

```
     mantem_threshold_constante
53   se threshold < LB: threshold ← LB;
54   se threshold > UB: threshold ← UB;
```

```
     mantem_alocacao_constante
55   enquanto threshold > δ(di) +
              Σxi ∈ Children t(di,x1) faça
56     escolha x1 ∈ Children onde
       ub(di,x1) > t(di,x1)
57     incrementa t(di,x1); fim faça;
58   enquanto threshold < δ(di) +
              Σxi ∈ Children t(di,x1) faça
59     escolha x1 ∈ Children onde
       t(di,x1) > lb(di,x1)
60     decrementa t(di,x1); fim faça;
61   envia(THRESHOLD,t(di,x1),CurrentContext)
     para cada agente filho x1
```

```
     mantem_child_threshold_constante
62   ∀d ∈ Di,x1 ∈ Children faça
63   enquanto lb(d,x1) > t(d,x1) faça
64     incrementa t(di,x1); fim faça; fim faça;
65   ∀d ∈ Di,x1 ∈ Children faça
66   enquanto ub(d,x1) > t(d,x1) faça
67   decrementa t(di,x1); fim faça; fim faça;
```

Pseudo-código do Opt-APO

Início

$d_i \leftarrow$ escolha aleatoriamente $d \in D_i$;

$F_i^* \leftarrow 0$; $p_i \leftarrow$ número de vizinhos;

$m_i \leftarrow$ ativo; *mediate* $\leftarrow$ nulo;

adiciona x_i a *good_list*;

envia (**INIT**,x_i,p_i,d_i,m_i,D_i,C_i,$path_{i,j}$) aos vizinhos;
initList $\leftarrow$ vizinhos;

Ao receber (**INIT**,x_j,p_j,d_j,m_j,D_j,C_j,$path_{j,i}$)

adiciona (x_j,p_j,m_j,D_j,C_j,*path*) na *agent_view*;

se x_j é vizinho de algum $x_k \in$ *good_list*

  adiciona x_j a *good_list*;

  adiciona na *good_list* todos os $x_l \in$ *agent_view* e que $x_l \notin$ *good_list*;
  $p_i \leftarrow$ tamanho da *good_list*; **fim se**;

se $x_j \notin$ *initList*

  envia (**INIT**,x_i,p_i,d_i,m_i,D_i,C_i,$path_{i,j}$) para x_j;
senão

remove x_j da *initList*;

verifica_agent_view; **fim se**;

Ao receber (**VALUE?**,x,p_j,d_j,m_j,c_j)

atualiza *agent_view* com x_j,p_j,d_j,m_j,c_j;

verifica_agent_view;

Ao receber (**EVALUATE?**,x_j,p_j,m_j)

atualiza *agent_view* com (x_j,p_j,m_j);

se (*mediate* $\neq$ nulo ou $\exists_k$ ($p_k > p_j$ ou m_k = ativo)) e m_j = ativo
  envia (**WAIT!**,x_i,p_i);

senão se *mediate* $\neq$ ativo

  mediate $\leftarrow m_j$;

  nomeia cada $d \in D_i$ com os nomes dos agentes e custos associados em $d_i \leftarrow d$;
  envia (**EVALUATE!**,x_i,p_i,*labeled* D_i);

fim se; **fim se**;

Ao receber (**ACCEPT!**,d,x_j,p_j,d_j,m_j)

$d_i \leftarrow d$; *mediate* $\leftarrow$ falso;

envia (**VALUE?**,x_i,p_i,d_i,m_i,c_i)
  $\forall x_j \in$ *agent_view*;
atualiza *agent_view* com (x_j,p_j,d_j,m_j);

verifica_agent_view;

Ao receber(**WAIT!**,x_j,p_j)

  counter--;

  se *counter* = 0

   escolha_solucao; **fim se**;

Ao receber(**EVALUATE!**,x_j,p_j,*labeled* D_j)

34 armazena(x_j,*labeled* D_j) em *preferences*;
    counter--;
35 **se** *counter* = 0
36   escolha_solucao; **fim se**;

---

mediar(m_i')

37 *preferences* $\leftarrow$ 0; *counter* $\leftarrow$ 0;
38 $\forall x_j \in$ *good_list* **faça**
39   envia(**EVALUATE?**,x_i,p_i,m_i') para x_j;
40   *counter*++; **fim faça**;
41 *mediate* $\leftarrow m_i'$; $m_i \leftarrow m_i'$;

---

verifica_agent_view

42 **se** *initList* $\neq$ 0 ou *mediate* $\neq$ nulo
43   retorna; **fim se**;
44 $c_i' \leftarrow \{x_j | \exists_{fi}\ x_j \in f_i$ e $f_i > f_i^*\}$;
45 $m_i' \leftarrow$ nulo;
46 **se** $F_i > F_i^*$ e $\exists_j$ ($f_i < f_i^*$ e $p_j \leq p_i$)
47    $m_i' \leftarrow$ ativo;
48 **senão se** $F_i > F_i^*$
49    $m_i' \leftarrow$ passivo; **fim se**; **fim se**;
50 **se** m_i' = ativo e $\neg\exists_j(p_j > p_i$ e m_j = ativo)
51   **se** $\exists(d_i' \in D_i)$ ($d_i' \cup$ *agent_view* $\rightarrow$ $F_i = F_i^*$) e as mudanças estão com Seus vizinhos de menor prioridade
52      $d_i \leftarrow d_i'$; $m_i \leftarrow$ nulo;
53      $c_i' \leftarrow \{x_j | \exists_{fi}\ x_i \in f_i$ e $f_i > f_j^*\}$;
54   envia (**VALUE?**,x_i,p_i,d_i,m_i,c_i)
        $\forall x_j \in$ *agent_view*;
55   **senão**
56   mediar(m_i'); **fim se**;
57 **senão se** m_i' = passivo
58   mediar(m_i'); **fim se**;
59 **senão se** $m_i \neq m_i'$ ou (m_i = nulo e $c_i \neq c_i'$)
60    $m_i \leftarrow m_i'$;
61   envia (**VALUE?**,x_i,p_i,d_i,m_i,c_i')
        $\forall x_j \in$ *agent_view*; **fim se**;
62 **senão se** m_i = nulo
63    $\forall x_j, x_k$ $x_j \in$ *agent_view* e $x_k \in c_j$ e $x_k \notin$ *good_list* e m_j = nulo **faça**
64      $\forall x_l$ no caminho de x_k e $x_l \notin$ *agent_view* **faça**
65   envia (**INIT**,x_i,p_i,d_i,m_i,D_i,C_i,$path_{i,1}$) para x_l; adiciona x_i em *initList*;
66   **fim faça**; **fim faça**; **fim se**; **fim se**;
67 $c_i \leftarrow c_i'$;

Pseudo-código do Opt-APO (continuação)

```
     escolha_solucao
68   selecione uma solução s usando uma busca
     BB que: 1) minimize o custos dos agentes
     na good_list; 2) minimize o custo que
     não estão na sessão;
69   F_i* ← cost(s);
70   F_i' ← F_i + custo corrente dos agentes
     que não estão na sessão;
71   F_s' ← F_i* + custo considerando s para os
     agentes que não estão na sessão;
72   se mediate = ativo e F_s' ≤ F_i'
73     d_i' ← d_i; fim se;
74   ∀x_j ∈ good_list faça
75   se x_j ∈ preferences
76     se d_j' ∈ s viola um x_k e
          x_k ∈ agent_view
77       envia (INIT,x_i,p_i,d_i,m_i,D_i,C_i,path_i,k)
           para x_j;
78       adiciona x_k na initList; fim se;
79     se mediate = ativo e F_s' ≤ F_i'
80       envia (ACCEPT!,d_j',x_i,p_i,d_i) para x_j;
         atualiza agent_view para x_j;
       senão se mediate = ativo e F_s' > F_i'
81       envia (ACCEPT!,d_j,x_i,p_i,d_i) para x_j;
82     fim se; fim se;
83   senão se mediate = ativo
84     envia (VALUE?,x_i,p_i,d_i,m_i,c_i) para x_j;
85   fim se; fim se; fim faça;
86   mediate ← nulo;
87   verifica_agent_view;
```

Pseudo-código do DPOP

```
Início

se Children(x_i) = 0

util_{x_i}(x_j) ← computa_util(x_j,x_k), onde:
   x_j é pai de x_i
   x_k é pseudo-pai de x_i;
envia (UTIL,util_{x_i}(x_j)) para x_j;
```

```
Ao receber (UTIL,util_{x_l}(x_i)), onde:
  x_l é filho de x_i;
armazena util_{x_l}(x_i);
se todas as mensagens de UTIL foram
   recebidas de seus filhos

  se x_j ≠ nulo

    v_i* ← escolha_valor_otimo(nulo);

    envia (VALUE,v_i*) ∀x_l;

  senão

    util_{x_i}(x_j) ← computa_util(x_j,x_k)

    envia (UTIL,util_{x_i}(x_j)) para x_j;

  fim se;

fim se;
```

```
   Ao receber (VALUE,value_{x_j}^{x_i})
14 adiciona todas as x_j ← v_j* ∈ value_{x_j}^{x_i} na
   agent_view;
15 v_i* ← escolha_valor_otimo(agent_view);
16 envia (VALUE,v_i*) ∀x_l;
```

```
   computa_util(x_j,x_k)
17 ∀ combinações de valores de x_k faça
18   compute um vetor util_{x_i}(x_j) para todos
     {util_{x_i}(v_i*(v_j),v_j)|v_j ∈ D_j};
19   monte um hiper-cubo util_{x_i}(x_j) com todos
     estes vetores (totalizando |x_k + 1|
     dimensões);
20 fim faça;
21 retorna util_{x_i}(x_j);
```

```
   escolha_valor_otimo(agent_view)
22 v_i* ← argmax_{v_i} Σ util_{x_l}(v_i,agent_view)
23 retorna v_i*;
```

Pseudo-código do NCBB

```
Início
se x_1 = nulo, onde x_1 é agente pai de x_i
  atualiza_contexto(); fim se;
enquanto
  busca();
  se x_1 = nulo ou atualiza_contexto = V
    break; fim se; fim enquanto;
costs[resultValue] ← 0;
∀x_j, onde x_j é agente filho de x_i,
buscaSubArvore(resultValue,x_j);
∀x_j envia (STOP) para x_j;
```

```
atualiza_contexto
enquanto recebe mensagem m de x_1;
  se recebida (SEARCH)
    bound ← m.BOUND; retorna F;
  senão se recebida (x_1 = d)
    context[x_1] ← d;
    envia LB(x_i,context,indx_1) −
      LB(x_i,context,indx_1 - 1) para x_1;
  fim se;
senão se recebida (STOP) retorna V;
fim se; fim enquanto;
```

```
   buscaSubArvore(d,child)
18 ∀descendants[child] envia (x_i = d);
19 anncdVals[c] ← d;
20 ∀x_k ∈ descendants[child] faça
21   recebe cost_k de x_k;
22   costs[d] ← costs[d] + cost_k;
23 fim faça;
24 se cost[d] > bound
25   prune(); anncdVals[child] ← 0;
26   retorna F;
27 senão
28   envia (SEARCH,BOUND ← cost[d]) para x_j;
29   retorna V;
30 fim se;
```

Pseudo-código do NCBB (continuação)

```
     busca

31   idle ← children; costs ← 0;
     unexplored ← 0; anncdVals ← 0;
     minCost ← LB(x_i,context,|ancestors|);

32   ∀d ∈ D e AgentCost(d,context) ≤ bound +
       minCost faça
33     costs[d] ← AgentCost(d,context) −
       minCost; fim faça;
34   ∀d ∈ D costs[d] ≠ 0 faça
       unexplored[d] ← children; fim faça;
35   enquanto unexplored ≠ 0 ou anncdVals ≠ 0

36   enquanto idle ≠ 0: escolha child ∈ idle;

37     idle ← (idle / child);
       escolha d: child ∈ unexplored[d];
       unexplored[d] ← (unexplored[d] /
                        child);
38     se ¬buscaSubArvore(d,child) ou
          (∃c ∈ D: child ∈ unexplored[c]

39       idle ← (idle ∪ child);
       fim se; fim enquanto;
41     recebe cost_j de todo x_j;
       d ← anncdVals[x_j]; anncdVals[x_j]
     ← 0;
       costs[d] ← costs[d] + cost_j;
42     se cost[d] > bound: prune();

43     senão se unexplored[d] ≠ 0 ou x_j ∉
              anncdVals[c]
44       bound ← costs[d]; resultValue
     ← d;
45       prune(); fim se; fim se;

46   se ∃c ∈ D: child ∈ unexplored[c]

47     idle ← (idle ∪ child); fim se;
     fim enquanto;


48   se x_1 = nulo
       envia (min_{d ∈ D} costs[d]) a x_1; fim
     se;
```