UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

ROBERT JUNGES

Comparação de Algoritmos para Otimização de Restrições Distribuídas em um Cenário de Controle Semafórico

Dissertação apresentada como requisito parcial
para a obtenção do grau de
Mestre em Ciência da Computação

Prof[a]. Dra. Ana Lucia Cetertich Bazzan
Orientador

Porto Alegre, novembro de 2007

CIP – CATALOGAÇÃO NA PUBLICAÇÃO

Junges, Robert

Comparação de Algoritmos para Otimização de Restrições Distribuídas em um Cenário de Controle Semafórico / Robert Junges. – Porto Alegre: PPGC da UFRGS, 2007.

103 f.: il.

Dissertação (mestrado) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR–RS, 2007. Orientador: Ana Lucia Cetertich Bazzan.

1. Sistemas multiagentes. 2. Simulação de tráfego. 3. Controle semafórico. 4. Otimização de restrições. I. Bazzan, Ana Lucia Cetertich. II. Título.

DEDICATÓRIA

Dedicado à memória de Rosa Junges e Avelino Mathias Junges.

SUMÁRIO

LISTA DE ABREVIATURAS E SIGLAS

ADOPT	*Asynchronous Distributed Optimization*
ATIS	*Advanced Traveler Information System*
ATMS	*Advanced Traffic Management System*
COP	*Constraint Optimization Problem*
CSP	*Constraint Satisfaction Problem*
DCOP	*Distributed Constraint Optimization Problem*
DisCSP	*Distributed Constraint Satisfaction Problem*
DPOP	*Dynamic Parameter Optimization Problem*
DTREE	*Dynamic Tree*
ITSUMO	*Intelligent Transportation System for Urban Mobility*
LO	Leste-Oeste
OPTAPO	*Optimal Asynchronous Partial Overlay*
NS	Norte-Sul
SCATS	*Sydney Coordinated Adaptive Traffic System*
SCOOT	*Split Cycle and Offset Optimization Technique*
SMA	Sistemas Multiagentes
TCP	*Transmission Control Protocol*
TRANSYT	*Traffic Network Study Tool*
UFRGS	Universidade Federal do Rio Grande do Sul
XML	*Extensible Markup Language*

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SÍMBOLOS

Símbolo	Descrição
n_v	número de variáveis usadas em problemas DCOP
V	conjunto das variáveis usadas em problemas DCOP
x_i	valor atribuído a uma variável v_i
D	conjunto de domínios para valores de variáveis DCOP
$f_{i,j} : d_i \times d_j \rightarrow N$	função de custo para o par de variáveis v_i e v_j
$Q_l : v_i -> a_j$	mapeamento distribuído de variáveis para agentes
O	conjunto de valores possíveis para as variáveis em DCOP
$O*$	conjunto de valores ótimos para as variáveis em DCOP
$F(O) = \sum f_{i,j}(v_i, v_j)$	função objetivo de um problemas DCOP
lb	*lower bound* local (ADOPT)
ub	*upper bound* local (ADOPT)
LB	*lower bound* global (ADOPT)
UB	*upper bound* global (ADOPT)
p_i	prioridade de um agente (OptAPO)
F_i	custo de um subproblema (OptAPO)
μ	indica o desejo de um agente mediar (OptAPO)
$R_{k,l}$	conjunto de relações (restrições) entre k e l (DPOP)
G	grafo (DPOP)
$P(s)$	pai do nodo s (DPOP)
$C(s)$	filho de um nodo s (DPOP)
$PP(s)$	os pseudo-pais de um nodo s (DPOP)
$PC(s)$	os pseudo-filhos de um nodo s (DPOP)
N_s	número de semáforos na formulação da abordagem proposta
S_l	conjunto de vetores que determinam o sentido do tráfego
D_{sp}	domínio de planos semafóricos
T_l	tamanho das vias de tráfego da rede
V_{max}	velocidade máxima dos veículos
$f_{l,n}$	definição do custo das restrições
$\beta_{l,n}$	definição de *bias*
$\rho_{l,n}$	densidade de veículos na via $l \rightarrow n$
ω	definição do valor do peso das restrições
α	magnitude do valor do custo das restrições
γ	concordância da ação dos agentes
τ	custo da combinação de planos entre semáforos vizinhos

RESUMO

Problemas de otimização de restrições distribuídas (DCOP - *Distributed Constraint Optimization Problem*) formam uma classe de problemas de grande interesse de estudo na ciência da computação em função da complexidade computacional. O presente trabalho tem o objetivo de comparar os três algoritmos mais populares em DCOP (ADOPT, OptAPO e DPOP) em termos de eficiência computacional e de solução proposta. Para tal estudo, é utilizado como domínio um problema de controle semafórico. Esse tipo de problema de controle é de fundamental importância para que se tenha uma administração eficiente do fluxo de veículos em uma malha viária. Além disso, envolve muitas interdependências entre variáveis da rede, como ocupação das vias e tempos de sinal verde dos semáforos, para que sejam determinadas as melhores configurações de controle. Nesse sentido, as estratégias devem fornecer bons resultados em nível de aplicação, e também em nível de computação, no que diz respeito ao uso da infra-estrutura computacional disponível, o que casa perfeitamente com os objetivos das implementações de DCOP. Ao longo deste trabalho, os temas relacionados à coordenação em sistemas multiagentes, otimização de restrições e controle de semáforos são estudados. Os modelos DCOP são utilizados com a finalidade de comparar os algoritmos. No que diz respeito aos resultados, percebe-se uma melhora no controle, obtida com o uso dos algoritmos DCOP em relação ao uso de controle fixo sincronizado e não sincronizado. Isso é verificado em nível de utilização das vias da rede. Além disso, outro tipo de resultado é verificado na execução dos algoritmos, tratando de questões como o tempo de execução. Foi possível estabelecer um comparativo entre os algoritmos e frente ao aumento do problema em quantidade de semáforos.

Palavras-chave: Sistemas multiagentes, Simulação de tráfego, Controle semafórico, Otimização de restrições.

Comparing Distributed Constraint Optimization Algorithms in a Traffic Control Scenario

ABSTRACT

Distributed Constraint Optimization Problems (DCOP) have a significant importance in Computer Science, due to its computational complexity. The objective of this work is to compare the three most popular algorithms for DCOP (ADOPT, OptAPO and DPOP) in terms of computational efficiency and quality of the proposed solution. In order to do that, a traffic control scenario is used. This kind of problem is very important when considering an efficient administration of the traffic network, which involves a lot of interdependencies among variables such as the occupation of the links and the split of the lights. The control strategies should be able to provide good results in terms of the domain application and consider the computational infrastructure available, matching exactly the objectives of the DCOP implementations. The present work is related to multiagent systems, constraint optimization and traffic light control. The DCOP models are used in order to perform the comparison among the algorithms. The results show that is possible to improve the efficiency of the control over the classic approaches of fixed traffic light timing. This is verified considering the utilization level of the network and the occurrence of traffic jams. Besides that, some computational issues are considered to compare the algorithms, for instance, the execution time.

Keywords: Multiagent Systems, Traffic Simulation, Traffic Light Synchronization, Constraint Optimization.

1 INTRODUÇÃO

Os problemas de otimização de restrições distribuídas (DCOP - *Distributed Constraint Optimization Problem*) constituem uma classe de problemas de grande interesse de estudo na comunidade acadêmica, e um dos principais motivos desse interesse é a complexidade associada às questões a serem tratadas. Essas questões de otimização envolvem diversas áreas de estudo na computação, como inteligência artificial, redes de computadores e processamento distribuído. Pode-se dizer em linhas gerais que um problema DCOP envolve uma lista de variáveis com interdependências, cujos valores devem ser atribuídos de forma a otimizar uma função objetivo. Assim é possível perceber o envolvimento das áreas de estudo citadas anteriormente, seja no processo de comunicação necessário para a atribuição dos valores das variáveis, seja no controle do processo distribuído ou na computação das funções de controle do resultado.

Esse problema pode ser abordado computacionalmente através da definição de sistemas multiagentes (SMA). Nesse tópico se tem um conjunto de unidades de processamento (agentes) para as quais são atribuídas tarefas de processamento e que devem interagir de forma a alcançar a solução de um problema. Sendo assim, um DCOP visto como um SMA implica a atribuição do conjunto de variáveis a um conjunto de agentes, que serão responsáveis pela atribuição dos valores, e o fazem com base em um processamento feito a partir de informações recebidas dinamicamente de outros agentes.

Entre as soluções existentes para DCOP, as mais populares são os algoritmos OptAPO (MAILLER; LESSER, 2004), ADOPT (MODI et al., 2003), e DPOP (PETCU; FALTINGS, 2005). No entanto, cada algoritmo tem as suas particularidades que envolvem os requisitos necessários para a execução, bem como vantagens ou deficiências. Em nível da aplicação real de qualquer uma dessas soluções, é muito importante se levar em consideração as características de execução de cada uma, de forma a escolher a mais adequada ao problema em questão. As tentativas de comparação (PETCU; FALTINGS, 2005) se limitam a questões intrínsecas dos algoritmos e não propriamente à eficácia e eficiência relacionada à aplicação, e além disso se restringem ao domínio de *meeting scheduling*.

Em (MAHESWARAN et al., 2004) um aplicação real de DCOP é apresentada, onde é necessário o uso de heurísticas para melhoria do processo de comunicação e pré-cálculos para o ADOPT. Os autores reportam que foi possível encontrar diferenças entre cenários abstratos (como coloração de grafo) e cenários concretos (como *meeting scheduling*).

O presente trabalho propõe uma comparação dos algoritmos citados e para isso faz uso de um estudo de caso de interesse comum e aplicação real: o controle semafórico.

Outra motivação para esse estudo de caso é que, atualmente, as condições de tráfego têm piorado devido à sobrecarga das vias causada pela crescente demanda de usuários, especialmente em horários específicos (onde aumenta a demanda). Nesse sentido é importante que se tenha uma administração do fluxo de veículos de maneira eficiente, tentando

diminuir os tempos de percurso, número de veículos parados, consumo de combustível e emissão de poluentes. O alto custo do combustível e a degradação ambiental causada pela poluição dos veículos, bem como outros aspectos sociais e econômicos envolvidos, constituem portanto, as principais motivações para um controle eficiente do fluxo. Esses problemas podem ser amenizados pela construção de novas opções de tráfego, seja pelo aumento da capacidade das vias ou pela adição de novas rotas. No entanto, opções mais custosas nem sempre podem ser consideradas, e nesse sentido uma solução, também mais imediata e flexível, precisa ser desenvolvida.

Dos modelos de gerenciamento de tráfego é requerido adaptabilidade e simplicidade para tratar a crescente demanda. Dito isso, um dos focos de preocupação passa a ser uma solução inteligente que trabalhe com a infra-estrutura de transporte existente. Sistemas inteligentes de controle de transporte, como apresentados em (PAPAGEORGIOU et al., 2003) são necessários para fornecer um alto nível de automação nos sistemas de malha viária. Isso é feito através do uso de tecnologias avançadas e adaptativas: sistemas avançados de gerenciamento de tráfego (ATMS - *Advanced Traffic Management System*) e sistemas de informação aos usuários (ATIS - *Advanced Traveler Information System*).

Também com o propósito de melhorar as condições de tráfego, o controle automatizado de sistemas de semáforo tem sido amplamente utilizado para otimizar os planos semafóricos, de forma a facilitar a movimentação de veículos. A atuação dos semáforos que operam baseados em planos fixos pré-determinados é determinada conforme alguns critérios de otimização e depois passados para o sistema real. Como exemplo de critério de otimização pode ser citado o tempo de cada sinal, conforme estimativas do fluxo nas vias. A determinação de quais planos são mais apropriados para cada hora do dia é uma tarefa complexa que requer contagens e estudos do fluxo de tráfego em diferentes pontos da rede de vias. SMA oferecem soluções mais flexíveis e robustas.

Dadas as questões levantadas nos parágrafos anteriores, será apresentado a seguir o objetivo desse trabalho, que relaciona o uso do formalismo de problemas de otimização de restrições distribuídas com o problema de controle semafórico.

1.1 Objetivo

O presente trabalho tem o objetivo de comparar os algoritmos mais populares em DCOP: OptAPO, ADOPT e DPOP. Para isso o problema de controle semafórico é visto sob o ponto de vista de um problema de otimização de restrições distribuídas (DCOP). Pretende-se avaliar o formalismo DCOP quanto a sua aplicabilidade ao problema de tráfego citado, criando para tal uma modelagem tratável pelos algoritmos DCOP. Essa comparação tem dois objetivos principais: permite um estudo da eficiência desses algoritmos em casos reais; investiga uma questão importante e em aberto na literatura sobre DCOP, pois não existem até este momento tentativas de comparar os algoritmos em um cenário relevante no mundo real, além do já citado problema de *meeting scheduling*.

De maneira geral, adota-se uma abordagem de sistemas multiagentes para modelar individualmente agentes controladores de semáforo. Considera-se, portanto, uma forma de controle descentralizado. As simulações objetivam estudar os impactos das soluções propostas, com medições sobre o fluxo de veículos, como por exemplo a densidade e velocidade média nas vias. Por meio dessas simulações se busca incorporar ao processo de tráfego, agentes capazes de intervir na configuração da malha viária, em nível de ajuste de planos semafóricos.

A partir disso será possível determinar um compromisso entre custos de computação

e grau de satisfação das restrições (oriundos do dinamismo do tráfego) para otimizar o desempenho da malha viária. Qualidade de tráfego, nesse caso, deve ser inferida em termos de eficiência de controle, calculada a partir de fatores locais ao semáforo, como por exemplo a densidade das vias.

Para tal, os algoritmos de otimização de restrições serão estudados. Os modelos de controle serão implementados e seus resultados avaliados usando como base a plataforma de simulação de tráfego ITSUMO (SILVA et al., 2006), onde será possível configurar a dinâmica do tráfego, bem como outros parâmetros individuais dos algoritmos, referentes ao processo iterativo.

Em resumo, o trabalho pretende unir as questões de DCOP e controle semafórico, fornecendo um mapeamento a ser feito para estudo dessas questões. Essencialmente se busca apontar as métricas que permitem fazer uma comparação da aplicabilidade e eficiência de DCOP em um problema de aplicação real e de interesse comum.

1.2 Organização do Trabalho

O trabalho apresenta o embasamento teórico dos tópicos tratados (simulação de tráfego, controle semafórico e problemas DCOP), os modelos DCOP utilizados, e o desenvolvimento dos experimentos comparativos.

O capítulo 2 trata do formalismo DCOP, apresentando sua fundamentação, e também uma descrição dos algoritmos. Adicionalmente é introduzida uma avaliação dos algoritmos retirada de outros trabalhos apresentados na literatura.

O capítulo 3 trata das questões relativas à simulação de tráfego: o modelo de movimentação utilizado na simulação de tráfego; o simulador utilizado e seu funcionamento; e controle semafórico de uma forma geral e no contexto do simulador.

O capítulo 4 apresenta o mapeamento dos problemas de controle semafórico para DCOP, descrevendo os modelos teóricos pesquisados e elaborados, bem como as métricas adotadas para posterior avaliação prática.

O capítulo 5 descreve os cenários e os parâmetros de configuração dos estudos de caso, tomando como base a abordagem proposta no capítulo 4 e visando os experimentos de simulação a serem conduzidos posteriormente.

O capítulo 6 descreve as simulações executadas, os resultados obtidos e uma análise destes, levando em consideração os aspectos qualitativos e quantitativos introduzidos nos capítulos 4 e 5.

Finalmente, no capítulo 7 são relatadas as conclusões gerais da condução do trabalho, e um direcionamento para possíveis trabalhos futuros.

2 PROBLEMAS DE RESTRIÇÕES DISTRIBUÍDAS

Antes de abordar os problemas de restrições distribuídas sob a perspectiva de otimização ou satisfação devemos referenciar esse tipo de problema no que diz respeito à questão de alocação distribuída dos recursos, ou seja, devem ser entendidas as questões relativas à distribuição do problema, para então apontar para as formas de tratamento das mesmas. A seção 2.1 trata dos problemas de alocação distribuída de recursos, servindo como base para descrição dos tipos de problema tratados nas seções seguintes a esta. A seção 2.2 trata dos problemas de satisfação de restrições (DisCSP *Distributed Constraint Satisfaction Problem*) (YOKOO et al., 1992), que em linhas gerais envolvem a satisfação de um conjunto de restrições de forma a atender o objetivo de um problema distribuído. A seção 2.3 trata dos problemas de otimização de restrições (DCOP *Distributed Constraint Optimization Problem*) (YOKOO; DURFEE, 1991), onde não apenas se deseja satisfazer um conjunto de restrições, mas sim escolher o conjunto que otimize essa escolha com base em uma avaliação geral das possíveis soluções.

2.1 Problemas de Alocação Distribuída de Recursos

Conforme apresentado em (MODI et al., 2003), os problemas de alocação distribuída de recursos servem como uma interpretação de alto nível para os problemas de satisfação e otimização de restrições, sendo esse último focado nesse trabalho. Essa abstração permite definir meios de mapeamento das características desses problemas nos formalismos propostos individualmente, fornecendo também dessa maneira uma classificação dos problemas entre as subclasses anteriormente citadas.

A primeira etapa consiste da identificação das características que determinam o grau de dinamismo e distribuição do problema. Um problema de alocação distribuída de recursos consiste de um conjunto de agentes capazes de executar um conjunto de operações e um conjunto de tarefas a serem completadas. Para isso, um subconjunto de agentes deve executar as operações necessárias para finalizar as tarefas. Uma solução para esse problema é a seleção do conjunto mínimo das tarefas existentes para execução de cada operação, tal que esses conjuntos não conflitem. Ou seja, como um agente pode executar uma operação por vez, não é possível que duas operações que estejam em subconjuntos diferentes sejam executadas simultaneamente.

2.1.1 Classificação dos Problemas em Nível de Relacionamentos

As definições abaixo, conforme (MODI, 2003) se referem à classificação de problemas de alocação de recursos de acordo com o tipo de conflito.

- SCF - *Strong Conflict-Free*: caracteriza os problemas onde não há duas tarefas que

possuem em comum uma operação do mesmo agente. Essa condição determina que qualquer conjunto de operações, entre as alternativas para uma tarefa, levará a uma solução onde todas as tarefas são completadas;

- WCF - *Weakly Conflict-Free*: ocorre quando há um conjunto de operações para cada uma das tarefas tal que todos os conjuntos de operações não são conflitantes. Essa condição é mais fraca que a SCF, pois requer que exista apenas uma solução;

- OC: significa *Over Constrained*. Um problema que não pode ser dito WCF é chamado de OC. Nos problemas OC, não necessariamente todas as tarefas são executadas concorrentemente por questão de limitação de recursos.

Pode-se dizer que um problema de alocação dinâmica de recursos é a seqüência de resoluções de problemas estáticos, assim sendo, um problema dinâmico é, no mínimo tão complexo quanto um estático. Em problemas OC pode não haver recursos suficientes para completar todas as tarefas presentes. Dessa forma, pode-se desejar encontrar uma solução maximizada. Ou seja, deseja-se escolher um subconjunto das tarefas de forma que um número máximo de tarefas seja solucionado. Esse é um problema NP-completo.

2.1.2 Mapeamento de Problemas de Alocação Distribuída em DCOPs

Os problemas OC se referem às questões de otimização e, portanto tratados por algoritmos DCOP. Um problema OC pode ser mapeado em um problema DCOP. Para isso assumimos uma função de custo que determina um valor referente a uma tarefa. O objetivo dos agentes, dessa forma, passa a ser achar um conjunto de tarefas tal que o valor somatório de custos das tarefas seja minimizado, e ainda assim haja um número suficiente de agentes para completar as mesmas.

2.2 Problemas de Satisfação de Restrições Distribuídas

Originado dos problemas de satisfação de restrições (CSP - *Constraint Satisfaction Problem*), o formalismo para problemas de satisfação de restrições **distribuídas** foi proposto com o objetivo de modelar o processamento das interações entre agentes, e seu processo local de decisão envolvidos na computação distribuída. Em DisCSP os agentes, controlando o valor de uma determinada variável, devem atribuir valores a essas variáveis de forma a satisfazer uma função global. Um conjunto de valores para as variáveis é considerado uma solução caso todas as restrições sejam satisfeitas. Entre as soluções propostas para essa classe de problemas se encontram as baseadas em técnicas de busca exaustiva como busca em profundidade (*Depth First Search*) e em largura (*Breadth First Search*).

A busca com retrocesso (*Backtracking*) é um método muito conhecido na resolução de problemas de satisfação de restrições. Resume-se como um método para resolver uma série de subproblemas onde cada um destes pode ter várias soluções possíveis, e cada solução escolhida afeta os resultados para os outros subproblemas. De acordo com (RUSSEL; NORVIG, 1995), para resolver o problema geral, é necessário achar uma solução para o primeiro subproblema e então, recursivamente, resolver os outros subproblemas baseado na primeira solução. Caso não seja possível, há um retrocesso e então se busca a próxima possível solução para o primeiro subproblema, e assim sucessivamente. O retrocesso termina quando não há mais soluções para o primeiro subproblema. Assume-se que os espaços X_i da seleção contêm apenas um número finito de valores distintos.

Os números finitos podem ser grandes e podem corresponder à discretização de intervalos contínuos. Assumindo M_i o número de valores distintos em X_i, e $M = \prod_{i=1}^{n} M_i$, a aproximação por *força bruta* significa formar cada um dos vetores possíveis da amostra de M, avaliar cada resultado de acordo com o critério de otimização e ver qual produz o maior valor. O algoritmo de busca por retrocesso é projetado para gerar a mesma resposta com menos tentativas de M e quando o conjunto M é grande (geralmente) isto se transforma em um ganho muito importante. Apesar de ser melhor que a geração de todas as soluções possíveis, segundo (KUMAR, 1992), para a maioria dos problemas práticos a busca com retrocesso simples temos tempo de execução exponencial.

A idéia básica deste tipo de busca é construir sobre um vetor da amostra um componente a cada momento e usar funções de critério para testar se o vetor que está sendo gerado ainda tem uma possibilidade de sucesso (GOLOMB; BAUMERT, 1965). Ainda, o algoritmo de busca com retrocesso bem construído é aquele que elimina grandes regiões do espaço cartesiano de busca. A eficiência do método depende muito de como a função de critério é modelada, já que uma função com poucos critérios faz com que um espaço muito grande de soluções seja considerado e uma função muito elaborada pode demorar muito tempo para ser executada em cada caso.

No caso de implementações distribuídas esse tipo de algoritmo implica maiores estudos, pois na busca de retrocesso o processamento alterna constantemente entre diferentes variáveis. Cada mudança de estados requer, pelo menos, uma mensagem (KASIF, 1986). Essa questão tem sido um grande problema para a aplicação de algoritmos distribuídos no *mundo real*, especialmente em problemas de otimização pelas questões relacionadas ao custo envolvido nas trocas de mensagens e tempo de processamento.

Esse tratamento de satisfação é considerado inadequado para problemas reais onde as possíveis soluções possam ser avaliadas em graus de qualidade, ou custo. Como exemplo, o problema de controle semafórico aqui tratado requer não apenas que uma configuração de planos semafóricos seja escolhida, mas também que essa solução seja a mais adequada, entre as possíveis, para a situação de tráfego. Além disso, pode ser impossível em alguns casos satisfazer todas as restrições, e então seria mais adequado **minimizar** o número de restrições não satisfeitas.

2.3 Problemas de Otimização de Restrições Distribuídas

Derivado de DisCSP, DCOP representa uma classe de problemas distribuídos com uma grande área de aplicação em SMA, onde são tratadas, entre outras, questões de coordenação. Como exemplo dessas aplicações pode ser citado o planejamento e agendamento distribuído (MAHESWARAN et al., 2004), distribuição de tarefas aplicado ao cenário de salvamento em desastres (KITANO et al., 1999), trabalho em equipe (TAMBE, 1997), robôs reconfiguráveis e distribuídos (SHEN; YIM, 2002), entre outros. De uma forma geral, DCOP fornece uma base para investigação de como os agentes podem coordenar seu processo de decisão em seu domínio de atuação.

Em geral, problemas de otimização são muito mais complexos que problemas de satisfação, justamente pelo motivo de não se estar interessado em achar **alguma solução**, mas sim a melhor delas, o que implica em uma maior exploração do espaço de estados. Baseado nisso, e na idéia de que é necessário haver um relacionamento entre os nodos da distribuição, pode-se dizer que um dos objetivos comuns a todos os algoritmos de otimização, visando desempenho, é otimizar o processo de comunicação (informações trocadas) necessário para encontrar uma solução.

Qualquer abordagem para aplicação com DCOP em problemas reais deve atender a alguns requisitos básicos. Primeiramente, por se tratar de domínios distribuídos, o método deve prever que os agentes possam otimizar alguma função global **de forma distribuída**, usando apenas **comunicação local** (agentes próximos ou dentro de um grupo). Ou seja, não é aceitável a existência de um agente central responsável por todo o processamento. O método também deve ser capaz de encontrar soluções de forma rápida com agentes operando assincronamente. Um agente não pode perder tempo esperando alguma resposta do sistema enquanto poderia estar trabalhando na solução. Finalmente, o método deve fornecer alguma garantia de qualidade.

Visto isso, percebe-se que um dos maiores obstáculos para resolução de DCOP é combinar assincronia com garantia de qualidade. Algumas abordagens falham em atender esses dois requisitos porque há uma grande dificuldade em garantir resultados quando os agentes estão mudando o valor de suas variáveis assincronamente. Como exemplo, o algoritmo *SynchBB* (HIRAYAMA; YOKOO, 1997) trata de DCOP onde os agentes concluem com segurança que a atual solução parcial não levará a um ótimo global comparando custos com um limite global. Essa abordagem falha em ser assíncrona e paralela porque computar um limite global requer que todos os custos das restrições sejam acumulados em um único agente, antes que as decisões sejam tomadas. Outra alternativa seria a aplicação repetida de algoritmos para problemas de satisfação de restrições (DisCSP), como o algoritmo ABT (YOKOO; HIRAYAMA, 1998). Um agente executando ABT conclui com certeza que a solução atualmente sendo explorada não levará a uma solução satisfatória global sempre que localmente for detectada uma restrição não satisfeita. Essa abordagem falha ao tentar generalizar DCOP porque se baseia na representação limitada de DisCSP, onde apenas uma restrição precisa ser quebrada para uma solução candidata ser inconsistente em nível global.

2.4 Definição formal para DCOP

Um problema DCOP é definido formalmente da seguinte maneira:

- um conjunto de n_a agentes, $A = \{a_1, a_2, ..., a_{n_a}\}$;
- um conjunto de n_v variáveis, $V = \{v_1, v_2, ..., v_{n_v}\}$;
- um conjunto de domínios $D = \{d_1, d_2, ..., d_{n_v}\}$, onde $v_i \in d_i$. Cada d_i é finito e discreto;
- um conjunto de funções de custo $f_{i,j} : d_i \times d_j \rightarrow N$, para o par de variáveis v_i e v_j. Funções de custo também podem ser chamadas de restrições;
- um mapeamento distribuído $Q_l : v_i \rightarrow a_j$ atribuindo cada variável a um agente. $Q(v_i) = a_j$ significa que o agente a_j é responsável por escolher um valor para v_i.
- uma função objetivo F, definida como uma agregação sobre o conjunto de restrições. O objetivo, com essa função, é encontrar um conjunto de valores O^* para as variáveis V, tal que a função objetivo seja minimizada ou maximizada. A função F objetivo é definida como $F(O) = \sum f_{i,j}(v_i, v_j)$, onde $v_i \leftarrow d_i, v_j \leftarrow d_j$ em O.

Em termos de SMA, um DCOP assume que as variáveis e restrições de um problema são distribuídas entre um grupo de agentes e, através da comunicação, devem achar a melhor atribuição de valores para essas variáveis. O objetivo dos agentes é escolher valores

para essas variáveis de forma que a função objetivo seja minimizada ou maximizada. Dois agentes cujas variáveis compartilham uma restrição são chamados vizinhos. Assume-se que os agentes podem mandar mensagens para qualquer agente do qual possui conhecimento e inicialmente os agentes apenas têm conhecimento sobre seus vizinhos.

2.5 Algoritmos mais Populares para DCOP

Foram desenvolvidos vários algoritmos distribuídos para tratar DCOP de forma ótima. Como exemplos: *Distributed Depth-First Branch and Bound* (DDBB) e *Distributed Iterative Deepening* (DID) (YOKOO; DURFEE, 1991), *Synchronous Branch and Bound* (SBB) e *Iterative Distributed Breakout* (IDB) (HIRAYAMA; YOKOO, 1997), o algoritmo *Optimal Asynchronous Partial Overlay* (OptAPO) (MAILLER; LESSER, 2004), o algoritmo *Asynchronous Distributed Optimization* (ADOPT) (MODI et al., 2003, 2005), e o algoritmo *Dynamic Parameter Optimization Problem* (DPOP) (PETCU; FALTINGS, 2005). Todos esses algoritmos têm alguns aspectos em comum: foram originados de algum algoritmo centralizado e todos mantêm total separação do conhecimento dos agentes durante o processo de solução do problema.

Os algoritmos ADOPT, OptAPO e DPOP representam os algoritmos mais populares em termos de tratamento de DCOP. São os mais referenciados na literatura, e apresentam resultados mais recentes sobre avanços na área tratada. Todos apresentam as duas características abaixo:

- são ditos completos: teoricamente garantem o retorno de uma solução ótima;
- são ditos assíncronos: permanecem corretos mesmo quando agentes executam concorrentemente, potencialmente com diferentes velocidades de execução.

Os algoritmos intercalam comunicação com computação. No entanto há algumas diferenças qualitativas.

O ADOPT garante o retorno de uma solução ótima ao mesmo tempo que permite aos agentes escolherem valores para as variáveis em paralelo. Para isso, realiza uma busca distribuída usando a comunicação de custos como forma de guiar os agentes no sentido de escolher valores ótimos para as variáveis. Os agentes comunicam seus valores atuais de variáveis para seus vizinhos, que por sua vez respondem com mensagens contendo limites mais baixos da função global condicionados aos valores escolhidos por outros agentes. Agentes de mais alta prioridade, que detêm mais conhecimento do problema (mais relacionamentos com outros agentes) respondem explorando novos valores. Conforme esse processo continua operando, limites mais baixos se tornam progressivamente mais apurados, até que o limite mais baixo da solução de custo mínimo se torna igual ao limite mais alto, indicando a solução ótima. Os agentes não comunicam diretamente suas restrições a outros agentes, mas apenas enviam mensagens entre vizinhos.

O OptAPO é uma abordagem alternativa para DCOP que usa a comunicação direta de restrições de forma a centralizar parcialmente o problema, através de um mediador. A escolha de um mediador é feira durante o processo de resolução do problema usando prioridades atribuídas aos agentes. Esse processo é denominado mediação cooperativa. O mediador usa um processo centralizado de otimização para encontrar uma solução ótima para sua porção do problema. O processo de otimização originalmente utilizado em (MAILLER; LESSER, 2004) é o algoritmo Branch and Bound (LAND; DOIG, 1960). O processo de comunicação de restrições usa uma técnica que evita a centralização quando

esta não é justificável pela estrutura do problema. Quando um agente recebe mensagens com restrições de outros agentes no problema ele adiciona o outro agente a uma estrutura de dados chamada *GoodList*. O tamanho dessa estrutura pode ser usado para medir o grau de centralização do algoritmo. Quando restrições são comunicadas entre agentes que não são vizinhos, um canal de comunicação é estabelecido entre estes.

Entre os três algoritmos aqui tratados, DPOP é a proposta mais recente para resolução de problemas de restrições distribuídas. DPOP é baseado em programação dinâmica e é um método de propagação de utilidade aplicado em redes de restrições construídas em árvore. Além disso, requer um número linear de mensagens, cujo tamanho máximo está relacionado à árvore de dependências construída para aplicação. Em função da complexidade e dos custos (trocas de mensagens) referentes à resolução de problemas de otimização, é necessário trabalhar com soluções que levem em consideração esse problemas na aplicação real. Por exemplo, um algoritmo que computa incrementalmente o conjunto de todas as soluções parciais para todas variáveis de acordo com certa ordem iria necessitar um número linear de mensagens.

Nas seções seguintes será apresentada uma revisão dos algoritmos mais populares para DCOP. Os algoritmos OptAPO, ADOPT e DPOP serão apresentados, bem como trabalhos de comparação já realizados.

2.6 Algoritmo ADOPT

ADOPT (Asynchronous Distributed Optimization) é um algoritmo para DCOP capaz de encontrar uma solução ótima ou uma solução com um determinado grau de proximidade ao ótimo, de acordo com o especificado pelo usuário, usando apenas comunicação assíncrona e local entre agentes. Comunicação local ocorre quando um agente não se comunica com todos outros agentes, e sim apenas com um determinado subgrupo de vizinhança. O ADOPT depende de um único agente central para agregar os limites de custo global e verificar a satisfação global para término da computação. Essa dependência gera um grau de centralização no algoritmo, ao mesmo tempo em que outras características, como a computação em paralelo dos agentes, o torna distribuído.

O ponto chave do algoritmo é alcançar a assincronia permitindo que os agentes mudem os valores das variáveis sempre que percebam que outra solução pode ser melhor que a atual. Sendo assim, garante computação assíncrona, pois o agente não precisa informação global para tomar suas decisões locais. Como essa idéia permite que soluções parciais sejam declinadas antes que a solução sub-ótima seja provada, essas soluções parciais precisam ser reconsideradas posteriormente.

Outro ponto importante do algoritmo é o mecanismo de detecção de término junto ao algoritmo. Os agentes terminam sempre que encontram uma solução completa cujo custo esteja abaixo de seu limite atual. Outros algoritmos assíncronos tipicamente requerem que um algoritmo para detecção de término seja executado separadamente, adicionando complexidade ao problema pela interação (mensagens) entre esses.

Basicamente, o algoritmo ADOPT se divide em três pontos: uma estratégia de busca assíncrona onde soluções podem ser abandonadas antes de serem consideradas sub-ótimas; uma reconsideração eficiente dessas soluções abandonadas; e detecção de término da execução.

2.6.1 Busca *Best-First*

Os agentes são considerados em ordem de prioridade em uma estrutura de árvore onde cada agente possui apenas um pai e vários filhos. Usando esse ordenamento, o ADOPT realiza uma busca distribuída usando uma estratégia *Best-First*, ou seja, cada agente sempre escolhe o valor de variável com o menor limite inferior. Essa estratégia contrasta com outras formas de busca distribuída como *Branch and Bound*, que é usado por outros algoritmos como OptAPO e SynchBB, e requer que os agentes tenham acesso a um limite superior global. Isso é vantajoso porque um limite inferior pode ser computado sem necessariamente acumular informações de custo global.

No ADOPT, um limite inferior inicial é computado baseado apenas no custo local, e a partir daí é refinado iterativamente enquanto novas informações são assincronamente recebidas de outros agentes. Essa estratégia de busca permite, como dito anteriormente, que os agentes abandonem soluções sub-ótimas.

2.6.2 Reconsideração de Soluções Abandonadas

Para permitir que os agentes reconstruam eficientemente soluções previamente exploradas, o ADOPT usa a idéia de adotar um limite inferior como forma de retornar ao ponto inicial (*backtrack threshold*). O processo de retrocesso é dado pelo algoritmo 2.2. A idéia principal determina que quando um agente sabe, por experiência passada nas buscas, que *lb* (*lower bound*) é um limite inferior para sua sub-árvore, ele deve informar aos agentes em sua sub-árvore para não desperdiçarem tempo buscando uma solução cujo custo é inferior a *lb*. Dessa forma, uma agente pai determina o valor do limite de busca (*backtrack threshold*) e comunica aos seus filhos, e estes por sua vez não mudam o valor de suas variáveis enquanto o custo é inferior ao limite de busca fornecido pelo pai. Uma vez que o limite de busca é calculado usando um limite inferior previamente conhecido, é garantido ser igual ou menor ao custo da solução ótima. Isso garante que a solução ótima não será ignorada. Usar limites de busca para reconstruir soluções previamente exploradas se torna mais difícil quando um agente tem múltiplos filhos. Um agente deve ser capaz de subdividir os limites de busca corretamente entre seus vários filhos. Isso é uma tarefa complexa, pois os agentes não podem lembrar como o custo, proveniente de seus filhos foi acumulado no passado, pelo menos sem requerer espaço de busca exponencial no pior caso. Essa dificuldade é tratada permitindo aos agentes subdividir o limite de busca arbitrariamente e corrigir essa subdivisão ao longo do tempo pelo recebimento de um *feedback* de seus filhos.

2.6.3 Detecção de término

Usa-se a idéia de limites para acompanhar o progresso em busca da solução ótima. Esses limites consistem de uma valor inferior (*lb*) e outro superior (*ub*) em função do custo da solução ótima. Quando o tamanho do intervalo atinge zero (limites inferior e superior iguais), o custo da solução ótima foi determinado e os agentes podem terminar a execução. Além disso, esses intervalos podem ser usados para determinar soluções próximas do ótimo, com um grau de proximidade determinado pelo tamanho do intervalo. Isso significa que os agentes podem encontrar soluções de forma mais rápida com garantias teóricas da qualidade da solução global.

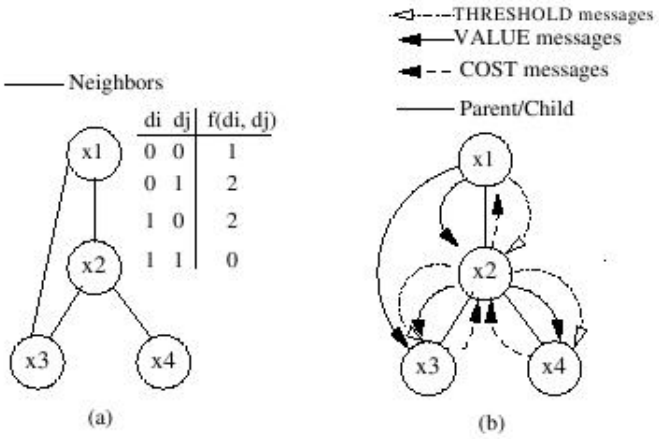


Figura 2.1: Árvore DFS a partir de um grafo de restrições (MODI et al., 2003)

2.6.4 Descrição do algoritmo ADOPT

Os agentes são inicialmente ordenados usando uma árvore de busca *Depth-First* (DFS), definindo relacionamentos pai-filho entre os agentes. O uso de árvores DFS é proposto no contexto de DisCSP. Para um dado grafo de restrições, uma árvore DFS é valida se não existem restrições entre agentes de diferentes sub-árvores. Restrições são permitidas apenas entre pais e filhos. Existem várias árvores DFS possíveis para um dado grafo de restrições. A figura 2.1, retirada de (MODI et al., 2003), mostra uma árvore DFS formada de um grafo de restrições. Nessa figura é assumido que pais e filhos são também vizinhos.

Cada agente é inicializado conforme descrito no algoritmo 2.1. A atribuição de valores a variáveis, através de mensagens *VALUE* são enviadas de pai a filho na árvore DFS enquanto o retorno (*feedback*) de custo percorre o caminho inverso, através de mensagens *COST*. Mensagens *THRESHOLD* são usadas para reduzir buscas redundantes e enviadas apenas de pai para filho. Essa comunicação também pode ser visualizada na figura 2.1.

Uma mensagem *COST* enviada de um filho para seus pais contém o custo calculado por esse agente mais os custos calculados de seus filhos. Uma mensagem *THRESHOLD* contém um simples valor representando o limite de busca (inicialmente zero).

Um contexto é uma solução parcial na forma de $(v_j, x_j), ..., (v_k, x_k)$, onde v_j representa a variável e v_j representa seu valor. Uma variável pode aparecer apenas uma vez em um contexto. Dois contextos são compatíveis se eles não discordam na atribuição de nenhuma variável. O *CurrentContext* é o contexto que armazena a visão do agente a_i sobre as variáveis de seus vizinhos.

Uma mensagem *COST* contém três campos:

- o contexto
- o *lb*: limite inferior
- o *ub*: limite superior

O campo contexto, enviado de um agente para seus pais contém o seu atual *CurrentContext*. Este campo é necessário porque os custos calculados são dependentes dos valores das variáveis, portanto um agente deve sempre enviar o contexto sobre o qual os custos foram calculados em cada mensagem *COST*. Quando a_i recebe uma mensagem

COST de seu filho a_l, e x é o valor de a_i no campo de contexto, então a_i guarda lb, indexado por x e a_l, na forma $lb(x,a_l)$. De forma similar, o campo ub é armazenado como $ub(x,a_l)$ e o campo de contexto é armazenado como $contexto(x,a_l)$.

Antes que qualquer mensagem *COST* seja recebida ou sempre que os contextos se tornam incompatíveis, ou seja *CurrenteContext* se torna incompatível com $context(x,a_l)$, o valor de $lb(x,a_l)$ é reinicializado para zero e $ub(x,a_l)$ é reinicializado para um valor máximo *Inf*.

O agente a_i calcula o custo como o custo local mais o custo recebido por *feedback* de seus filhos. O custo local cl para um determinado valor de v_i pertencente a d_i, é a soma dos custos das restrições entre a_i e seus vizinhos.

Quando a_i recebe uma mensagem *COST*, ele adiciona $lb(x,a_l)$ ao seu custo local $c(x)$ para calcular um limite inferior para o valor x, denotado por $lb(x)$. De forma similar, a_i adiciona $ub(x,a_l)$ ao seu custo local $cl(x)$ para calcular um limite superior para o valor x, denotado por $ub(x)$. O menor limite inferior sobre todos os valores escolhidos para v_i é o limite inferior para a variável v_i, denotado por LB. De forma similar o menor limite superior sobre todos os valores escolhidos para v_i é o limite superior para a variável v_i, denotado por UB.

O agente a_i envia LB e UB para seus pais nos campos lb e ub de uma mensagem *COST*. De forma intuitiva, $LB = k$ indica que não é possível a soma dos custos locais em cada agente da sub-árvore a partir de a_i ser menor que k, dado que todos agentes escolheram os valores no *CurrentContext*. De forma similar, $UB = k$ indica que o custo ótimo na sub-árvore a partir de a_i não será maior que k, dado que todos agentes escolheram os valores no *CurrentContext*. Vale lembrar que um agente folha não tem sub-árvore, então $cl(x) = LB(x) = UB(x)$ para todas as escolhas de valores para v. Se a_i não é uma folha e ainda não recebeu nenhuma mensagem *COST* de seus filhos, UB é igual ao valor máximo *Inf* e LB é igual ao menor custo local $cl(x)$ entre todas as escolhas de valores para v pertencente a d_i.

O valor do threshold de busca para a_i é armazenado na variável *threshold*, inicializada em zero. Seu valor é atualizado de duas formas:

- o valor pode ser aumentado sempre que a_i determinar que o valor de custo da solução ótima em sua sub-árvore deve ser maior que o valor atual de *threshold*;

- se a_i determina que o custo da solução ótima em sua sub-árvore deve necessariamente ser menor que o valor atual de *threshold*, então o *threshold* diminui.

Essas duas atualização são feitas comparando *threshold* com LB e UB. A atualização do threshold é sintetizada nas seguintes invariantes:

• *ThresholdInvariant*: $LB \leq threshold \leq UB$. O *threshold* no custo para a sub-árvore a partir de a_i não pode ser maior que UB ou menor que LB. Um pai também pode mudar o valor do *threshold* de um filho mandando uma mensagem *THRESHOLD*. A razão para isso é que em alguns casos, os pais podem determinar um limite no custo ótimo de uma solução em uma sub-árvore do agente, mas o agente pode não conhecer esse limite. Sendo assim o pai informa através de uma mensagem *THRESHOLD*. Um agente pai pode configurar corretamente o valor de *threshold* de seus filhos subdividindo seu próprio *threshold* entre seus filhos, e então usando as duas equações seguintes para recalcular ao longo do tempo enquanto um *feedback* é retornado dos filhos. Denotamos $t(x,a_l)$ como o *threshold* no custo alocado pelo pai a_i para o filho a_l, dado que a_i escolhe x. Então os valores de $t(x,a_l)$ estão sujeitos às seguintes invariantes:

• *AllocationInvariant*: para o valor atual de v_i pertencente a d_i, *threshold*$= cl(x) + \sum x_l \in Childrent(x,x_l)$. O *threshold* no custo para a_i deve ser igual ao custo local, ao escolher, mais a soma dos *thresholds* alocados para os filhos de a_i.

• *ChildThresholdInvariant*: para qualquer $x \in d_i$, para qualquer $a_l \in Children$, $lb(x,a_l) \leq t(x,a_l) \leq ub(x,a_l)$. O *threshold* alocado ao filho a_l pelo pai a_i não pode ser menor que o limite inferior ou maior que o limite superior enviado por a_l a a_i.

Com essas invariantes, um agente pode usar seu próprio *threshold* para determinar limites sobre o custo da solução ótima na sub-árvore de seus filhos. O valor *threshold* é usado para determinar quando mudar o valor da variável. Sempre que $lb(v_i)$ excede *threshold*, a_i muda o valor de suas variáveis para um com um valor de limite inferior menor. Vale lembrar que a_i não pode provar que o seu valor atual é definitivamente sub-ótimo porque é possível que o *threshold* seja menor que o custo da solução ótima. No entanto, ele muda o valor para um com um custo menor de qualquer forma, para isso usando a estratégia de busca *Best-First*, descrita na seção 2.6.1.

Algoritmo 2.1 ADOPT - Inicialização

$threshhold \leftarrow 0$; $CurrentContext \leftarrow []$;
para todo $d \in D_i; x_l \in Children$ **faça**
  $lb(d,x_l) \leftarrow 0; t(d,x_l) \leftarrow 0$;
  $ub(d,x_l) \leftarrow Inf; context(d,x_l) \leftarrow []$;
fim para
$d_i \leftarrow d$ que minimiza $LB(d)$;
$backtrack$;

Algoritmo 2.2 ADOPT - Retrocesso (*backtrack*)

se $threshold == UB$ **então**
  $d_i \leftarrow d$ que minimiza $UB(d)$;
senão se $LB(d_i)\, threshold$ **então**
  $d_i \leftarrow d$ que minimiza $LB(d)$;
fim se
$SEND(VALUE,(x_i,d_i))$ para cada vizinho de menor prioridade;
$maintainAllocationInvariant$;
se $threshold == UB$ **então**
  se recebeu TERMINATE do pai ou x_i é *root* **então**
    $SEND(TERMINATE, CurrentContext \cup (x_i,d_i))$ para cada filho;
    termina execução;
  fim se
fim se
$SEND(COST,x_i,CurrentContext,LB,UB)$ ao pai;

2.7 Algoritmo OptAPO

O algoritmo OptAPO (MAILLER; LESSER, 2004) é uma abordagem para DCOP baseada na idéia de mediação cooperativa. Como dito anteriormente, O OptAPO (assim como o ADOPT) é um algoritmo completo. Os agentes podem estender o contexto usado

para tomada de decisão local para um grafo de relacionamento. Através do processo denominado **mediação cooperativa**, um dos agentes deve atuar como mediador, computando a solução para este contexto estendido, e como resultado disso recomendando valores para as variáveis dos agentes associados ao processo mediativo.

Conforme (MAILLER; LESSER, 2004), o algoritmo OptAPO, assim como sua variante para satisfação de problemas distribuídos, o APO (MAILLER; LESSER, 2003), garante a resolução de problemas de forma rápida, distribuída e assíncrona sem a explosão no *overhead* de comunicação normalmente associada com os algoritmos distribuídos.

Isto é possível porque os agentes geram uma lista, também denominada *GoodList*, que armazena o nome dos agentes que se relacionam direta ou indiretamente com o dono da lista, e uma lista denominada *AgentView*, que armazena os nomes, valores, domínios e as relações funcionais com os agentes relacionados. No processo de solução do problema cada agente tenta melhorar o valor de seu subproblema (o problema que o agente resolve utilizando seu grafo de relacionamentos).

A prioridade para assumir o papel de mediador é dada ao agente que detém mais informação sobre o problema. Um grafo modela o DCOP onde cada nodo é um agente e as relações são as restrições do problema. Cada restrição ou relação funcional tem um custo associado.

No decorrer da solução do problema, cada agente busca melhorar o valor qualitativo da solução do seu subproblema, centralizado em sua *GoodList*, ou então justificar seu custo, identificando no grafo de relacionamentos as estruturas que contribuem para isso.

Para esse processo, os agentes fazem uso da função de mediador, computam o valor ótimo de seus subsistemas, e buscam mudar os valores das variáveis envolvidas na sessão de mediação, com o objetivo de otimizar seus valores. Todavia, essa otimização não é alcançada sem implicar custos para os agentes fora da sessão. O mediador assume que estes agentes fora da sessão estão envolvidos em estruturas similares para justificativa de custos. O processo continua até que cada um dos agentes tenha justificado os custos de seu subproblema centralizado.

Para facilitar a solução do problema, cada agente tem uma prioridade dinâmica, que é baseada no tamanho de sua *GoodList*. Essas prioridades são usadas pelos agentes para decidir a ordem na qual um ou mais agentes farão a mediação quando houver necessidade. Essa ordenação por prioridades é importante por duas razões:

- a prioridade garante que o agente com maior conhecimento tome as decisões. Isso melhora a eficiência do algoritmo, diminuindo os efeitos de decisões com visão parcial (agentes com menos visão do problema - miopia);
- a prioridade melhora a eficácia do processo de mediação. Agentes de baixa prioridade esperam que agentes de alta prioridade façam a mediação.

Os três estágios do algoritmo são descritos abaixo:

2.7.1 Inicialização

Durante a inicialização, aos agentes é fornecido um valor (pode ser aleatório, caso não haja atribuição), e as funções sobre suas variáveis. Sendo assim, cada agente a_i envia uma mensagem de inicialização para cada um de seus vizinhos a_j. Essa mensagem de inicialização contém o nome da variável (v_i), prioridade (p_i), valor corrente(x_i), domínio(d_i), e relação funcional(R_i). A mensagem também contém a variável μ, que indica o desejo de mediar do agente, e uma lista de agentes existentes entre i e j no grafo

de relacionamentos. A lista *InitList* guarda o nome dos agentes para os quais a mensagem de inicialização foi mandada.

Quando um agente recebe uma mensagem de inicialização (tanto pela inicialização quanto por uma requisição posterior), ele guarda a informação em sua *AgentView* e adiciona a variável à lista *GoodList*. Um agente somente é adicionado a *GoodList* se ele está ligado, por um relacionamento funcional, a um agente que já está na lista. Isso garante que o grafo de agentes criado pela *GoodList* sempre permaneça conectado.

A *InitList* é então usada para verificar se a mensagem é um pedido de conexão ou uma resposta a um pedido de conexão. Se o agente não está na *InitList* isso significa que é um pedido de conexão, então uma resposta é gerada e enviada. Se um agente está na *InitList*, a mensagem é uma resposta a um pedido de conexão previamente enviado. Nesse caso uma mensagem de resposta não é enviada (evitando um /textitloop infinito de comunicação entre agentes).

Os agentes na *GoodList* são um subconjunto dos agentes contidos na *AgentView*. Assim é tido para manter a integridade da *GoodList* e permite que as conexões sejam bidirecionais.

Podemos considerar o caso onde um agente tem sido repetidamente o mediador e estendeu seu subproblema local por um longo caminho no grafo de relacionamentos. À medida que isso acontece, ele se conecta com agentes que podem ter uma visão muito limitada e não tem conhecimento de sua ligação indireta com o mediador. Para que a conexão seja bidirecional, quem recebe o pedido de conexão precisa guardar o nome do agente que pede a conexão, mas não pode adicioná-lo a *GoodList* até que um caminho entre ambos seja identificado. Para garantir resultados ótimos ao algoritmo, cada um dos agentes não finaliza até que todas as relações funcionais entre um agente em sua *AgentView* apareçam na *GoodList*. Durante períodos de inatividade, os agente buscam essa condição estável de relacionamentos aumentando sua *GoodList* para incluir esses relacionamentos parciais.

2.7.2 Verificação da Visão do Agente

Após receber todas as mensagens de inicialização, os agentes executam um procedimento para verificar a *AgentView*. O objetivo do agente é melhorar a solução de seu subproblema (representado por F_i). Com isso, durante o segundo estágio a *AgentView* é usada para calcular o custo corrente F_i no sub-grafo de relações dado pela *GoodList* de i. Se $F_i > F_i^*$ (F_i^* como sendo o valor ótimo do subsistema), então o agente i conduz uma seção para mediação passiva ou ativa. Essa verificação garante que, ao término do algoritmo, todos agente terão otimizado o custo dado por F_i. Na inicialização, o valor de F_i^* é sempre 0, significando que a melhor resposta obtida até então não teve custo. Esse valor muda à medida que o agente centraliza estruturas que associam um custo à resposta ótima. Os agentes podem determinar esse custo em função da centralização usada no processo de mediação. Sempre que um agente media, ele recalcula o valor de F_i^* executando uma busca centralizada em sua *GoodList*. Agentes decidem entre uma mediação passiva ou ativa baseados na existência ou não de outro agente com prioridade igual ou menor a dele nas relações funcionais sub-ótimas em sua *GoodList*. Se um dos agentes possui prioridade menor ou igual, a mediação vai ser ativa, caso contrário vai ser passiva.

As duas principais diferenças entre mediação passiva e ativa são:

- durante uma mediação ativa, o agente receptor ativa seu *flag* de mediação, evitando que ele entre em outro processo de mediação sem ter acabado o atual. Isso causa

uma estabilidade no sistema, que permite que a mediação tenha efeito completo, mas reduz paralelismo, porque evita que outros agentes medeiem;

- a intenção da mediação passiva é verificar e entender os resultados que agentes de alta prioridade obtiveram sem interferir em suas ações ou mudando sua solução corrente.

Em outras palavras, a intenção da mediação passiva é mudar o valor de F_i^*, e a intenção da mediação ativa é mudar o valor de F_i^* e F_i.

Mediação passiva aumenta o paralelismo do processo de resolução do problema e permite que os agentes ganhem em termos de contexto (aumentando sua visão) sem causar instabilidade ao sistema. Se um agente decide que quer mediar ativamente, ele só pode fazê-lo quando um agente de mais alta prioridade não o fizer. Os agentes no sistema sabem quando um agente de mais alta prioridade quer mediar em função da *flag* μ, mencionado anteriormente. Sempre que um agente verifica sua *AgentView* ele recalcula o valor dessa *flag*, que determina seu desejo de mediar no futuro, caso a oportunidade seja dada. Essa informação é compartilhada com cada um dos agentes na sua *AgentView* sempre que os valores mudam.

Como um mediador ativo, um agente primeiro busca retificar a condição sub-ótima mudando suas próprias variáveis. Isso previne que sessões ocorram sem necessidade, estabilizando o sistema e poupando mensagens e tempo. Se o mediador acha um valor que torna $F_i = F_i^*$ e ele percebe que a relação funcional sendo melhorada é compartilhada apenas com agentes de baixa prioridade, ele executa a alteração e envia uma mensagem *value?* para os agentes em sua *AgentView*. Se ele não encontra esse valor que torna $F_i = F_i^*$, então ele inicia uma sessão de mediação ativa descrita no próximo item. Uma mensagem *value?* é simular uma mensagem de inicialização. Ela contém informações sobre prioridade, valor corrente, etc. sobre uma variável. Ao contrário da mensagem de inicialização, no entanto, uma mensagem *value?* contém uma lista c_i, que possui o nome de qualquer agente que compartilha uma relação com custo maior que 0 com o agente que envia a mensagem. Usando essa lista, um agente pode dizer quando há conflitos que precisam ser incluídos em sua *GoodList*. Isso permite que os agentes previnam centralização desnecessária quando a situação indica que não há benefícios diretos. Em outras palavras, não há necessidade de adicionar um agente ao subsistema local quando ele não está envolvido em uma relação que causa aumento do custo global.

2.7.3 Mediação

A sessão de mediação é a porção mais complexa do protocolo. Como dito anteriormente, nessa seção, um agente decide mediar se ele acha $F_i > F_i^*$. A sessão de mediação ocorre com o mediador enviando uma mensagem *evaluate?* para cada um dos agentes em sua *GoodList*. O objetivo disso é:

- informar aos agentes receptores que uma mediação está prestes a iniciar. Se a mediação for ativa tenta obter um *lock* do agente;

- obter informações do agente sobre os efeitos de efetuar mudanças em suas variáveis locais. Obtendo essas informações o mediador toma conhecimento sobre variáveis e relacionamentos fora de sua visão local sem precisar diretamente e imediatamente conectar a outros agentes.

Algoritmo 2.3 Algoritmo *Branch and Bound*

se $Variaveis == vazio$ **então**
  $Melhor\_solucao \leftarrow Caminho\_de\_busca$;
  $N \leftarrow Distancia$;
  se $N \leq S$ **então**
    retorna "acabado";
  senão
    retorna "continue a procura";
  fim se
senão se $Valores == vazio$ **então**
  retorna "continue a procura";
senão se $Distancia = N$ **então**
  retorna "continue a procura";
senão
  $Valor\_Atual \leftarrow$ primeiro valor em $Valores$;
  $Nova\_Distancia \leftarrow Distancia$;
  tenta escolhas no $Caminho\_de\_busca$ com $Nova\_Distancia < N$:
  se a escolha é inconsistente com o $Valor\_Atual$ **então**
    $Nova\_Distancia \leftarrow Nova\_Distancia + 1$;
  fim se
  se $Nova\_distancia < N$ e
  Branch and Bound($Caminho\_de\_busca$ mais $Valor\_Atual$, $Nova\_Distancia$, $Variaveis$ menos a primeira, $Valores$ da segunda variável em $Variaveis$) = "acabado"
  então
    retorna "acabado";
  senão
    retorna **Branch and Bound**($Caminho\_de\_busca$, $Distancia$, $Variaveis$, $Valores$ menos $Valor\_atual$);
  fim se
fim se

Quando um agente recebe um pedido de mediação ele irá responder com uma mensagem *wait!* ou *evaluate!*. Os agentes respondem com uma mensagem *wait!* sempre que o pedido for para uma sessão ativa e o agente ou está envolvido em outra sessão ou está esperando um pedido para uma sessão ativa de outro agente que é de mais alta prioridade que o atual. Todos os pedidos de mediação passiva são retornados, e pedidos para mediação passiva são declinados somente quando absolutamente necessário. Quando um mediador tiver recebido respostas de todos agentes para os quais ele enviou pedidos, ele escolhe uma solução. Agentes que enviaram uma mensagem *wait!* são removidos da mediação. O mediador então conduz uma busca *Branch and Bound* (conforme algoritmo 2.3, adaptado de (FREUDER; WALLACE, 1992)), na qual, como critério primário, é minimizado o custo do subproblema na *GoodList*, e como critério secundário minimiza o custo para agentes fora da sessão. O resultado de minimizar o critério primário, sendo o valor ótimo para o problema na *GoodList*, é armazenado como F_i^*. Os agentes usam duas táticas especiais durante a busca:

- os valores são ordenados de forma que a primeira porção da busca seja a solução corrente. Isso geralmente limita a valores próximos de F_i^* após ser explorado to-

mando vantagem do trabalho previamente feito no problema;

- a busca é terminada antes sempre que o limite é igual ao atual F_i^* e o custo para os agentes fora da mediação é 0. Esse método funciona porque o F_i^* atual é sempre um limite baixo no F^* atual e F_i^* apenas aumenta em buscas sucessivas por causa da natureza monotônica do número de variáveis na busca e função de agregação.

Na conclusão da busca local, o mediador computa dois valores, $F_i^{'}$ e $F_s^{'}$. O valor $F_i^{'}$ é o custo, dado o atual conjunto de variáveis e seus valores, para o subsistema estendido composto dos agentes na *GoodList* junto com qualquer outro agente que aparece nas informações de preferência retornadas nas mensagens *evaluate!*. O valor $F_s^{'}$ é o valor para esse mesmo subsistema estendido dada a solução s retornada pela busca centralizada. Idealmente, esses valores vão ser os mesmos. Em outras palavras, s tem um efeito não negativo no estado atual do problema global. Como agentes agem miopicamente quando eles computam s, algumas vezes eles escolhem uma solução que parece boa, mas no geral tem efeito negativo. Sempre que isso ocorre, o mediador ignora s (mantendo mudanças a F_i^*) e reverte para a solução atual, evitando efetivamente que faça uma decisão local ótima que possui conseqüências globais ruins. O efeito geral é similar à mediação passiva no sentido que mantém estabilidade no sistema enquanto agentes ganham em contexto. Após computar esses valores, o mediador conecta (envia mensagens de inicialização) com qualquer outro agente que não está em sua *AgentView* e foi forçado a aumentar seu custo em função de s. Isso ocorre mesmo que o mediador escolha não usar s. Esse passo aumenta a *GoodList* do agente para que na próxima vez que ele mediar, não repita o mesmo erro. O mediador conclui a sessão enviando mensagens *accept!* (se a sessão for ativa) para os agentes na sessão, que por sua vez adotam a resposta proposta e enviam mensagens *value?* para todos outros agentes em sua *AgentView*.

De um forma geral, o processo de escolha do conjunto solução, durante o processo de mediação, pode ser seqüenciado no algoritmo 2.4.

Algoritmo 2.4 OptAPO - Escolha da Solução

seleciona uma solução s com o algoritmo *Branch and Bound* que:
1. minimize o custo para os agentes na *GoodList*;
2. minimize o custo para os agentes fora da seção;

$F_i^* \leftarrow custo(s)$;
$F_i' \leftarrow F_i +$ custo atual para os agentes fora da sessão;
$F_s' \leftarrow F_i^* +$ custo de usar a solução s para os agentes fora da sessão;
se *mediação* $==$ **ativa** e $F_s' \leq F_i'$ **então**
  $v_i \leftarrow v_i'$;
fim se
para todo $x_j \in AgentView$ **faça**
  se $x_j \in GoodList$ **então**
    se $d_j' \in s$ viola um x_k e $x_k \notin AgentView$ **então**
      envia(Inicialização, $(x_i, p_i, v_i, m_i, D_i, C_i, caminho_{i,k})$) para x_k;
      adiciona x_k na *InitList*;
    fim se
    se *mediação* $==$ **ativa** e $F_s' \leq F_i'$ **então**
      envia (aceita!, (d_j', x_i, p_i, v_i)) para x_j;
      atualiza a *AgentView* para x_j;
    senão se *mediação* $==$ **ativa** e $F_s' > F_i'$ **então**
      envia (aceita!, (d_j, x_i, p_i, v_i)) para x_j;
    fim se
  senão se *mediação* $==$ **ativa então**
    envia (value?, $(x_i, p_i, v_i, m_i, c_i)$) para x_j;
  fim se
fim para
mediação $\leftarrow$ **nenhuma**;
verifica a visão do agente;

2.8 Algoritmo DPOP

DPOP (Dynamic Parameter Optimization Problem) é um método para resolução de problemas de restrições distribuídas baseado em programação dinâmica. Caracteriza-se pela utilização de um método de propagação de utilidade aplicado em redes de restrições construídas em árvore. Além disso, requer um número linear de mensagens, cujo tamanho máximo está relacionado à árvore desenhada para aplicação.

Em função da complexidade e dos custos (trocas de mensagens) referentes à resolução de problemas de otimização, é necessário trabalhar com soluções que levem essas questões em consideração na aplicação real. Por exemplo, um algoritmo que computa incrementalmente o conjunto de todas as soluções parciais para todas variáveis, de acordo com certa ordem, iria necessitar um número linear de mensagens. No entanto, as mensagens poderiam crescer exponencialmente em tamanho, e o algoritmo não teria paralelismo.

Recentemente, o algoritmo *sum-product* (KSCHISCHANG; FREY; LOELIGER, 2001) foi proposto para problemas de satisfação de restrições. Esse algoritmo apresenta um compromisso aceitável entre a exploração de um espaço de estados, baseada em programação dinâmica com mensagens de tamanho fixo, e pode facilmente ser implementado de forma distribuída. No entanto, é aplicado apenas para redes de restrições modeladas em forma de árvores. Sendo assim, DPOP se apresenta como uma extensão desse algoritmo para

topologias arbitrárias usando um arranjo de árvores para a representação de grafo do problema, isso considerando o problema de otimização.

2.8.1 Otimização de Restrições Distribuídas para Redes Estruturadas em Forma de Árvores

Para redes estruturadas em forma de árvore foram desenvolvidos métodos de otimização de tempo polinomial. Entre eles, o algoritmo *sum-product* e o algoritmo DTREE (PETCU; FALTINGS, 2004). No caso do algoritmo DTREE, os agentes enviam mensagens *UTIL* (vetores de utilidade) para seus pais. Um filho s_i do nodo s_k envia para s_k um vetor com as utilidades ótimas que podem ser alcançadas pela sub-árvore baseada em s_i, mais $R_{k,l}$ (relações entre k e l, onde l indexa os filhos de k) e são compatíveis com cada valor x_k de s_k. Para os nodos folhas a computação desses valores é imediata, investigando as restrições que eles têm com seus vizinhos, assim iniciando o processo. Cada nodo s_i responde a essas mensagens de acordo com o processo abaixo:

1. Espera por mensagens *UTIL* de todos seus filhos. Como todas sub-árvores são disjuntas, somando elas, s_i computa quanta utilidade cada um de seus valores resulta para toda sua sub-árvore. Isso, junto com as relações entre s_i e seu pai s_j, habilita s_i a computar exatamente quanta utilidade pode ser alcançada por toda sub-árvore baseada em s_i, considerando a compatibilidade com cada um dos valores de s_j. Ainda, s_i pode enviar a s_j suas mensagens *UTIL*. Também, s_i guarda seus valores ótimos em relação a cada valor de s_j;

2. Se for um nodo raiz, s_i pode computar o valor de utilidade ótimo geral correspondente a cada um de seus valores (baseado em todas as mensagens de *UTIL* recebidas), escolher o ótimo, e mandar uma mensagem *VALUE* a seus filhos, informando sua decisão. Até o recebimento da mensagem *VALUE* de seu pai, cada nodo está apto a escolher o valor ótimo para si (como o valor ótimo previamente armazenado referente ao valor que seu pai escolheu), e passar isso aos seus filhos. Nesse ponto, o algoritmo finaliza para s_i.

O funcionamento desse algoritmo é provado em (PETCU; FALTINGS, 2004), bem como o fato de requerer número linear de mensagens.

2.8.2 Otimização de Restrições Distribuídas para Redes em Geral

Para aplicar um algoritmo como o DTREE em um grafo cíclico, é necessário primeiramente transformar o grafo em uma pseudo-árvore.

2.8.2.1 Pseudo-árvores

O arranjo de um grafo G em forma de pseudo-árvore forma uma árvore com os mesmos vértices de G e a propriedade que vértices adjacentes no grafo original caiam na mesma porção da árvore. A idéia principal de seu uso em busca, é que, devido à independência relativa de seus nodos em diferentes porções da pseudo-árvore, torna-se possível realizar busca em paralelo nessas porções independentes. A figura 2.2 retirada de (PETCU; FALTINGS, 2005) mostra um exemplo de uma pseudo-árvore. Esta consiste de arcos de árvore, mostrados como linhas sólidas, e arcos de fundo, mostrados como linhas pontilhadas, que não são parte da árvore criada. Um caminho formado somente por arcos de árvore é chamado de caminho de árvore. Um caminho de árvore associado a um arco de fundo é o caminho de árvore conectando os dois nodos envolvidos no arco de fundo.

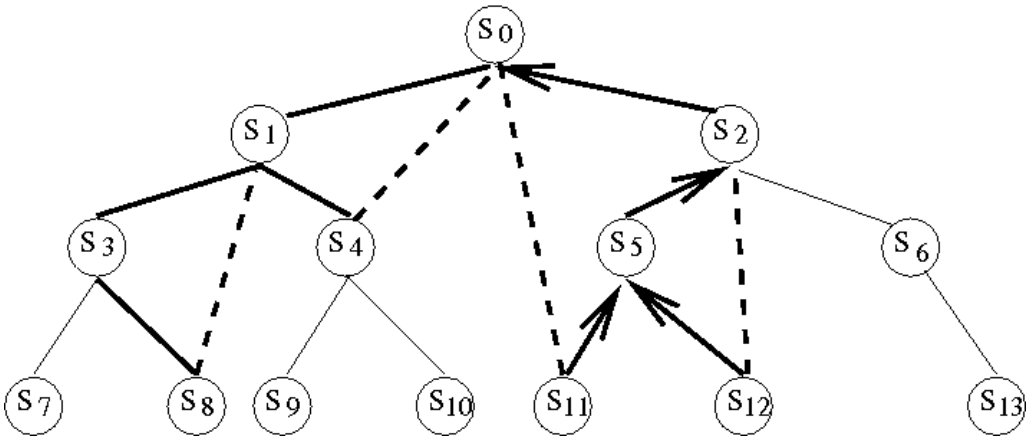


Figura 2.2: Exemplo da estruturação de uma pseudo-árvore (PETCU; FALTINGS, 2005)

Para cada arco de fundo, o nodo mais alto envolvido é chamado de *handler* do caminho de fundo. Também é definido:

- $P(s)$: pai do nodo s. O nodo acima na hierarquia da pseudo-árvore que está conectada com o nodo s diretamente através de um arco de árvore.
- $C(s)$: filho de um nodo s.
- $PP(s)$: os pseudo pais de um nodo s. Conjunto de nodos superiores na pseudo-árvore que estão conectados ao nodo s diretamente através de arcos de fundo.
- $PC(s)$: os pseudo filhos de um nodo s. O conjunto de nodos inferiores na hierarquia de uma pseudo-árvore que estão conectados aos nodos s diretamente através de arcos de fundo.

2.8.3 O Algoritmo DPOP

A execução possui três fases. Primeiramente os agentes estabelecem uma estrutura de pseudo-árvore a ser usada nas fases seguintes. As duas fases seguintes são as propagações de *UTIL* e *VALUE*, similares ao que foi mostrado para o caso do algoritmo DTREE. O pseudo código é exibido no algoritmo 2.5.

2.8.3.1 Propagação UTIL

Como no DTREE, a propagação *UTIL* começa nas folhas da pseudo-árvore, somente através dos arcos de árvore.

Em uma rede em forma de árvore, uma mensagem *UTIL* enviada por um nodo a seu pai é dependente apenas da sub-árvore baseada no respectivo nodo, e nas restrições entre o nodos e seus pais. Por exemplo, ainda na figura 2.2, considerando a mensagem $s_6 \rightarrow s_2$, percebe-se que essa mensagem é dependente apenas na variável alvo s_2, uma vez que não há ligação entre s_6 ou s_{13} e qualquer nodo acima de s_2.

Em uma rede com ciclos (cada arco de fundo produz um ciclo), uma mensagem enviada por um nodo a seu pai pode depender também de variáveis acima do seu pai. Isso acontece quando existe um arco de fundo conectando o nodo que envia com tal variável. Por exemplo, considerando a mensagem s_8 para s_3 na figura 2.2 podemos ver que as utilidades que a sub-árvore baseada em s_8 pode alcançar não são dependentes apenas de seu pai s_3. Como s_8 está conectado com s_1 através do arco de fundo $s_8 \rightarrow s_1$, s_8 deve

contar essa dependência quando enviar suas mensagens a s_3. Nesse momento se faz uso da programação dinâmica: s_8 vai computar as utilidade ótimas que sua sub-árvore pode alcançar para cada valor de combinação da tupla $[s_3;s_1]$. Então irá montar a mensagem como um hipercubo com duas dimensões (uma para a variável alvo s_1 e outra para o *handler* de arco de fundo s_1) e enviar para s_3. Essa é a diferença chave entre DTREE e DPOP: mensagens circulando na rede do DTREE sempre têm uma dimensão (são lineares no tamanho de domínio da variável alvo), enquanto no DPOP as mensagens têm múltiplas dimensões (uma para a variável alvo e outra para a variável de contexto).

2.8.3.2 Combinando Mensagens - Aumento Dimensional

Considerando o exemplo: s_5 recebe e mensagens de seus filhos s_{11} e s_{12}; a mensagem de s_{11} tem s_0 como contexto, e a mensagem de s_{12} tem s_2 como contexto. Ambas têm uma dimensão para s_5 (variável alvo) e uma dimensão para suas variáveis de contexto, portanto duas dimensões cada. s_5 precisa enviar suas mensagem para seu pai s_2. s_5 considera todas as possibilidades de valores de s_2 e para cada uma delas todas as combinações de valores para as variáveis de contexto (s_0 e s_2) e s_5 são consideradas; os valores de s_5 são sempre escolhidos de forma a otimizar as utilidades para cada tupla $< s_0, s_2, s_2 >$. Como s_2 é tanto uma variável alvo como de contexto, a mensagem reduz a duas dimensões, não três. Esse processo pode ser pensado como o produto das mensagens $s_{11} \rightarrow s_5$ e $s_{12} \rightarrow s_5$ resultando em um hipercubo com dimensões s_0, s_2 e s_5, seguido de uma projeção no eixo s_5, que guarda as utilidades ótimas para as tuplas $< s_0, s_2 >$.

2.8.3.3 Agrupando Mensagens - Diminuição Dimensional

Sempre que uma mensagem multidimensional *UTIL* alcança uma variável alvo que ocupa uma dimensão na mensagem, a variável alvo *se otimiza* tirando do contexto, e a mensagem que segue perde uma dimensão. Por exemplo s_1, que está inicialmente presente no contexto da mensagem $s_8 \rightarrow s_3$: uma vez que a mensagem chega em s_1, como s_1 não tem mais influência nas partes acima da árvore, s_1 pode otimizar simplesmente escolhendo o melhor valor para si, para cada valor de seu pai s_0 (processo DTREE normal). Assim, um *handler* de arco de fundo (s_1 nesse caso) aparece como uma dimensão extra nas mensagens trafegando através do caminho de árvore associado com o arco de fundo ($s_8 \rightarrow s_3 \rightarrow s_1$).

2.8.3.4 Propagação VALUE

Similar ao DTREE. Agora, adicionalmente ao valor de seu pai, a mensagem *VALUE*$s_j P(s_j)$ que um nodo s_j recebe de seu pai também contém os valores de todas variáveis que estavam presentes no contexto da mensagem *UTIL* de s_j ao seu pai.

Algoritmo 2.5 Algoritmo DPOP

Cada agente executa:
Fase 1: criação da pseudo-árvore
elege líder entre todos $s_j \in S$
o líder eleito inicia criação da pseudo-árvore
após isso, s_i conhece $P(s_i), PP(s_i), C(s_i) ePC(s_i)$
(Fase 2: propagação da mensagem UTIL)
se s_i não é um nodo folha **então**
  $UTIL_{s_i}(P(s_i) \leftarrow computa\_utilidade(P(s_i, PP(s_i))$;
  envia_mensagem($P(s_i, UTIL_{s_i}(P(s_i)))$
fim se
ativa função gerencia_mensagens_UTIL()

2.9 Comparação Qualitativa

Essa seção apresenta resultados de comparações entre os algoritmos ADOPT, OptAPO e DPOP conforme a pesquisa realizada. A centralização do problema pode ter um impacto significativo na quantidade de computação necessária por um agente, porém a métrica mais tradicional para avaliação desses algoritmos, o número de ciclos de execução, falha na análise desse ponto.

Um ciclo é definido como uma unidade de progresso algorítmico, onde todos agentes, em paralelo, processam suas mensagens recebidas, realizam qualquer computação necessária, e enviam suas respostas. Uma mensagem enviada no ciclo i não é recebida antes do ciclo $i+1$. Embora essa métrica não seja perfeita, pois não considera que os tempos de execução podem variar, ela fornece um método conveniente para medir o desempenho de algoritmos assíncronos.

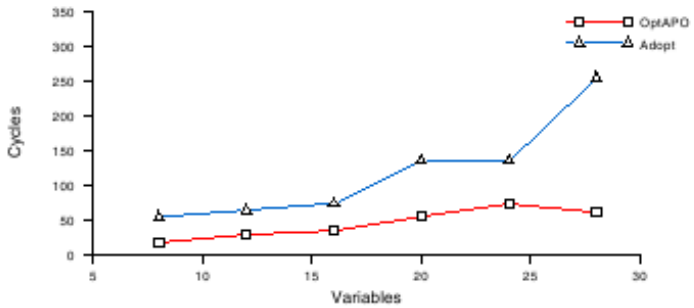
Uma execução verdadeiramente assíncrona em um algoritmo DCOP é difícil de ser medida confiavelmente por causa do número exponencial de execuções possíveis que diferem significativamente em seus tempos de execução. Portanto, para resultados que possam ser reproduzidos, é mais adequado executar o algoritmo de uma forma síncrona em apenas um computador, e assim, a métrica do número de ciclos fornece uma boa maneira de medir a execução.

Uma grande diferença entre os algoritmos é que no OptAPO e DPOP os agentes comunicam suas restrições, permitindo aos agentes que as recebem fazer uma avaliação. Essa comunicação tem implicações significativas na quantidade de computação de cada agente na resolução do problema. Isso ocorre pois à medida que o subproblema de um agente cresce com o acúmulo de restrições, mais computação local (busca) é necessária para encontrar a solução ótima para os subproblemas maiores. Portanto, quando restrições são comunicadas entre agentes a carga computacional para cada agente pode aumentar durante a resolução do problema. Por outro lado, em um algoritmo que não comunica as restrições, como o ADOPT, podemos esperar que a carga computacional para cada agente permaneça constante ao longo do tempo.

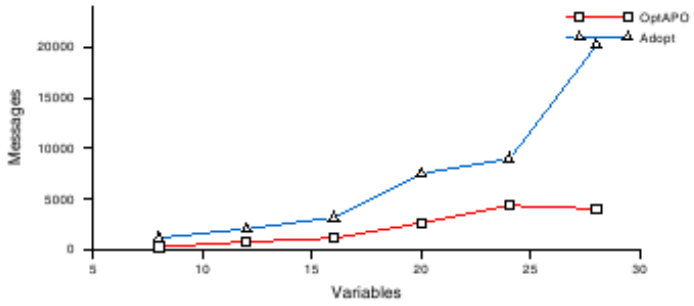
Vale ressaltar o trabalho feito em (MAILLER; LESSER, 2004), onde foi realizada uma comparação entre ADOPT e OptAPO através da aplicação dos dois algoritmos a um problema de coloração de grafos. Foram feitos testes com diferentes números de agentes (8, 12, 16, 20, 24 e 28) para três cores possíveis no grafo. Duas séries de testes foram executadas com instâncias do problema com muitas e poucas restrições. Como métrica foi usado o número de mensagens, o número de ciclos e o tempo total consumido para se

obter a solução (medido em segundos). As principais conclusões foram:

- O OptAPO superou o ADOPT com relação ao número de ciclos (uma rodada de troca de mensagens entre os agentes), número de mensagens e tempo de execução. Ou seja, foram necessários menos ciclos;
- OptAPO na realidade executa mais rápido que a busca *Branch and Bound* que ele usa internamente. Isso mostra que as características de execução do algoritmo não são um subproduto da busca centralizada.



(a) Cycles



(b) Messages

Figura 2.3: Comparação do número de ciclos e mensagens entre ADOPT e DPOP para um problema de coloração de grafos (DAVIN; MODI, 2005)

A figura 2.3 mostra gráficos referentes ao número de mensagens e ciclos utilizado por ambos os algoritmos no problema descrito acima.

Conforme (DAVIN; MODI, 2005), pode ser dito que o problema usado para comparação é, por natureza, um problema de satisfação de restrições e não de otimização de restrições. As cores não podem ser adjacentes neste caso, não havendo valores suficientes para que se busque a otimização.

Os resultados dessa análise mostraram que um algoritmo não centralizado, como o ADOPT, usa mais ciclos de comunicação, porém tem um custo computacional menor por ciclo. OptAPO, um algoritmo parcialmente centralizado, tem relativamente poucos ciclos de comunicação, mas maior custo de computação por ciclo.

Concluindo, enquanto OptAPO requer menos ciclos que o ADOPT, os ciclos do OptAPO são mais longos pois requerem mais computação. Para domínios com baixa latência de comunicação comparado ao tempo para efetuar a computação, ADOPT tem um melhor desempenho que OptAPO pois nesses domínios os agentes podem se comunicar eficientemente e, sendo assim, ADOPT consegue tirar vantagem disso por distribuir mais igualmente o trabalho.

DPOP é um trabalho posterior aos dois outros algoritmos. A característica mais importante de DPOP é que, diferentemente dos métodos antecessores, ele necessita de um número linear de mensagens para encontrar a solução ótima, gerando um *overhead* menor de comunicação (PETCU, 2006). Isso é verificado em (PETCU; FALTINGS, 2005), onde resultados experimentais com problemas de *meeting scheduling* e alocação de recursos mostram que DPOP pode tratar problemas desse tipo com maior ordem de magnitude, em termos de número de agentes, em relação aos seus antecessores.

3 CONTROLE SEMAFÓRICO E SIMULAÇÃO DE TRÁFEGO

Nesse capítulo são apresentados os conceitos de controle semafórico usados como base para desenvolvimento deste trabalho. Também são referenciadas algumas abordagens para tratamento desse tipo de controle. Além disso, é apresentada uma descrição do simulador de tráfego ITSUMO, bem como sua relação com este trabalho.

3.1 Controle Semafórico

Quando a carga (número de veículos) em uma rede de tráfego se aproxima de sua capacidade máxima (número total de veículos suportados), percebe-se o início da formação de congestionamentos. Por aplicação de controle de semáforos se entende o conjunto de ações de configuração do controle de forma a melhorar o fluxo de veículos em uma malha viária, evitando a formação de congestionamentos. Nesse caso, entende-se como melhora todo benefício no sentido de diminuir tempos de tráfego dos motoristas. Esse benefício tem relação direta com outros pontos observáveis na avaliação de uma rede, como o bom aproveitamento das capacidades das vias, e as conseqüências sócio-econômicas.

O uso de semáforos nos cruzamentos de uma malha viária é a medida mais comum em nível de controle de tráfego. Por cruzamento se tem o conjunto de vias que se cruzam em uma determinada área. Cada via tem uma direção e pode ter dois sentidos. O controle, pelo uso de semáforos tem os seguintes objetivos: ordenar o movimento do tráfego, aumentar a capacidade do cruzamento, reduzir a freqüência de acidentes, interromper o tráfego principal a fim de permitir o tráfego secundário, e eventualmente sincronizar semáforos para permitir um movimento contínuo em uma determinada via.

Os componentes principais de um cruzamento são a área de intersecção (entre as vias) e as pistas de aproximação deste. Estas pistas são agrupadas em movimentos de tráfego, fases ou estágios não conflitantes (em termos da geometria do cruzamento). Ou seja, todos os veículos presentes nas pistas de um mesmo movimento devem poder trafegar no cruzamento simultaneamente sem conflitos.

Os controladores de semáforos determinam o comportamento desejado para um determinado cruzamento através da configuração de um conjunto de parâmetros. Essa configuração necessita o entendimento dos conceitos de plano semafórico e fase. Uma fase determina um conjunto de movimentos permitidos. Um plano semafórico é um conjunto único de configurações de temporização para as fases, com isso determinando quais fases serão ativadas, em que ordem e por quanto tempo. Dessa forma, a permissão de passagem no cruzamento é garantida a todos os movimentos, de forma seqüencial. A figura 3.1 mostra um exemplo de fases de um plano semafórico, de acordo com os movimentos do cruzamento. Além disso, define-se um ciclo como a repetição de uma série de combinações de sinais em um cruzamento. A duração desse ciclo é chamada tempo

de ciclo. Uma fase é uma parte desse ciclo, na qual um conjunto de movimentos não conflitantes usa o cruzamento.

A figura 3.2 mostra dois exemplos de planos semafóricos para um mesmo cruzamento. A diferença está no início do tempo de cada fase.

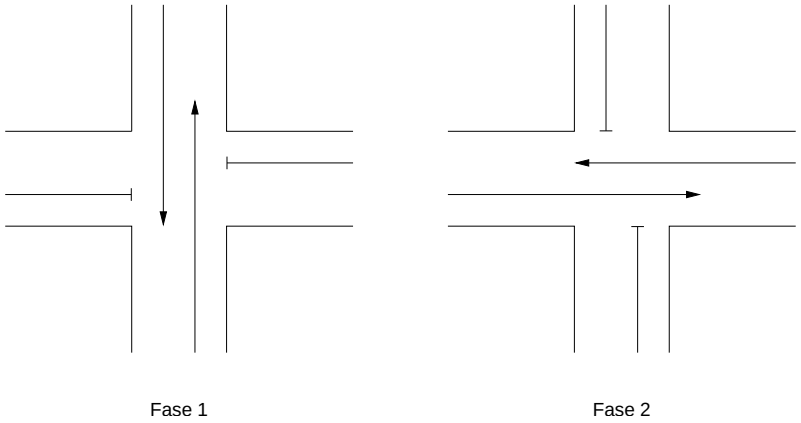


Figura 3.1: Exemplo de fases de um plano semafórico

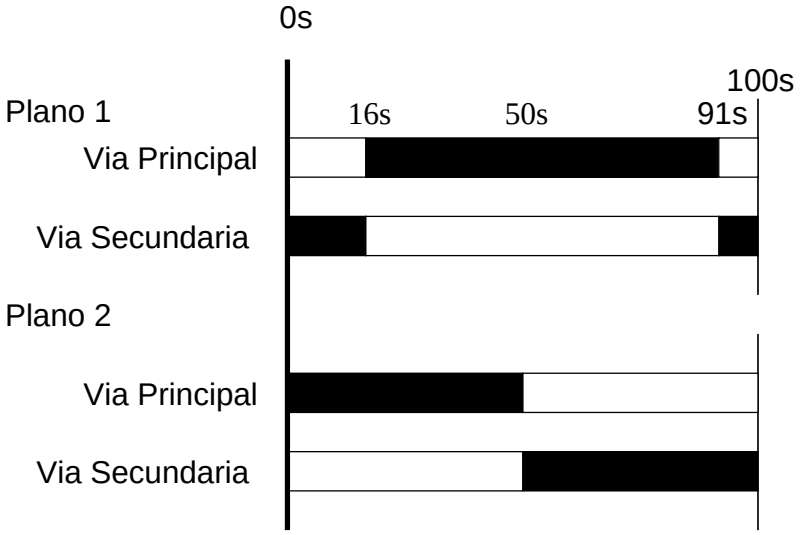


Figura 3.2: Planos semafóricos básicos (a linha em preto representa o tempo de verde)

Existem quatro possibilidades de influenciar as condições de tráfego através de operações de semáforos:

- Especificação das fases: para cruzamentos complexos não é trivial determinar o melhor número e a constituição das fases de um ciclo. Essas definições podem ter um grande impacto na capacidade e eficiência do tráfego do cruzamento;
- Divisão do tempo: referente à porção do tempo total de sinal verde para cada fase. Essa divisão deve ser otimizada em função da demanda dos fluxos envolvidos;
- Tempo de ciclo: tempos de ciclo maiores aumentam a capacidade de tráfego do cruzamento, porém pode aumentar desnecessariamente o tempo de sinal vermelho em cruzamentos que não estão saturados;

- Defasagem (*Offset*): é a diferença de tempo entre os inícios da mesma fase em semáforos adjacentes. Essa configuração pode dar origem a uma onda verde, ou seja, uma seqüência de fases tal que um veículo trafegue em sinal verde pelo trecho da via compreendido pelos semáforos que compõem a onda verde.

3.1.1 Sincronização e Coordenação de Semáforos

É clara a variação de tráfego aplicada em uma determinada malha viária em função de diferentes horários do dia, ou então em função de imprevistos que causam mudanças repentinas do comportamento esperado, como um acidente ou a execução de uma obra. Com essa motivação é necessário considerar a adaptação do controle aplicado por um semáforo ou conjunto de semáforos.

Por sincronização se entende a configuração dos parâmetros de **temporização** de um conjunto de semáforos, de forma que a atuação de semáforos adjacentes possibilite um fluxo de tráfego contínuo entre os semáforos envolvidos na sincronização. No que diz respeito a sistemas de controle de tráfego urbano, existem essencialmente dois tipos de temporização: fixa ou adaptativa. A sincronização com temporização fixa é a mais comumente utilizada e a mais simples, já que não requer o uso de equipamento especiais, como detectores de tráfego. Detectores são capazes de indicar a presença de uma massa metálica (um veículo) dentro de seu campo de atuação (campo elétrico). No caso da temporização adaptativa, esses detectores, bem como um sistema computacional mais complexo se fazem necessários, para que, através da leitura de informações das vias de tráfego se possa alterar a forma de atuação dos semáforos no sentido de criar uma adequação aos fluxos.

Pode-se dizer que a sincronização com temporização fixa tem um custo baixo, mas ao mesmo tempo seu desempenho (em termos de eficiência de controle) é de baixo a médio. A temporização dinâmica tem um custo mais alto, no entanto seu desempenho é de médio a alto, variando conforme o grau de centralização da solução.

Atualmente, as abordagens de sincronização de semáforos baseadas no uso de planos fixos pré-determinados, visam à otimização de rotas (geralmente rotas principais) e a rede como um todo nem sempre obtém uma melhora significativa na qualidade do tráfego. Aqui entra o conceito de onda verde, onde há uma configuração de semáforos tal que os veículos trafegando em uma determinada via obtenham uma seqüência de sinais verdes, permitindo realizar um trajeto sem interrupções. Há uma preocupação de se gerar uma onda verde em uma determinada via, e como conseqüência é gerada uma série de engarrafamentos nas vias transversais a essa. Uma abordagem mais adaptativa deve fazer uso de dados colhidos da via para configurar dinamicamente ondas verdes nas diversas opções de rotas da rede, para isso diminuindo o tempo ou abrangência da onda.

A otimização das variáveis de temporização configura o conjunto de planos semafóricos que quando aplicados conseguem reduzir o tempo de tráfego, e com isso o tempo de espera nos semáforos. Todos os planos coordenados devem ter o mesmo tempo de ciclo. Em engenharia de tráfego se costuma fazer um estudo que é uma função da geometria do cruzamento e com base nisso se define o tempo de ciclo ou defasagem usando um valor mínimo e máximo.

Há sincronização quando dois ou mais semáforos estão executando os mesmos planos semafóricos de modo que permita um veículo passar pelos semáforos sincronizados sem paradas. A figura 5.3 exemplifica o conceito de sincronização, mostrando as defasagens dos planos usados no semáforos de uma via, permitindo o fluxo contínuo por esta. O maior problema na sincronização de semáforos é encontrar os tempos de fases considerando variações nos tempos de ciclo e nas velocidades dos veículos. De acordo com

(ROBERTSON; BRETHERTON, 1991), uma rede com 30 a 40 semáforos demanda um ano de trabalho-homem.

Em um âmbito mais geral se estabelece o conceito de coordenação semafórica. Nesse conceito, um grupo de agentes tem sua configuração feita de forma a sincronizar seus planos semafóricos de acordo com a necessidade do fluxo. Essa tarefa de coordenação prevê a escolha ou configuração dos planos utilizados pelos semáforos de forma a efetivar essa sincronização. A configuração necessita uma avaliação geral da rede formada pelos semáforos envolvidos, pois diferentes direções e sentidos devem ser coordenados, e um semáforo pode coordenar com semáforos adjacentes porém em direções diferentes.

Neste trabalho o problema de controle semafórico é interpretado como um problema de coordenação semafórica, onde se deseja, pela escolha dos planos semafóricos a serem utilizados por cada semáforo, minimizar o impacto da carga de veículos na avaliação da rede. Essa escolha prevê a disponibilização de um conjunto de planos pré-definidos, onde cada plano tem o objetivo de sincronizar com uma das direções possíveis. Dinamicamente, pela avaliação da rede, as necessidade de sincronização são consideradas no relacionamento de semáforos adjacentes para atribuição dos planos semafóricos. Os detalhes desse modelo de coordenação são dados no capítulo 4.

Em (BAZZAN, 2005) uma abordagem baseada em SMA é descrita, onde cada semáforo é modelado como um agente. Cada agente possui planos pré-definidos para coordenação com agentes adjacentes. Planos diferentes podem ser escolhidos para haver coordenação em diferentes direções de acordo com a hora do dia. Os principais benefícios dessa abordagem são: os agentes podem criar subgrupos de sincronização para melhor atender às necessidades do fluxo em alguma direção; não há necessidade de um controle central; e não há comunicação nem negociação direta entre os agentes. No entanto, são necessárias matrizes de pagamento (*pay–off matrices*) e essas matrizes devem ser formalizadas explicitamente pelo projetista do sistema. Isto faz com que a abordagem consuma tempo quando diferentes opções de coordenação são possíveis e/ou a rede de trafego é muito complexa (grande número de vias e cruzamentos).

Outra abordagem (OLIVEIRA; BAZZAN; LESSER, 2005), considera a comunicação entre agentes e é baseada em mediação cooperativa. Esta trata o problema de forma a manter um compromisso entre a abordagem clássica centralizada e a abordagem com agentes totalmente autônomos e comunicação implícita. Para tal é usado um algoritmo distribuído de otimização de restrições em um cenário dinâmico, caracterizando assim a realidade do tráfego. Essa abordagem, usando processos de mediação para coordenação, é capaz de reduzir a freqüência dos problemas resultantes da falta de coordenação. Para tal é usado o algoritmo OptAPO (*Optimal Asynchronous Partial Overlay*) (MAILLER; LESSER, 2004). No entanto, uma desvantagem relacionada a este processo é o custo associado à comunicação entre os agentes, necessária para que o relacionamento se estabeleça e haja informações sobre a rede disponível a todos agentes de forma a realizar inferências e decisões de mediação. Este custo de troca de informações entre agentes, e o processamento destas pode não ser interessante em redes mais complexas.

Abaixo são tratados alguns dos sistemas (algoritmos ou programas) de sincronização de semáforos atualmente em uso no mercado.

3.1.2 TRANSYT

O *Traffic Network Study Tool* (TRANSYT (TRANSYT-7F, 1988)) é um dos programas de temporização de fases mais amplamente utilizados e mais antigos. É executado de maneira *offline* para determinar o tempo ótimo de semáforos coordenados em qual-

quer rede de vias na qual o fluxo médio de veículos seja conhecido, utilizando modelos macroscópicos e determinísticos de simulação. Embora o TRANSYT seja geralmente utilizado como ferramenta de otimização *offline*, pode ser utilizado de um modo *online* se fazendo a atualização dos dados da rede em pequenos intervalos de tempo e realimentando a rede com os resultados obtidos no simulador. O modelo a ser simulado deve ter entradas de veículos e probabilidades de mudança de direção constantes ao longo de todo o período de simulação.

Os critérios de otimização utilizados são: tamanho da fila, tamanho da banda da onda verde e quantidade de paradas. O programa otimiza as fases e as defasagens relativas dado um conjunto de tempos de ciclo, realizando diversas interações entre o módulo de simulação de tráfego e o módulo de otimização de semáforos. Os resultados são calculados em relação a uma rota específica na rede definida pelo usuário.

3.1.3 SCOOT

O *Split Cycle and Offset Optimization Technique* (SCOOT) (ROBERTSON; BRETHERTON, 1991) é um modelo centralizado de controle de tráfego desenvolvido pelo *Transportation Road Research Laboratory* no Reino Unido. O SCOOT usa detectores instalados nas vias para medir perfis do fluxo de tráfego em tempo real e, juntamente com tempos de percurso e graus de saturação (ocupação relativa à capacidade nominal da via) pré-determinados, prediz filas em intersecções. O sistema de previsão de filas se baseia no uso de **Padrões Cíclicos de Fluxo** (PCFs). Um PCF é o fluxo médio de veículos em uma direção em qualquer ponto da via durante o tempo de sinal verde. O sistema utiliza os mesmos critérios de otimização utilizados pelo TRANSYT. A partir das informações coletadas via os detectores, o sistema central envia instruções para os equipamentos localizados nos semáforos. Essas instruções são interpretadas e verificadas pelo semáforo. Caso as instruções sejam realizáveis, ele envia uma mensagem de aceitação, caso contrário, envia uma mensagem indicando uma falha na instrução. SCOOT tem três procedimentos de otimização para sincronizar os semáforos: o de tempos de fase, o de defasagem e o do tempo de ciclo. Cada procedimento otimizador estima o efeito de uma mudança incremental pequena dos tempos no desempenho total da rede de tráfego da região. Um índice de desempenho é calculado baseado em predições sobre paradas e velocidade dos veículos. Os ajustes do tempo de ciclo, tempos de fase e as defasagens são feitos em conjunto para operar num grau de saturação pré-definido (geralmente 90%). Os testes mostraram que SCOOT é mais eficaz quando a demanda se aproxima da capacidade, onde a demanda é imprevisível, e quando as distâncias entre intersecções são curtas (ROBERTSON; BRETHERTON, 1991).

3.1.4 SCATS

O *Sydney Coordinated Adaptive Traffic System* (SCATS) (LOWRIE, 1982), foi inicialmente desenvolvido na Austrália para aplicação em Sydney e em outras cidades australianas. Atualmente está instalado em mais de 50 cidades no mundo. É um sistema dinâmico de controle de semáforos com uma arquitetura descentralizada. A otimização do sistema se dá através de mudanças no tamanho do ciclo da fase e da defasagem, além disso, permite também que algumas fases não sejam executadas. O sistema possui uma biblioteca de planos que podem ser selecionados.

3.2 Simulação de Tráfego e o Simulador ITSUMO

O simulador do projeto ITSUMO (SILVA et al., 2006) utiliza um modelo de simulação do tipo microscópico e, portanto, ideal para simulação *bottom-up* ou baseada em agentes.

O modelo microscópico de movimentação física é baseado em um autômato celular. Esta escolha se deve a questões de eficiência computacional. Este tipo de modelo leva em consideração veículos individualmente, e é relativamente mais complexo que um modelo macroscópico. Neste modelo, cada veículo possui uma velocidade e se movimenta de acordo com relações bastante simples. Portanto, o modelo utilizado no ITSUMO é baseado no autômato celular proposto por Nagel-Schreckenberg, mas inclui algumas extensões ao modelo original, as quais são detalhadas em (SILVA et al., 2006). Através do simulador é possível reproduzir situações clássicas como acidentes e outras perturbações sobre a via, bem como o que atualmente é conhecido como a causa mais freqüente de congestionamentos que surgem sem um motivo aparente: o fato de que, seguindo-se uma desaceleração aleatória (alguém que observa a paisagem, telefona ao volante, etc.), ocorre uma reação em cadeia de desacelerações, cada uma mais forte que a que a causou, de modo que em pouco tempo se forma um congestionamento de grandes proporções.

No projeto do sistema ITSUMO estão previstos os seguintes módulos:

- Aquisição de dados
- Configuração dos componentes (malha viária, motoristas, e controladores)
- Simulação microscópica
- Controle de Tráfego
- Visualização de dados

O controle de tráfego (configuração semafórica) pode ser modelado de acordo com diversas abordagens, desde as abordagens clássicas (planos pré-definidos), como novas abordagens, baseadas em negociação, aprendizagem por reforço, etc.

3.3 Funcionamento da Simulação no ITSUMO

Desde a concepção dos parâmetros de uma simulação até a simulação de fato, seguem alguns passos necessários pra criação dos elementos que a compõem.

Inicialmente deve ser criada a topologia, especificando todos os objetos da malha viária, e a seguir as configurações que serão aplicadas pelo simulador como parâmetros de desenvolvimento. Tanto os dados de configuração da simulação, quanto os dados referentes à configuração da rede de tráfego (próxima seção) são armazenados em arquivos XML, passados como parâmetro para as interfaces e programas do simulador.

3.3.1 Componentes

Na representação da rede de tráfego, cada via é descrita como a composição de nodos e arcos. Cada arco é composto por um conjunto de seções conectadas (*sections*). Uma seção representa uma porção da via que é delimitada por dois nodos (*nodes*). Cada seção é formada por um conjunto de vias com o mesmo sentido (*lanesets*). A estrutura de mais baixo nível na representação é a faixa (*lane*). Estas são divididas em células de tamanho fixo, que podem estar vazias ou ocupadas por um veículo. O tamanho da célula

é escolhido de forma a representar o espaço médio ocupado por um veículo em trânsito, mas esse valor pode ser ajustado. É importante fazer esse ajuste, de forma a usar valores discretos que representem de forma fiel as medidas reais.

Um nodo é um componente importante da rodovia, pois representa a conexão entre duas seções, e é a abstração para um cruzamento na vida real. Pode conter três objetos diferentes: injetores, sumidouros e semáforos. Injetores (*sources*) e sumidouros (*sinks*) são usados para ajustar os parâmetros de simulação para valores desejados de fluxo de tráfego. Ou seja, eles inserem e retiram veículos da rede a fim de reproduzir o fluxo de tráfego que se deseja observar. Desta forma, o engenheiro de tráfego ou usuário responsável pela simulação pode configurar um nodo de forma em que os veículos são inseridos na rede de tráfego de acordo com regras fixas. Atualmente, nodes equipados com um objeto injetor podem gerar veículos de acordo com:

- fluxo constante
- probabilidade constante
- probabilidade variável
- fluxo variável

Além disso, o simulador suporta dois outros objetos: coletores de informação e semáforos. O propósito do primeiro é coletar todo tipo de informação sobre o cenário simulado, como: taxa de ocupação da via, média de velocidade dos veículos, fluxo de veículos em uma via específica, etc. semáforos controlam um conjunto de planos semafóricos que refletem as restrições de movimentação do tráfego real.

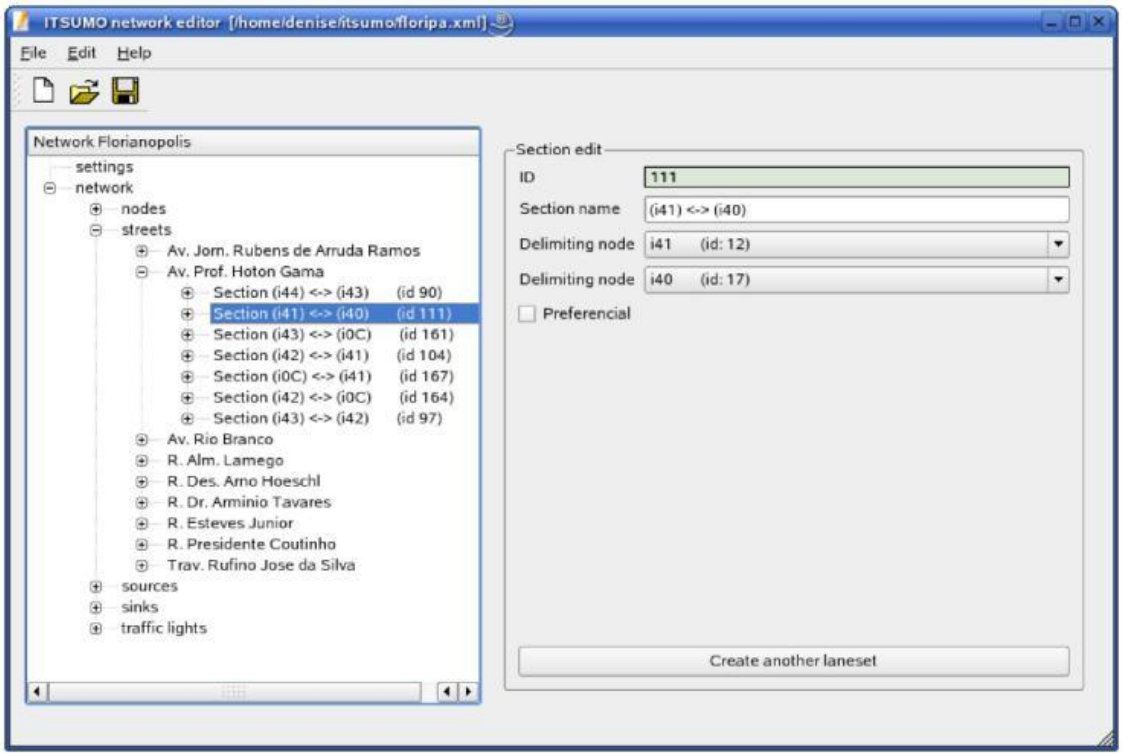


Figura 3.3: Interface de criação de malhas viárias

O simulador dispõe de uma ferramenta para criação e configuração da malha viária, onde os objetos citados acima são instanciados para formação da malha. A figura 3.3 mostra essa interface de configuração. A estrutura de dados do banco de dados, atualmente representado por um arquivo XML, reflete os objetos componentes do modelo, citadas acima:

- *General Settings*: composto de parâmetros como nome da topologia, tamanho da célula, orientação, freqüência da medição dos sensores, hora de início da simulação e probabilidade de desaceleração padrão;
- *Network*;
- *Node*: coordenadas cartesianas do nodo e a rede associada;
- *Street*: nome da rodovia e a rede associada;
- *Section*: nome da seção, tipo (preferencial ou não), nodos delimitadores, e via associada;
- *Laneset*: tamanho e seção associada;
- *Lane*: velocidade máxima, largura, e *laneset* associada;
- *Laneset Probability*: conjunto de probabilidades de mover um veículo para uma *laneset* específica passando por um nodo e via associado onde essas probabilidades são válidas;
- *Turning Probability*: conjunto de movimentos válidos para o veículo de acordo com sua via atual;
- *Traffic Light*: conjunto de planos semafóricos, associados a um nodo;
- *Signal Plan*: conjunto de movimentos permitidos em uma ordem específica e ciclo de tempo;
- *Sink*: probabilidade de remoção de um veículo e o nodo associado;
- *Source*: comportamento de inserção, *laneset* onde os veículos devem ser inseridos e o nodo que contém o injetor.

Os objetos componentes podem ser visualizados na figura 3.4.

3.3.2 Configuração

De uma maneira sucinta, as etapas percorridas pelo simulador para iniciar uma simulação podem ser enumeradas como a seguir:

1. Leitura dos arquivos XML de configuração;
2. Criação do objeto *Settings*, com informações da topologia a ser simulada;
3. Criação da malha viária;
4. Criação dos nodos: objetos que representam os cruzamentos;

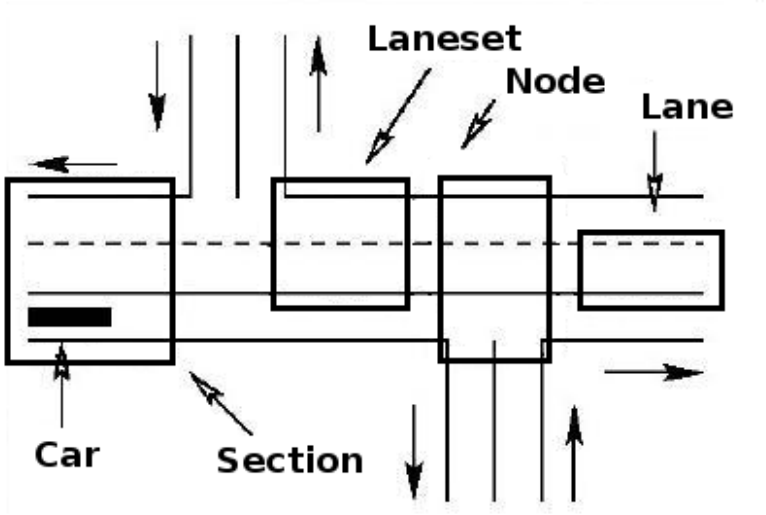


Figura 3.4: Objetos componentes da estrutura do simulador ITSUMO

5. Criação das vias: abstração para o conceito de ruas;
6. Criação dos semáforos (opcional);
7. Criação dos injetores de veículos: objetos que inserem motoristas na malha;
8. Criação dos sumidouros de veículos: objetos que retiram motoristas da malha;
9. Ajuste dos pesos de cada direção, conforme cruzamentos;
10. Ajuste das possibilidades de direções a serem adotadas em cada cruzamento;
11. Instanciação dos modelos de motoristas (opcional);
12. Liberação para início da simulação.

Após a liberação do início da simulação, os eventos ocorrem na seguinte ordem:

- Efetuar as ações pendentes nos cruzamentos;
- Informar aos veículos que solicitem novas decisões de movimentação aos seus motoristas;
- Solicitar aos veículos que apliquem as decisões solicitadas pelos motoristas;
- Informar aos agentes semafóricos estado da rede (caso definidos);
- Solicitar alterações aos agentes semafóricos (caso definidos).

Na construção de um motorista são determinadas as ações que configurarão seu comportamento, para que durante a simulação seja possível aplicar e avaliar suas decisões perante o fluxo de veículos.

3.3.3 Iterações e Saídas

A cada passo de tempo, é realizada a atualização do sistema, baseada nas decisões de movimentação dos motoristas, tratamento dos relacionamentos, colisões e de acordo com o tipo de coletores de informação usados, os resultados e valores medidos são armazenados em um arquivo.

O simulador também dispõe de um módulo de visualização dos resultados da simulação. A visualização pode ser em nível macroscópico. Em nível macroscópico, é considerado apenas os dados que refletem o comportamento geral da malha viária, abstraindo detalhes e constituindo uma ferramenta muito importante para entender o que está acontecendo em um cenário específico.

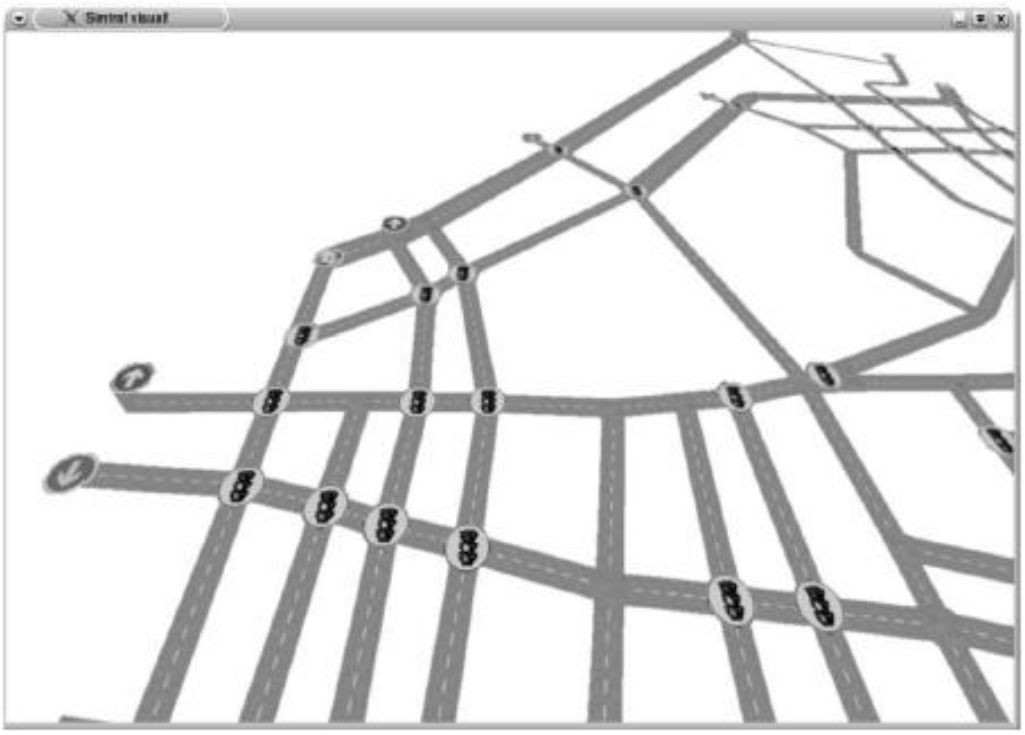


Figura 3.5: Visualização da malha viária

Um exemplo da visualização pode ser vista na figura 3.5.

3.4 Agentes Semafóricos no ITSUMO

Com o intuito de facilitar a tarefa de desenvolvimento de controladores de tráfego, foi criado neste trabalho um conjunto de classes básicas para modelagem de agentes semafóricos. Pela implementação, a definição desses agentes é tida em uma estrutura de dados e de código fora do simulador ITSUMO. Isso viabiliza uma independência desse módulo em relação ao código do simulador e não torna obrigatório o conhecimento da estrutura interna do simulador ou então a manipulação direta dessa quando se deseja criar novos agentes.

Para tal foi criada uma interface entre os agentes e o simulador. Essa interface faz uso de *sockets* e do protocolo TCP para estabelecer o canal de comunicação que permite que o estado da simulação seja passado aos agentes e também que os agentes possam comunicar ao *kernel* do simulador suas decisões a respeito de ações de controle nos semáforos.

Aos agentes cabe interpretar as informações recebidas, fazer inferências sobre essas, de acordo com a lógica programada, e após comunicar sua decisão. Toda essa comunicação ocorre usando um protocolo definido para esse módulo.

Outro aspecto importante é a independência de implementação. Uma vez respeitado o protocolo definido, o controle dos agentes e sua lógica de controle das ações dos semáforos podem ser codificadas usando qualquer linguagem de programação ou ferramenta externa.

Na simulação, cada agente é executado em um processo (*thread*) independente, e é possível definir que um agente controla um semáforo ou um conjunto de semáforos da malha viária. A identificação dos semáforos controlados é obtida por meio de um número *ID*, definido pela interface de edição de topologias (também do simulador ITSUMO).

Abaixo, um resumo do protocolo operado entre os agentes e o simulador:

1. O simulador inicia a execução e espera por um tempo configurável ou uma interrupção do teclado por pedidos de conexão vindos dos agentes;

2. Os agentes se conectam ao simulador através do envio de uma mensagem de conexão. Essa mensagem contém um identificador de conexão e a identificação dos semáforos que deseja controlar;

3. Após o ITSUMO receber essa mensagem de conexão é retornada uma mensagem resposta confirmando a conexão. Essa resposta é composta pelos dados dos semáforos para os quais o agente solicitou controle;

4. Com os passos 2 e 3 a conexão é estabelecida;

5. A cada *sensor interval*, o simulador envia uma mensagem a cada um dos agentes informando o estado atual da simulação (referente aos cruzamentos identificados pelos semáforos controlados por cada agente, ou seja, não de toda a rede);

6. Os agentes confirmam o recebimento da mensagem de estado com uma mensagem contendo **OK**;

7. Com um período pré-definido o simulador solicita uma ação dos agentes;

8. Quando o agente recebe a mensagem descrita no item anterior é feito um processamento sobre os dados armazenados e deve ser enviada uma mensagem com as mudanças (ou ações) desejadas.

As estruturas de modelagem de agentes semafóricos citadas acima foram utilizadas, com o propósito de controle semafórico em (OLIVEIRA et al., 2007), onde foi modelado um controle semafórico individual baseado em técnicas de aprendizado por reforço.

Nos capítulos 4 e 5 são descritas a configuração dos modelos de tráfego e de semáforos utilizados neste trabalho. Para isso são referenciados os conceitos de sincronização e coordenação anteriormente introduzidos, bem como a relação e aplicação desses modelos com o simulador e suas facilidades descritas nas últimas seções.

4 ESTUDO DE CONTROLE DE TRÁFEGO COMO UM PROBLEMA DCOP

Este capítulo apresenta o estudo feito para a formulação de um problema de controle semafórico como um DCOP (conforme capítulo 2). Esse estudo prevê um conjunto de métricas para avaliação da aplicação dos algoritmos, bem como as características necessárias a um problema de controle de tráfego, o qual se deseja modelar na forma de DCOP.

Na seção 2.1 foi apresentada uma discussão sobre as características gerais e classificação de problemas genéricos de alocação distribuída de recursos. A partir dessa classe de problemas se derivam os problemas de otimização de restrições. Nas seções posteriores são dados os detalhes da modelagem e do estudo dos problemas de controle semafórico como DCOPs. Sendo assim, consiste da execução das seguintes etapas:

1. Definição das variáveis da malha viária pertinentes ao modelo DCOP (seção 4.1);

2. Construção dos planos semafóricos (seção 4.2);

3. Construção do modelo DCOP (seção 4.3);

4. Construção das restrições do problema (seção 4.4);

5. Uso dos algoritmos DCOP, apresentados nas seções 2.6, 2.7 e 2.8 para aplicação nas simulações a serem realizadas (seção 4.5);

6. Avaliação do algoritmo DCOP, conforme métricas definidas (seção 4.7);

7. Avaliação da evolução do tráfego na malha viária, conforme métricas definidas (seção 4.8).

As simulações aqui realizadas foram realizadas através da ferramenta ITSUMO. Nessas simulações, interativamente, o estado do tráfego da malha viária (ocupação das vias) é passado aos algoritmos estudados, que retornam as soluções encontradas ao simulador para aplicação no controle dos semáforos. As informações da execução dos algoritmos, bem como o impacto das ações no tráfego são considerados para análise, onde é apresentada uma comparação frente às métricas definidas nas seções 4.7 e 4.8. A aplicação desse estudo é apresentada no capítulo 5 para o estudo de caso proposto.

4.1 Definição das Variáveis da Malha Viária

Para que seja possível mapear o problema de configuração de semáforos (problema relatado na seção 3.1.1) em um problema DCOP necessariamente devem ser definidas as variáveis enumeradas abaixo:

- Número de semáforos (N_s): o número de semáforos vai determinar o número de agentes envolvidos no problema DCOP. Adota-se o mapeamento de um agente por semáforo, ou seja, neste problema cada agente é um controlador semafórico, responsável pela atribuição do plano semafórico a ser usado no semáforo de um cruzamento específico (seção 3.4);

- Conjunto de vetores que determinam o sentido do tráfego (S_l): o sentido do tráfego determina os relacionamentos usados para construção das restrições. Ou seja, um determinado agente só possui uma restrição com agentes *que enviam veículos* na malha. O vetor é dado pelo par origem/destino do trecho da via, ou seja, com isso é possível determinar, para um par de agentes (semáforos) vizinhos (delimitam uma via) qual deles recebe o fluxo de veículos do outro. Esse conjunto será utilizado na construção das restrições do problema, detalhada na seção 4.4;

- Comprimento das vias da rede (T_l): número de células em cada via l da rede. Esse valor é usado na definição do conjunto de planos semafóricos (D_{sp}), pois é importante para a definição da coordenação;

- Domínio de planos semafóricos (D_{sp}): conjunto de planos disponíveis para cada agente. Cada plano tem uma função específica, que diz respeito à direção que está sendo sincronizada, e de acordo com isso e com o estado da malha seu uso pode ser determinado pelo agente. A definição dos planos a serem utilizados se faz na seção 4.2;

- Velocidade máxima dos veículos (V_{max}): esse valor é definido em termos de células por iteração e também é importante na definição do conjunto de planos semafóricos (D_{sp}) coordenados, pois influencia no tempo necessário para percorrer uma via da malha, ou seja, define a defasagem.

4.2 Construção dos Planos Semafóricos

Essa seção tem por objetivo detalhar a construção dos planos semafóricos a serem disponibilizados para cada agente. Cabe então, a cada agente, atuar dinamicamente ao longo da simulação na escolha do plano a ser utilizado de acordo com a situação da malha. Essa escolha se faz pela execução dos algoritmos DCOP. Conforme conceitos de controle semafórico apresentados na seção 3.1, aqui é apresentada a forma de definição dos planos a ser utilizada.

A definição desse conjunto se faz necessária para otimizar o controle (seção 3.1), visto que se existisse apenas um plano (por exemplo), os agentes não teriam liberdade de ação para adaptarem o controle ao tráfego.

Além disso, cada plano é composto por um conjunto de fases. Essas fases determinam o tempo de sinal verde em cada um dos possíveis movimentos. Cada movimento é determinado por uma via de origem e por uma via de destino. Uma fase não pode ser composta por movimentos conflitantes, ou seja, que fluxos de veículos se sobreponham.

Utiliza-se um valor de tempo de ciclo t_{ciclo}, definido para cada cenário especificamente, e esse tempo deve ser dividido entre as fases de forma que todas os sentidos possíveis sejam atendidos. Nesse trabalho é utilizado um tempo de fase t_{fase} igual para cada movimento possível em um cruzamento.

Sendo assim a determinação dos tempos de fase leva em conta o tempo total de ciclo, as distâncias entre os semáforos a serem coordenados e a velocidade máxima, que em conjunto determinarão a defasagem (*offset*) usada para ajustar os tempos de fase, de forma que possa ser gerada uma onda verde (sincronização) entre os semáforos.

4.3 Variáveis DCOP

O número de agentes é determinado pelo número N_s de semáforos presentes na rede. É atribuído um agente para cada semáforo. Além disso, cada agente é responsável pelo controle de seu semáforo, ou seja, pela atribuição do valor de sua variável, aqui entendido como um valor do conjunto D_{sp}.

Esse problema de otimização de restrições distribuídas, formalmente é escrito pela tupla $< A, D, F >$ tal que:

- $A = \{a_1, ..., a_{n_s}\}$ é o conjunto de agentes/variáveis;
- $D = D_{sp}$ é o conjunto domínio das variáveis, dado por uma seqüência finita de valores, representando as possibilidades de atribuição de planos semafóricos possíveis de serem escolhidos;
- $F = \{f_1, ..., f_p\}$ é um conjunto de restrições entre as variáveis, onde a restrição f_i denota o custo associado à restrição entre as variáveis. Cada restrição prevê um valor de custo para um determinado par de valores do domínio de cada variável envolvida.

4.4 Construção das Restrições

As restrições ocorrem entre dois agentes, e são quantificadas (custo) pelo fluxo de veículos circulando entre estes no momento de sua construção, no tempo t. Cada semáforo vai gerar restrições com os semáforos (nodos) que o alimentam (enviam veículos). Sendo assim, as restrições são dadas pela ação de controle do semáforo no instante t em função do fluxo de veículos e ponderadas também por uma relação que define o grau de concordância das ações dos semáforos envolvidos na construção da restrição.

Definimos o custo da restrição entre dois semáforos quaisquer l e n, para o conjunto de valores possíveis no domínio de suas variáveis de controle como sendo:

$$f_{l,n} = (1 - \beta_{l,n}) \times \omega$$

Onde:

- $\beta_{l,n} = \rho_{l,n} / \sum_z \rho_{z,n}$, onde z = conjunto de todos os nodos que inserem veículos no nodo n;
- $\rho_{l,n}$ = densidade de veículos no trecho da via entre os nodos l e n;

- $\omega = \tau \times \alpha \times \gamma$, onde :

  α é um artefato (multiplicador) para trabalhar com números inteiros pois os algoritmos DCOP não usam valores decimais para o custo das restrições. Foi usado $\alpha = 10$;

  γ determina a concordância dos agentes em termos da direção de sincronização adotada. Usa-se o valor 1.5 para agentes sincronizados e o valor 2 para agentes não sincronizados. O valor é maior no segundo caso, pois se deseja priorizar a sincronização;

  τ é um fator de concordância do plano semafórico com o fluxo de veículos, e quantificado na tabela 4.1.

O valor de τ é dado para a concordância das ações entre os dois semáforos: calcula-se o β individualmente para os dois nodos envolvidos e com base nisso é possível determinar qual sentido possui uma maior demanda de veículos (maior β). Se β concorda com o sentido do plano que o agente está executando então se diz que o agente executa o melhor plano em função do fluxo (concordante com o fluxo). Dito isso, o peso mostrado da tabela 4.1 considera a combinação de possibilidades de planos nos dois agente envolvidos. A tabela considera fluxo de *l* para *n*.

As situações são:

1. agentes concordantes: ambos estão executando plano de sincronização na mesma direção. As restrições que se encaixam nessa classificação possuem um peso menor, pois é a situação ideal;

2. agentes não concordantes: não estão sincronizando na mesma direção. Nesse caso a restrição possui um peso maior que o caso anterior, pois se deseja que os agentes operem coordenadamente.

Os pesos usados no cálculo das restrições são dados conforme o caso que caracteriza o controle dos agentes (qual sentido estão priorizando) e as combinações de valores possíveis para as variáveis, classificadas conforme os casos acima. Seu valor varia conforme o grau de concordância do plano sendo usado, com a situação atual do tráfego, considerando os dois agentes envolvidos.

Tabela 4.1: Pesos das associações de planos semafóricos

Plano do agente *l*	Plano do agente *n*	τ
Concordante	Concordante	0
Não concordante	Concordante	1
Concordante	Não concordante	1.5
Não concordante	Não concordante	2

Caso os dois agentes estejam executando planos que priorizem a direção com maior demanda, então o custo é zero. Em qualquer situação diferente dessa é aplicado um peso referente à combinação das ações dos agentes.

A aplicação dos pesos e o valor de β considerado em cada restrição define o custo da restrição. Essa combinação de valores determina para um semáforo um conjunto de

restrições que levam em consideração as demandas de tráfego (β) em cada direção e um peso referente à situação de sincronização. A questão fica em definir em qual direção sincronizar (configuração de planos semafóricos) com base no fluxo de veículos.

4.5 Aplicação dos Algoritmos DCOP

Tipicamente o simulador ITSUMO não varia os planos, ou então conforme configuração pré-definida, sem que haja algum tipo de inteligência para identificação do melhor plano para a situação da rede. Neste trabalho são implementados algoritmos cujo objetivo é aplicar controle nos cruzamentos com semáforos, com base nas informações sobre o estado da malha fornecidas pelo simulador.

Para tal foi usada a interface de agentes semafóricos do simulador (construída neste trabalho) com o propósito de permitir a comunicação (e intervenção) de agentes externos nos processos de simulação do ITSUMO. Nesse caso o interesse é no controle semafórico, ou seja, agentes capazes de realizar um processamento independente do simulador, de forma que suas ações sejam repassadas ao simulador para serem de fato aplicadas. Para isso, conforme exposto na seção 3.4 esses agentes realizam dois tipos de troca de informação com o simulador:

1. comunicação do estado da via: o simulador informa dados (densidade, velocidade média, número de veículos parados e plano semafórico em execução, conforme 3.4) a respeito da via;

2. comunicação de ações: os agentes enviam requisições de atuação nos semáforos ao simulador.

O uso dessa estrutura é de fundamental importância na aplicação dos algoritmos DCOP. Através dela, esses agentes são capazes de fazer a interface entre o simulador e os algoritmos. Nos estudos propostos por esse trabalho, foi definido um agente para cada semáforo presente na malha viária, e esses agentes são responsáveis por receber do simulador informações do tráfego (como densidade das vias e velocidade média), e através dessas informações montar o conjunto de variáveis e restrições utilizadas pelos algoritmos DCOP. Além disso, esses agentes colhem estatísticas do avanço da simulação e trocam informações com os algoritmos DCOP (ou seja, executam os algoritmos usando o estado da rede como parâmetro de entrada) obtendo os resultados para aplicação na simulação.

Os três algoritmos de DCOP usados são listados abaixo, bem como os créditos por suas implementações.

- DPOP: usado o algoritmo disponibilizado na ferramenta *Frodo* pelo autor do DPOP em `http://liawww.epfl.ch/frodo/`;

- ADOPT: usado o algoritmo disponibilizado em `http://teamcore.usc.edu/dcop/`;

- OptAPO: foi implementado nesse trabalho.

4.6 Métricas de Avaliação dos Algoritmos de DCOP

Quando se considera a aplicação dos algoritmos DCOP, devem-se levar em conta suas características de execução. Essas questões precisam ser mapeadas à realidade de infra-estrutura disponível para sua aplicação real. Ou seja, características como execução distribuída e capacidade de comunicação devem ser ponderadas na escolha da melhor solução. Conforme trabalho apresentado em (MODI et al., 2005; MODI, 2003), podemos descrever dois grandes entraves relativos ao sucesso da execução de algoritmos DCOP, os quais são discutidos a seguir.

4.6.1 Limitação de Tempo

Em domínios onde o tempo de execução é peça chave, como é o caso do controle semafórico, as limitações de tempo determinam limiares que devem ser respeitados para a disponibilização das decisões tomadas. Essas limitações podem inviabilizar a busca pela solução ótima. Isso se torna fundamental pois, sabido que a classe DCOP é NP-completa, o tempo disponível pode não ser suficiente no pior caso.

Dado o problema em questão, algumas soluções podem fazer uso de métodos locais de busca de soluções que tipicamente são eficientes na redução de tempo, porém ao mesmo tempo abandonam as garantias teóricas a respeito da qualidade da solução. Conforme (MODI, 2003), métodos locais incompletos não podem garantir soluções ótimas, seja qual for o tempo disponibilizado. Nesse caso, um método local não é capaz de fazer uso do tempo disponível para garantir uma solução de melhor qualidade, pelo fato de não estarem relacionando a qualidade global da solução. Com isso não é possível aos agentes realizar qualquer tipo de processamento sobre como ou quando devem parar de buscar soluções melhores.

4.6.2 Qualidade da Comunicação

Em geral, ao tratar da aplicação de algoritmos DCOP, assume-se que o meio de comunicação é perfeito. Ou seja, há recursos suficientes para garantir o volume de comunicação (velocidade, tamanho de banda para tráfego de dados), e a confiabilidade da transmissão (tolerância a falhas).

Porém, no mundo real, nem sempre é possível contar com essa certeza, e é comum encontrar problemas de limitação de canal de comunicação, perda de dados na transferência ou dados corrompidos. Nesse sentido é importante que se tenha consciência disso ao avaliar a solução proposta por um algoritmo DCOP em específico. Isso levando em consideração o problema a ser tratado e os recursos disponíveis.

Em geral, espera-se da infra-estrutura de comunicação características de:

- Confiabilidade: toda mensagem enviada será recebida eventualmente;
- Atomicidade: a ordem das mensagens é preservada na transmissão de dados entre agentes.

4.7 Descrição das Métricas de Avaliação para DCOP

Com base nas descrições apontadas na seção anterior, percebe-se a importância da correta avaliação dos fatores que determinam a execução de um determinado algoritmo aplicado a um determinado problema. Isso quando se mantém a idéia de explorar a qualidade global de solução proposta pelos algoritmos.

As métricas de avaliação utilizadas neste trabalho são:

1. Número de ciclos de execução: número de iterações necessárias entre os agentes para disponibilização da solução do problema. Esse valor está relacionado com as necessidades de tempo de execução e capacidade de comunicação existente. Com um maior número de mensagens trocadas, maior será a necessidade de tempo para execução e maior será a dependência na estabilidade da comunicação, tanto em termos de capacidade de transmissão quanto em termos de tolerância a falhas. Esta é uma métrica clássica usada para avaliação de DCOP, já desenvolvida em (MAILLER; LESSER, 2004), (DAVIN; MODI, 2005) e (PETCU; FALTINGS, 2005);

2. Tempo total de execução: tempo necessário para disponibilização da solução do problema; Pela avaliação dessa variável é possível determinar a viabilidade da aplicação do algoritmo frente à dinâmica do problema.

3. Número de mensagens trocadas: número médio de mensagens trocadas entre os agentes ao longo da execução do algoritmo. Afeta as questões de análise de tempo de execução e capacidade de comunicação. Esse valor é diretamente proporcional ao número de ciclos de execução e ao tempo total de execução;

4. Tamanho das mensagens trocadas: tamanho médio das mensagens trocadas entre os agentes durante as iterações para solução do problema. Esse valor afeta a análise da capacidade de comunicação necessária para execução do algoritmo. Ou seja, a capacidade de transmissão de dados necessária. E afeta também as considerações de tolerância a falhas;

5. Qualidade da solução: ver próxima seção.

Cada problema tem suas características particulares, e embora se deseje em geral o máximo de desempenho e qualidade na execução dos algoritmos, há limiares a serem considerados quando se leva em consideração as necessidades reais de aplicação, a complexidade do problema, e relevância da solução ótima frente ao seu custo e a infra-estrutura disponível para implantação de uma solução.

4.8 Métricas de Avaliação da Rede: Estudo de Caso

Não há uma forma de se usar apenas uma métrica que avalie o desempenho global de uma malha viária. Muitas variáveis são consideradas e o relacionamento entre elas se torna complexo. Sendo assim, com o intuito de avaliar o desempenho da rede, um conjunto de métricas é utilizado como forma de comparação entre situações de tráfego. São descritas:

1. Número total de veículos parados: representa o número de veículos parados ao longo de toda malha viária. Esse valor dá uma idéia dos congestionamentos existentes na malha em nível geral;

2. Densidade média: representa a densidade média de todas as vias componentes da malha viária;

3. Velocidade média: representa a velocidade média de todos os veículos presentes na malha viária;

4. Grau de concordância dos semáforos: representado pelo número grupos de coordenação e o tamanho médio desses grupos. Diz respeito ao número de agentes vizinhos que estão executando o mesmo plano.

Neste capítulo foram descritos os conceitos usados na modelagem do problema de controle semafórico aqui tratado. Essa modelagem será referenciada no próximo capítulo onde o problema será instanciado no estudo de caso a ser proposto. Esse estudo de caso tem seus resultados relatados e analisados no capítulo 6.

5 DESCRIÇÃO DO ESTUDO DE CASO

Nesse capítulo serão tratadas as questões referentes ao estudo de caso. Esse estudo considera as configurações de parâmetros e a modelagem adotada considerando o formalismo de DCOP.

5.1 Descrição da Malha Viária

Para este trabalho foi adotada uma configuração de grade (visível na figura 5.1) para a rede de tráfego. Esse formato de rede permite uma fácil manipulação das variáveis que determinam o fluxo e fornece simetria na dinâmica de tráfego em todos os nodos. Ou seja, qualquer nodo tem o mesmo número de vias de entrada e vias de saída, com as mesmas orientações. Nesse formato de grade se pode analisar as possibilidade de controle em um determinado ponto sabendo e controlando diretamente os fatores que determinam o tráfego naquele ponto. Sendo assim o foco passa a ser o controle em si, e não a rede de relacionamentos que pode levar a uma determinada configuração de tráfego. Os objetos injetores e sumidouros (apresentados na seção 3.3.1) estão dispostos simetricamente e a dinâmica da rede é configurada alterando apenas possibilidades de fluxo e valor de inserção de veículos nos injetores.

Nesse trabalho são consideradas quatro configurações distintas de malha viária em nível de número de cruzamento (agentes). Usamos 9, 25, 49 e 81 cruzamentos com semáforos. Nas explicações a seguir usaremos o caso da rede de 25 nodos, nomeados de A1 até E5, compondo as interseções da rede (cruzamentos). Essa rede pode ser visualizada na figura 5.1. Sendo assim, definimos um número de semáforos $N_s = 25$. A malha viária também contém um total de 10 nodos do tipo injetor (6 injetores para 9 nodos, 14 injetores para 49 nodos e 18 injetores para 81 nodos), dispostos de forma intercalada com outros 10 nodos do tipo sumidouro (6 sumidouros para 9 nodos, 14 sumidouros para 49 nodos e 18 sumidouros para 81 nodos). Entre os nodos do tipo injetor ou sumidouro e os nodos principais de entrada foi configurada uma distância de 300 metros, para que a vazão de entrada da rede (no caso de um injetor) possa ser absorvida sem que os veículos fiquem retidos antes do primeiro cruzamento. Entre os demais nodos (nodos principais) foi configurada a mesma distância de 300 metros, constituindo uma distância razoável para que haja possibilidade de movimentação, sem que se force congestionamento por motivo de distâncias demasiadamente pequenas. De acordo com o modelo de autômato celular usamos células de 5 metros, ou seja, cada via é formada por 60 células. Com essa definição construímos o tamanho de trecho de via $T_l = 60$ células, onde l indica a via entre dois nodos, ou seja, um dos trechos que compõem a via. Além disso, a velocidade *máxima* é de 3 células por iteração, o que corresponde a 15 m/s ou aproximadamente 54 Km/h, valor adequado para simulação de tráfego urbano.

Cada via possui apenas um sentido de tráfego, dado pelo vetor injetor → sumidouro, que pode ter duas direções: horizontal e vertical. Cada cruzamento possui um semáforo controlando os fluxos de acordo com alguns planos pré-determinados, detalhados na seção 5.2.

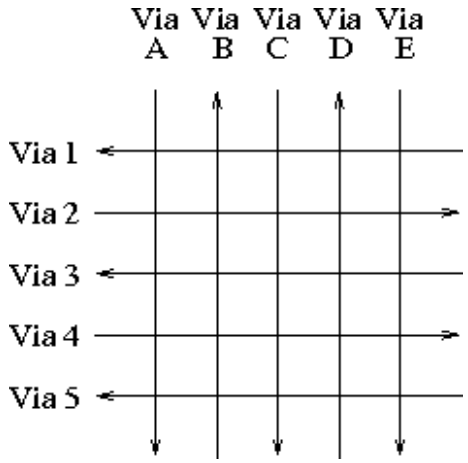


Figura 5.1: Visualização da malha viária.

5.2 Construção dos Planos Semafóricos

Os planos semafóricos foram construídos levando em consideração que cada semáforo na rede deve poder sincronizar em qualquer uma das duas direções: horizontal ou vertical. Sendo assim, foram disponibilizados dois planos semafóricos, um para sincronização horizontal e outro para sincronização vertical. Cabe ao controlador definir quanto utilizar um ou outro, ou seja, quando se deve sincronizar com uma direção específica. Sendo assim o tamanho do domínio de planos semafóricos é igual a dois, e temos $D_{sp} = 1,2$, onde 1 representa sincronização horizontal e 2 representa sincronização vertical.

Foi definido um tempo de ciclo $t_{ciclo} = 60$ segundos, e as fases foram divididas em tempos $t_{fase} = 30$ segundos. Sendo assim, a diferença entre os planos está na defasagem (*offset*) de aplicação das fases e, consequentemente, na direção da onda verde.

Com a definição de uma velocidade máxima $V_{max} = 3$ células/iteração temos um tempo de 22 segundos para que um motorista percorra uma via de $T_l = 60$ células (tamanho de todas as vias neste cenário), partindo da velocidade zero (aqui se considera o tempo de aceleração até a velocidade máxima) e um tempo de 20 segundos partindo da velocidade máxima e mantendo essa. Esses tempos consideram fluxo livre para o motorista em questão.

Com base nesses tempos a sincronização proposta prevê em cada via a seqüência de fases prevendo as defasagens (*offsets*) supracitadas.

Tomando como exemplo a via 2 (mostrada na figura 5.1), com direção horizontal e sentido Oeste para Leste temos o seguinte plano de sincronização horizontal:

Semáforo	2A	2B	2C	2D	2E
Fase 1	NS - 0 a 21 e NS - 52 a 59	LO - 0 a 11 e LO - 42 a 59	NS - 0 a 1 e NS - 32 a 59	NS - 0 a 21 e NS - 52 a 59	LO - 0 a 11 e LO - 42 a 59
Fase 2	LO - 22 a 51	NS - 12 a 41	LO - 2 a 31	LO - 22 a 51	NS - 12 a 41

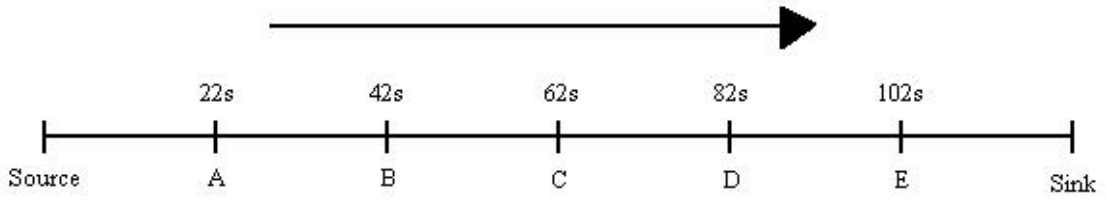


Figura 5.2: Trajeto dos motoristas. São indicados os nomes dos nodos e o tempo em que os motoristas os cruzam.

Considerando o tempo de ciclo $t_{ciclo} = 60$ segundos, cada item da tabela mostra o instante (em segundos) inicial e final de cada fase para sincronização em onda verde, conforme o trajeto mostrado na figura 5.2.

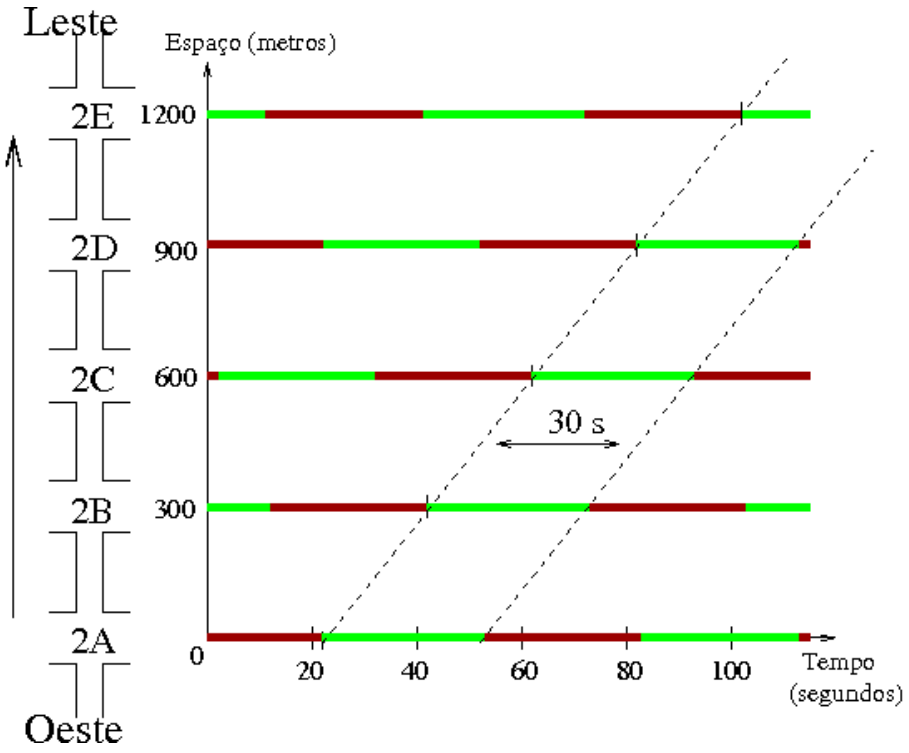


Figura 5.3: Diagrama espaço-tempo da sincronização dos semáforos.

Tomando como exemplo a via 2 (formada pelos cruzamentos 2A, 2B, 2C, 2D e 2E), com direção horizontal e sentido Oeste → Leste, podemos visualizar na figura 5.3 os planos coordenados que permitem que veículos que iniciam a viagem dentro da banda de tempo indicada pelas linhas pontilhadas trafeguem nessa via 2 sem parar.

Esses planos se repetem nos semáforos de todas as vias como sincronização na direção horizontal. Há outro plano para sincronização na direção vertical, onde são usadas as mesmas fases do plano apresentado anteriormente porém são invertidos os sentidos de cada fase.

5.3 Modelagem DCOP

Considerando a definição do formalismo para DCOP apresentada no capítulo 2 e a descrição do cenário de tráfego apresentada nas seções 5.1 e 5.2, é modelado a seguir

o cenário de estudo como um problema de otimização de restrições distribuídas. Para isso serão definidas as entidades correspondentes aos agentes, suas variáveis e a lógica de construção das restrições usadas na execução dos algoritmos.

De acordo com a construção da rede definida na seção 5.1 há $N_s = 25$ ($N_s = 9$, $N_s = 49$ e $N_s = 81$) semáforos e é atribuído um agente por semáforo. Além disso, cada agente é responsável pelo controle de seu semáforo, ou seja, pela atribuição do valor de sua variável, aqui entendido como a escolha do plano semafórico a ser utilizado. Como descrito anteriormente, o valor da variável do plano semafórico pode variar entre 1 e 2 , onde cada plano representa uma diferente configuração de defasagens.

Esse problema de otimização de restrições distribuídas, formalmente é escrito pela tupla $< A, D, F >$, conforme definido na seção 2.4.

Aqui são consideradas apenas restrições binárias, ou seja, entre duas variáveis/agentes ao mesmo tempo.

5.4 Construção das Restrições

As restrições são construídas dinamicamente conforme formulação apresentada na seção 4.4. Ou seja, as inferências de densidade nos trechos das vias são obtidas no momento da atuação dos agentes, e leva em consideração o estado da malha naquele instante. Esses valores determinam os custos das restrições.

5.5 Configuração das Simulações

Detalhes dos valores de configuração são dados a seguir, e na descrição dos cenários usados nas simulações. Foram abordados, para as simulações, dois tipos de alimentação da rede: alimentação estática e alimentação dinâmica. Esse fator influencia diretamente no comportamento refletido nos nodos da rede. Para cada tipo de alimentação há três cenários, distintos pelo tipo de controle representado: controle não sincronizado, controle sincronizado, e controle por DCOP.

5.5.1 Alimentação da Rede

Por alimentação da rede se entende a configuração do valor da taxa de inserção dos injetores da rede. Os dois casos propostos nas próximas seções visam atender a dois parâmetros de estudo, caracterizados pelo comportamento estático desses elementos e pelo comportamento variável em função do tempo.

5.5.1.1 Alimentação Estática

Nesse caso todos os injetores da rede são configurados com o mesmo valor de taxa de inserção, ou seja, a probabilidade de que um veículo seja inserido em determinado instante. Essa forma prevê igualdade na alimentação da rede em todos os sentidos ao longo do tempo.

Nesse caso, usamos os seguintes parâmetros de simulação:

- Número de passos: 9000 passos = 9000 segundos = 2.5 horas de tráfego;
- Freqüência da coleta dos dados da rede: 1 passo = cada 1 segundo;
- Freqüência de atuação dos agentes (para o cenário com DCOP): 720 passos (12 minutos);

- Probabilidade de desaceleração: 0% e 5%.

O valor de taxa de inserção usada é 0.3, ou seja, 30% de chance de inserir um veículo em um instante de tempo. Valores menores que esse não representam uma variação significativa na avaliação da rede, conforme estudos feitos no decorrer do trabalho, e portanto foram omitidos dos experimentos finais.

5.5.1.2 Alimentação Dinâmica

A configuração do cenário com alimentação estática viabiliza estudar as características de adaptabilidade das configurações de controle a serem estudadas. A alimentação varia conforme a tabela 5.1, e com isso são geradas situações distintas de tráfego:

1. **Madrugada**: baixa taxa de inserção nos injetores. Caracteriza uma situação de baixo volume de tráfego na rede;

2. **Manhã**: maior taxa de inserção nos injetores das vias de direção horizontal;

3. **Tarde**: maior taxa de inserção nos injetores das vias de direção vertical;

4. **Hora do Rush**: mesma taxa de inserção em todos injetores, com valores variando de médio a alto.

Espera-se relatar com essa variação as diferenças encontradas no caso de não haver sincronização de semáforos, haver sincronização em uma determinada direção, e haver uma forma de controle adaptativa atuando na seleção dos planos semafóricos mais adequados (uso dos algoritmos de DCOP).

Usamos os seguintes parâmetros:

- Número de passos: 35000 passos = 35000 segundos = aproximadamente 10 horas de tráfego;

- Freqüência de atuação dos sensores (coleta dos dados da rede): 1 passo = cada 1 segundo;

- Freqüência de atuação dos agentes (para o cenário com DCOP): 720 passos (12 minutos);

- Probabilidade de desaceleração: 0% e 5%.

Dentro daquele tempo de simulação (10 horas), usamos as situações exibidas na tabela 5.1.

5.6 Cenários

Os cenários aqui tratados dizem respeito às formas de controle usadas nas simulações, e descritas nas seções abaixo.

Tabela 5.1: Taxas de inserção - Alimentação Dinâmica

Situação	Intervalos	Taxa NS	Taxa LO
Madrugada I	0 a 5000	0.10	0.10
Manhã	5000 a 10000	0.10	0.40
Tarde	10000 a 15000	0.40	0.10
Hora do *rush* I	15000 a 25000	0.3	0.3
Hora do *rush* II	25000 a 30000	0.5	0.5
Madrugada II	30000 a 35000	0.10	0.10

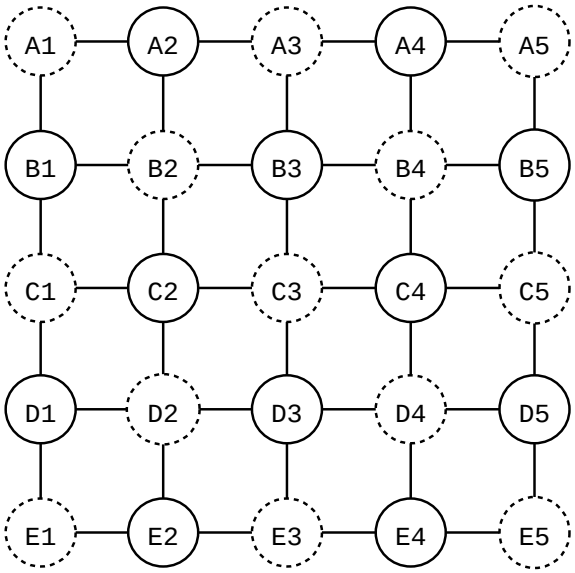


Figura 5.4: Cenário 1: os cruzamentos com círculos pontilhados sincronizam na direção vertical e os com linha cheia sincronizam na direção horizontal.

5.6.1 Cenário 1: configuração não coordenada de planos

Propositadamente, a configuração dos planos faz com que cada semáforo não coordene a direção de sincronização com nenhum de seus vizinhos. Essa situação pode ser visualizada na figura 5.4, e serve como exemplo dos efeitos de uma má coordenação.

5.6.2 Cenário 2: uso de planos sincronizados

São usados os planos definidos na seção 5.2 para geração de onda verde na direção horizontal. O objetivo é comparar esse tipo de controle fixo porém sincronizado, e comumente adotado na prática, aos demais métodos usados. Este cenário serve para mostrar os efeitos de uma sincronização fixa quando o tráfego muda, passando a ficar em desacordo com a sincronização planejada.

5.6.3 Cenário 3: uso da intervenção dos algoritmos DCOP na simulação

A fim de solucionar o problema do desacordo do tráfego com a sincronização planejada, este trabalho propõe a escolha de planos sincronizados de forma adaptativa utilizando DCOP. Com uma freqüência definida é executado um dos algoritmos DCOP (OptAPO, DPOP ou ADOPT).

No próximo capítulo são relatados os resultados dos experimentos e a análise, a partir dos parâmetros e casos definidos anteriormente.

6 RESULTADOS E COMPARAÇÃO

Nesse capítulo serão apresentados os resultados das simulações realizadas bem como a sua análise. O capítulo está dividido em duas seções principais: a seção 6.1 trata dos resultados observados, e em relação à qualidade da solução, conforme seção 4.7 (densidade das vias, velocidade média, veículos parados). A seção 6.2 trata dos resultados sob o ponto de vista das métricas para comparação de algoritmos DCOP, introduzidas na seção 4.8. Na seção 6.3 são apresentadas as considerações finais sobre as simulações relacionando os resultados das seções citadas anteriormente.

Os valores apresentados nas tabelas e gráficos são o resultado do cálculo da **média para 10 repetições** de cada caso. Todas as simulações foram executadas em um computador processado por Intel Centrino Core Duo de 1.6GHz, com 512MB de memória RAM e executando o sistema operacional Suse Linux 10.2. A tabela 6.1 apresenta um resumo dos parâmetros de simulação usados, já citados no capítulo anterior.

Tabela 6.1: Resumo dos parâmetros de simulação usados

Parâmetro	Alimentação Estática	Alimentação Dinâmica
No. de repetições	10	10
Número de passos de simulação	9000	35000
Freqüência de leitura de dados	1	1
Freqüência de atuação dos agentes	720	720
Taxa de inserção	0.3	0.1 até 0.5 (ver tabela 5.1)
Probabilidade de desaceleração	0% e 5%	0% e 5%

6.1 Avaliação da Rede - Médias de Densidade, Velocidade e Número de Veículos Parados

Foram usados os parâmetros definidos no capítulo anterior, mais especificamente na seção 5.5, e os valores medidos são relatados nas seções abaixo.

6.1.1 Alimentação Estática

Da tabela 6.2 até a tabela 6.5 são mostrados os valores de velocidade média, densidade média e média de número de veículos parados para as redes simuladas considerando uma probabilidade de desaceleração de 0%. Da mesma forma, da tabela 6.6 até a tabela 6.9 são mostradas essas mesmas métricas considerando 5% de probabilidade de

desaceleração. Quanto maior a probabilidade de desaceleração, mais tráfego se terá em um dado momento. Os valores foram obtidos pela média de cada métrica ao logo dos passos de simulação, excluindo-se os primeiros 2000 passos, onde a rede ainda não está suficientemente povoada de veículos.

Tabela 6.2: Valores médios das métricas da rede para alimentação estática e 9 semáforos - desaceleração 0%

Experimento	Veículos parados	Densidade	Velocidade Média
Cenário 1	4.16 (±0.80)	0.17 (±0.02)	1.70 (±0.22)
Cenário 2	2.12 (±0.55)	0.13 (±0.02)	2.09 (±0.34)
Cenário 3 (ADOPT)	0.97 (±0.56)	0.08 (±0.01)	2.05 (±0.40)
Cenário 3 (OptApo)	0.91 (±0.50)	0.08 (±0.01)	2.15 (±0.43)
Cenário 3 (DPOP)	1.42 (±0.71)	0.09 (±0.02)	1.86 (±0.40)

Tabela 6.3: Valores médios das métricas da rede para alimentação estática e 25 semáforos - desaceleração 0%

Experimento	Veículos parados	Densidade	Velocidade Média
Cenário 1	3.83 (±0.98)	0.17 (±0.01)	1.80 (±0.25)
Cenário 2	2.10 (±0.72)	0.13 (±0.01)	2.10 (±0.29)
Cenário 3 (ADOPT)	1.04 (±0.63)	0.08 (±0.02)	1.98 (±0.26)
Cenário 3 (OptApo)	1.15 (±0.59)	0.08 (±0.01)	2.03 (±0.25)
Cenário 3 (DPOP)	1.70 (±0.40)	0.09 (±0.02)	1.77 (±0.21)

Tabela 6.4: Valores médios das métricas da rede para alimentação estática e 49 semáforos - desaceleração 0%

Experimento	Veículos parados	Densidade	Velocidade Média
Cenário 1	4.02 (±0.70)	0.16 (±0.01)	1.73 (±0.22)
Cenário 2	2.21 (±0.23)	0.13 (±0.01)	1.88 (±0.50)
Cenário 3 (ADOPT)	0.97 (±0.52)	0.08 (±0.01)	2.06 (±0.28)
Cenário 3 (OptApo)	1.02 (±0.47)	0.09 (±0.01)	2.03 (±0.25)
Cenário 3 (DPOP)	1.43 (±0.28)	0.09 (±0.01)	1.87 (±0.17)

Tanto com 9 quanto com 25, 49 ou 81 semáforos é possível perceber uma diminuição do número de veículos parados usando DCOP. A velocidade média não é a melhor métrica a ser considerada nesse caso devido à discretização do modelo de movimentação adotado no simulador. O modelo impõe que a velocidade seja um número inteiro, e portanto de baixa granularidade. Por essa razão, serão omitidos os gráficos referentes à velocidade média, exceto pela figura 6.1, que exemplifica essa métrica para o cenário de alimentação estática na rede com 25 agentes e 5% de probabilidade de desaceleração . O cenário 1, com semáforos não sincronizados, representa a pior situação de controle possível na rede, como esperado, já que a falta de coordenação foi propositadamente introduzida. A leitura dessas tabelas permite interpretar que os valores das métricas para o cenário 1 ficaram muito aquém dos cenários com sincronização, ou seja, os cenários 2 e 3. Em todas as situações isso se deve à otimização do fluxo imposta pela sincronização, que permite aos motoristas trafegarem por mais tempo em sinal verde, dando vazão à alimentação dos injetores.

Tabela 6.5: Valores médios das métricas da rede para alimentação estática e 81 semáforos - desaceleração 0%

Experimento	Veículos parados	Densidade	Velocidade Média
Cenário 1	4.78 (±0.36)	0.14 (±0.01)	1.20 (±0.10)
Cenário 2	1.88 (±0.82)	0.10 (±0.01)	1.90 (±0.22)
Cenário 3 (ADOPT)	1.29 (±0.87)	0.08 (±0.01)	1.78 (±0.31)
Cenário 3 (OptAPO)	1.30 (±0.85)	0.09 (±0.01)	1.80 (±0.42)
Cenário 3 (DPOP)	1.54 (±0.72)	0.09 (±0.01)	1.68 (±0.49)

Tabela 6.6: Valores médios das métricas da rede para alimentação estática e 9 semáforos - desaceleração 5%

Experimento	Veículos parados	Densidade	Velocidade Média
Cenário 1	10.21 (±2.28)	0.29 (±0.06)	1.21 (±0.23)
Cenário 2	6.72 (±1.37)	0.23 (±0.03)	1.40 (±0.20)
Cenário 3 (ADOPT)	1.88 (±0.53)	0.11 (±0.02)	1.91 (±0.30)
Cenário 3 (OptApo)	1.97 (±0.60)	0.11 (±0.01)	1.89 (±0.34)
Cenário 3 (DPOP)	2.42 (±0.46)	0.13 (±0.02)	1.71 (±0.23)

Em relação ao cenário 3, os algoritmos DCOP também apresentaram melhoria adicional ao cenário 2. Porém para o caso de 9 e 81 semáforos essa melhoria não foi muito expressiva. Nesse último caso pode ser verificada uma dificuldade dos algoritmos DCOP no que diz respeito à otimização desse número de variáveis e suas restrições. Pode-se perceber que se torna difícil manter o compromisso de otimização com esse número de restrições a serem consideradas. No caso do cenário de 9 agentes, pela simplicidade do cenário não há espaço para um controle mais sofisticado.

Na seqüência, os valores resumidos nas tabelas anteriores são exibidos graficamente comparados ao decorrer dos passos de simulação, para o valor de densidade média.

6.1.2 Alimentação Dinâmica

No caso **Manhã** temos uma situação oposta ao controle proposto para a sincronização fixa do cenário 2, pois a alimentação maior vem dos injetores das vias verticais, enquanto a sincronização ocorre na direção horizontal. Isso pode ser visualizado pela leitura das taxas de inserção listadas na tabela 5.1. Já no caso **Tarde** há uma taxa maior de inserção na direção da sincronização (ainda para o cenário 2). Em geral os algoritmos DCOP (cenário 3) se mostraram mais flexíveis às variações na taxas de inserção (que geram instabilidades no efeito da sincronização). Isso pode ser visto pelo acompanhamento da densidade média das vias ao longo da simulação comparando o cenário 3 (DCOP) com os demais. As figuras 6.10 até 6.13 mostram graficamente esses valores considerando 0% de probabilidade de desaceleração.

Tabela 6.7: Valores médios das métricas da rede para alimentação estática e 25 semáforos - desaceleração 5%

Experimento	Veículos parados	Densidade	Velocidade Média
Cenário 1	9.22 (±2.36)	0.28 (±0.04)	1.32 (±0.26)
Cenário 2	6.95 (±1.82)	0.23 (±0.03)	1.48 (±0.22)
Cenário 3 (ADOPT)	1.78 (±0.80)	0.11 (±0.01)	1.97 (±0.23)
Cenário 3 (OptApo)	1.90 (±0.59)	0.12 (±0.01)	1.94 (±0.26)
Cenário 3 (DPOP)	3.08 (±0.40)	0.14 (±0.02)	1.70 (±0.27)

Tabela 6.8: Valores médios das métricas da rede para alimentação estática e 49 semáforos - desaceleração 5%

Experimento	Veículos parados	Densidade	Velocidade Média
Cenário 1	8.91 (±2.29)	0.27 (±0.04)	1.29 (±0.22)
Cenário 2	5.82 (±1.31)	0.21 (±0.02)	1.52 (±0.17)
Cenário 3 (ADOPT)	1.97 (±0.50)	0.12 (±0.01)	1.93 (±0.18)
Cenário 3 (OptApo)	1.91 (±0.49)	0.12 (±0.01)	1.94 (±0.17)
Cenário 3 (DPOP)	2.90 (±0.62)	0.14 (±0.01)	1.80 (±0.19)

Tabela 6.9: Valores médios das métricas da rede para alimentação estática e 81 semáforos - desaceleração 5%

Experimento	Veículos parados	Densidade	Velocidade Média
Cenário 1	8.05 (±2.17)	0.14 (±0.01)	1.20 (±0.10)
Cenário 2	5.35 (±0.97)	0.10 (±0.01)	1.90 (±0.22)
Cenário 3 (ADOPT)	2.04 (±0.77)	0.12 (±0.01)	1.90 (±0.21)
Cenário 3 (OptAPO)	1.98 (±0.85)	0.11 (±0.01)	1.92 (±0.18)
Cenário 3 (DPOP)	2.88 (±0.72)	0.13 (±0.02)	1.77 (±0.20)

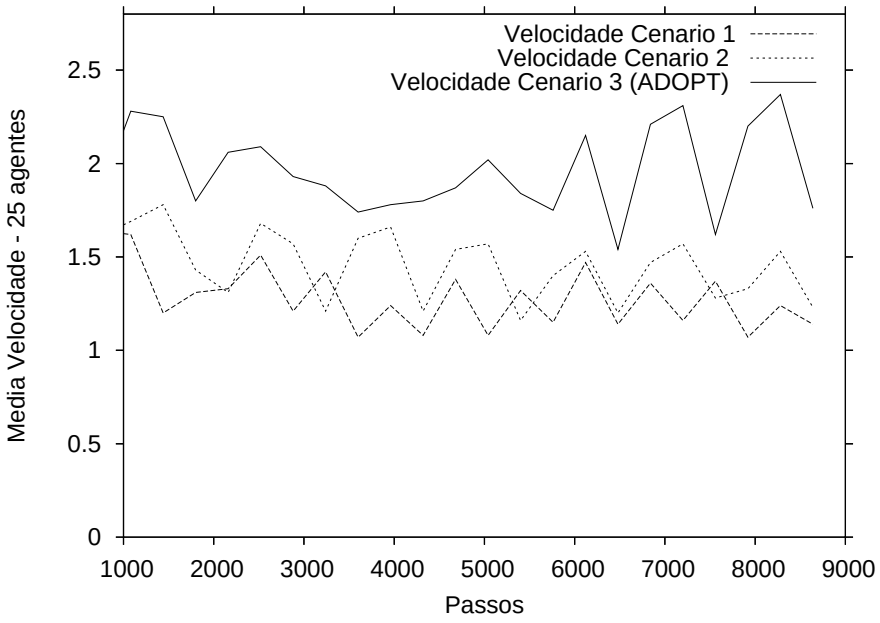


Figura 6.1: Velocidade média das vias da rede executando ADOPT no caso de alimentação estática, 25 agentes e 0% de probabilidade de desaceleração.

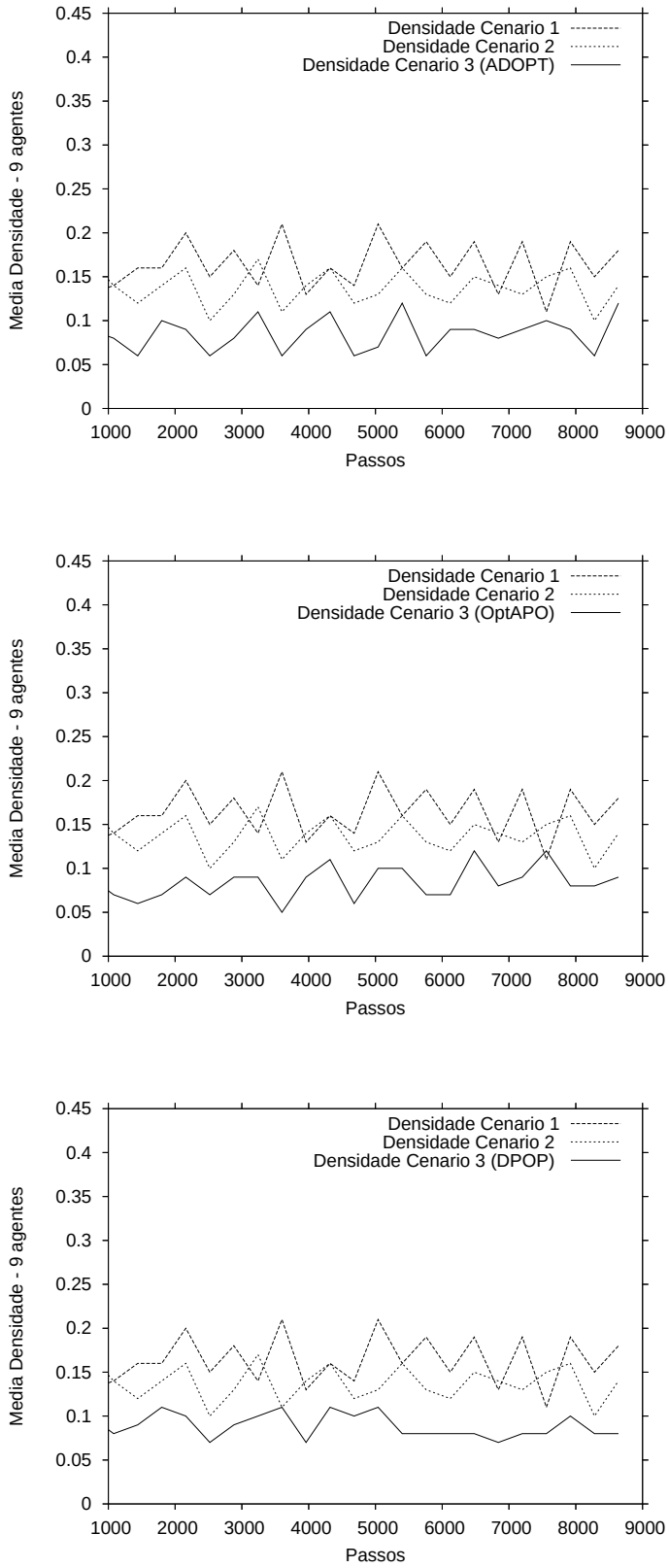


Figura 6.2: Densidade média das vias da rede no caso de alimentação estática, 9 agentes e 0% de probabilidade de desaceleração, executando ADOPT (acima), OptAPO (meio) e DPOP (abaixo).

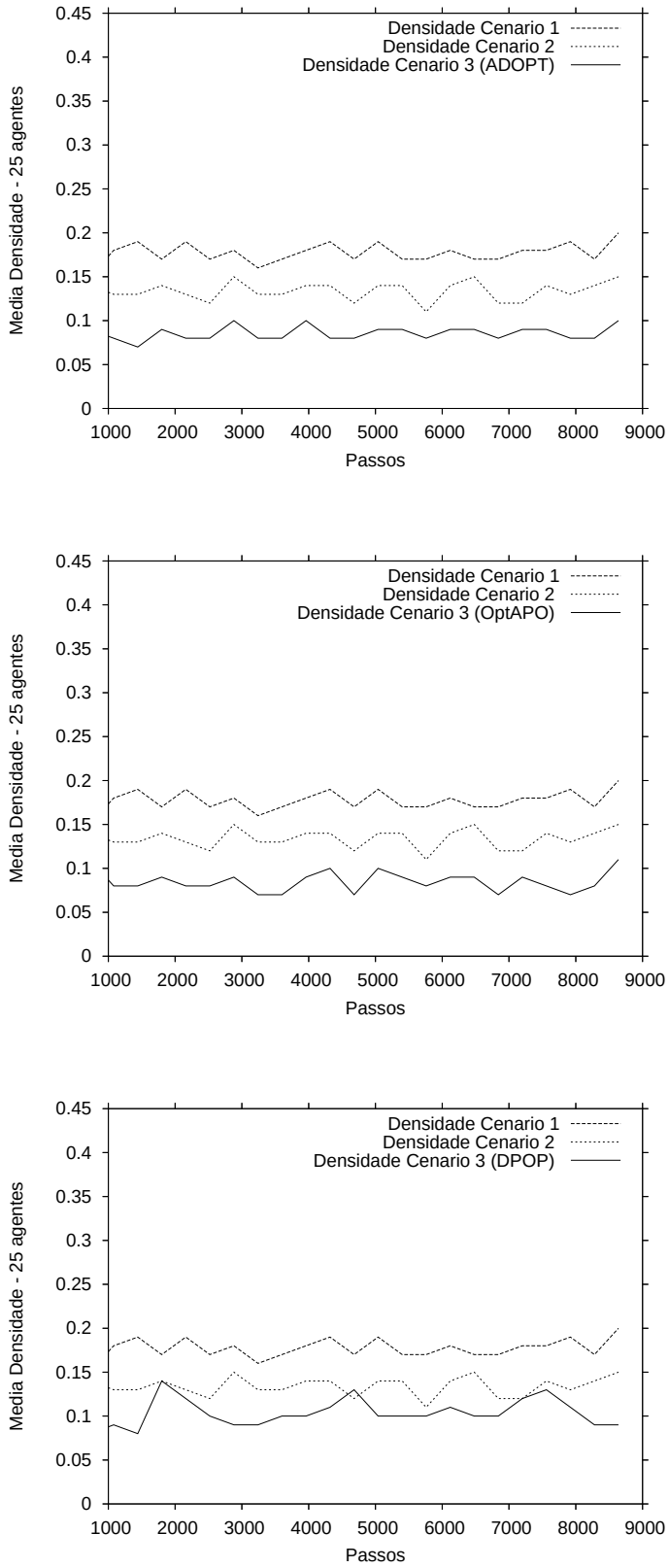


Figura 6.3: Densidade média das vias da rede no caso de alimentação estática, 25 agentes e 0% de probabilidade de desaceleração, executando ADOPT (acima), OptAPO (meio) e DPOP (abaixo).

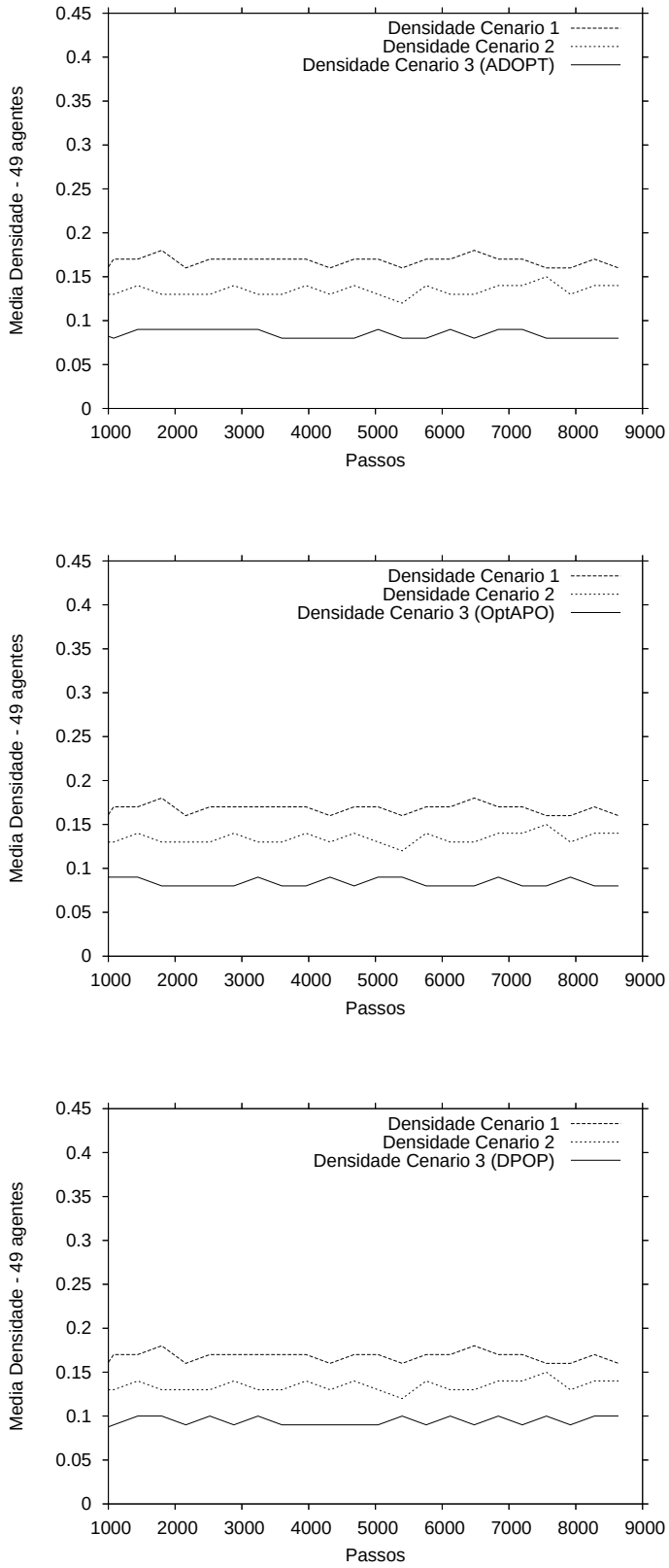


Figura 6.4: Densidade média das vias da rede no caso de alimentação estática, 49 agentes e 0% de probabilidade de desaceleração, executando ADOPT (acima), OptAPO (meio) e DPOP (abaixo).

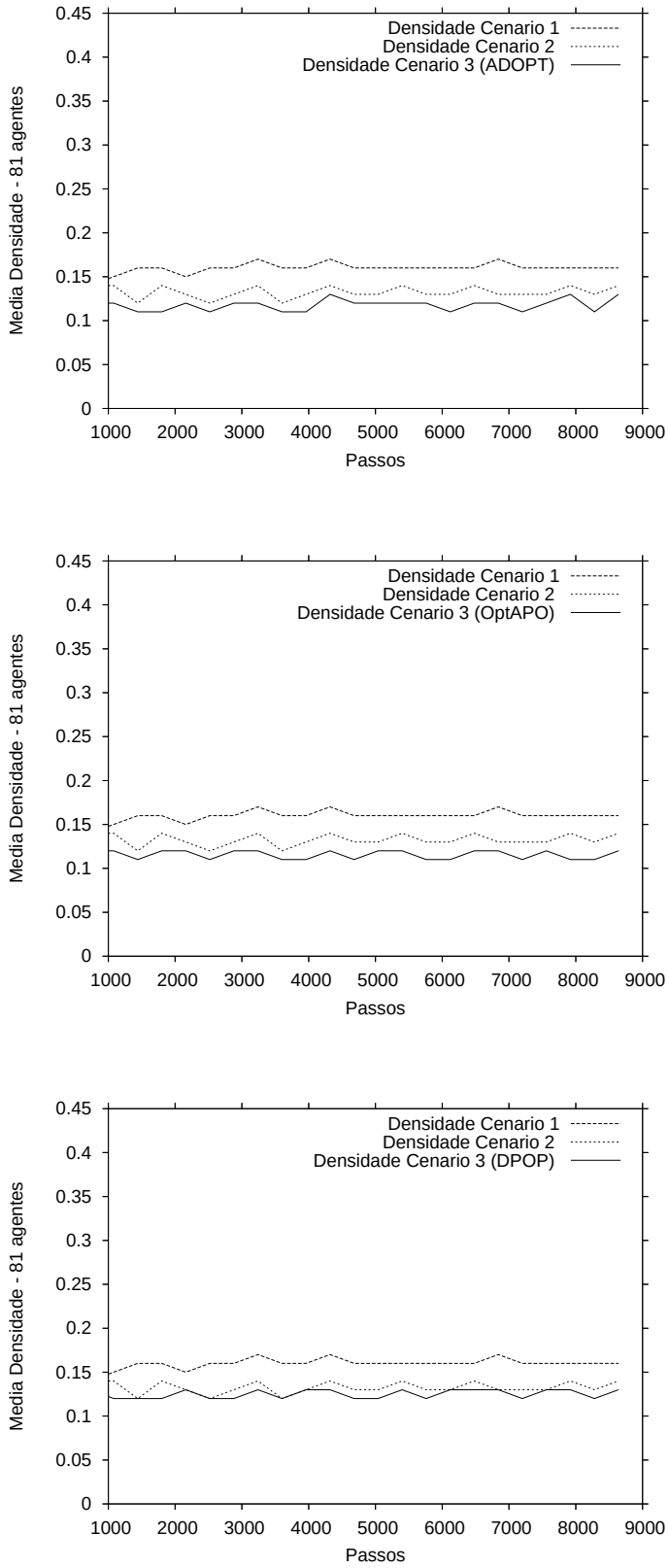


Figura 6.5: Densidade média das vias da rede no caso de alimentação estática, 81 agentes e 0% de probabilidade de desaceleração, executando ADOPT (acima), OptAPO (meio) e DPOP (abaixo).

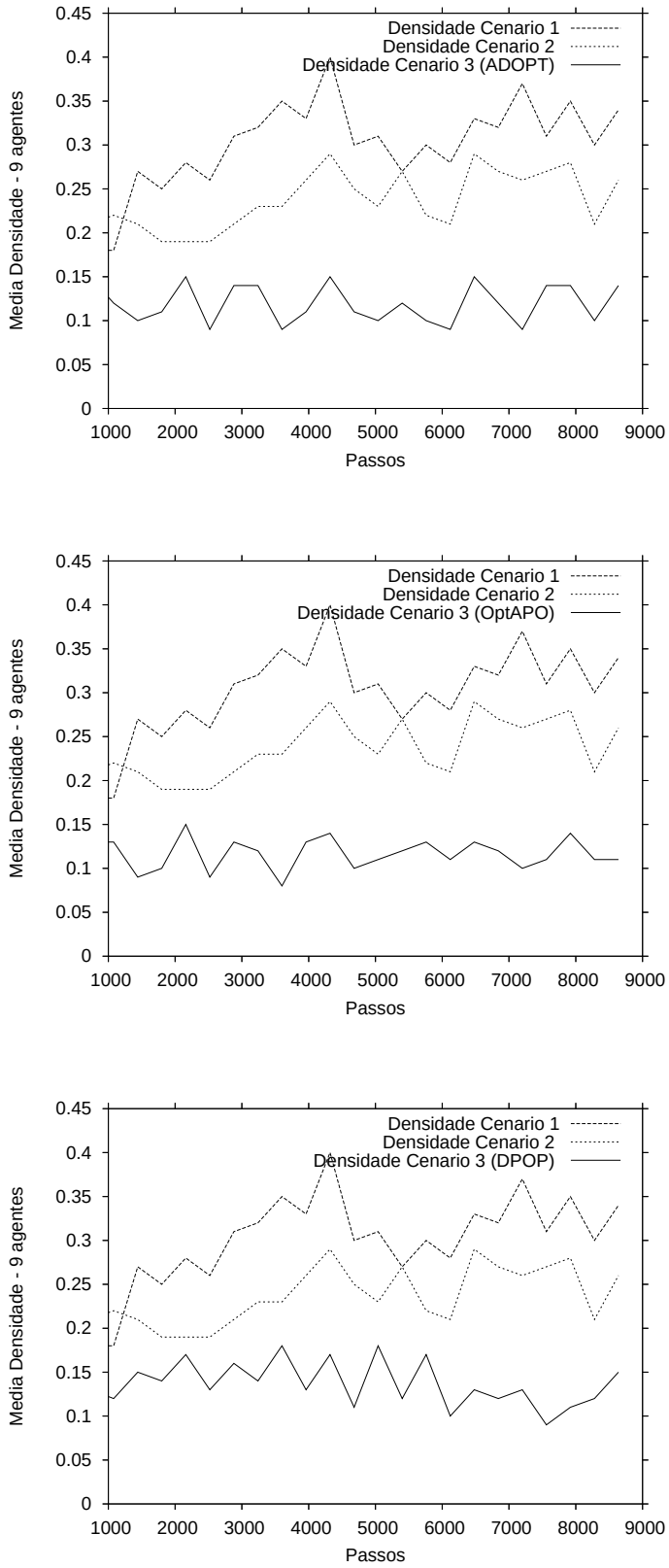


Figura 6.6: Densidade média das vias da rede no caso de alimentação estática, 9 agentes e 5% de probabilidade de desaceleração, executando ADOPT (acima), OptAPO (meio) e DPOP (abaixo).

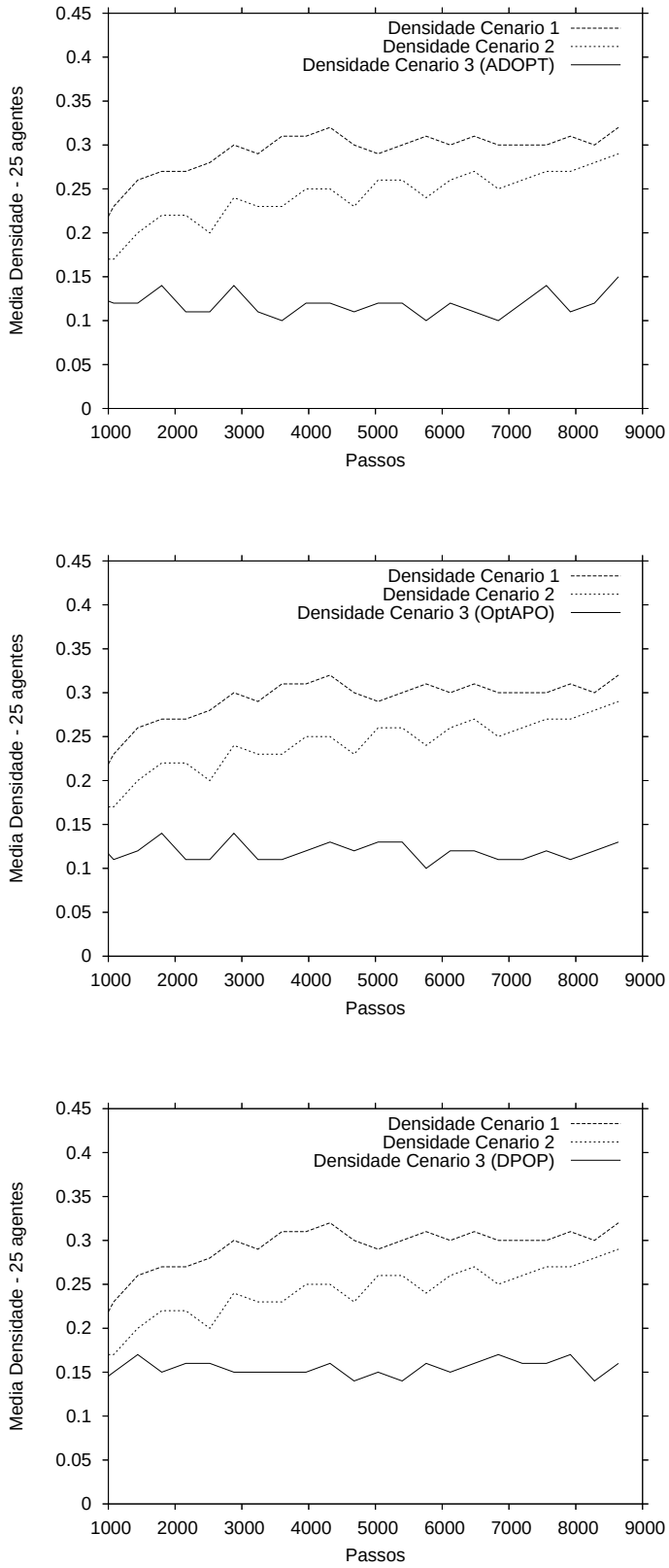


Figura 6.7: Densidade média das vias da rede no caso de alimentação estática, 25 agentes e 5% de probabilidade de desaceleração, executando ADOPT (acima), OptAPO (meio) e DPOP (abaixo).

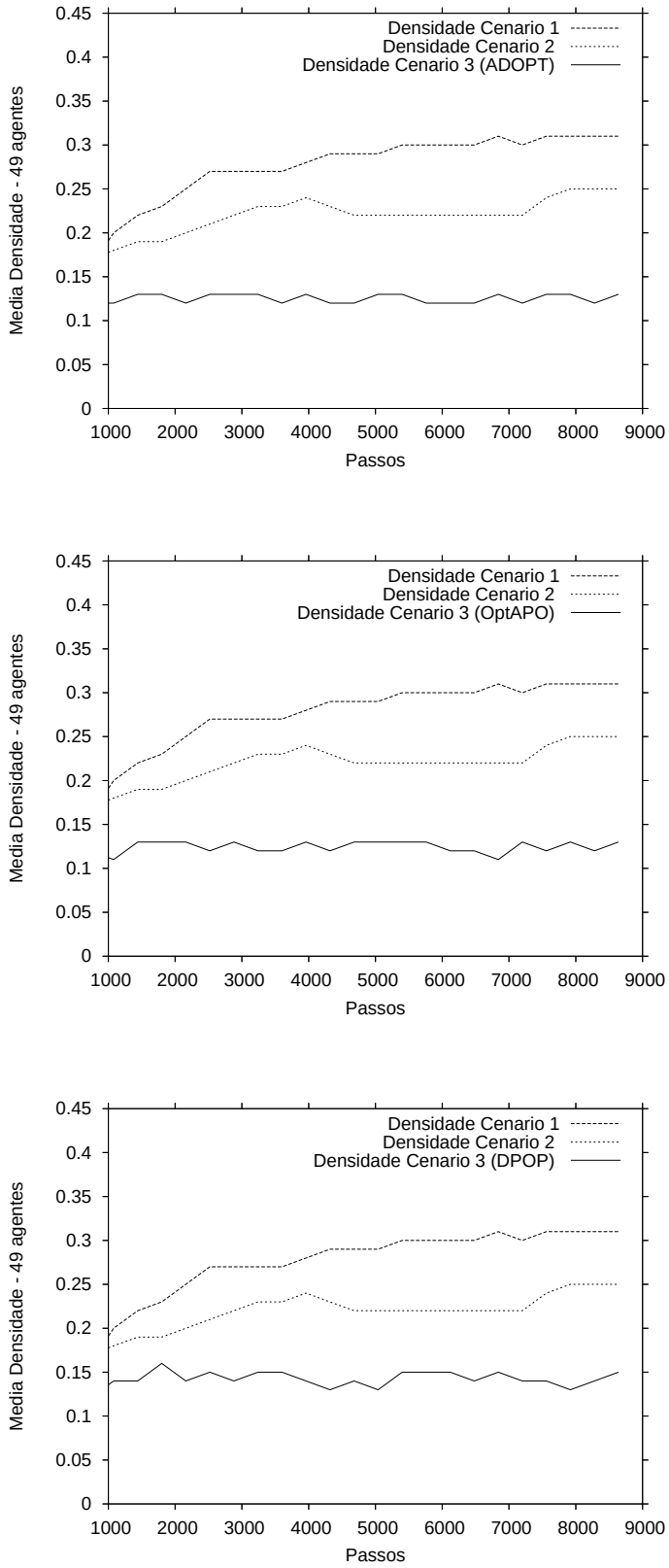


Figura 6.8: Densidade média das vias da rede no caso de alimentação estática, 49 agentes e 5% de probabilidade de desaceleração, executando ADOPT (acima), OptAPO (meio) e DPOP (abaixo).

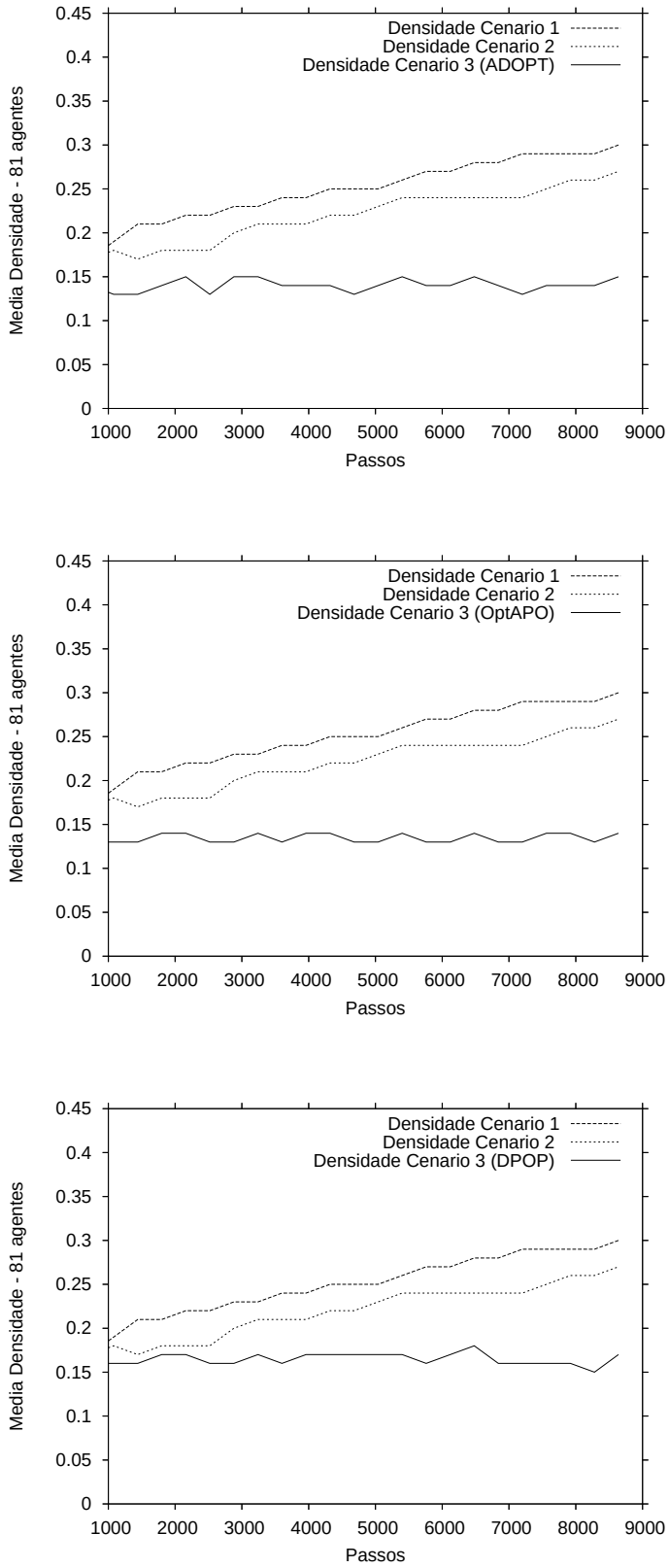


Figura 6.9: Densidade média das vias da rede no caso de alimentação estática, 81 agentes e 5% de probabilidade de desaceleração, executando ADOPT (acima), OptAPO (meio) e DPOP (abaixo).

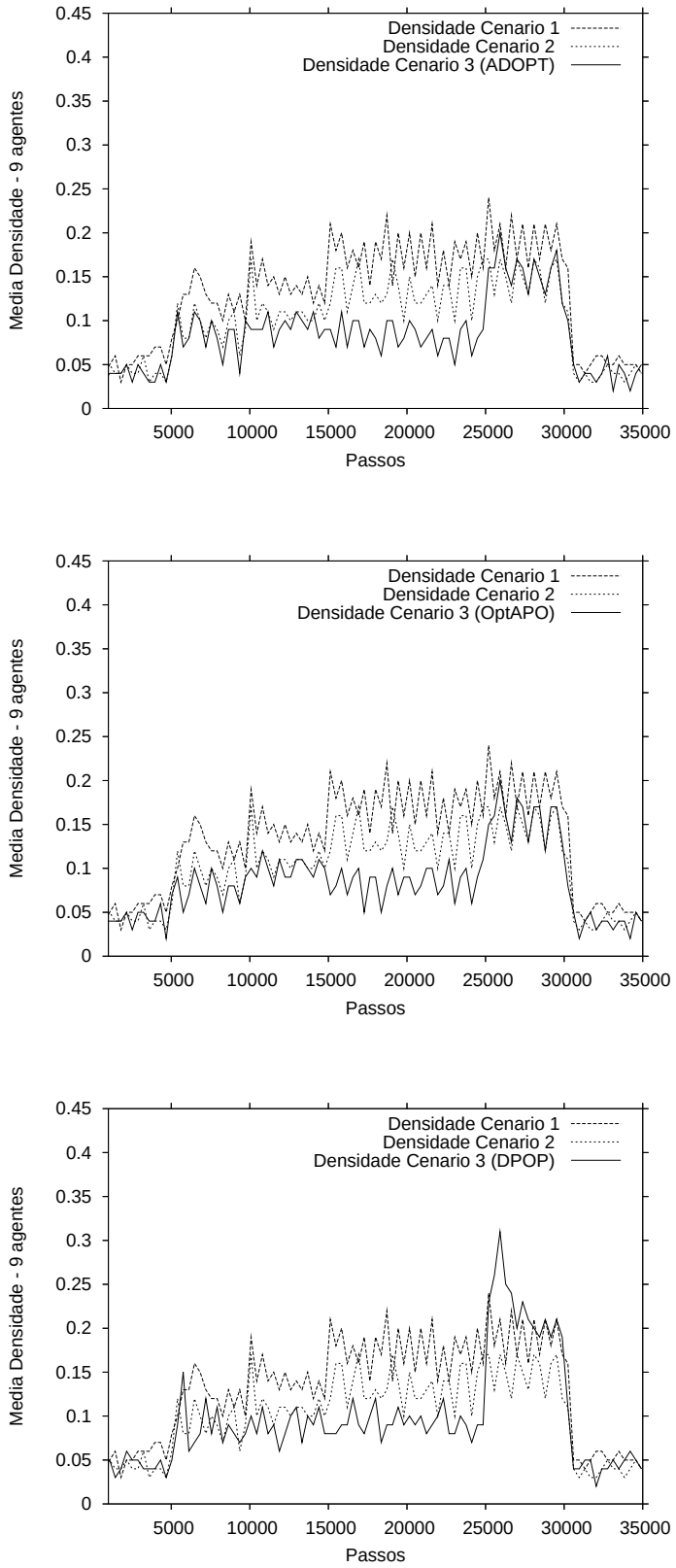


Figura 6.10: Densidade média das vias da rede no caso de alimentação dinâmica, 9 agentes e 0% de probabilidade de desaceleração, executando ADOPT (acima), OptAPO (meio) e DPOP (abaixo).

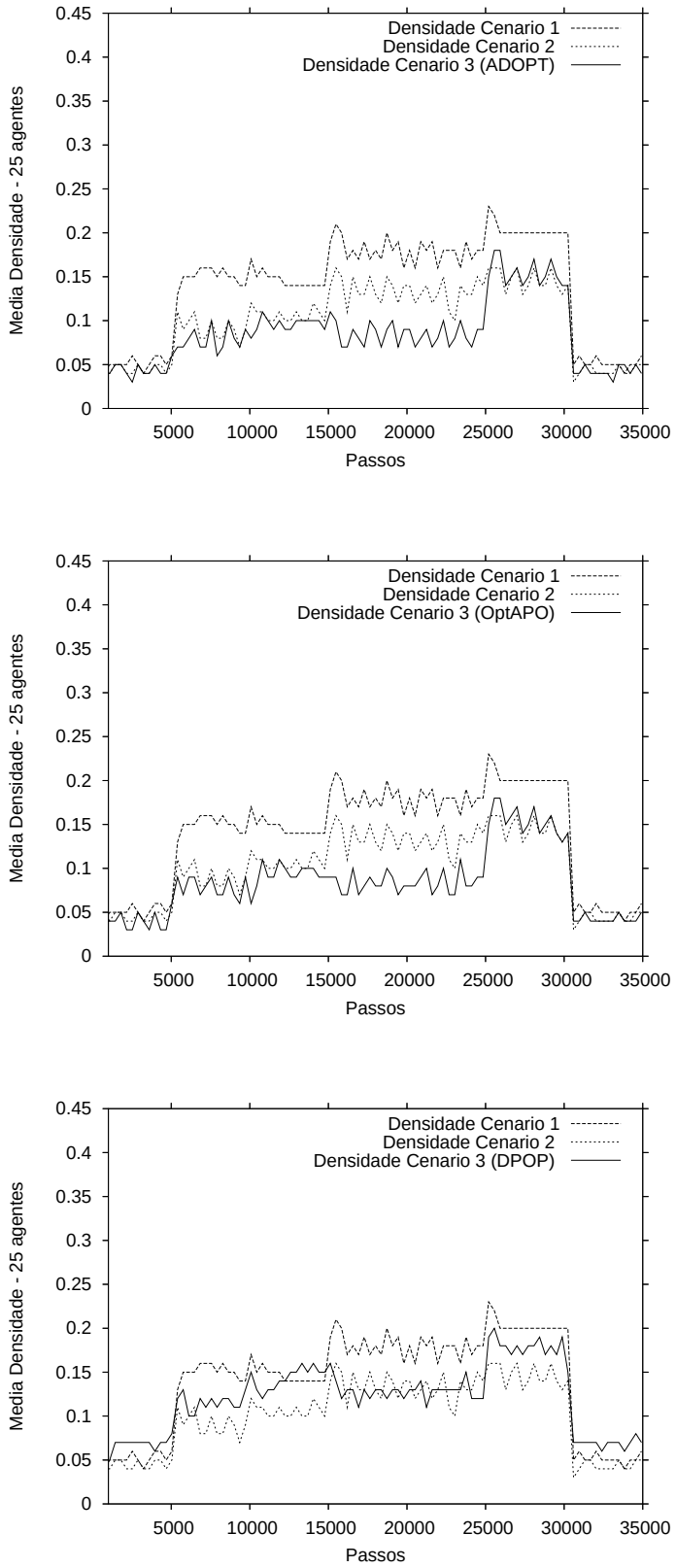


Figura 6.11: Densidade média das vias da rede no caso de alimentação dinâmica, 25 agentes e 0% de probabilidade de desaceleração, executando ADOPT (acima), OptAPO (meio) e DPOP (abaixo).

Da mesma forma, as figuras 6.14 até 6.17 mostram graficamente os valores de densidade média considerando 5% de probabilidade de desaceleração.

As conclusões apresentadas anteriormente, para o cenário de alimentação estática, também são válidas para o caso de alimentação dinâmica do cenário. O cenário 2, apesar de mais eficiente que o cenário 1 na situação que a alimentação favorecia a direção de sincronização (**Tarde**), não apresentou a mesma ordem de desempenho nos demais casos. Isso acontece pois a sincronização favorece apenas uma direção, sendo que as demais são prejudicadas por isso, e também se percebe assim o efeito dos congestionamentos nas vias secundárias (não priorizadas pela sincronização). No caso **Hora do Rush**, onde o cenário 2 foi desfavorecido, ou seja, menos eficiente (frente ao cenário 3, com DCOP), por estar priorizando as vias principais, quando as secundárias também estavam alimentando a rede da mesma forma que as principais (em termos de taxa de inserção dos injetores). Todos os cenários apresentaram desempenho equivalente no caso **Madrugada**, pois com uma taxa de inserção muito baixa em todos injetores, nenhuma forma de controle é requerida. Vale ressaltar o desempenho de DCOP para a situação dinâmica frente aos demais pela característica de adaptabilidade às variações de inserção de veículos.

O detalhamento dos valores das métricas de avaliação da rede, na forma de tabelas, para o caso de alimentação dinâmica é apresentado a seguir.

Tabela 6.10: Valores médios da densidade para a rede com 9 semáforos, alimentação dinâmica e probabilidade de desaceleração de 0%

Cenário	Madrugada	Manhã	Tarde	Hora do *Rush*
Cenário 1	0.05 ($\pm$ 0.01)	0.12 ($\pm$ 0.02)	0.14 ($\pm$ 0.01)	0.17 ($\pm$ 0.02)
Cenário 2	0.04 ($\pm$ 0.01)	0.09 ($\pm$ 0.01)	0.11 ($\pm$ 0.01)	0.13 ($\pm$ 0.02)
Cenário 3 (ADOPT)	0.03 ($\pm$ 0.01)	0.08 ($\pm$ 0.02)	0.09 ($\pm$ 0.01)	0.08 ($\pm$ 0.01)
Cenário 3 (OptAPO)	0.04 ($\pm$ 0.01)	0.07 ($\pm$ 0.01)	0.10 ($\pm$ 0.01)	0.08 ($\pm$ 0.01)
Cenário 3 (DPOP)	0.04 ($\pm$ 0.01)	0.08 ($\pm$ 0.02)	0.09 ($\pm$ 0.01)	0.09 ($\pm$ 0.01)

Tabela 6.11: Valores médios da velocidade para a rede com 9 semáforos, alimentação dinâmica e probabilidade de desaceleração de 0%

Cenário	Madrugada	Manhã	Tarde	Hora do *Rush*
Cenário 1	1.63 ($\pm$ 0.31)	1.75 ($\pm$ 0.23)	1.59 ($\pm$ 0.25)	1.69 ($\pm$ 0.16)
Cenário 2	1.76 ($\pm$ 0.40)	1.98 ($\pm$ 0.29)	1.96 ($\pm$ 0.42)	2.04 ($\pm$ 0.36)
Cenário 3 (ADOPT)	1.90 ($\pm$ 0.34)	1.97 ($\pm$ 0.32)	1.89 ($\pm$ 0.55)	2.03 ($\pm$ 0.43)
Cenário 3 (OptAPO)	1.94 ($\pm$ 0.48)	2.01 ($\pm$ 0.39)	1.96 ($\pm$ 0.38)	2.09 ($\pm$ 0.39)
Cenário 3 (DPOP)	1.70 ($\pm$ 0.46)	1.78 ($\pm$ 0.33)	1.71 ($\pm$ 0.33)	1.65 ($\pm$ 0.41)

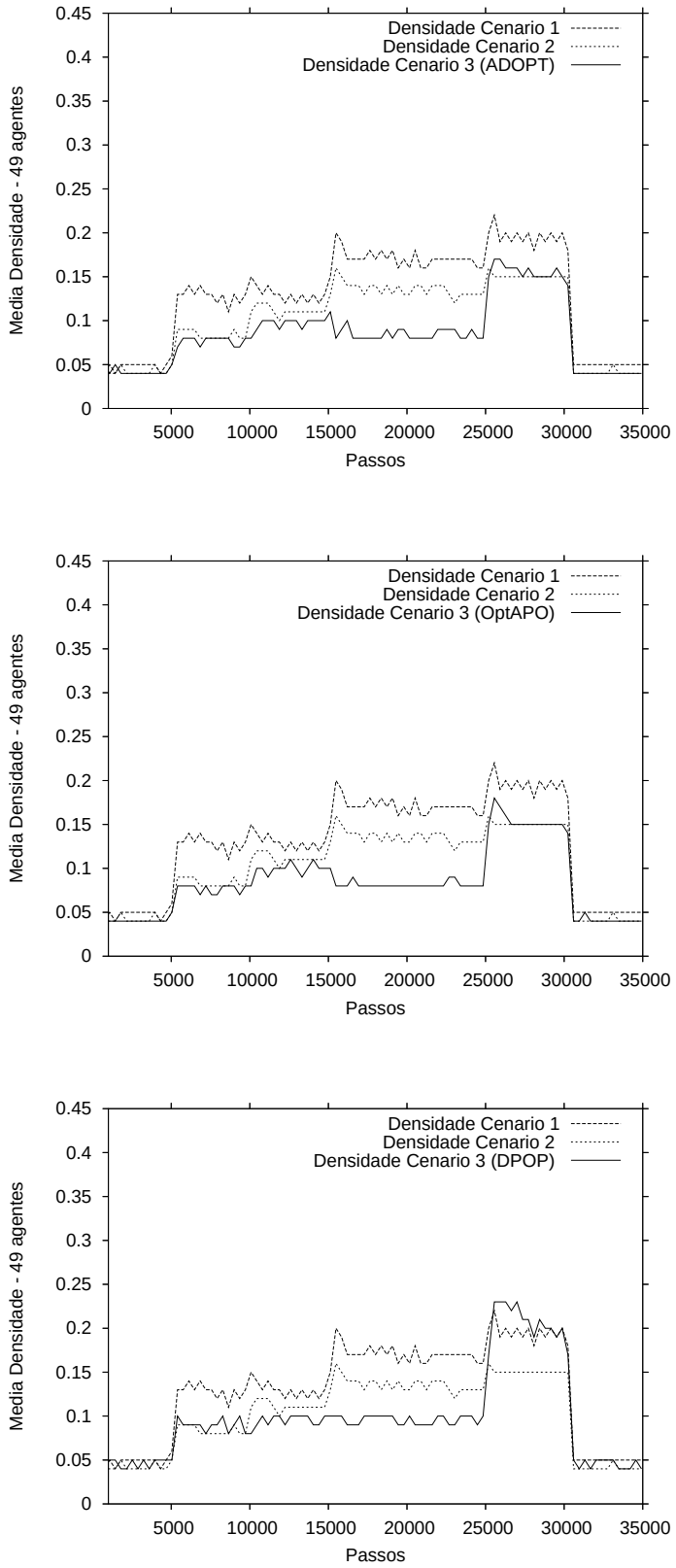


Figura 6.12: Densidade média das vias da rede no caso de alimentação dinâmica, 49 agentes e 0% de probabilidade de desaceleração, executando ADOPT (acima), OptAPO (meio) e DPOP (abaixo).

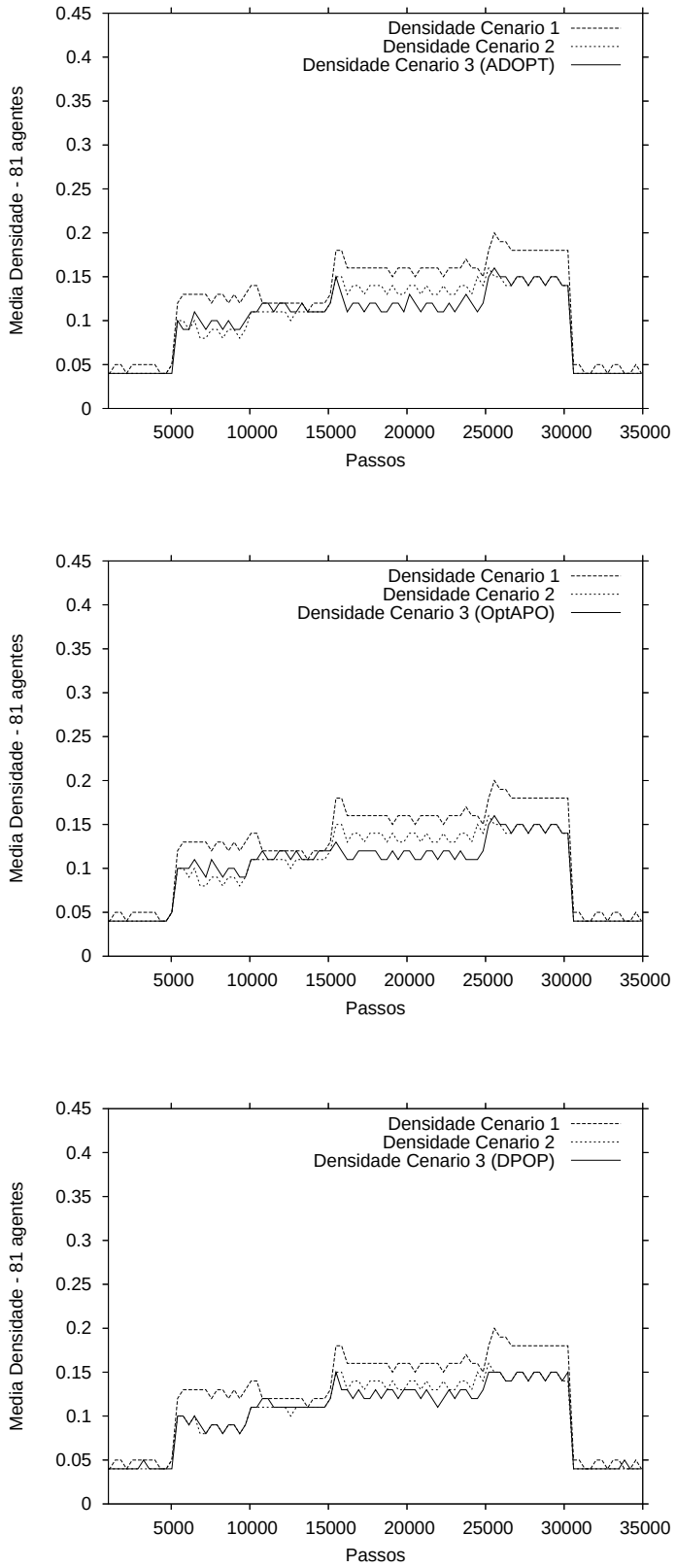


Figura 6.13: Densidade média das vias da rede no caso de alimentação dinâmica, 81 agentes e 0% de probabilidade de desaceleração, executando ADOPT (acima), OptAPO (meio) e DPOP (abaixo).

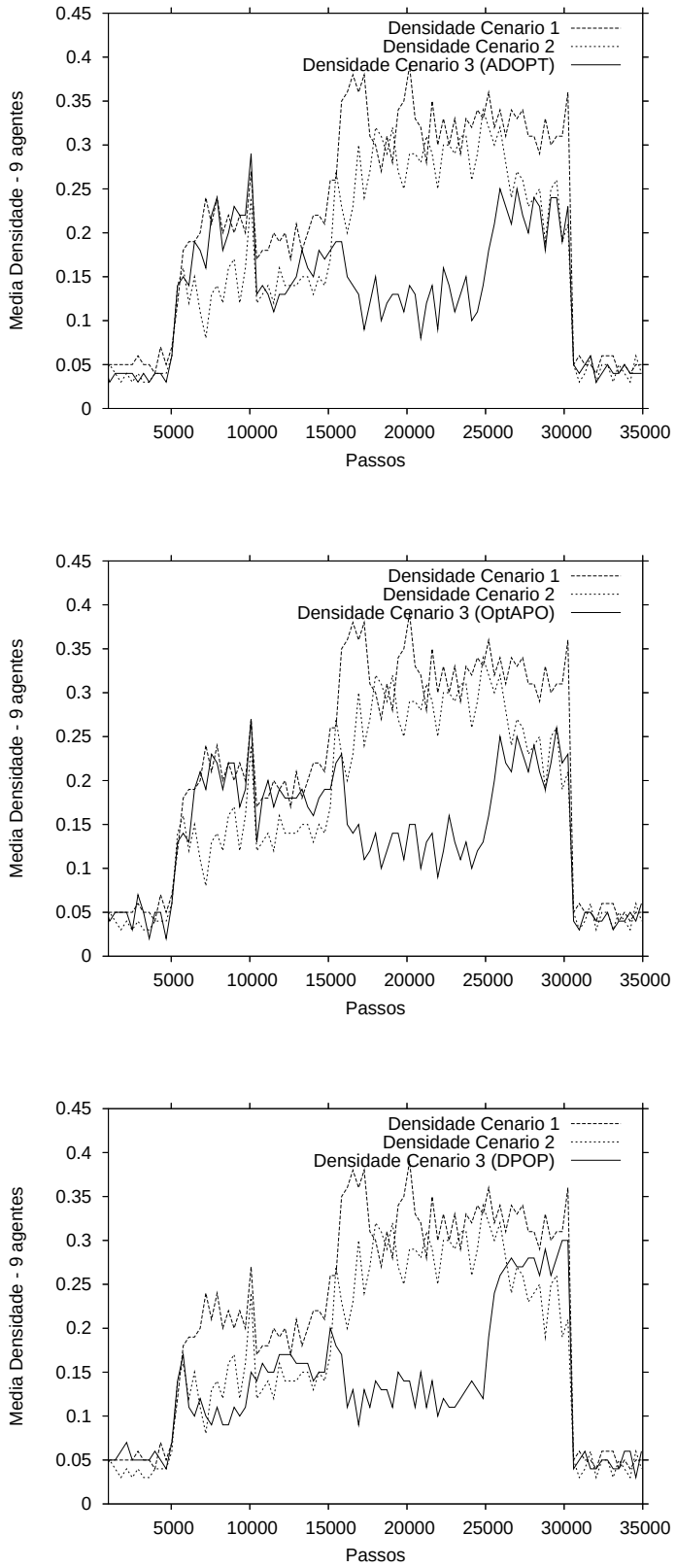


Figura 6.14: Densidade média das vias da rede no caso de alimentação dinâmica, 9 agentes e 5% de probabilidade de desaceleração, executando ADOPT (acima), OptAPO (meio) e DPOP (abaixo).

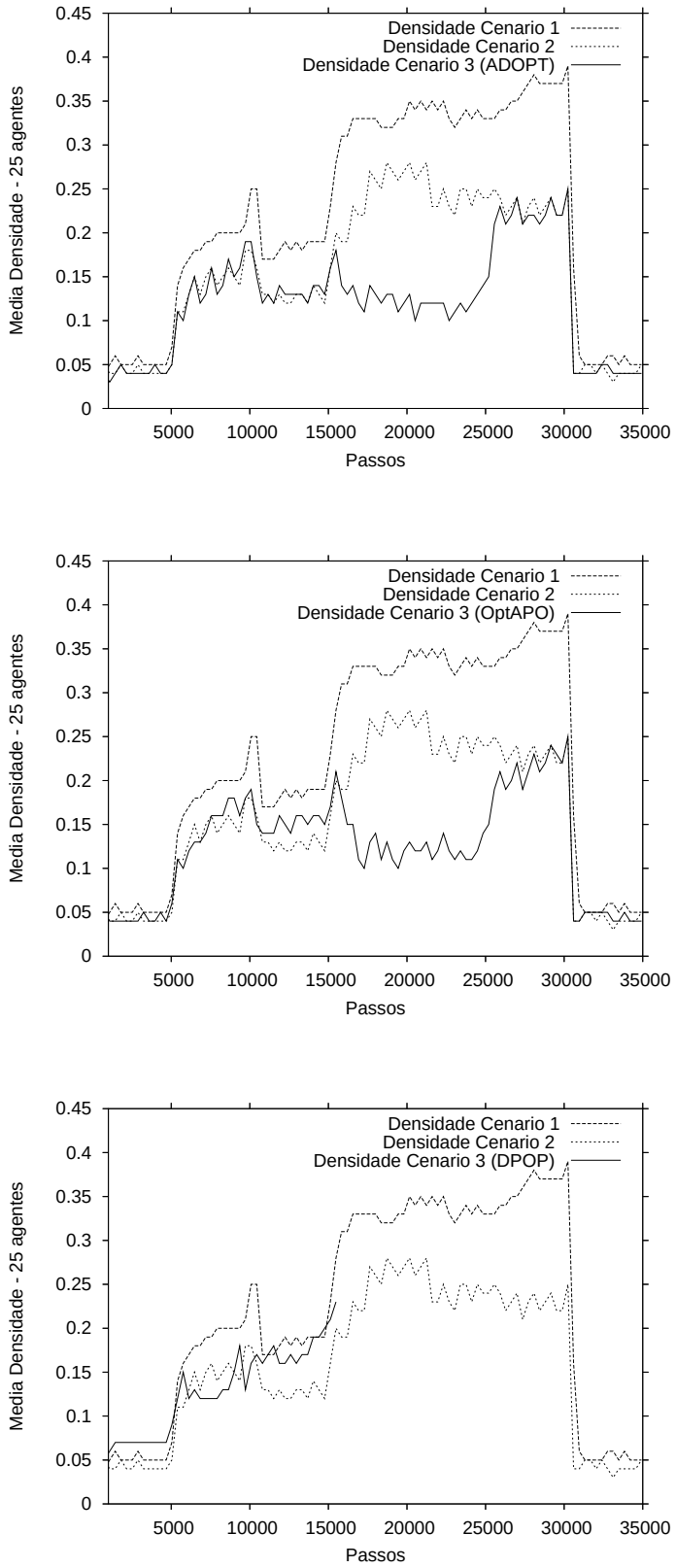


Figura 6.15: Densidade média das vias da rede no caso de alimentação dinâmica, 25 agentes e 5% de probabilidade de desaceleração, executando ADOPT (acima), OptAPO (meio) e DPOP (abaixo).

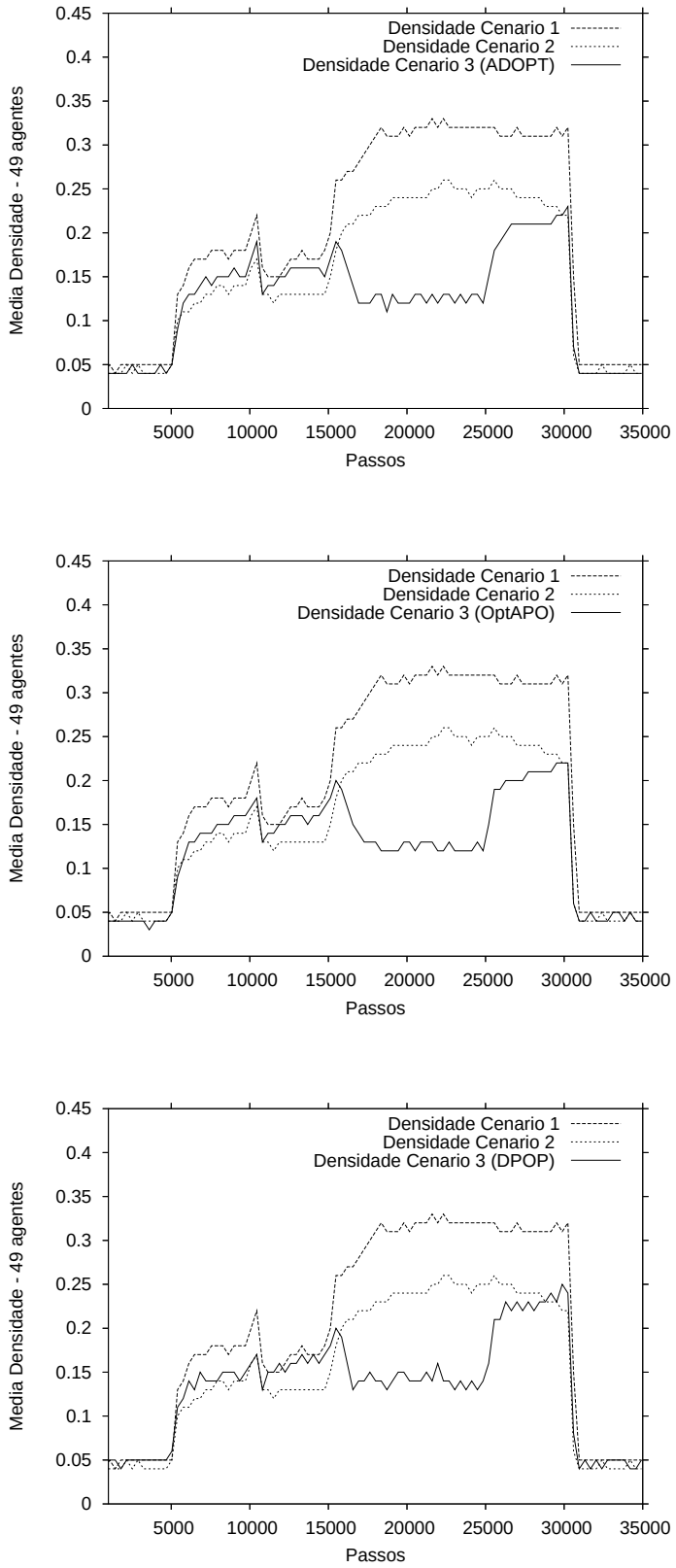


Figura 6.16: Densidade média das vias da rede no caso de alimentação dinâmica, 49 agentes e 5% de probabilidade de desaceleração, executando ADOPT (acima), OptAPO (meio) e DPOP (abaixo).

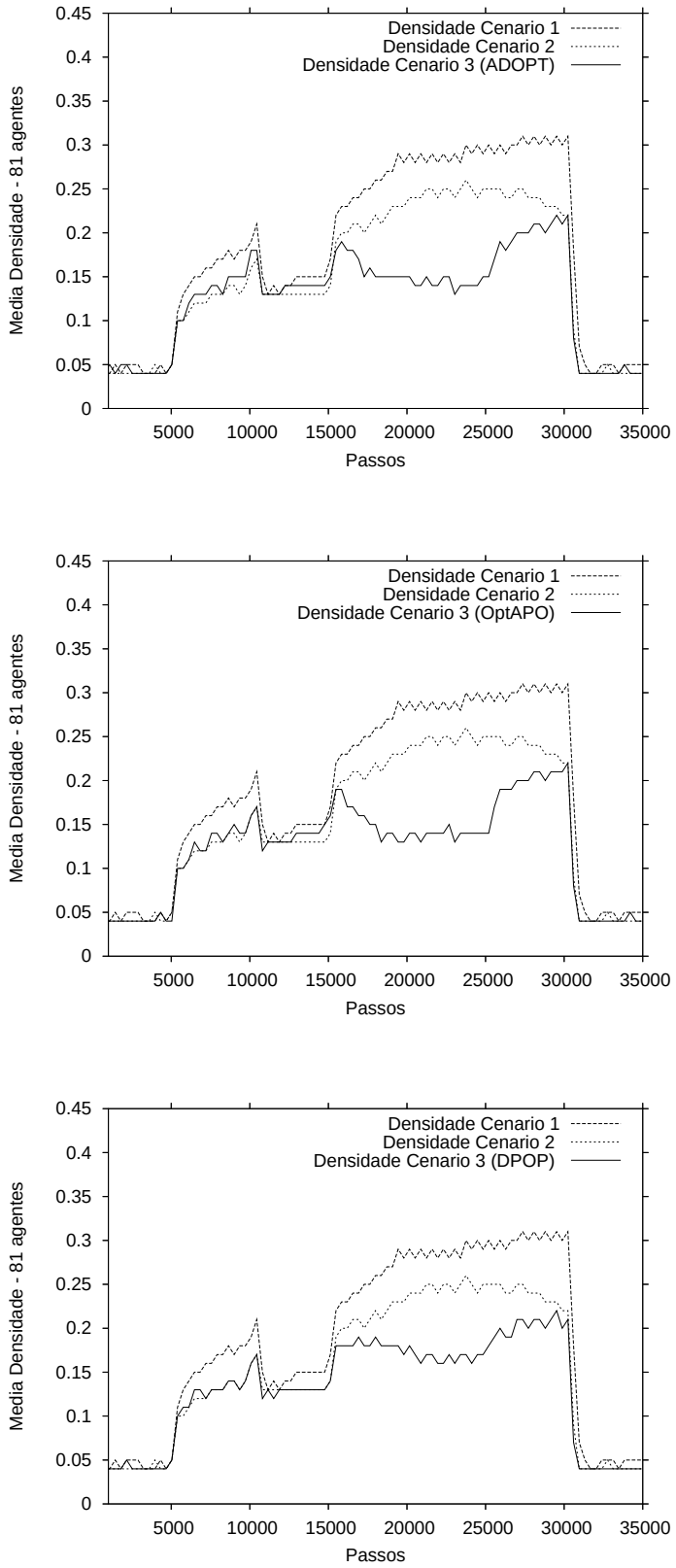


Figura 6.17: Densidade média das vias da rede no caso de alimentação dinâmica, 81 agentes e 5% de probabilidade de desaceleração, executando ADOPT (acima), OptAPO (meio) e DPOP (abaixo).

Tabela 6.12: Valores médios de veículos parados para a rede com 9 semáforos, alimentação dinâmica e probabilidade de desaceleração de 0%

Cenário	Madrugada	Manhã	Tarde	Hora do *Rush*
Cenário 1	0.96 (± 0.32)	2.53 (± 0.99)	4.14 (± 0.53)	4.46 (± 0.52)
Cenário 2	0.50 (± 0.39)	0.73 (± 0.44)	2.01 (± 0.40)	1.89 (± 0.40)
Cenário 3 (ADOPT)	0.41 (± 0.28)	0.66 (± 0.34)	1.75 (± 0.62)	0.99 (± 0.48)
Cenário 3 (OptAPO)	0.51 (± 0.30)	0.56 (± 0.36)	1.70 (± 0.64)	0.97 (± 0.46)
Cenário 3 (DPOP)	0.59 (± 0.42)	1.39 (± 0.76)	1.72 (± 0.91)	1.80 (± 0.59)

Tabela 6.13: Valores médios da densidade para a rede com 9 semáforos, alimentação dinâmica e probabilidade de desaceleração de 5%

Cenário	Madrugada	Manhã	Tarde	Hora do *Rush*
Cenário 1	0.05 (± 0.01)	0.19 (± 0.04)	0.20 (± 0.02)	0.32 (± 0.03)
Cenário 2	0.03 (± 0.01)	0.13 (± 0.03)	0.14 (± 0.03)	0.27 (± 0.03)
Cenário 3 (ADOPT)	0.03 (± 0.01)	0.18 (± 0.04)	0.15 (± 0.04)	0.13 (± 0.02)
Cenário 3 (OptAPO)	0.04 (± 0.01)	0.17 (± 0.04)	0.18 (± 0.03)	0.13 (± 0.03)
Cenário 3 (DPOP)	0.05 (± 0.01)	0.10 (± 0.02)	0.15 (± 0.01)	0.13 (± 0.02)

Tabela 6.14: Valores médios da velocidade para a rede com 9 semáforos, alimentação dinâmica e probabilidade de desaceleração de 5%

Cenário	Madrugada	Manhã	Tarde	Hora do *Rush*
Cenário 1	1.55 (± 0.36)	1.26 (± 0.22)	1.28 (± 0.22)	1.13 (± 0.17)
Cenário 2	1.81 (± 0.41)	1.63 (± 0.27)	1.69 (± 0.37)	1.27 (± 0.26)
Cenário 3 (ADOPT)	1.80 (± 0.31)	1.36 (± 0.34)	1.58 (± 0.32)	1.87 (± 0.31)
Cenário 3 (OptAPO)	1.89 (± 0.34)	1.37 (± 0.26)	1.48 (± 0.28)	1.89 (± 0.31)
Cenário 3 (DPOP)	1.53 (± 0.17)	1.54 (± 0.42)	1.27 (± 0.31)	1.39 (± 0.21)

Tabela 6.15: Valores médios de veículos parados para a rede com 9 semáforos, alimentação dinâmica e probabilidade de desaceleração de 5%

Cenário	Madrugada	Manhã	Tarde	Hora do *Rush*
Cenário 1	0.98 (± 0.30)	6.49 (± 2.26)	7.00 (± 0.99)	11.63 (± 1.8)
Cenário 2	0.42 (± 0.27)	3.17 (± 1.42)	3.79 (± 1.20)	8.97 (± 1.88)
Cenário 3 (ADOPT)	0.37 (± 0.28)	5.69 (± 2.65)	4.56 (± 2.01)	2.23 (± 0.99)
Cenário 3 (OptAPO)	0.38 (± 0.24)	5.52 (± 2.67)	5.89 (± 1.32)	2.35 (± 1.42)
Cenário 3 (DPOP)	1.02 (± 0.37)	2.52 (± 1.28)	4.67 (± 0.83)	2.81 (± 0.95)

Tabela 6.16: Valores médios da densidade para a rede com 25 semáforos, alimentação dinâmica e probabilidade de desaceleração de 0%

Cenário	Madrugada	Manhã	Tarde	Hora do *Rush*
Cenário 1	0.05 (± 0.01)	0.14 (± 0.02)	0.14 (± 0.01)	0.18 (± 0.01)
Cenário 2	0.04 (± 0.01)	0.08 (± 0.01)	0.10 (± 0.01)	0.13 (± 0.01)
Cenário 3 (ADOPT)	0.04 (± 0.01)	0.07 (± 0.01)	0.09 (± 0.01)	0.08 (± 0.01)
Cenário 3 (OptAPO)	0.04 (± 0.01)	0.07 (± 0.01)	0.09 (± 0.01)	0.08 (± 0.01)
Cenário 3 (DPOP)	0.06 (± 0.01)	0.11 (± 0.01)	0.14 (± 0.01)	0.12 (± 0.01)

Tabela 6.17: Valores médios da velocidade para a rede com 25 semáforos, alimentação dinâmica e probabilidade de desaceleração de 0%

Cenário	Madrugada	Manhã	Tarde	Hora do *Rush*
Cenário 1	1.52 (± 0.18)	1.59 (± 0.15)	1.53 (± 0.11)	1.76 (± 0.21)
Cenário 2	1.77 (± 0.24)	1.86 (± 0.22)	1.95 (± 0.24)	2.05 (± 0.24)
Cenário 3 (ADOPT)	1.79 (± 0.21)	1.88 (± 0.24)	1.94 (± 0.21)	1.99 (± 0.26)
Cenário 3 (OptAPO)	1.76 (± 0.22)	1.86 (± 0.20)	1.98 (± 0.24)	2.02 (± 0.26)
Cenário 3 (DPOP)	1.42 (± 0.31)	1.38 (± 0.22)	1.42 (± 0.19)	1.45 (± 0.23)

Tabela 6.18: Valores médios de veículos parados para a rede com 25 semáforos, alimentação dinâmica e probabilidade de desaceleração de 0%

Cenário	Madrugada	Manhã	Tarde	Hora do *Rush*
Cenário 1	0.89 (± 0.29)	3.72 (± 0.96)	3.75 (± 0.43)	4.25 (± 0.89)
Cenário 2	0.48 (± 0.39)	0.82 (± 0.38)	1.82 (± 0.75)	1.86 (± 0.71)
Cenário 3 (ADOPT)	0.41 (± 0.33)	0.64 (± 0.46)	1.52 (± 0.78)	1.00 (± 0.63)
Cenário 3 (OptAPO)	0.45 (± 0.40)	0.66 (± 0.34)	1.53 (± 0.83)	0.96 (± 0.62)
Cenário 3 (DPOP)	1.93 (± 0.73)	3.66 (± 0.51)	4.28 (± 0.78)	3.65 (± 0.52)

Tabela 6.19: Valores médios da densidade para a rede com 25 semáforos, alimentação dinâmica e probabilidade de desaceleração de 5%

Cenário	Madrugada	Manhã	Tarde	Hora do *Rush*
Cenário 1	0.05 (± 0.01)	0.17 (± 0.03)	0.19 (± 0.02)	0.32 (± 0.02)
Cenário 2	0.04 (± 0.02)	0.13 (± 0.03)	0.13 (± 0.01)	0.24 (± 0.03)
Cenário 3 (ADOPT)	0.04 (± 0.01)	0.13 (± 0.03)	0.13 (± 0.01)	0.12 (± 0.01)
Cenário 3 (OptAPO)	0.04 (± 0.01)	0.14 (± 0.03)	0.15 (± 0.01)	0.12 (± 0.02)
Cenário 3 (DPOP)	0.06 (± 0.01)	0.12 (± 0.02)	0.17 (± 0.01)	0.22 (± 0.01)

Tabela 6.20: Valores médios da velocidade para a rede com 25 semáforos, alimentação dinâmica e probabilidade de desaceleração de 5%

Cenário	Madrugada	Manhã	Tarde	Hora do *Rush*
Cenário 1	1.59 (± 0.21)	1.43 (± 0.14)	1.30 (± 0.10)	1.19 (± 0.09)
Cenário 2	1.79 (± 0.21)	1.48 (± 0.24)	1.72 (± 0.21)	1.43 (± 0.22)
Cenário 3 (ADOPT)	1.76 (± 0.23)	1.53 (± 0.22)	1.66 (± 0.17)	1.89 (± 0.23)
Cenário 3 (OptAPO)	1.75 (± 0.20)	1.50 (± 0.26)	1.61 (± 0.21)	1.85 (± 0.18)
Cenário 3 (DPOP)	1.43 (± 0.28)	1.32 (± 0.16)	1.23 (± 0.15)	1.26 (± 0.12)

Tabela 6.21: Valores médios de veículos parados para a rede com 25 semáforos, alimentação dinâmica e probabilidade de desaceleração de 5%

Cenário	Madrugada	Manhã	Tarde	Hora do *Rush*
Cenário 1	0.78 (± 0.27)	5.51 (± 1.76)	6.25 (± 1.04)	11.41 (± 1.34)
Cenário 2	0.40 (± 0.31)	3.42 (± 1.41)	3.18 (± 0.95)	7.10 (± 1.71)
Cenário 3 (ADOPT)	0.42 (± 0.24)	3.49 (± 1.59)	3.39 (± 1.06)	1.92 (± 0.83)
Cenário 3 (OptAPO)	0.44 (± 0.30)	3.68 (± 1.89)	4.39 (± 1.02)	2.11 (± 1.11)
Cenário 3 (DPOP)	1.95 (± 0.66)	4.36 (± 0.77)	5.76 (± 0.72)	7.41 (± 0.54)

Tabela 6.22: Valores médios da densidade para a rede com 49 semáforos, alimentação dinâmica e probabilidade de desaceleração de 0%

Cenário	Madrugada	Manhã	Tarde	Hora do *Rush*
Cenário 1	0.04 (± 0.01)	0.12 (± 0.02)	0.13 (± 0.01)	0.17 (± 0.01)
Cenário 2	0.04 (± 0.01)	0.08 (± 0.01)	0.11 (± 0.01)	0.13 (± 0.01)
Cenário 3 (ADOPT)	0.04 (± 0.01)	0.07 (± 0.01)	0.09 (± 0.01)	0.08 (± 0.01)
Cenário 3 (OptAPO)	0.04 (± 0.01)	0.07 (± 0.01)	0.09 (± 0.01)	0.08 (± 0.01)
Cenário 3 (DPOP)	0.04 (± 0.01)	0.08 (± 0.01)	0.09 (± 0.01)	0.09 (± 0.01)

Tabela 6.23: Valores médios da velocidade para a rede com 49 semáforos, alimentação dinâmica e probabilidade de desaceleração de 0%

Cenário	Madrugada	Manhã	Tarde	Hora do *Rush*
Cenário 1	1.42 (± 0.21)	1.46 (± 0.09)	1.49 (± 0.08)	1.67 (± 0.12)
Cenário 2	1.80 (± 0.24)	1.99 (± 0.25)	1.92 (± 0.11)	2.02 (± 0.20)
Cenário 3 (ADOPT)	1.79 (± 0.28)	1.97 (± 0.23)	1.87 (± 0.13)	2.02 (± 0.26)
Cenário 3 (OptAPO)	1.74 (± 0.26)	1.95 (± 0.26)	1.92 (± 0.14)	2.00 (± 0.26)
Cenário 3 (DPOP)	1.49 (± 0.23)	1.77 (± 0.15)	1.68 (± 0.10)	1.70 (± 0.09)

Tabela 6.24: Valores médios de veículos parados para a rede com 49 semáforos, alimentação dinâmica e probabilidade de desaceleração de 0%

Cenário	Madrugada	Manhã	Tarde	Hora do *Rush*
Cenário 1	0.88 (± 0.27)	3.16 (± 0.68)	3.34 (± 0.24)	4.26 (± 0.44)
Cenário 2	0.42 (± 0.30)	0.60 (± 0.28)	2.10 (± 0.52)	1.98 (± 0.55)
Cenário 3 (ADOPT)	0.43 (± 0.30)	0.55 (± 0.30)	1.75 (± 0.44)	0.96 (± 0.43)
Cenário 3 (OptAPO)	0.40 (± 0.28)	0.59 (± 0.29)	1.79 (± 0.41)	0.92 (± 0.43)
Cenário 3 (DPOP)	0.75 (± 0.27)	1.29 (± 0.37)	1.78 (± 0.38)	1.72 (± 0.32)

Tabela 6.25: Valores médios da densidade para a rede com 49 semáforos, alimentação dinâmica e probabilidade de desaceleração de 5%

Cenário	Madrugada	Manhã	Tarde	Hora do *Rush*
Cenário 1	0.04 (± 0.01)	0.16 (± 0.03)	0.17 (± 0.01)	0.30 (± 0.02)
Cenário 2	0.04 (± 0.01)	0.12 (± 0.02)	0.13 (± 0.01)	0.23 (± 0.02)
Cenário 3 (ADOPT)	0.04 (± 0.01)	0.13 (± 0.02)	0.15 (± 0.01)	0.13 (± 0.01)
Cenário 3 (OptAPO)	0.03 (± 0.01)	0.13 (± 0.03)	0.15 (± 0.01)	0.13 (± 0.02)
Cenário 3 (DPOP)	0.04 (± 0.01)	0.13 (± 0.02)	0.15 (± 0.01)	0.14 (± 0.01)

Tabela 6.26: Valores médios da velocidade para a rede com 49 semáforos, alimentação dinâmica e probabilidade de desaceleração de 5%

Cenário	Madrugada	Manhã	Tarde	Hora do *Rush*
Cenário 1	1.45 (± 0.20)	1.29 (± 0.10)	1.29 (± 0.10)	1.17 (± 0.09)
Cenário 2	1.85 (± 0.22)	1.62 (± 0.28)	1.62 (± 0.12)	1.46 (± 0.16)
Cenário 3 (ADOPT)	1.77 (± 0.22)	1.57 (± 0.25)	1.54 (± 0.11)	1.85 (± 0.16)
Cenário 3 (OptAPO)	1.77 (± 0.17)	1.58 (± 0.27)	1.58 (± 0.15)	1.84 (± 0.17)
Cenário 3 (DPOP)	1.63 (± 0.27)	1.55 (± 0.07)	1.56 (± 0.13)	1.65 (± 0.14)

Tabela 6.27: Valores médios de veículos parados para a rede com 49 semáforos, alimentação dinâmica e probabilidade de desaceleração de 5%

Cenário	Madrugada	Manhã	Tarde	Hora do *Rush*
Cenário 1	0.82 (± 0.22)	5.04 (± 1.49)	5.42 (± 0.73)	10.69 (± 1.46)
Cenário 2	0.38 (± 0.24)	2.76 (± 1.20)	3.25 (± 0.54)	6.67 (± 1.24)
Cenário 3 (ADOPT)	0.44 (± 0.27)	3.36 (± 1.44)	4.41 (± 0.75)	2.31 (± 0.98)
Cenário 3 (OptAPO)	0.43 (± 0.24)	3.28 (± 1.49)	4.44 (± 0.67)	2.42 (± 1.08)
Cenário 3 (DPOP)	0.69 (± 0.34)	3.61 (± 1.04)	4.77 (± 0.62)	3.50 (± 0.77)

Tabela 6.28: Valores médios da densidade para a rede com 81 semáforos, alimentação dinâmica e probabilidade de desaceleração de 0%

Cenário	Madrugada	Manhã	Tarde	Hora do *Rush*
Cenário 1	0.04 (± 0.01)	0.12 (± 0.02)	0.12 (± 0.01)	0.15 (± 0.01)
Cenário 2	0.04 (± 0.01)	0.08 (± 0.01)	0.10 (± 0.01)	0.13 (± 0.01)
Cenário 3 (ADOPT)	0.04 (± 0.01)	0.09 (± 0.01)	0.11 (± 0.01)	0.11 (± 0.01)
Cenário 3 (OptAPO)	0.04 (± 0.01)	0.09 (± 0.01)	0.11 (± 0.01)	0.11 (± 0.01)
Cenário 3 (DPOP)	0.04 (± 0.01)	0.08 (± 0.01)	0.11 (± 0.01)	0.12 (± 0.01)

Tabela 6.29: Valores médios da velocidade para a rede com 81 semáforos, alimentação dinâmica e probabilidade de desaceleração de 0%

Cenário	Madrugada	Manhã	Tarde	Hora do *Rush*
Cenário 1	1.37 (± 0.23)	1.46 (± 0.10)	1.43 (± 0.12)	1.66 (± 0.05)
Cenário 2	1.71 (± 0.23)	1.93 (± 0.23)	1.92 (± 0.18)	2.02 (± 0.20)
Cenário 3 (ADOPT)	1.72 (± 0.22)	1.90 (± 0.24)	1.90 (± 0.17)	2.04 (± 0.22)
Cenário 3 (OptAPO)	1.72 (± 0.23)	1.87 (± 0.25)	1.88 (± 0.21)	2.03 (± 0.22)
Cenário 3 (DPOP)	1.68 (± 0.24)	1.91 (± 0.23)	1.92 (± 0.16)	2.03 (± 0.22)

Tabela 6.30: Valores médios de veículos parados para a rede com 81 semáforos, alimentação dinâmica e probabilidade de desaceleração de 0%

Cenário	Madrugada	Manhã	Tarde	Hora do *Rush*
Cenário 1	0.80 (± 0.24)	2.94 (± 0.66)	3.08 (± 0.35)	3.80 (± 0.35)
Cenário 2	0.47 (± 0.31)	0.72 (± 0.33)	1.97 (± 0.48)	1.95 (± 0.36)
Cenário 3 (ADOPT)	0.48 (± 0.35)	0.93 (± 0.39)	2.16 (± 0.44)	1.52 (± 0.39)
Cenário 3 (OptAPO)	0.47 (± 0.33)	1.04 (± 0.42)	2.17 (± 0.43)	1.51 (± 0.44)
Cenário 3 (DPOP)	0.47 (± 0.31)	0.80 (± 0.37)	2.08 (± 0.45)	1.69 (± 0.41)

Tabela 6.31: Valores médios da densidade para a rede com 81 semáforos, alimentação dinâmica e probabilidade de desaceleração de 5%

Cenário	Madrugada	Manhã	Tarde	Hora do *Rush*
Cenário 1	0.04 (± 0.01)	0.15 (± 0.03)	0.15 (± 0.02)	0.26 (± 0.03)
Cenário 2	0.04 (± 0.01)	0.11 (± 0.02)	0.13 (± 0.01)	0.22 (± 0.02)
Cenário 3 (ADOPT)	0.04 (± 0.01)	0.12 (± 0.02)	0.14 (± 0.01)	0.15 (± 0.01)
Cenário 3 (OptAPO)	0.04 (± 0.01)	0.12 (± 0.02)	0.13 (± 0.01)	0.14 (± 0.01)
Cenário 3 (DPOP)	0.04 (± 0.01)	0.12 (± 0.02)	0.13 (± 0.01)	0.17 (± 0.01)

Tabela 6.32: Valores médios da velocidade para a rede com 81 semáforos, alimentação dinâmica e probabilidade de desaceleração de 5%

Cenário	Madrugada	Manhã	Tarde	Hora do *Rush*
Cenário 1	1.39 (± 0.24)	1.25 (± 0.11)	1.25 (± 0.05)	1.18 (± 0.08)
Cenário 2	1.73 (± 0.17)	1.59 (± 0.24)	1.65 (± 0.15)	1.46 (± 0.13)
Cenário 3 (ADOPT)	1.74 (± 0.23)	1.56 (± 0.26)	1.62 (± 0.15)	1.77 (± 0.14)
Cenário 3 (OptAPO)	1.72 (± 0.21)	1.58 (± 0.24)	1.65 (± 0.14)	1.80 (± 0.16)
Cenário 3 (DPOP)	1.75 (± 0.20)	1.58 (± 0.24)	1.64 (± 0.15)	1.68 (± 0.15)

Tabela 6.33: Valores médios de veículos parados para a rede com 81 semáforos, alimentação dinâmica e probabilidade de desaceleração de 5%

Cenário	Madrugada	Manhã	Tarde	Hora do *Rush*
Cenário 1	0.76 ($\pm$ 0.24)	4.56 ($\pm$ 1.45)	4.77 ($\pm$ 0.83)	9.11 ($\pm$ 1.36)
Cenário 2	0.43 ($\pm$ 0.28)	2.62 ($\pm$ 1.04)	3.32 ($\pm$ 0.54)	6.31 ($\pm$ 1.21)
Cenário 3 (ADOPT)	0.42 ($\pm$ 0.27)	2.98 ($\pm$ 1.22)	3.69 ($\pm$ 0.67)	2.92 ($\pm$ 0.69)
Cenário 3 (OptAPO)	0.45 ($\pm$ 0.27)	2.72 ($\pm$ 1.12)	3.58 ($\pm$ 0.55)	2.71 ($\pm$ 0.80)
Cenário 3 (DPOP)	0.43 ($\pm$ 0.27)	2.71 ($\pm$ 1.02)	3.20 ($\pm$ 0.54)	3.77 ($\pm$ 0.58)

6.1.3 Grupos de Coordenação

Outra questão relevante na avaliação da rede é a ocorrência de grupos de coordenação. Um grupo de coordenação, nesse caso, é caracterizado pelo sequenciamento de agentes vizinhos executando o mesmo plano semafórico. Ou seja estão coordenando suas ações de sincronização. O número de grupos de coordenação, nas vias verticais ou nas vias horizontais, bem como o tamanho desses grupos (frente à dimensão da rede), serve de métrica para avaliar se o controle está coerente com o tráfego.

A tabela 6.34 mostra a adequação do controle para o caso de alimentação estática (seção 5.5.1.1). Nessa tabela são considerados grupos de coordenação de tamanho maior que a metade do tamanho das vias na rede (em termos de número de cruzamentos), para cada caso. Por exemplo, na rede com 25 semáforos, o tamanho das vias é de 5 semáforos, e portanto são considerados os grupos com no mínimo 3 semáforos. A tabela exibe a média do número de grupos, o tamanho médio dos grupos e o tamanho máximo de grupo verificado (nesse caso o limite é o tamanho da via) para as vias com orientação horizontal (LO) e orientação vertical (NS). Percebe-se a adequação do controle, pela formação dos grupos de coordenação, ou seja os semáforos estão sincronizando, pela ocorrência de grupos de coordenação em número próximo à quantidade de vias na rede e pelo tamanho médio dos grupos acima do tamanho médio de semáforos em cada via.

Tabela 6.34: Grupos de Sincronização: média do número de grupos e tamanho de grupos, e tamanho máximo de grupos

	No. Grupos		Tamanho Grupos		Tamanho Máximo	
	LO	NS	LO	NS	LO	NS
ADOPT 25	3.25	3.75	3.25	3.50	4	4
OptAPO 25	3.75	3.75	3.25	3.25	4	4
DPOP 25	3.00	3.25	3.0	2.75	4	3
ADOPT 49	6.0	4.75	4.5	4.25	5	5
OptAPO 49	6.25	5.50	4.25	4.50	5	5
DPOP 49	5.25	5.00	3.25	3.50	5	4
ADOPT 81	7	7.5	5.5	5.75	7	6
OptAPO 81	7	7.25	5.75	5.25	7	6
DPOP 81	6.75	7	5.25	5.0	7	6

6.2 Avaliação Computacional dos Algoritmos DCOP

Foi possível perceber as características de desempenho particulares de cada algoritmo, refletidas na leitura das métricas apresentadas na seção 4.7. Essa leitura se reflete em nível de tempo de execução, número de mensagens trocadas e número de ciclos. Os valores médios foram obtidos das simulações do caso de alimentação dinâmica. Vale ressaltar que a implementação do algoritmo OptAPO não prevê uma arquitetura totalmente distribuída, como é feito nos demais algoritmos, e portanto isso beneficia o tempo computacional pois não há o *overhead* causado pelo processamento de rede necessário para a troca de mensagens entre as *threads*.

As tabelas 6.35, 6.37, 6.39 e 6.41 mostram as médias de tempo de execução e número de ciclos. Nas tabelas 6.36, 6.38, 6.40 e 6.42 são exibidos os valores do tamanho e quantidade de mensagens trocadas, incluindo os totais de tamanho de banda necessária para execução dos algoritmos (diz respeito às trocas de mensagens envolvidas no processamento da solução). Observa-se que, apesar da redução no número de mensagens necessárias, do algoritmo OptAPO para o DPOP, no total o tamanho médio das mensagens tem peso negativo contra o DPOP. O algoritmo ADOPT apresentou os valores mais altos para tempo de execução e número de ciclos.

Tabela 6.35: Valores médios das métricas DCOP: 9 agentes

Alg.	Tempo(s)	No. de Ciclos
ADOPT	0.65 ($\pm$0.15)	2186.13 ($\pm$648.03)
OptAPO	0.16 ($\pm$0.39)	29.08 ($\pm$23.20)
DPOP	0.07 ($\pm$0.01)	18

Tabela 6.36: Mensagens para DCOP: 9 agentes

Alg.	No. de Msgs	Tam. Msgs(bytes)	Tam. Total das Msgs(MB)
ADOPT	298.25 ($\pm$63.37)	47.39 ($\pm$14.90)	13.80
OptAPO	203.58 ($\pm$153.05)	6.80 ($\pm$1.98)	1.35
DPOP	36	55.54 ($\pm$2.66)	1.95

Tabela 6.37: Valores médios das métricas DCOP: 25 agentes

Alg.	Tempo(s)	No. de Ciclos
ADOPT	16.07 ($\pm$2.29)	6524.81 ($\pm$1095.12)
OptAPO	3.90 ($\pm$1.11)	56.54 ($\pm$15.76)
DPOP	0.22 ($\pm$0.02)	50

O principal ponto negativo do algoritmo ADOPT se encontra no número de ciclos e mensagens necessárias para o processamento da otimização. O algoritmo DPOP não enfrenta o mesmo problema, porém as mensagens trocadas entre os agentes são muito maiores. Elimina-se a necessidade maior de troca de mensagens pois cada mensagem contém mais informação que não precisa ser repetidamente transmitida. Porém há uma dependência maior, no caso de uma aplicação real, na questão da largura de banda disponível.

O algoritmo OptAPO representa uma boa solução DCOP em termos de tempo de execução e comunicação. No entanto, para as redes com um maior número de agentes,

Tabela 6.38: Mensagens para DCOP: 25 agentes

Alg.	No. de Msgs	Tam. Msgs(bytes)	Tam. Total das Msgs(MB)
ADOPT	4123.70 (±2130.45)	72.68 (±12.73)	292.68
OptAPO	766.36 (±141.46)	25.22 (±6.44)	18.87
DPOP	112	295.10 (±3.80)	33.05

Tabela 6.39: Valores médios das métricas DCOP: 49 agentes

Alg.	Tempo(s)	No. de Ciclos
ADOPT	112.90 (±33.39)	12387.84 (±3098.56)
OptAPO	21.16 (±17.53)	75.33 (±14.02)
DPOP	13.59 (±0.01)	98

onde o número de restrições é muito maior assim como seus custos (em função de taxas de inserção mais altas), o processo de mediação se torna mais freqüente. Essa mediação implica uma centralização parcial do problema, o que contraria o objetivo de distribuição do problema. Esse processo de mediação teria sido mais visível nos resultados se estivéssemos usando de fato um ambiente distribuído, onde os custos da troca de mensagens nesses momentos afetaria mais significativamente o tempo de execução.

Com o aumento do número de agentes envolvidos na computação é possível perceber que o algoritmo DPOP teve um melhor desempenho no que diz respeito a tempo de execução (ver figura 6.18) e número de ciclos. O mesmo não vale para o tamanho das mensagens, onde os demais algoritmos mantiveram valores muito mais baixos, com aumento apenas no número de mensagens trocadas. Na totalização das mensagens, o tamanho das mensagens foi desfavorável na avaliação do DPOP (ver figuras 6.19, 6.20 e 6.21). DPOP teria um desempenho muito mais significativo em relação aos demais algoritmos caso pudesse manter o baixo número de mensagens e tempo de execução e, diminuir o tamanho das mensagens trocadas. Esse tipo de melhoria no uso de memória é proposta em (PETCU; FALTINGS, 2007).

6.3 Considerações Finais

Foi possível observar uma atuação positiva dos algoritmos DCOP frente aos demais cenários simulados. Ainda assim, deve-se apontar para um futuro trabalho de aplicação em um cenário mais complexo, ou seja, com mais interseções, vias e assimetria na geometria dos cruzamentos.

Quanto à computação dos algoritmos DCOP, verificou-se a eficiência previamente relatada pelos trabalhos estudados, na seção 2.9. E assim fica visível a característica de adaptabilidade da solução DCOP quando se observa os valores da situação de alimentação dinâmica. Isso tem um valor muito positivo, visto que, tipicamente o comportamento de uma malha viária não é fixo, e sim variável conforme as horas do dia e em função de fatores externos. Além disso vale destacar que todas as soluções cabem em uma aplicação real no que diz respeito ao tempo de execução, ou seja, dentro do tempo de atuação proposto para os agentes (nesse caso 720 segundos) há espaço para execução do controle DCOP proposto, exceto no cenário com ADOPT para 81 agentes.

Tabela 6.40: Mensagens para DCOP: 49 agentes

Alg.	No. de Msgs	Tam. Msgs(bytes)	Tam. Total das Msgs(MB)
ADOPT	15972.91 (±4503.92)	86.08 (±13.02)	1342.72
OptAPO	1202.16 (±202.07)	57.23 (±5.51)	67.18
DPOP	228	19451.22 (±222.26)	4330.93

Tabela 6.41: Valores médios das métricas DCOP: 81 agentes

Alg.	Tempo(s)	No. de Ciclos
ADOPT	6727.46 (±182.98)	325512.42 (±10108.67)
OptAPO	295.75 (±31.56)	174.30 (±18.41)
DPOP	180.71 (±15.35)	152.00

Tabela 6.42: Mensagens para DCOP: 81 agentes

Alg.	No. de Msgs	Tam. Msgs(bytes)	Tam. Total das Msgs(MB)
ADOPT	358962.40 (±1690.40)	125.40 (±20.79)	43958.27
OptAPO	4015.66 (±68.70)	93.45 (±8.15)	366.46
DPOP	322	69286.78 (±721.12)	21787.44

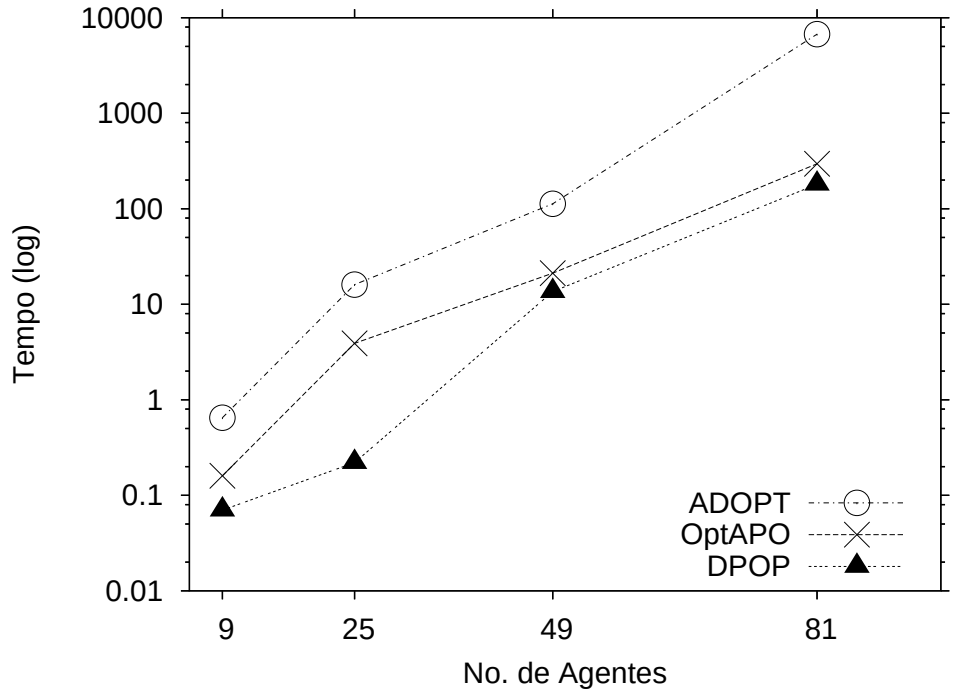


Figura 6.18: Tempo de execução dos algoritmos para 9, 25, 49 e 81 agentes.

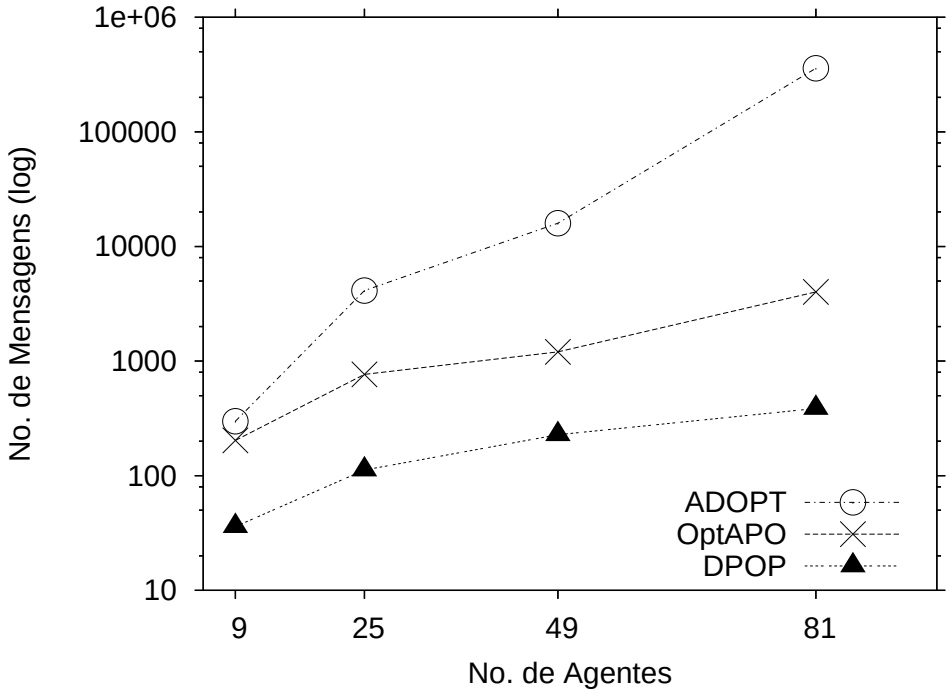


Figura 6.19: Número de mensagens dos algoritmos para 9, 25, 49 e 81 agentes.

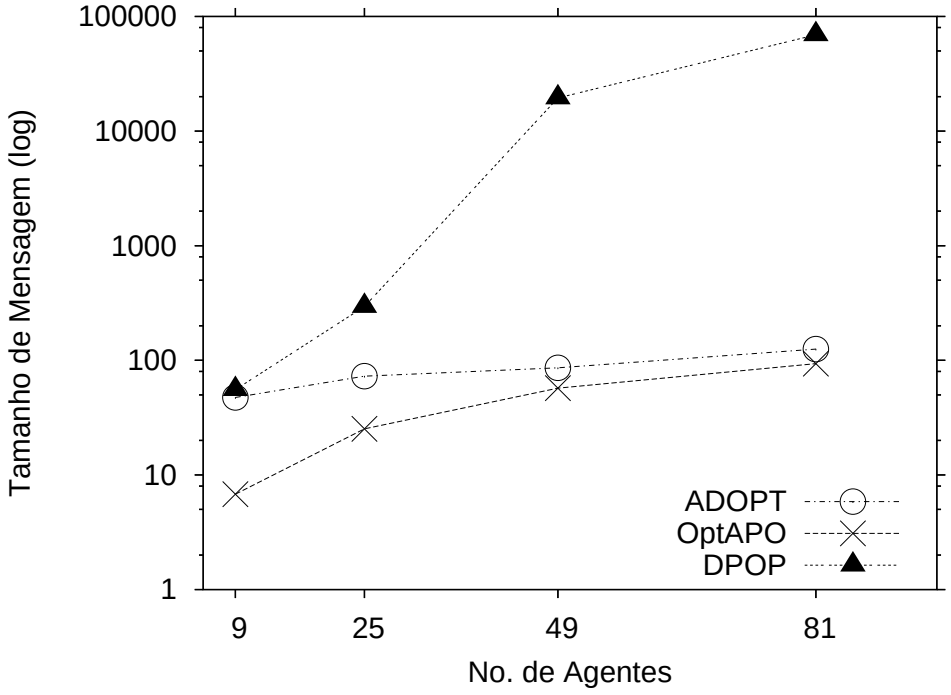


Figura 6.20: Tamanho das mensagens dos algoritmos para 9, 25, 49 e 81 agentes.

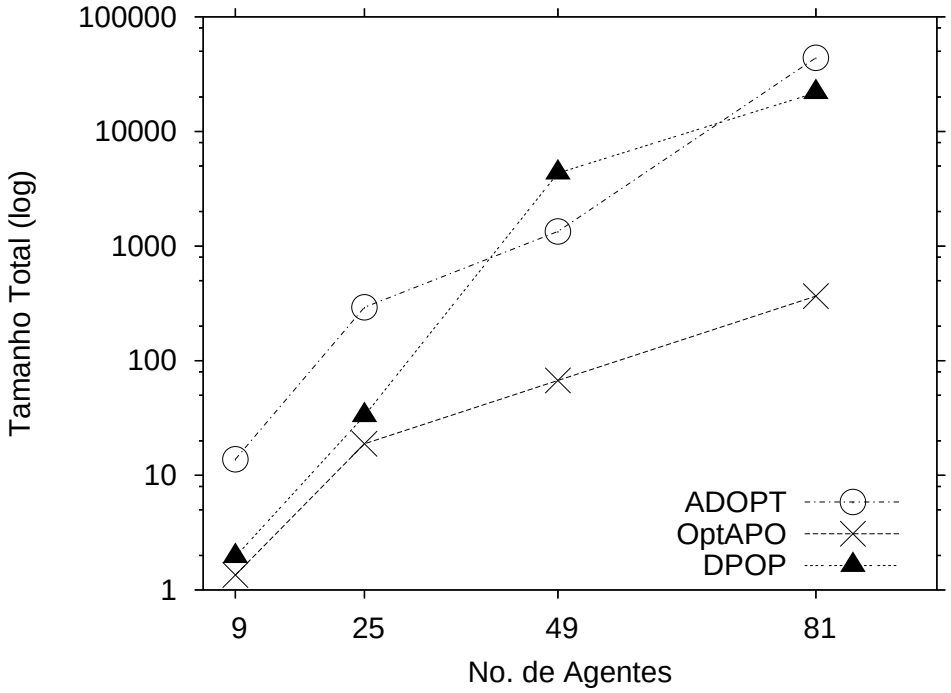


Figura 6.21: Tamanho total das mensagens dos algoritmos para 9, 25, 49 e 81 agentes.

7 CONCLUSÕES E TRABALHOS FUTUROS

Este trabalho traçou seus objetivos no sentido de abordar o problema de controle semafórico através de um modelo formal amplamente estudado: o dos problemas de otimização de restrições distribuídas. De grande interesse da comunidade científica na área de inteligência artificial, esse tipo de problema remete a uma modelagem focada na dinâmica (do tráfego) e que visa fornecer adaptabilidade aliada a eficiência de controle. Isso permite estabelecer um caminho para aplicação real dessa abordagem para solução de problemas reais e em cenários reais.

Dessa forma, como principais contribuições desse trabalho podem ser elencadas: a aplicação do formalismo DCOP ao problema de controle semafórico, através da proposta de modelagem apresentada; o estudo de DCOP, em nível de eficiência de controle e de execução no caso de um problema real e comum de tráfego; e a comparação dos algoritmos mais populares em DCOP nesse cenário proposto.

Embora seja sabido que os algoritmos estudados nesse trabalho garantem a solução ótima do problema, ainda assim é válida a comparação por se tratar da aplicação em um problema real. Além disto, tal comparação nunca havia sido realizada em ambientes dinâmicos onde existe a necessidade de que os algoritmos se adaptem às mudanças do ambiente. Foi possível observar uma atuação de forma positiva dos algoritmos DCOP frente aos demais cenários propostos.

Quanto à eficiência computacional dos algoritmos DCOP, percebe-se uma relação de melhoria nas questões de tempo computacional e complexidade imposta pela resolução. Vale lembrar que, em uma aplicação real, os tempos verificados na simulação sofrerão um acréscimo impulsionado pelo custo dos protocolos de rede e da transmissão em si.

Quanto à questão de eficácia dos algoritmos frente ao problema real proposto, fica visível a característica de adaptabilidade dos algoritmos de DCOP, no caso de mudança nos volumes de tráfego. Este é um ponto importante visto que, tipicamente, o comportamento de uma malha viária não é fixo, e sim variável conforme as horas do dia e em função de fatores como acidentes e condições meteorológicas.

No entanto, trabalhos futuros podem e devem ser propostos, visando uma sintonia do modelo adotado e da sua direta aplicação a um cenário real, pelo uso, ainda, de ferramentas de simulação como o ITSUMO. Nesse sentido a idéia é modelar um cenário real, ou seja, uma topologia real encontrada em alguma cidade, e a partir disso, inferir a complexidade necessária ao modelo e à solução, para que se façam valer as características de dinamismo exploradas.

Também são propostas outras formas de modelar os pesos do equacionamento do modelo, permitindo flexibilizar o controle pelo uso de diferentes valores de custo para as combinações de planos semafóricos. Não considerar apenas critérios de volume de tráfego e coordenação de semáforos, mas também imposições que se façam necessárias a

um engenheiro de tráfego, como por exemplo a eventual necessidade de desviar o tráfego em determinado momento para uma via específica. Isso se faz interessante em situações de acidentes ou imprevistos que possam ser contornados ou minimizados pelo controle do fluxo de veículos.

Outra proposta é a implementação totalmente distribuída do algoritmo OptAPO, que para esse trabalho foi implementado de uma forma centralizada, sem que processos independentes, efetivamente, fizessem os papéis de agente. Essa implementação distribuída daria uma idéia melhor dos custos de rede para a troca de mensagens, e serviria também, num contexto maior, para que se fizessem experimentos futuros onde os agentes estivessem de fato separados fisicamente em máquinas para simular a rede no cenário de tráfego proposto.

Também é considerada a reprodução de outras formas de controle já apresentadas na literatura de controle de tráfego nesses cenários estudados e nos cenários propostos futuramente. Abordagens como a de aprendizado por reforço podem contribuir no sentido de comparar a aplicação desses algoritmos.

REFERÊNCIAS

BAZZAN, A. L. C. A Distributed Approach for Coordination of Traffic Signal Agents. **Autonomous Agents and Multiagent Systems**, [S.l.], v.10, n.1, p.131–164, March 2005.

DAVIN, J.; MODI, P. J. Impact of Problem Centralization in Distributed Constraint Optimization Algorithms. In: INTERNATIONAL JOINT CONFERENCE ON AUTONOMOUS AGENTS AND MULTIAGENT SYSTEMS, 4., 2005. **Proceedings...** New York: ACM Press, 2005. p.1057–1066.

FREUDER, E. C.; WALLACE, R. J. Partial constraint satisfaction. **Artificial Intelligence**, [S.l.], v.58, n.1–3, p.21–70, 1992.

GOLOMB, S. W.; BAUMERT, L. D. Backtrack Programming. **J. ACM**, [S.l.], v.12, n.4, p.516–524, 1965.

HIRAYAMA, K.; YOKOO, M. Distributed Partial Constraint Satisfaction Problem. In: INTERNATIONAL CONFERENCE ON PRINCIPLES AND PRACTICE OF CONSTRAINT PROGRAMMING, CP, 3., 1997. **Proceedings...** Berlin: Springer, 1997. p.222–236. (Lecture Notes in Computer Science, v.1330).

KASIF, S. On the parallel complexity of some constraint satisfaction problems. In: NATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE, AAAI, 1986. **Proceedings...** [S.l.: s.n.], 1986. p.349–353.

KITANO, H.; TODOKORO, S.; NODA, I.; MATSUBARA, H.; TAKAHASHI, T. Robocup rescue: search and rescue in large-scale disaster as a domain for autonomous agents research. In: IEEE INTERNATIONAL CONFERENCE ON SYSTEMS, MAN, AND CYBERNETICS, 1999, Tokyo, Japan. **Proceedings...** [S.l.: s.n.], 1999. v.6, p.739–743.

KSCHISCHANG, F.; FREY, B.; LOELIGER, H. Factor graphs and the sum-product algorithm. **IEEE Transactions On Information Theory**, [S.l.], v.47, n.2, p.498–519, Feb. 2001.

KUMAR, V. Algorithms for Constraints Satisfaction problems: a survey. **AI Magazine**, [S.l.], v.13, n.1, p.32–44, 1992.

LAND, A. H.; DOIG, A. G. An Automatic Method for Solving Discrete Programming Problems. **Econometrica**, [S.l.], n.28, p.497–520, 1960.

LOWRIE, P. The Sydney Coordinated Adaptive Traffic System – Principles, Methodology, Algorithms. In: INTERNATIONAL CONFERENCE ON ROAD TRAFFIC SIGNALLING, 1982, Sydney, Australia. **Proceedings...** London: IEE, 1982. p.67–70.

MAHESWARAN, R. T.; TAMBE, M.; BOWRING, E.; PEARCE, J. P.; VARAKANTHAM, P. Taking DCOP to the Real World: efficient complete solutions for distributed multi-event scheduling. In: INTERNATIONAL JOINT CONFERENCE ON AUTONOMOUS AGENTS AND MULTIAGENT SYSTEMS, AAMAS, 3., 2004. **Proceedings...** New York: IEEE Computer Society, 2004. v.1, p.310–317.

MAILLER, R.; LESSER, V. A Mediation Based Protocol for Distributed Constraint Satisfaction. In: INTERNATIONAL WORKSHOP ON DISTRIBUTED CONSTRAINT REASONING, 4., 2003, Acapulco, Mexico. **Proceedings...** [S.l.: s.n.], 2003. p.49–58.

MAILLER, R.; LESSER, V. Solving Distributed Constraint Optimization Problems Using Cooperative Mediation. In: INTERNATIONAL JOINT CONFERENCE ON AUTONOMOUS AGENTS AND MULTIAGENT SYSTEMS, 3., 2004. **Proceedings...** New York: IEEE Computer Society, 2004. p.438–445.

MODI, P. J. **Distributed Constraint Optimization for Multiagent Systems**. 2003. Tese (Doutorado em Ciência da Computação) — University of Southern California.

MODI, P. J.; SHEN, W.-M.; TAMBE, M.; YOKOO, M. An asynchronous complete method for distributed constraint optimization. In: INTERNATIONAL JOINT CONFERENCE ON AUTONOMOUS AGENTS AND MULTIAGENT SYSTEMS, 2., 2003. **Proceedings...** New York: ACM Press, 2003. p.161–168.

MODI, P. J.; SHEN, W.; TAMBE, M.; YOKOO, M. ADOPT: asynchronous distributed constraint optimization with quality guarantees. **Artificial Intelligence**, [S.l.], v.161, p.149–180, Jan. 2005.

OLIVEIRA, D.; BAZZAN, A. L. C.; LESSER, V. Using Cooperative Mediation to Coordinate Traffic Lights: a case study. In: INTERNATIONAL JOINT CONFERENCE ON AUTONOMOUS AGENTS AND MULTI AGENT SYSTEMS, AAMAS, 4., 2005. **Proceedings...** New York: ACM, 2005. v.1, p.463–470.

OLIVEIRA, D.; BAZZAN, A. L. C.; SILVA, B. C.; BASSO, E. W.; NUNES, L.; ROSSETTI, R. J. F.; OLIVEIRA, E. C.; SILVA, R.; LAMB, L. C. Reinforcement learning based control of traffic lights in non-stationary environments: a case study in a microscopic simulator. In: EUROPEAN WORKSHOP ON MULTI-AGENT SYSTEMS, EUMAS, 4., 2007. **Proceedings...** Germany: Technical University of Aachen, 2007. p.31–42.

PAPAGEORGIOU, M.; DIAKAKI, C.; DINOPOULOU, V.; KOTSIALOS, A.; WANG, Y. Review of Road Traffic Control Strategies. **Proceedings of the IEEE**, [S.l.], v.91, n.12, p.2043–2067, December 2003.

PETCU, A. **Recent Advances in Dynamic, Distributed Constraint Optimization**. [S.l.: s.n.], 2006.

PETCU, A.; FALTINGS, B. A distributed, complete method for multi-agent constraint optimization. In: INTERNATIONAL WORKSHOP ON DISTRIBUTED CONSTRAINT REASONING, DCR, 5., 2004. **Proceedings...** [S.l.: s.n.], 2004.

PETCU, A.; FALTINGS, B. A Scalable Method for Multiagent Constraint Optimization. In: INTERNATIONAL JOINT CONFERENCE ON ARTIFICIAL INTELLIGENCE, 19., 2005, Edinburgh, Scotland. **Proceedings...** [S.l.: s.n.], 2005. p.266–271.

PETCU, A.; FALTINGS, B. MB-DPOP: a new memory-bounded algorithm for distributed optimization. In: INTERNATIONAL JOINT CONFERENCE ON ARTIFICIAL INTELLIGENCE, IJCAI, 20., 2007, Hyderabad, India. **Proceedings...** [S.l.: s.n.], 2007.

ROBERTSON, D. I.; BRETHERTON, R. D. Optimizing Networks of Traffic Signals in Real Time – The SCOOT Method. **IEEE Transactions on Vehicular Technology**, [S.l.], v.40, n.1, p.11–15, Feb. 1991.

RUSSEL, S.; NORVIG, P. **Artificial Intelligence**: a modern approach. [S.l.]: Prentice–Hall, 1995.

SHEN, W. M.; YIM, M. Self-reconfigurable robots. **IEEE/ASME Transactions on Mechatronics**, New York, v.7, n.4, p.401–402, 2002.

SILVA, B. C.; JUNGES, R.; OLIVEIRA, D.; BAZZAN, A. L. C. ITSUMO: an intelligent transportation system for urban mobility. In: INTERNATIONAL JOINT CONFERENCE ON AUTONOMOUS AGENTS AND MULTIAGENT SYSTEMS, AAMAS, 5., 2006. **Proceedings...** New York: ACM Press, 2006. p.1471–1472.

TAMBE, M. Towards flexible teamwork. **Journal of Artificial Intelligence Research (JAIR)**, [S.l.], n.7, p.93–124, 1997.

TRANSYT-7F. **TRANSYT-7F User's Manual**. [S.l.]: Transportation Research Center, University of Florida, 1988.

YOKOO, M.; DURFEE, E. H. **Distributed constraint optimization as a formal model of partially adversarial cooperation**. Ann Arbor, Michigan: University of Michigan, 1991. (CSE-TR-101-91).

YOKOO, M.; DURFEE, E. H.; ISHIDA, T.; KUWABARA, K. Distributed Constraint Satisfaction for Formalizing Distributed Problem Solving. In: INTERNATIONAL CONFERENCE ON DISTRIBUTED COMPUTING SYSTEMS, 1992. **Proceedings...** [S.l.: s.n.], 1992. p.614–621.

YOKOO, M.; HIRAYAMA, K. Distributed constraint satisfaction algorithm for complex local problems. In: INTERNATIONAL CONFERENCE ON MULTIAGENT SYSTEMS, 1998. **Proceedings...** [S.l.: s.n.], 1998.