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

ABEL CORRÊA

Alocação de tarefas de desastre na plataforma *RMASBench*: uma abordagem baseada em passagem de mensagens e formação de grupos

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

Orientador: Prof. Dr. Ana Lucia Cetertich Bazzan

Porto Alegre
2015

CIP — CATALOGAÇÃO NA PUBLICAÇÃO

Corrêa, Abel

Alocação de tarefas de desastre na plataforma *RMASBench*: uma abordagem baseada em passagem de mensagens e formação de grupos / Abel Corrêa. – Porto Alegre: PPGC da UFRGS, 2015.

80 f.: il.

Dissertação (mestrado) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR–RS, 2015. Orientador: Ana Lucia Cetertich Bazzan.

1. Sistemas multiagente. 2. Problema de otimização. 3. Passagem de mensagens. 4. Modelos gráficos. 5. Formação de grupos. 6. Desastre urbano. I. Bazzan, Ana Lucia Cetertich. II. Título.

Para Vó Ertília e Vera Lúcia, minhas primeiras orientadoras,

À Rejane pela amizade.

AGRADECIMENTOS

Agradeço à minha família por ter dado todo o apoio e acreditarem em mim. À minha mãe Nica e a Cíntia, minha companheira, sempre presentes ao meu lado dando todo o apoio para eu seguir adiante. Ao meu pai Leonel, Mára e aos meus irmãos, Ricardo e Taís, pela compreensão, companheirismo e disponibilidade. Às minhas sobrinhas pelo carinho e por compreenderem o motivo de eu não ter conseguido ficar ao lado delas como eu gostaria. À minha prima Gabriela pela amizade.

Sou muito grato à professora Ana Bazzan pela orientação, pela confiança depositada em mim e pelas lições que tive a oportunidade de aprender durante todo o período do curso, desde a disciplina de inteligência artificial avançada. Sua orientação foi fundamental para o desenvolvimento desse trabalho.

Agradeço aos amigos: Anderson Tavares, Cristiano Galafassi, Fabiane Penteado, Fernando dos Santos, Gabriel Ramos, Gabriela Callo, Jorge Samatelo, Luiz Henrique Dias, Marcelo Souza, Mariana Mendoza, Matthew Vallis, Ricardo Grunitzki, Rodrigo Batista, Sérgio Montazzolli entre outros que, porventura, posso ter esquecido. Agradeço a todos vocês pelos mais diversos motivos: amizade, conselhos, companhia, revisões, incentivo, sushi, churrascos e a cervejinha para descontrair. Enfim, por abrirem espaço para me aceitarem como amigo. Gostaria de ter estado mais próximo de vocês no laboratório.

Agradeço aos meus amigos, dos quais não pude estar próximo durante esse período, por compreenderem o motivo da minha ausência e pelo apoio que me deram. À ASSUFRGS e aos colegas do movimento de greve por lutarem pelos trabalhadores que ajudam a tornar a UFRGS uma faculdade de excelência.

Agradeço ao Instituto de Informática e ao Programa de Pós - Graduação em Computação pela oportunidade de estudar e aprender junto a pessoas tão competentes. Agradeço também à Secretaria de Educação a Distância e ao Centro de Processamento de Dados pela liberação concedida para realização das disciplinas e por todo o apoio que foi dado.

Por fim, agradeço à Universidade Federal do Rio Grande do Sul pelo incentivo para que possamos crescer junto a instituição, auxiliando no desenvolvimento de uma faculdade de excelência.

“Gaal Dornick, using non-mathematical concepts, defined psychohistory to be that branch of mathematics which deals with the reactions of human conglomerates to fixed social and economic stimuli...

...The necessary size of such a conglomerate may be determined by Seldon’s First Theorem...”

— ENCYCLOPEDIA GALACTICA

RESUMO

Em ambientes de desastre urbano, grupos de agentes de resgate devem resolver tarefas de modo a minimizar os danos que podem ocorrer na cidade. Tais ambientes são dinâmicos e parcialmente observáveis, com características que dizem respeito à distância espacial, quantidade de recursos, à dificuldade da tarefa de desastre e à capacidade do agente de atendê-la. A comunicação entre os agentes pode ser ruidosa ou inexistente. Os sistemas multiagente são desenvolvidos para resolver problemas complexos e abrangentes, que estão além da capacidade de um único agente. Nesse contexto, os agentes são elementos computacionais autônomos que são responsáveis por uma parte da solução do problema. Os agentes são situados em um ambiente e podem ter habilidade social, interagindo com outros agentes para resolver as tarefas. Comumente, o domínio de desastre urbano é formalizado como um problema de alocação de tarefas e modelado como um problema de otimização de restrições distribuídas entre agentes heterogêneos, onde eles têm que escolher as tarefas que maximizam suas utilidades individuais ou minimizem seus custos individuais. Essa dissertação de mestrado propõe um modelo para formação de grupos de agentes baseado na minimização de uma métrica de distância. O modelo é formalizado como um problema de otimização de restrições distribuídas, usando algoritmos para troca de mensagens entre os agentes. O modelo chamado Formação de Grupos pela Minimização da Distância (FGMD) tem agentes autônomos que tem a capacidade de se auto-organizar sem a necessidade de um controle centralizado. Aplicamos o FGMD na plataforma *RMASBench*, que é um simulador para situações de desastre urbano. Comparou-se o FGMD com os algoritmos mais recentes de passagem de mensagens, tendo sido verificado que o FGMD use menos computação não-paralela. Com respeito a minimização dos danos na cidade, mostrou-se que é possível obter resultados melhores que as abordagens do estado-da-arte com leve aumento no esforço computacional.

Palavras-chave: Sistemas multiagente. problema de otimização. passagem de mensagens. modelos gráficos. formação de grupos. desastre urbano.

Allocation of disaster tasks in the *RMASBench* platform: an approach based on message passing and group formation

ABSTRACT

In urban disaster environments, groups of rescue agents must solve tasks in order to minimize the damage that can occur in a city. Such environments are dynamic and partially observable, with features that correspond to spatial distance, amount of resources, difficulty of the disaster task, and the capability of the agent to handle it. The communication between the agents can be noisy or non-existent. Multiagent systems are developed to solve complex and comprehensive problems, that are beyond the capability of one single agent. In this context, the agents are autonomous computational elements that are responsible for a piece of the solution of the problem. The agents are situated in an environment, and may have social ability, interacting with other agents to solve the tasks. Commonly, the urban disaster domain is formalized as a task allocation problem, and modelled as a constraint optimization problem distributed among heterogeneous agents, where they have to choose the tasks that maximize their individual utilities or minimize their individual costs. This master thesis proposes a model for formation of groups of agents based in the minimization of a distance. The model is formalized as a distributed constraint optimization problem, using algorithms to exchange messages between agents. The model called Formation of Groups by Minimization of Distance (FGMD) has self-organizing autonomous agents without a centralized control. We applied the FGMD in the *RMASBench* platform, that is a simulator for urban disaster situations. We compare the FGMD with the most recent message passing algorithms, verifying that FGMD uses less non-parallel computation. With respect to the minimization of the damage in the city, we show that it is possible to obtain better results than the state-of-art approaches, with slightly increase of computational effort.

Keywords: multiagent systems, optimization problem, message passing, graphical models, group formation, urban disaster.

LISTA DE ABREVIATURAS E SIGLAS

RCRS	*RoboCup Rescue Simulator*
USAR	*Urban Search And Rescue*
USARSim	*Unified System for Automation and Robot Simulation*
BMS	*Binary Max-Sum*
CFST	*Coalition Formation with Spatial and Temporal constraints*
COP	*Constraint Optimization Problem*
DCOP	*Distributed Constraint Optimization Problem*
DSA	*Distributed Stochastic Algorithm*
FGMD	Formação de Grupos pela Minimização da Distância
GAP	*Generalized Assignment Problem*
E-GAP	*Extended Generalized Assignment Problem*
GIS	*Geographical Information System*
kNCA	*k-Neighbourhood Component Analysis*
MS	*Max-Sum*
MTTA	*Multi-Team Task Allocation*
RCR	Projeto *RoboCup Rescue*
SMA	*Sistemas Multiagente*
SNE	*Stochastic Neighbourhood Embedding*
TAP	*Task Allocation Problem*
HOP	*High Order Potential*
THOP	*Tractable High Order Potential*

LISTA DE SÍMBOLOS

$A = \{a_1, \cdots, a_i, \cdots, a_n\}$	Conjunto de agentes;
$X = \{x_1, \cdots, x_i, \cdots, x_n\}$	Conjunto de variáveis, assumindo que cada agente a_i é responsável por uma variável x_i;
$D_i = \{d_{i1}, \cdots, d_{ij}, \cdots, d_{ik}\}$	Domínio de um dado agente a_i, onde k é a cardinalidade do domínio;
d_{ij}	Tarefa d_j no domínio do agente a_i;
δ_{ij}	Valor/utilidade de uma tarefa d_{ij};
I_j	Valor que representa a dificuldade de uma tarefa d_j, denominado *fieryness*;
I_{max}	Valor máximo de *fieryness*, de modo que a tarefa não possa mais ser resolvida;
$dist(i, j)$	Distância Euclideana entre as coordenadas de agente a_i e uma tarefa d_j;
ν	Parâmetro para ponderar a distância e a dificuldade de uma tarefa;
$p_j(\mathbf{x})$	Função de custo para um conjunto de variáveis;
$q_{ij}(x_i)$	Função de custo de uma variável x_i dado um valor d_j;
W_j	Parâmetro que representa a carga de trabalho de uma tarefa d_j;
n	Número de agentes em um grupo;
$Z = \{z_{ij}, \forall x_i \in X, \forall d_j \in D_i\}$	Conjunto de variáveis binárias que representam os valores no domínio dos agentes;
k	Parâmetro de poda do domínio, de modo que um agente tenha no máximo k tarefas no domínio.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ALGORITMOS

SUMÁRIO

1 INTRODUÇÃO

Agentes são sistemas computacionais capazes de agir de forma autônoma e interagir com outros agentes para satisfazer os objetivos para os quais foram programados. Em ambientes orientados a tarefas, o objetivo dos agentes pode ser representado como tarefas que cada agente deverá resolver, na qual o sucesso na realização das tarefas é avaliado através de uma medida de desempenho (SANTOS, 2009). Os agentes têm capacidade de percepção e de ação em um ambiente. Os sistemas multiagente (SMA) são construídos quando um objetivo está além da capacidade de um único agente. Ambientes reais podem ter um grande número de tarefas e agentes. Esses ambientes são parcialmente observáveis e dinâmicos, pois os agentes tem uma percepção limitada de um conjunto de tarefas com características variáveis ao longo do tempo como, por exemplo, dificuldade na realização da tarefa ou distância até ela.

Os desastres urbanos são um dos problemas mais graves que envolvem agentes heterogêneos e tarefas em ambientes hostis, tendo impacto ambiental, social e econômico onde ocorrem. As equipes de agentes nessas situações precisam cooperar e se coordenar para tomar decisões rapidamente. Em janeiro de 1995, um terremoto atingiu a cidade de Kobe no Japão. A destruição atingiu milhares de construções, soterrando pessoas, obstruindo ruas e dando início a focos de incêndio que se alastraram pelas construções da cidade. A infraestrutura de energia, água e comunicação foram extremamente danificadas, e as equipes de resgate e os paramédicos tiveram grande dificuldade em agir devido aos danos causados pelo terremoto nas estradas e prédios.

O terremoto de Kobe motivou a busca por um sistema robusto e dinâmico para operações de busca e resgate, como robôs de resgate ou assistentes digitais para auxílio na localização de vítimas (KITANO; TADOKORO, 2001). Nesse contexto, foi fundado o Projeto *RoboCup Rescue* (RCR), onde são centralizados esforços na busca desse sistema. Através das ligas de robos de resgate e simulação de resgate, o projeto RCR promove a pesquisa e desenvolvimento de agentes robóticos de busca e resgate em situações de desastre urbano. A liga de robos de resgate, denominada *Urban Search and Rescue* (USAR), visa avaliar implementações robóticas em ambientes representativos. A liga de simulação de resgate se divide nas competições de robôs virtuais e simulação de agentes. A competição de robôs virtuais utiliza um simulador de robôs autônomos em um ambiente baseado na plataforma *UnrealTournament*, chamada *USARSim*. Por fim, a simulação de agentes visa descobrir novas formas de coordenação autônoma entre times de agentes em situações de desastre urbano simulados (AKIN et al., 2013). A liga de simulação de agentes utiliza o simulador de desastre urbano *RoboCup Rescue Simulator*

(RCRS) e, atualmente, a plataforma de avaliação *RMASBench* que permite o desenvolvimento de algoritmos de coordenação para uso no RCRS (KITANO; TADOKORO, 2001; KLEINER et al., 2013).

No contexto do domínio de desastre urbano, algumas abordagens formalizam o problema como alocação/atribuição de tarefas para agentes (do inglês, *Task Allocation Problem* - TAP). O problema de atribuição generalizada, do inglês *Generalized Assignment Problem* (GAP) e sua extensão, E-GAP, são utilizados para formalizar alguns aspectos do problema de atribuição de tarefas (SCERRI et al., 2005). A solução para esses modelos consiste em encontrar um conjunto de atribuições de tarefas para agentes, que maximizem as competências do grupo de agentes que as realizam, respeitando o limite de recursos que cada agente possui. Contudo, tais modelos impõem uma restrição de que cada tarefa deverá ser atribuída para um agente apenas. Intuitivamente, deve ser assumido que um único agente seja capaz de realizar qualquer tarefa no ambiente. Formalmente, tal restrição inviabiliza a formação de grupos para alocação de tarefas.

Em Santos (2009) é proposto um algoritmo distribuído de formação de agrupamentos baseado em inteligência de enxames. O algoritmo *Bee Clustering* é inspirado no processo de recrutamento observado em colônias de abelhas, onde uma abelha dança para recrutar outras abelhas para coletarem alimento em uma fonte de nectar. No cenário de desastres, o *Bee Clustering* foi aplicado ao problema de formação de grupos de agentes que realizam tarefas de desastre. Uma das ideias do *Bee Clustering* é a de abordar o problema de alocação de tarefas de uma maneira mais genérica, através da formação de grupos de agentes baseado na minimização de uma métrica de distância entre os agentes e as tarefas que serão realizadas por eles. De acordo com Santos (2009, p.85) observou-se uma limitação quanto ao uso de uma métrica baseada no elemento central do grupo (centróide).

Trabalhos mais recentes formalizam o domínio de desastre urbano como um problema de otimização de restrições distribuídas (do inglês, *Distributed Constraint Optimization Problem* - DCOP). No DCOP, os agentes devem escolher valores para suas variáveis de modo que uma função objetivo seja otimizada. Comumente, assume-se que cada agente é responsável por uma única variável que representa alguma tarefa de desastre que será realizada por ele em um dado momento, sendo que o domínio do agente é composto de todas as tarefas que ele consegue perceber no ambiente. Os modelos DCOP utilizam uma representação em grafo, onde os vértices do grafo representam agentes ou tarefas no ambiente e as arestas representam as relações ou restrições entre eles. A avaliação dos algoritmos DCOP pode ser feita através de alguma métrica empírica de desempenho relacionada ao problema, uso de comunicação (troca

de mensagens entre os agentes) e verificação de restrições (computação não-paralelizada realizada por cada agente). Essa forma de representação tem sido utilizada com sucesso no domínio de desastre urbano e um modelo DCOP pode ser otimizado por qualquer algoritmo DCOP do estado-da-arte.

Ramchurn et al. (2010) apresentaram um modelo DCOP representado em grafo para o problema de formação de coalizões baseado em restrições espaciais e temporais em cenários de desastres. O objetivo do modelo é maximizar o número de tarefas realizadas considerando todas as combinações possíveis de alocação de tarefas para agentes. Os autores utilizam o algoritmo de passagem de mensagens *max-sum* (MS), onde os agentes enviam e recebem mensagens através das arestas do grafo de restrições, de modo a otimizar o valor de suas variáveis. Contudo, o espaço de busca da solução do problema é exponencial considerando um grande número de agentes e tarefas no ambiente. Pujol-Gonzalez et al. (2014) apresentaram o algoritmo *binary max-sum* (BMS), para otimizar em tempo polinomial um modelo DCOP representado em grafo, aplicado ao problema de coordenação de times de agentes em ambientes de desastres. No BMS, a representação do problema aumenta exponencialmente, sendo que para o modelo de coordenação em ambientes de desastres abordado pelos autores, observou-se que o algoritmo utiliza muita comunicação e processamento não-paralelo.

1.1 Contribuições

A contribuição da presente dissertação consiste em apresentar um modelo para formação de grupos para alocação de tarefas em ambientes dinâmicos, utilizando uma métrica empírica baseada na distribuição da carga de trabalho entre grupos de agentes. O modelo chamado Formação de Grupos pela Minimização da Distância (FGMD) usa o algoritmo de passagem de mensagens *binary max-sum* para otimizar uma função objetivo decomposta em funções locais distribuídas para agentes heterogêneos.

A eficiência do algoritmo é avaliada na plataforma *RMASBench*, que atua sobre o simulador de desastre urbano *RoboCup Rescue Simulator*. Nos experimentos realizados, buscou-se reproduzir as condições de avaliação apresentadas em Pujol-Gonzalez et al. (2014), e os resultados obtidos indicam que o FGMD utiliza menos comunicação e computação não-paralelizada, apresentando diferenças significativas entre os modelos comparados na maioria das condições experimentadas.

1.2 Organização do Trabalho

Esse trabalho está organizado como segue:

Capítulo 1 Introduz os conceitos que serão abordados neste trabalho, motivação e contribuições;

Capítulo 2 Apresenta conceitos de sistemas multiagente, domínio de desastre urbano, a plataforma de simulação *RoboCup Rescue Simulator* e o *RMASBench*;

Capítulo 3 Apresenta a fundamentação teórica, abordando o problema de otimização de restrições distribuídas e os algoritmos de passagem de mensagens;

Capítulo 4 Mostra trabalhos relacionados ao domínio de desastre urbano;

Capítulo 5 Apresenta o modelo Formação de Grupos pela Minimização da Distância, apresentando a representação do problema e a aplicação do algoritmo *binary max-sum*;

Capítulo 6 Descreve os experimentos e apresenta os resultados;

Capítulo 7 Conclui o trabalho, abordando os aspectos positivos e negativos do modelo, apresentando possíveis direções para trabalhos futuros.

2 SISTEMAS MULTIAGENTE E O DOMINIO DE DESASTRE URBANO

Os sistemas multiagente (SMA) são construídos para resolver problemas que estão além da competência de um único agente. Esse capítulo apresentará algumas definições e conceitos relacionados a agentes inteligentes, como construí-los e definir seus objetivos. A seguir, será apresentado o domínio de desastre urbano que é um dos problemas mais complexos que exige coordenação de agentes.

2.1 Agentes Inteligentes

Um agente é um sistema computacional que está situado em um ambiente, sendo capaz de realizar ações autônomas para satisfazer os objetivos para o qual foi programado (WOOLDRIDGE, 2009, p. 21). Franklin e Graesser (1997) traçam um paralelo entre agentes humanos, animais e robôs, no intuito de definir o que são os agentes. Do mesmo modo que os agentes do mundo real vivem em um ambiente real, os agentes artificiais "vivem" em um ambiente artificial e podem ser classificados conforme as propriedades que possuem. Wooldrige e Jennings (1995) sugerem que um agente inteligente apresente algumas das seguintes características:

Reatividade: Capacidade de perceber o ambiente e reagir de forma rápida às mudanças;

Pró-Atividade: Capacidade de exibir comportamento direcionado a tarefas, mostrando iniciativa de modo a satisfazer seus objetivos;

Habilidade Social: Capacidade de interagir com outros agentes ou seres humanos.

O agente percebe o ambiente no qual está inserido e produz, como saída, uma ação que será realizada por ele. O ambiente onde estão situados os agentes pode ser classificado conforme algumas das seguintes propriedades (RUSSEL; NORVIG, 1995 apud WOOLDRIDGE, 2009):

Completamente ou parcialmente observáveis: um ambiente é completamente observável quando cada agente possui informação completa e atualizada sobre os estados do ambiente, caso contrário, o ambiente é parcialmente observável;

Determinístico ou estocástico: Um ambiente determinístico é aquele em que qualquer ação terá um único efeito garantido, enquanto em um ambiente estocástico há incerteza sobre o resultados das ações realizadas;

Estático ou dinâmico: Um ambiente estático permanece inalterado, exceto pelo resultados das ações dos agentes, e em um ambiente dinâmico ocorrem mudanças independente das

ações dos agentes;

Discreto ou contínuo: Um ambiente é discreto se há um número finito e fixo de ações sobre ele, caso contrário, é contínuo.

Objetivos complexos e abrangentes podem requerer a atuação de mais de um agente. Um sistema multiagente é formado por um certo número de agentes que interagem entre si para resolver um objetivo. De acordo com Jennings, Sycara e Wooldrige (1998 apud SANTOS, 2009), um SMA pode apresentar as seguintes características:

Agentes heterogêneos: Os agentes podem possuir diferentes competências para realizar uma mesma tarefa;

Percepção limitada: Os agentes não tem conhecimento completo do ambiente, sendo que as informações estão distribuídas e é necessário interação entre os agentes para satisfazer os objetivos do sistema;

Autonomia: A computação é assíncrona, ou seja, os agentes devem atuar de forma assíncrona e autônoma. A sincronia pode ser obtida através das interações entre os agentes;

Inexistência de um controle global: Não há uma entidade central com percepção global do sistema, capaz de definir um comportamento adequado para os agentes.

A abordagem proposta nesse trabalho assume que os agentes são heterogêneos e têm percepção limitada do ambiente, não havendo controle centralizado sobre o comportamento dos agentes. Os agentes são autônomos e devem tomar decisões que beneficiem o sistema, para isso a interação entre eles é um fator determinante.

2.2 Alocação de Tarefas

Agentes inteligentes devem realizar ações de modo a satisfazer seus objetivos. Comumente, a definição desses objetivos se dá através da especificação de tarefas em um ambiente orientado à tarefas. O sucesso na realização das tarefas pelos agentes pode ser verificado através de uma métrica de desempenho (SANTOS, 2009).

Scerri et al. (2005) introduziram o termo *extreme teams* para designar um SMA com centenas de agentes e tarefas. Os agentes que compõem um *extreme teams* possuem competências definidas e recursos limitados para a realização das tarefas. Um ambiente de alocação de tarefas para *extreme teams* possui as seguintes características:

- O ambiente é dinâmico e as características dos agentes e das tarefas pode sofrer variações

com o passar do tempo;

- Os agentes podem realizar várias tarefas ao mesmo tempo, limitado pela quantidade de recursos disponível;
- Os agentes podem possuir funcionalidades sobrepostas, isto é, podem estar aptos a realizar a mesma tarefa, mas com diferentes níveis de competência;
- Pode existir inter-relacionamentos entre as tarefas, de modo que certas tarefas devam ser resolvidas simultaneamente.

Para formalizar o problema de alocação de tarefas, os autores propõem o E-GAP, uma variação do modelo matemático GAP (do inglês, *Generalized Assignment Problem*). Para mais detalhes sobre o modelo GAP, recomenda-se consultar (SCERRI et al., 2005) e (SANTOS, 2009, pg. 17).

Com relação ao modelo E-GAP, seja $V = \{d_1, \cdots, d_m\}$ o conjunto de tarefas no ambiente e $A = \{a_1, \cdots, a_n\}$ o conjunto de agentes para realizá-las. Cada agente $a_i \in A$ possui competência para realizar cada tarefa $d_j \in V$ e uma quantia limitada de recursos para realizar todas as suas tarefas. A competência reflete quão bom será o resultado da alocação de uma tarefa para um agente, sendo utilizada como uma medida de recompensa que o time de agentes recebe por realizar a tarefa. A capacidade de um agente para uma tarefa é dada por $Cap(a_i, d_j) \rightarrow [0, 1]$, se $Cap(a_i, d_j) > 0$, diz-se que a_i é capaz de resolver d_j. Para resolver uma tarefa d_j, um agente a_i deve gastar uma quantia em recursos, representada por $Res(a_i, d_j)$. Define-se uma matriz de alocação M, tal que m_{ij} é o valor da i-ésima linha e j-ésima coluna da matriz, com valores de acordo com a Equação 2.1.

$$m_{ij} = \begin{cases} 1 & \textit{se } a_i \textit{ está realizando } d_j \\ 0 & \textit{caso contrário} \end{cases} \tag{2.1}$$

O inter-relacionamento entre tarefas é representado por um conjunto de subconjuntos de tarefas, denotado $\bowtie$. Seja $\bowtie = \{\theta_1, \cdots, \theta_p\}$, onde $\theta_k = \{d_{k1}, \cdots, d_{kq}\}$ representa o k-ésimo subconjunto de tarefas inter-relacionadas. O número de tarefas que estão sendo realizadas no subconjunto θ_k é calculado conforme a Equação 2.2.

$$x_k = \sum_{a_i \in A} \sum_{d_{kj} \in \theta_k} m_{ikj} \tag{2.2}$$

O valor da utilidade um agente a_i realizando uma tarefa d_j, considerando o conjunto de

tarefas inter-relacionadas, $\bowtie$, é dado pela Equação 2.3.

$$Val(a_i, d_j, \bowtie) = \begin{cases} Cap(a_i, d_j) \times m_{ij} & \textit{se } \forall \theta_k \in \bowtie, d_j \notin \theta_k \\ Cap(a_i, d_j) \times m_{ij} & \textit{se } \exists \theta_k \in \bowtie | d_j \in \theta_k \wedge x_k = |\theta_k| \\ 0 & \textit{caso contrário} \end{cases} \tag{2.3}$$

O objetivo do E-GAP consiste em encontrar o maior valor da alocação de tarefas para agentes ao longo do tempo, considerando uma função de custo, $DC^t(d_j^t)$, sobre as tarefas que não estão sendo realizadas.

$$f(M) = \sum_t \sum_{a_i^t \in A^t} \sum_{d_j^t \in V^t} (Val^t(a_i^t, d_j^t, \bowtie^t) \times m_{ij}^t) - \sum_t \sum_{d_j^t \in V^t} (1 - \sum_{a_i^t \in A^t} m_{ij}^t) \times DC^t(d_j^t) \tag{2.4}$$

Formalmente, o objetivo do E-GAP é maximizar o valor da função de alocação, representada pela Equação 2.4, respeitando o limite de recursos dos agentes, $Res(a_i^t)$, e a restrição de que cada tarefa deve ser alocada por apenas um agente, representados, respectivamente, nas Equações 2.5 e 2.6.

$$\forall t, \forall a_i^t \in A^t, \sum_{d_j^t \in V^t} Res^t(a_i^t, d_j^t) \times m_{ij}^t \leq Res(a_i^t) \tag{2.5}$$

$$\forall t, \forall d_j^t \in V^t, \sum_{a_i^t \in A^t} m_{ij}^t \leq 1 \tag{2.6}$$

O modelo E-GAP é utilizado para formalizar o problema de alocação de tarefas para agentes em ambientes orientados à tarefas. Contudo, a restrição de exclusão mútua apresentada na Equação 2.6 é um limitador para o problema de formação de grupos de agentes, pois considera que um agente seja capaz de resolver qualquer tarefa no ambiente

2.3 Domínio de Desastre Urbano

O domínio de desastre urbano em sistemas multiagente é um dos problemas sociais mais graves que requer a coordenação e planejamento de agentes heterogêneos, e podem en-

volver um grande número de agentes e tarefas. Os agentes representam equipes de resgate e salvamento, bombeiros, policiais e civis, e as tarefas representam extinção de incêndios em prédios, atendimento médico em civis, evacuação de civis em prédios em chamas ou com perigo de desabamentos, remoção de bloqueios em ruas, entre outros.

Algumas das dificuldades deste cenário são que, em geral, os recursos são escassos, a comunicação é ruidosa e os agentes não possuem conhecimento completo do ambiente. A coordenação e a comunicação são fatores importantes para o controle da situação de desastre. Em alguns casos podem haver dificuldades na comunicação, devido à ausência de redes de energia ou falha na rede de dados e telecomunicações. Além disso, o tempo necessário para a realização das tarefas e a quantidade de recursos requeridos podem não ser conhecidos de forma exata.

O ambiente de desastre urbano possui características que o torna um dos cenários mais complexos para sistemas multiagente, tais como:

- o ambiente é **parcialmente observável** em relação aos agentes;
- o ambiente é **estocástico**, pois não há certeza sobre o resultado das ações dos agentes;
- o ambiente é **dinâmico** podendo haver modificações em cada estado, ou seja, novas tarefas podem surgir ou desaparecer e as características das tarefas podem se alterar ao longo do tempo.

A Subseção 2.3.1 apresentará o *RoboCup Rescue Simulator* (RCRS)[1], um simulador de busca e resgate para situações de desastres. O RCRS provê um ambiente simulado de desastre urbano com prédios em chamas, situações de desabamento, bloqueios em ruas e resgate de civis, criando um ambiente complexo que requer a coordenação de agentes heterogêneos com restrições de movimentação, comunicação e recursos. Além disso possui código-fonte aberto, o que permite o desenvolvimento de novas funcionalidades e simuladores.

Na Subseção 2.3.2 será apresentado o *RMASBench*, que é uma plataforma executada sobre o RCRS para comparação de algoritmos de coordenação multiagente. O *RMASBench* permite criar um ambiente controlado para simulações de desastre.

2.3.1 Projeto *RoboCup Rescue*

O Projeto *RoboCup Rescue* (RCR) objetiva a transferência de tecnologias, desenvolvidas através de suas atividades, para ambientes de desastre urbano do mundo real. O *RoboCup*

[1] http://www.robocuprescue.org/

Rescue Simulator (RCRS) estabelece um domínio com características dinâmicas, agentes heterogêneos, diferentes classes de tarefas, examinando os princípios fundamentais do trabalho em equipe e sistemas multiagente em tempo real, integrando infraestrutura, robótica e projetos de simulação (KITANO; TADOKORO, 2001).

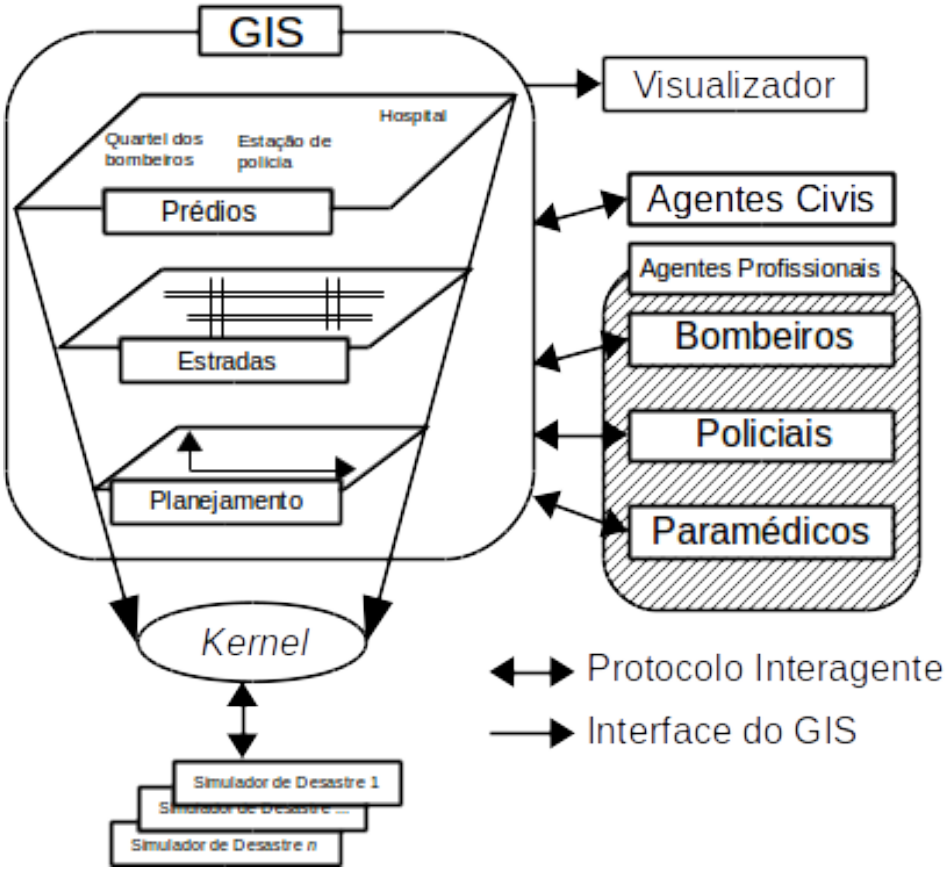
A arquitetura do RCRS é composta de módulos independentes que interagem com um *kernel* (Figura 2.1). O *kernel* é a parte central do sistema, que recebe, processa e encaminha informações para os outros módulos. Um conjunto de simuladores independentes realizam tarefas que irão influenciar diretamente no ambiente e nas decisões dos agentes. O simulador de incêndio faz com que o fogo tenha início nas construções, evoluindo no cenário de acordo com as ações realizadas pelos agentes. O simulador de colapso gera os efeitos de desabamento e bloqueio das ruas, causados por um terremoto. O sistema de informações geográficas (GIS) inicializa a configuração inicial do mundo, inserindo as construções, ruas, e mantendo um registro de eventos da simulação. Os *viewers* mostram em tempo real o que está acontecendo em cada *timestep* (unidade de tempo) de simulação. O módulo de agentes é dividido em *civilian*, *platoon* e *center agents*. Esse módulo detalha como os agentes irão atuar no cenário, respondendo a cada situação que lhes é apresentada durante a simulação. Os *platoon agents* são os responsáveis por tentar controlar as situações de desastres diretamente, agindo no ambiente, realizando tarefas ou se movimentando pelo cenário.

Há três tipos de *platoon agents*: bombeiros (*fire brigade*), policiais (*police force*) e as equipes de paramédicos (*ambulance team*). O objetivo dos bombeiros é localizar e extinguir incêndios em prédios. Os policiais devem explorar o ambiente procurando e removendo bloqueios das ruas e entradas dos prédios, facilitando o deslocamento no cenário. Os paramédicos devem procurar por civis soterrados ou feridos para levá-los até os refúgios para que fiquem em segurança. Os agentes centrais (*center agents*) são as centrais de serviço dos agentes. Por exemplo, o quartel de bombeiros, hospital e a central de polícia. As centrais podem ser utilizadas para auxiliar na coordenação entre os agentes, bem como centralizar as informações sobre o ambiente. Uma vez que cada agente está restrito a visualizar apenas o que está dentro do seu raio de visão, eles podem se comunicar entre si por rádio, voz, ou através do envio de informações para a central de forma que esta possa comunicar aos outros o estado atual do sistema.

O RCRS proporciona um ambiente adequado para realização de pesquisas como:

- Planejamento multiagente;
- Alocação de tarefas;
- Aprendizado multiagente em ambiente parcialmente observável;

Figura 2.1: Arquitetura do RCRS



Fonte: Adaptado de (KITANO; TADOKORO, 2001)

- Roteamento;
- Planejamento sobre uso de recursos;
- Formação de coalizões ou times de agentes;

O RCRS simula um ambiente dinâmico, onde os agentes observam apenas parte das tarefas, a comunicação é incerta e a realização das tarefas deve ocorrer de modo a minimizar os danos no cenário.

2.3.2 *RMASBench*

O *RMASBench* é uma plataforma desenvolvida sobre o *RoboCup Rescue Simulator* (RCRS) para avaliação de sistemas baseados em coordenação multiagente em ambientes de desastre urbano. Ele possibilita a avaliação empírica de algoritmos em cenários realísticos sem a necessidade de desenvolvimento de elementos de baixo-nível que podem ser irrelevantes (KLEINER et al., 2013).

No *RMASBench* a comunicação é garantida por uma camada central que reúne as infor-

mações sobre o ambiente, agentes e tarefas. No início de cada *timestep* de simulação, a camada central do *RMASBench*, chamada *center agent*, cria uma instancia do problema com informações sobre os agentes, tarefas e informações relevantes sobre o estado do ambiente, obtidos do RCRS. Uma matriz $n \times m$, onde n indica o total de agentes e m indica o total de tarefas, é utilizada para representar as utilidades dos agentes para cada *timestep*. Por fim, o módulo *solver* encarrega-se de executar o algoritmo que será usado para a tomada de decisão individual dos agentes. Ao final da execução do algoritmo, o *center agent* comunica para o *kernel* do RCRS quais tarefas cada agente irá executar naquele *timestep* de simulação. Uma heurística de poda pode ser utilizada para restringir o domínio dos agentes para k tarefas de utilidade mais alta, caso contrário, o domínio dos agentes será composto de todas as tarefas no ambiente.

O *RMASBench* provê métricas sobre o desempenho dos agentes, uso de comunicação, troca de mensagens, computação não-paralelizada e métricas sobre o cenário de desastre, como incêndios iniciados, resolvidos, percentual de dano no mapa e tempo de extinção das chamas.

Uma diferença importante entre o RCRS e o *RMASBench* é a troca de mensagens. No RCRS, quando um agente envia uma mensagem, ela será recebida apenas no *timestep* seguinte. Já no *RMASBench*, as trocas de mensagem podem ocorrer em cada iteração do algoritmo. Além disso, os agentes podem se comunicar utilizando o *center agent* como canal de comunicação, enquanto que no RCRS não há garantia de que a mensagem seja recebida.

Conforme Kleiner et al. (2013), o *RMASBench* pode ser usado para realizar comparações de algoritmos em dois tipos de cenários: coordenação de agentes bombeiros para extinguir incêndios (com ou sem bloqueios) e evacuação de multidões. Nesse trabalho será abordado o cenário de coordenação de agentes bombeiros para extinção de incêndios, sendo que em alguns experimentos foram incluídos bloqueios para aumentar a dificuldade nas ações dos bombeiros. Em cenários com bloqueios, os agentes policiais são responsáveis por desbloquear as ruas e garantir a mobilidade dos bombeiros no cenário.

2.4 Resumo

Este capítulo apresentou os sistemas multiagente (SMA), um sistema composto de elementos computacionais conhecidos como agentes. Um agente possui autonomia para alcançar os objetivos para o qual foi desenvolvido, podendo possuir habilidade social para interagir com outros agentes e solucionar problemas de forma cooperativa.

A Seção 2.2 apresentou o *extreme teams* e o E-GAP, utilizado para modelar aspectos do problema de alocação de tarefas para agentes em ambientes dinâmicos.

A Seção 2.3 introduziu o domínio de desastre urbano, apresentando o projeto *RoboCup Rescue Simulator* (Subseção 2.3.1) e o *RMASBench* (Subseção 2.3.2). O *RoboCup Rescue Simulator* (RCRS) é um simulador de código-fonte aberto, que é amplamente utilizado para realização de pesquisas na área de inteligência artificial e sistemas multiagente e o *RMASBench* é uma plataforma para comparação de algoritmos no RCRS. O RCRS simula um cenário mais complexo, com comunicação insuficiente e agentes distribuídos com informação parcial do ambiente. Já o *RMASBench* utiliza o *center agent* para garantir a comunicação entre os agentes e realização de experimentos em cenários controlados.

3 BUSCA DISTRIBUÍDA SOBRE RESTRIÇÕES

Um problema de otimização de restrições distribuídas (do inglês, *distributed constraint optimization problem*, ou DCOP) requer a otimização de uma função objetivo global, distribuída como funções locais que representam as restrições dos agentes (MODI et al., 2003). O DCOP é uma forma de modelar problemas onde as soluções têm graus de qualidade. Um algoritmo para um problema modelado como um DCOP é um procedimento de busca de uma atribuição completa de valores para variáveis que minimize ou maximize uma função objetivo global, distribuída em funções locais para agentes.

Na Seção 3.1 será introduzido o problema de otimização de restrições para agentes distribuídos (DCOP). Os problemas modelados como um DCOP podem ser representados como um grafo de restrições, com vértices representando os agentes e as arestas representando as restrições entre eles. Trabalhos mais recentes abordam a representação em grafo-fator como uma forma de distribuir um problema para múltiplos agentes. Um grafo-fator é um grafo bipartido composto de vértices que representam as variáveis do DCOP e vértices que representam interações entre subconjuntos de variáveis. Por fim, na Seção 3.3 será apresentado o *max-sum*, um algoritmo de passagem de mensagens que é utilizado para otimização em modelos representados como um grafo-fator. Recentemente, os algoritmos de passagem de mensagens têm sido aplicados para coordenação multiagente em cenários de desastre urbano.

3.1 DCOP: Otimização Distribuída entre Agentes com Restrições

Conforme Modi et al. (2003), um problema de otimização de restrições distribuídas, do inglês *distributed constraint optimization problem* (DCOP), consiste de variáveis $X = \{x_1, \cdots, x_i, \cdots, x_n\}$, onde cada variável $x_i \in X$ possui um domínio finito e discreto representado por D_i. Apenas os agentes que controlam as variáveis podem escolher um valor e conhecer o seu domínio, e o objetivo dos agentes é escolher o valor para sua variáveis, de modo que uma função objetivo seja minimizada ou maximizada. Geralmente, assume-se que cada agente é responsável por atribuir valor para uma variável apenas. Em Meisels (2008, p. 14), são descritas duas maneiras de representar agentes de múltiplas variáveis: **i)** definir um estado de atribuição composta para todas as variáveis dos agentes; ou, **ii)** definir agentes virtuais, sendo um para cada variável do agente.

A função objetivo global do DCOP consiste da soma de funções menores que representam as restrições entre os agentes. Seja $\mathbf{X}$ uma atribuição completa de valores para todas

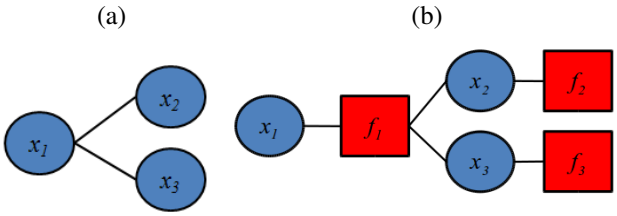
as variáveis do conjunto X, o objetivo do DCOP consiste em encontrar uma atribuição completa $\mathbf{X}^{\star}$ de valores para todas as variáveis, que minimiza ou maximiza a função objetivo $F(\mathbf{X})$, sendo que $F(\mathbf{X})$ pode ser decomposta na soma de funções menores $f_j(\mathbf{x})$ (PETCU, 2007, p.10). A Equação 3.1 apresenta a função objetivo do DCOP para minimização, onde $f_j(\mathbf{x})$ representa uma função de custo que recebe como argumento uma atribuição de valores para variáveis $\mathbf{x}$, tal que $\mathbf{x} \subseteq \mathbf{X}$.

$$\mathbf{X}^{\star} = \arg\min_{\mathbf{X}} F(\mathbf{X}) = \left[\sum_{f_j \in F} f_j(\mathbf{x}) \right] \tag{3.1}$$

Geralmente, um DCOP é representado como um *grafo de interação* com variáveis representadas como círculos, e uma aresta entre pares de variáveis indica que elas possuem uma restrição (FARINELLI; ROGERS; JENNINGS, 2014). Nesse trabalho será utilizada a representação em grafo-fator, que tem sido aplicada em trabalhos de coordenação em sistemas multiagente no domínio de desastre urbano, redes de sensores, etc.

Um grafo-fator consiste da decomposição de problemas complexos na soma de funções locais representadas como fatores. Ele é um grafo bipartido com nós que correspondem às variáveis do DCOP (geralmente representados como círculos) e nós que representam as restrições como funções locais sobre os valores de variáveis com restrições (geralmente representados como quadrados). A Figura 3.1 ilustra um modelo representado como um grafo de interação (Figura 3.1a) e um modelo equivalente representado como um grafo-fator (Figura 3.1b).

Figura 3.1: Modelo DCOP representado como um grafo de interação (3.1a) e representado como um grafo-fator (3.1b)



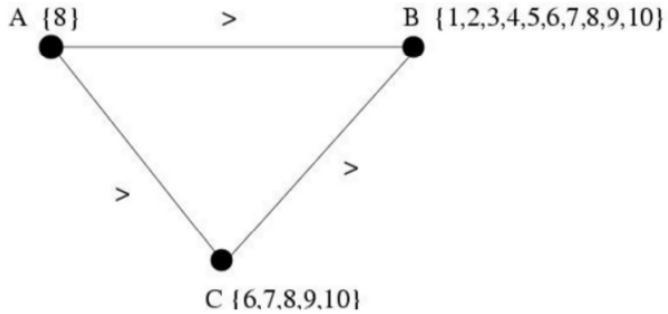
Diferente dos grafos de interação, que geralmente modelam restrições entre pares de agentes, um grafo-fator pode modelar restrições *n*-árias como funções de custo entre subconjuntos de váriaveis. Assumindo que os agentes do DCOP são responsáveis por atribuir valor para uma única variável, a busca pela solução ótima consiste na troca de mensagens entre os agentes, de modo a localizar a atribuição de valor que maximiza/minimiza uma função objetivo global decomposta em funções locais (ver Equação 3.1).

3.2 Métricas para Avaliação de Algoritmos DCOP

Comumente, os algoritmos de busca distribuída para modelos DCOP são avaliados pela quantidade de mensagens trocadas entre os agentes, verificação de restrições não-concorrentes (do inglês, *non-concurrent constraint checks* - NCCCs) e alguma métrica de desempenho dependente do domínio de aplicação. Em cenários de desastre urbano, o dano sofrido na cidade pode ser utilizado como métrica de desempenho do algoritmo

A troca de mensagens visa procurar uma solução ótima considerando os valores dos agentes vizinhos. Em ambientes reais, uma mensagem não chega instantaneamente ao seu destino, havendo atraso de acordo com as propriedades da rede de comunicação utilizada. Essa métrica representa a "carga da rede" utilizada para encontrar uma solução e é significante quando todos os algoritmos produzem mensagens de tamanho comparável (PETCU, 2007, p.241). Ainda, é possível avaliar um algoritmo DCOP utilizando o número de ciclos de simulação (iterações). Para isso, assume-se o uso de um simulador, o qual em cada ciclo de simulação, cada agente lê as mensagens recebidas. Essa métrica conta o número de iterações para resolver um problema.

Figura 3.2: Exemplo de grafo de restrições em um DCOP



Fonte: (MEISELS, 2008, p.107)

A métrica de verificação de restrições diz respeito ao esforço computacional realizado por um agente, ou seja, a busca pela atribuição de valor ótima para a variável de um agente em uma iteração. Para o entendimento dessa métrica de avaliação, considere o problema ilustrado na Figura 3.2 apresentado em Meisels (2008, p.107). Assumindo que os agentes A, B e C, com domínio representado pelos valores entre chaves, executam um algoritmo de busca seguindo essa mesma ordem e com arestas representando suas restrições. O agente A realiza uma atribuição e envia uma mensagem para os agentes B e C. Por sua vez, B e C devem atribuir os valores, respeitando a restrição de que seus valores devem ser maiores que o informado pelo

agente A. Como o problema é distribuído, os agentes B e C recebem a mensagem do agente A simultaneamente.

Durante a execução do algoritmo, os agentes realizam as seguintes ações:

- O agente A atribui o valor 8 e envia mensagem para B e C;
- O agente B recebe uma mensagem com a atribuição de A e percorre seu domínio até encontrar um valor que satisfaça a restrição de A (B > A). Depois de nove verificações, o valor 9 é atribuído por B e uma mensagem é enviada para C;
- Ainda na mesma iteração, o agente C recebe a mensgem de A contendo o valor 8 e depois de quatro verificações atribui o valor 9. Note que o agente B só receberá a mensagem de C na próxima iteração;
- Na iteração seguinte, o agente C recebe a mensagem de B com o valor 9 e realiza uma verificação adicional atribuindo o valor 10 para sua variavel e, assim, a primeira solução é encontrada.

A verificação de restrições de um agente é chamada de *constraint check* (CC), como o problema é distribuído e os agentes executam o procedimento de busca ao mesmo tempo, a métrica *non-concurrent constraint checks* (NCCCs) representa o maior valor de verificação de restrições entre todos os agentes em uma única iteração.

Figura 3.3: Gráfico de NCCCs para o problema representado pelo grafo de restrições ilustrado na Figura 3.2

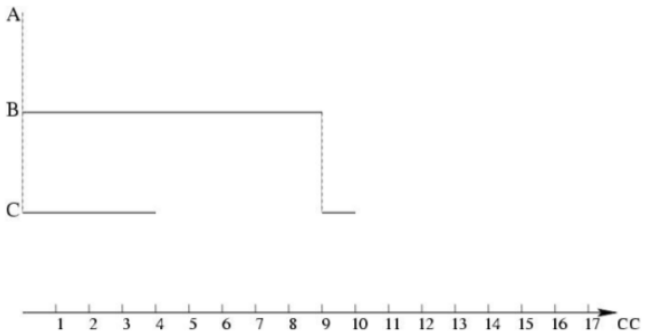


Fig. 10.2. Time plot of computations

Fonte: (MEISELS, 2008, p.108)

3.3 Algoritmo de passagem de mensagens *max-sum*

Como visto na Seção 3.1, um DCOP pode ser representado como um grafo-fator, um grafo bipartido composto de vértices que representam variáveis (nós variáveis) e vértices que representam as restrições do problema como funções sobre os valores das variáveis (nós fatores). Um grafo-fator representa uma função objetivo global de muitas variáveis, fatorada como o produto ou a soma de funções locais (KSCHISCHANG; FREY; LOELIGER, 2001).

Considere um conjunto de n agentes, onde o estado de cada agente pode ser representado por uma variável discreta x_i. Cada agente interage localmente com agentes vizinhos através de uma função de utilidade, $f_i(\mathbf{x})$, cujo valor é dependente da atribuição conjunta da variável do agente e das variáveis de seus vizinhos. A atribuição conjunta de um agente e seus vizinhos, no escopo de uma função, é denotada $\mathbf{x}$. Com isso, deseja-se encontrar uma atribuição completa, $\mathbf{X}^\star$, de valores para todas as variáveis dos agentes, de modo que a função objetivo do sistema, a soma das utilidades dos agentes, seja maximizada (ver Equação 3.2).

$$\mathbf{X}^\star = \arg\max_{\mathbf{X}} F(\mathbf{X}) = \left[\sum_{i=1}^{n} f_i(\mathbf{x})\right] \tag{3.2}$$

O *Max-Sum* (MS) é um algoritmo baseado em passagem de mensagens, que pode ser aplicado em um problema de otimização de restrições representado como um grafo-fator, onde os nós são capazes de transmitir e receber mensagens através de suas arestas. O MS utiliza duas mensagens que são propagadas pelos nós do grafo-fator, a mensagem de um nó variável i para um nó fator j, para cada valor no seu domínio, denotada $q_{i\to j}(x_i)$; e uma mensagem de um nó fator j para um nó variável i para cada valor da variável x_i, denotada $r_{j\to i}(x_i)$.

Na Equação 3.3, a mensagem de um nó fator j para um nó variável i representa a maior utilidade que pode ser obtida para cada valor da variável x_i, onde $\mathcal{N}_j$ é um vetor de índices das variáveis que estão conectadas ao nó fator j e $\mathbf{x}_j\backslash i \equiv \{x_k : k \in \mathcal{N}_j\backslash i\}$ (FARINELLI; ROGERS; JENNINGS, 2014). Do ponto de vista de um agente, representa o valor que ele atribui ao grupo, denotado por $f_j(\mathbf{x})$; somado ao retorno do grupo, representado pela soma das mensagens recebidas dos membros no grupo, $\sum_{k\in\mathcal{N}_j\backslash i} q_{k\to i}(x_k)$.

$$r_{j\to i}(x_i) = \max_{\mathbf{x}_j\backslash i}\left[f_j(\mathbf{x}_j) + \sum_{k\in\mathcal{N}_j\backslash i} q_{k\to i}(x_k)\right] \tag{3.3}$$

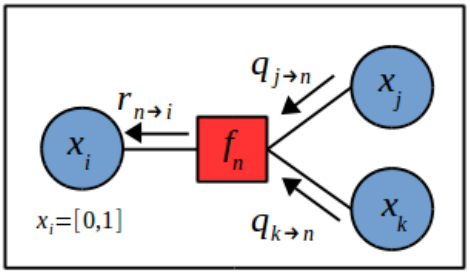
Na Equação 3.4, a mensagem de um nó variável i para um nó fator j representa a maior utilidade dos nós fatores vizinhos de i, onde $\mathcal{M}_i$ é um vetor de índices dos nós fatores conec-

tados ao nó variável i; e α_{ij} é uma escalar para normalizar as mensagens, de modo que a soma dos valores no domínio do agente seja igual a zero, $\sum_{x_i} q_{i\to j}(x_i) = 0$. O uso da escalar α_{ij} evita o crescimento infinito do valor das mensagens em grafos cíclicos.

$$q_{i\to j}(x_i) = \alpha_{ij} + \sum_{k\in\mathcal{M}_i\setminus j} r_{k\to i}(x_i) \tag{3.4}$$

Seja n um nó fator, vizinho dos nós variáveis i, j e k. Considere que x_i é uma variável discreta com domínio $[0, 1]$. O procedimento de passagem de mensagem do nó n para cada valor x_i, do nó i, é ilustrada na Figura 3.4 e escala exponencialmente, dado o número de vizinhos do nó fator n e a cardinalidade do domínio das variáveis. A complexidade da mensagem do nó fator para o nó variável é $O(D^{\eta+1})$, onde D refere-se a maior cardinalidade dentre as variáveis e η refere-se ao número de vizinhos do fator.

Figura 3.4: Exemplo de fluxo de mensagens do nó fator n para o nó variável i



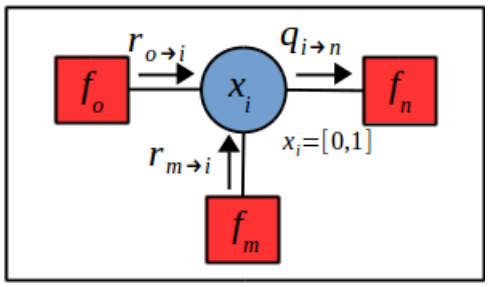
$$r_{n\to i}(0) = max_{(x_j,x_k)}\left[f_n(0, x_j, x_k) + q_{j\to n}(x_j) + q_{k\to n}(x_k)\right]$$
$$r_{n\to i}(1) = max_{(x_j,x_k)}\left[f_n(1, x_j, x_k) + q_{j\to n}(x_j) + q_{k\to n}(x_k)\right]$$

Seja i um nó variável, vizinho dos nós fatores n, m e o. O procedimento de passagem da mensagem, de cada valor do nó i para o nó n, é ilustrada na Figura 3.5 e tem complexidade linear $O(D\eta)$.

A propagação das mensagens atualiza a utilidade de cada valor das variáveis, terminando ao convergir, ou seja, quando a utilidade não se alterar de uma iteração para outra, ou quando for alcançado o número limite de iterações do algoritmo. A decisão sobre qual valor atribuir para a variável do agente depende da avaliação da função de marginalização $z_i(x_i)$, apresentada na Equação 3.5, para cada valor de x_i. O agente irá escolher o valor que maximiza $z_i(x_i)$, denotado por $\arg\max_{x_i\in D_i} z_i(x_i)$.

$$z_i(x_i) = \sum_{j\in\mathcal{M}_i} r_{j\to i}(x_i) \tag{3.5}$$

Figura 3.5: Exemplo de fluxo de mensagens do nó variável i para o nó fator n



$$q_{i\to n}(0)=\left[r_{m\to i}(0)+r_{o\to i}(0)\right]$$
$$q_{i\to n}(1)=\left[r_{m\to i}(1)+r_{o\to i}(1)\right]$$

Os algoritmos de passagem de mensagens podem utilizar um coeficiente da *damping*, para tornar a convergência do algoritmo mais robusta em grafos cíclicos. O fator de *damping* acrescenta uma ponderação entre as mensagens antigas e novas conforme a Equação 3.6.

$$\begin{aligned} q_{i\to j} &= q^{old}_{i\to j} \times damping + q^{new}_{i\to j} \times (1 - damping) \\ r_{j\to i} &= r^{old}_{j\to i} \times damping + r^{new}_{j\to i} \times (1 - damping) \end{aligned} \tag{3.6}$$

O Algoritmo 1 apresenta o pseudo-código das operações que cada agente realiza para implementar o *max-sum*. Comumente, consideram-se condições de término do *loop while* (Linha 3) um número pré-determinado de iterações, quando as mensagens não se alteram de uma iteração para outra; ou quando todos os agentes não alteram o valor da sua variável. As Linhas 5 e 8 referem-se, respectivamente, aos cálculos das mensagens dos nós fatores para os nós variáveis, conforme apresentado na Equação 3.3, e das variáveis para os fatores, conforme apresentado na Equação 3.4. As Linhas 6 e 9 referem-se, respectivamente ao envio da mensagem para o agente a_i, que possui a variável x_i e cujo recebimento da mensagem é apresentado nas Linhas 10 e 11. Por fim, a Linha 12 refere-se ao procedimento de alteração do valor da variável, onde cada agente irá alterar o valor da sua variável para a que maximiza o valor resultante da função apresentada na Equação 3.5.

Algoritmo 1 *Max-Sum* - Adaptado de (FARINELLI; ROGERS; JENNINGS, 2014)

1: $Q \leftarrow \emptyset$ # Inicializa um vetor de mensagens dos nós variáveis para os nós fatores
2: $R \leftarrow \emptyset$ # Inicializa um vetor de mensagens dos nós fatores para os nós variáveis
3: **while** condição de término não for alcançada **do**
4: **for** $i \in \mathcal{N}_j$ **do**
5: $r_{j \to i}(x_i) =$ CALCULARMENSAGEMPARAVARIAVEIS(x_i, f_j, Q)
6: ENVIARMENSAGEM$(r_{j \to i}(x_i), a_i)$
7: **for** $j \in \mathcal{M}_i$ **do**
8: $q_{i \to j}(x_i) =$ CALCULARMENSAGEMPARAFATORES(x_i, f_j, R)
9: ENVIARMENSAGEM$(q_{i \to j}(x_i), a_i)$
10: $Q \leftarrow$ RECEBERMENSAGENSDOSFATORES( )
11: $R \leftarrow$ RECEBERMENSAGENSDASVARIAVEIS( )
12: $x_i^{\star} =$ ATUALIZARVALORATUAL(x_i, R)

3.4 *Binary Max-Sum* e Fatores de Alta Ordem

Pesquisas mais recentes introduziram modelos representados como grafo-fator com variáveis binárias e nós fatores especiais chamados *high order potentials* (HOPs), em tradução livre, fatores de alta ordem (TARLOW; GIVONI; ZEMEL, 2010). Um HOP modela interações não locais entre variáveis discretas e aumenta o poder de representação dos nós fatores, possibilitando representar *soft* ou *hard constraints*. Outra classe de fatores de alta ordem, são os *tractable high order potentials* (THOPs), em tradução livre, fatores tratáveis de alta ordem. Um THOP é um potencial de alta ordem que modela interações não locais em tempo polinomial, utilizando variáveis binárias, *cache* e reuso de informação. No domínio de desastre urbano, um modelo com variáveis binárias e THOPs foi apresentado, por Pujol-Gonzalez et al. (2014), aplicado ao problema de alocação de tarefas de desastre para agentes.

Nos modelos baseados em grafo-fator com variáveis binárias e THOPs, cada variável $x_i \in X$ é uma variável de decisão sobre um conjunto de variáveis binárias z_{ij}, onde $z_{ij} = 1$ se $x_i = d_j$ ou 0 caso contrário. Considere a seguinte notação:

- $X = \{x_1, \cdots, x_i, \cdots, x_n\}$ é o conjunto de variáveis de decisão, onde i pode variar de 1 até n;
- Para uma variável x_i, $D_i = \{d_{i1}, \cdots, d_{ij}, \cdots, d_{ik}\}$ é um conjunto finito de valores no seu domínio, onde k representa a cardinalidade do domínio de x_i;
- $Z = \{\forall x_i \in X, d_j \in D_i, z_{ij} \in \{0, 1\}\}$ é o conjunto de variáveis binárias que representam as combinações dos valores d_j para cada agente x_i;
- **z** representa um subconjunto de variáveis binárias no escopo de um THOP;

- F representa o conjunto de fatores que representam as restrições do problema.

Na Seção 3.3 foi mencionado que a complexidade do *max-sum* recai sobre o procedimento de maximização no cálculo das mensagens dos fatores para as variáveis, representado na Equação 3.3, possuindo complexidade de ordem exponencial. Pujol-Gonzalez et al. (2013) usaram o algoritmo de passagem de mensagens *binary max-sum* (BMS), que pode ser aplicado sobre um problema de otimização de restrições representado como um grafo-fator com variável binárias e THOPs, reduzindo a complexidade do cálculo das mensagens para polinomial. Para isso, o problema deve ser formalizado com variáveis binárias representando o domínio dos agentes.

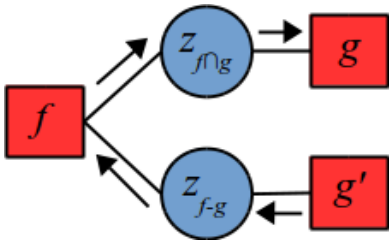
No BMS, considerando os nós fatores f e g conectados por uma variável binária, duas mensagens devem ser passadas entre f e g, sendo uma para cada estado da variável binária (0 e 1). Contudo, uma simplificação pode ser realizada de forma que seja enviado um valor único que corresponde a diferença entre os dois estados (ver Equação 3.7).

$$v_{f \to g} = \mu_{f \to g}(1) - \mu_{f \to g}(0) \tag{3.7}$$

O cálculo da mensagem de um nó fator f para um nó fator g é apresentado na Equação 3.8, cujo fluxo da mensagem pode ser ilustrado conforme a Figura 3.6. A função $f(.)$ tem como argumento o conjunto de variáveis binárias no escopo do fator f e o resultado é dependente do domínio do problema e do THOP que o fator representa.

$$\mu_{f \to g}(\mathbf{z}_{f \cap g}) = \max_{\mathbf{z}_{f-g}} \left[f(\mathbf{z}_{f \cap g}, \mathbf{z}_{f-g}) + \sum_{g' \in \eta(f)-g} \mu_{g' \to f(\mathbf{z}_{g' \cap f})} \right] \tag{3.8}$$

Figura 3.6: Fluxo da mensagem do nó fator f para o nó fator g no *binary max-sum*



3.5 *Tractable High Order Potentials* - THOPs

Um THOP é um nó fator com variáveis binárias z_{ij}, ao qual o *binary max-sum* pode computar as mensagens em tempo polinomial. Eles permitem representar restrições distintas para um problema de otimização de restrições distribuídas (DCOP). Considerando independência entre os agentes de um DCOP, cada agente pode calcular localmente as mensagens e enviá-las para os agentes vizinhos, ou seja, que possuem o mesmo THOP. Seja x_i um agente qualquer, o primeiro passo para representar o problema com THOPs consiste em dividir o domínio do agente em k variáveis binárias ($|D_i| = k$).

Um fator de seleção, também chamado de fator *OneAndOnlyOne*, impõe ao problema a restrição de que apenas uma das variáveis binárias esteja ativa (PUJOL-GONZALEZ et al., 2013). Esse fator pode ser usado para problemas de alocação de tarefas para agentes, para impor a restrição de que um agente deve resolver uma tarefa por vez (PUJOL-GONZALEZ et al., 2014). Considerando um valor de utilidade para cada valor no domínio de uma variável, o fator de seleção ordena os dois melhores valores, de forma que esse valor possa ser acessado diretamente pela variável. Para um dado problema de maximização, seja s_i um fator de seleção para um conjunto $\mathbf{z}$ de variáveis binárias de um agente x_i. A Equação 3.9 representa o cálculo da mensagem desse fator, cuja complexidade é $O(k)$, onde k é o número de variáveis binárias no seu escopo.

$$s_i(\mathbf{z}) = \begin{cases} 0 & \textit{se } \sum_{j=1}^{k} z_{ij} = 1 \\ -\infty & \textit{caso contrário} \end{cases} \tag{3.9}$$

Seja δ_{ij} um valor de utilidade de um valor d_j que pode ser atribuído por uma variável x_i. A utilidade é representada por um fator f_{ij}, cuja função $f_{ij}(z_{ij})$ retorna a utilidade do valor, quando a variável binária está ativa. A mensagem do fator f_{ij} pode ser acessada com complexidade $O(1)$.

$$f_{ij}(z_{ij}) = \begin{cases} \delta_{ij} & \textit{se } z_{ij} = 1 \\ 0 & \textit{caso contrário} \end{cases} \tag{3.10}$$

A utilidade total de um valor d_j pode ser representada como a soma das utilidades indi-

viduais de todas as variáveis que estão atribuindo d_j (ver Equação 3.11).

$$f_j(\mathbf{z}) = \sum_{i=1}^{n} f_{ij}(z_{ij}) \tag{3.11}$$

Como abordado na seção anterior, o *binary max-sum* permite um processo de simplificação da mensagem (ver Equação 3.7), de modo que a mensagem passada pelo grafo seja um valor único que corresponda a diferença entre os dois estados das variáveis binárias (PUJOL-GONZALEZ et al., 2013). A Equação 3.12 apresenta a mensagem simplificada que corresponde ao cálculo do fator de utilidade $f_{ij}(z_{ij})$ (Equação 3.10).

$$\begin{aligned} v_{f_{ij}\to s_i} &= \mu_{f_{ij}\to s_i}(1) - \mu_{f_{ij}\to s_i}(0) \\ v_{f_{ij}\to s_i} &= \delta_{ij} - 0 \\ v_{f_{ij}\to s_i} &= \delta_{ij} \end{aligned} \tag{3.12}$$

A mensagem do fator de seleção s_i para o fator de utilidade f_{ij} pode ser obtida através da Equação 3.8, conforme os passos apresentados na Equação 3.13. Considera-se que $\mu_{s_i\to f_{ij}}(1)$ é o valor da função de seleção quando a restrição é satisfeita.

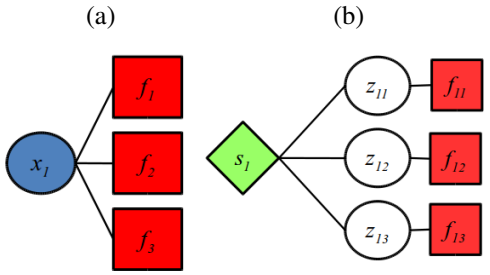
$$\begin{aligned} \mu_{s_i\to f_{ij}}(1) &= 0 \\ \mu_{s_i\to f_{ij}}(0) &= \max_{k\in\eta(s_i)\setminus j} \delta_{ik} \\ v_{s_i\to f_{ij}} &= \mu_{s_i\to f_{ij}}(1) - \mu_{s_i\to f_{ij}}(0) \\ v_{s_i\to f_{ij}} &= 0 - \max_{k\in\eta(s_i)\setminus j} \delta_{ik} \\ v_{s_i\to f_{ij}} &= - \max_{k\in\eta(s_i)\setminus j} \delta_{ik} \end{aligned} \tag{3.13}$$

A troca de mensagem entre os fatores de seleção e utilidade tem o objetivo de atualizar o valor da melhor escolha para um agente x_i. O valor que é enviado corresponde ao segundo melhor valor no domínio de um agente. Para calcular de forma eficiente, o fator de seleção mantém os dois melhores valores no domínio, representados pela tupla $\langle v^\star, v^{\star\star}\rangle$. Por fim, resume-se a mensagem do fator de seleção conforme a Equação 3.14.

$$v_{s_i\to f_{ij}} = \begin{cases} -v^\star & \textit{se } v_{f_{ij}\to s_i} \neq v^\star \\ -v^{\star\star} & \textit{se } v_{f_{ij}\to s_i} = v^\star \end{cases} \tag{3.14}$$

A Figura 3.7 ilustra a diferença entre um modelo DCOP representado como um grafo-fator tradicional (Figura 3.7a) e o modelo equivalente representado como um grafo-fator com variáveis binárias, o fator de seleção e o fator de utilidade (Figura 3.7b).

Figura 3.7: Modelo DCOP representado como um grafo-fator tradicional (3.7a) e modelo equivalente representado como um grafo-fator com variáveis binárias e THOPs (3.7b)



Em modelos com variáveis binárias, a utilidade total (Equação 3.11) pode ser representada por um fator de cardinalidade. A mensagem do fator de cardinalidade representa uma função definida sobre um conjunto de variáveis binárias ativas no seu escopo. Utilizando um algoritmo de programação dinâmica apresentado em Tarlow, Givoni e Zemel (2010), é possível calcular a mensagem do fator de cardinalidade em $O(n \log n)$, onde n é o número de variáveis binárias ativas no seu escopo.

O Algoritmo 2 apresenta o pseudo-código do algoritmo proposto por Tarlow, Givoni e Zemel (2010). Inicialmente, todas as mensagens são recebidas e ordenadas, tal que $\mu_1 \leq \mu_b \leq \mu_n$ (Linha 1). Uma função $r(b)$ retorna o índice reverso j da mensagem (Linha 2). Em um passo linear são computadas as somas cumulativas, representadas por u, u_{-1}, u_0 e u_1 (Linha 3), que representam, respectivamente, a soma total das utilidades, a soma com $b-1$, b e $b+1$ variáveis binárias ativas. As somas cumulativas armazenam o resultado da função de cardinalidade para cada combinação possível das variáveis binárias. Calculam-se, também, os vetores de máximos cumulativos (ou mínimos cumulativos dependendo do problema), representados por m_{-1}, m_{0r}, m_{0l} e m_1 (Linha 17). Por fim, com os máximos/mínimos cumulativos, calculam-se as mensagens que serão enviadas para os fatores vizinhos (Linha 28).

Algoritmo 2 Cálculo da mensagem do fator de cardinalidade - Adaptado de (TARLOW; GIVONI; ZEMEL, 2010; PUJOL-GONZALEZ et al., 2014)

$\mu = \{\mu_1, \cdots, \mu_b, \cdots, \mu_n\}$
$r(b) = j$
for $b = 0$ *até* $b = n$ **do**
    if $b = 0$ **then**
        $u[b] = 0$
    else
        $u[b] = u[b-1] + \mu_b$
    $u_0[b] = u[b] + f(b)$
    if $b > 0$ **then**
        $u_{-1}[b] = u[b] + f(b-1)$
    else
        $u_{-1}[b] = \infty$
    if $b < n$ **then**
        $u_1[b] = u[b] + f(b+1)$
    else
        $u_1[b] = \infty$
for $b = 0$ *até* n **do**
    if $b = 0$ **then**
        $m_1[b] = u_1[b]$
        $m_{0l}[b] = u_0[b]$
        $m_{0r}[n-b] = u_0[n-b]$
        $m_{-1}[n-b] = u_{-1}[n-b]$
    else
        $m_1[b] = \min(u_1[b], m_1[b-1])$
        $m_{0l}[b] = \min(u_0[b], m_{0l}[b-1])$
        $m_{0r}[n-b] = \min(u_0[n-b], m_{0r}[n-b+1])$
        $m_{-1}[n-b] = \min(u_{-1}[n-b], m_{-1}[n-b+1])$
for $b = 0$ *até* n **do**
    $j = r(b)$ # índice reverso
    $msg_0 = msg_1 = \infty$
    if $j > 0$ **then**
        $msg_0 = \min(msg_0, m_{0l}[j-1])$
        $msg_1 = \min(msg_1, m_1[j-1])$
    $msg_0 = \min(msg_0, m_{-1}[j+1] - \mu_j)$
    $msg_1 = \min(msg_1, m_{0r}[j+1] - \mu_j)$
    $msg = msg_1 - msg_0$
    ENVIARMENSAGEM(msg, j) # mensagem do fator de cardinalidade para j

3.6 Resumo

Esse capítulo introduziu o problema de otimização de restrições distribuídas (DCOP), um *framework* teórico utilizado para formalizar problemas de agendamento, quebra-cabeças, atribuição de tarefas, entre outros. Um DCOP é representado na forma de um grafo de restrições, sendo que abordagens mais recentes tem utilizado a representação em grafo-fator. Uma das motivações para a representação em grafo-fator é a garantia de convergência de algoritmos para inferência em grafos acíclicos. Contudo, em grafos cíclicos há evidências de que é possível encontrar boas soluções aproximadas (KSCHISCHANG; FREY; LOELIGER, 2001). A representação baseada em grafo também é bastante útil para modelar as interações entre grupos de agentes, que é o foco deste trabalho.

A Seção 3.3 apresentou o *max-sum*, que é um dos algoritmos de passagem de mensagem do estado-da-arte que pode ser utilizado em problemas representados como grafo-fator. A decomposição do problema em funções locais para agentes permite otimizar a função objetivo de maneira distribuída, de modo que cada variável escolha um valor para sua variável e propague uma mensagem de utilidade através dos vértices vizinhos. A Seção 3.4 apresentou a representação em grafo-fator com variáveis binárias e fatores tratáveis de alta ordem (THOP), bem como o cálculo das mensagens que são propagadas pelos nós do grafo.

4 TRABALHOS RELACIONADOS

Neste capítulo serão apresentadas abordagens relacionadas ao domínio de desastre urbano. Serão apresentadas abordagens aplicadas ao simulador *RoboCup Rescue Simulator* e a plataforma *RMASBench*, foco desse trabalho.

4.1 Abordagens inspiradas no modelo GAP

O problema de alocação de tarefas em ambientes de desastre urbano pode ser formalizado como um problema de atribuição generalizada, do inglês *Generalized Assignment Problem* (GAP), ou sua extensão E-GAP (SCERRI et al., 2005). Neste modelo, o objetivo consiste em encontrar a atribuição de tarefas para agentes que maximiza a recompensa esperada do sistema, que é obtida a partir das recompensas individuais dos agentes. Esse modelo possui duas restrições:

a) Cada agente pode resolver um número limitado de tarefas com base na quantidade de recursos disponíveis;
b) Cada tarefa deve ser resolvidas por no máximo um agente.

Scerri et al. (2005) apresentaram o E-GAP, uma extensão do modelo GAP que incorpora as características existentes em cenários dinâmicos. O modelo E-GAP amplia o escopo do GAP para incluir temporalidade, agentes heterogêneos e tarefas inter-relacionadas, ou seja, tarefas que precisam satisfazer um conjunto de restrições (e.g. um conjunto de tarefas que precisa ser resolvida em um mesmo instante). Diversos algoritmos para DCOP são inviáveis para resolver o E-GAP devido ao tempo e espaço necessário para computação da solução ótima (SANTOS, 2009, p.29). Scerri et al. (2005) propuseram o algoritmo LA-DCOP (*Low-communication Approximate DCOP*) que se propõe a resolver o E-GAP com uso de pouca comunicação. O LA-DCOP utiliza um protocolo de comunicação baseado em *tokens* para assegurar que a comunicação seja a menor possível, sendo que cada *token* é associado a uma tarefa e pertence unicamente ao agente que a encontrou. Quando a tarefa faz parte de um conjunto de tarefas inter-relacionadas, o agente toma para si todo o conjunto de tarefas, criando um *token* para cada, e começa um procedimento de recrutamento até que os agentes tenham aceitado resolver todas as tarefas no conjunto. Apesar do algoritmo ter sido proposto para resolver o E-GAP, no trabalho de Scerri et al. (2005) não foram abordados agentes heterogêneos ou tarefas inter-relacionadas.

Santos (2009) e Santos e Bazzan (2009) propuseram o algoritmo *eXtreme-ants* para o problema de alocação de tarefas considerando todas as características existentes no modelo E-GAP. Os agentes no *eXtreme-ants* decidem quais tarefas irão realizar baseado no modelo de divisão de trabalho existente em insetos sociais. A decisão de realizar ou não uma tarefa é probabilística com base na tendência do agente de realizá-la, sendo que a tendência é representada por uma função que considera um limiar de resposta do agente e o estímulo associado à tarefa. A função de tendência tem o objetivo de fornecer ao agente uma visão geral do ambiente com respeito as demais tarefas no ambiente. Assim como o LA-DCOP, o *eXtreme-ants* utiliza um protocolo de comunicação baseado em *token*. O agente possuidor do *token* tem o direito de decidir se irá alocar a tarefa associada a ele. Para tratar do problema de alocação de tarefas inter-relacionadas, os agentes no algoritmo *eXtreme-ants* utilizam um processo de recrutamento. O processo de recrutamento pode ser baseado em comunicação, ou comunicação e movimentação. No processo de recrutamento baseado em comunicação, um agente envia mensagens de recrutamento para agentes escolhidos aleatoriamente e aguarda a decisão destes, referente a aceitação, ou não, da tarefa. No processo por comunicação e movimentação a diferença é que os agentes podem movimentar-se pelo ambiente e executar outras tarefas enquanto aguardam por decisões de outros agentes. Uma característica que não é tratada no *eXtreme-ants* refere-se a como as tarefas não realizadas podem afetar a tendência da tarefa escolhida. No modelo proposto, os valores de estimativa das tarefas não realizadas são propagadas como mensagens para os grupos de agentes.

Um dos maiores problemas dos modelos baseados em GAP e E-GAP está na restrição de que uma mesma tarefa não pode ser resolvida por um grupo de agentes. Em ambientes de desastre urbano, isso pode implicar no entendimento de que um único agente deve ser capaz de resolver uma única tarefa. Outro ponto é que as abordagens apresentadas possuem um protocolo de comunicação baseado em *tokens*. O *token* associado a tarefa faz com que a tomada de decisão sobre a sua realização seja síncrona. No pior caso, cada *token* deverá percorrer todos os agentes até que seja resolvida.

4.2 Alocação de tarefas baseada em formação de agrupamentos

Quando um único agente não é suficiente para realizar uma tarefa é necessário formar grupos de agentes. A formação de grupos de agentes com base nas similaridades entre as suas características pode ser visto como um problema de formação de agrupamentos. O algoritmo mais simples para formação de agrupamentos é o k-médias, do inglês *k-means*, que posiciona

centróides (pontos centrais) para indicar os agrupamentos com base na média das instâncias do conjunto de dados. No entanto, o algoritmo é centralizado, requer conhecimento sobre a quantidade de grupos existentes e o resultado é sensível à inicialização dos centróides.

O *bee clustering* (SANTOS, 2009; SANTOS; BAZZAN, 2010; SANTOS; BAZZAN, 2012) é um algoritmo distribuído de formação de grupos inspirado no processo de recrutamento observado entre colônias de abelhas operárias. Cada agente representa um objeto que precisa ser agrupado. Especificamente no ambiente de desastre urbano, o *bee clustering* foi implementado como uma soluação para o problema de alocação de tarefas baseado em formação de agrupamentos. Neste cenário, os grupos representam as tarefas que devem ser resolvidas e os agentes devem decidir quais grupos irão participar. A tomada de decisão dos agentes é baseada nas ações: dançar, visitar e assistir. Um agente que está dançando tem o objetivo de recrutar/convidar outros agentes a fazerem parte de seu grupo. Um agente que está visitando outro deverá decidir se participará do grupo ou não. A ação de assistir representa a exploração do cenário, onde ocorre a procura de outros grupos de agentes que estão dançando para decidir sobre visitá-los ou não.

Uma limitação mencionada em Santos (2009) é o tempo de convergência do algoritmo, quando utilizada uma métrica baseada no cálculo do centróide sobre os atributos dos agentes de um grupo. Ainda, uma vez que o algoritmo não usa comunicação explícita, o desempenho do algoritmo irá depender dos mecanismos de exploração do cenário.

4.3 Modelos baseados em grafo-fator

As pesquisas mais recentes relacionados a ambientes de desastre urbano utilizam a representação em grafo-fator. Esta forma de representação possui garantia de convergência para uma solução ótima em grafos acíclicos. Nesta seção, será apresentado o modelo de formação de coalizões com restrições espaciais e temporais (CFST) e detalhado o modelo de alocação de tarefas para times.

4.3.1 Formação de Coalizões com Restrições Espaciais e Temporais (CFST)

No problema de formação de coalizões com restrições espaciais e temporais (RAMCHURN et al., 2010), do inglês *coalition formation with spatial and temporal constraints* (CFST), os agentes são representados como nós variáveis e as tarefas são representadas como

nós fator em um grafo-fator. Assume-se que cada agente é responsável por uma única variável, cujo domínio é o conjunto de tarefas que o agente consegue perceber em um período de tempo. Um agente (nó variável) estará conectado a uma tarefa (nó fator) se ela estiver em seu domínio e ele puder chegar nela em tempo hábil de resolvê-la.

Para a tomada de decisão, os agentes baseiam-se em funções de custo que representam a viabilidade da atribuição conforme as características de *deadline* e *workload* da tarefa. O *deadline* representa o tempo máximo da tarefa até que ela não possa mais ser resolvida. O *workload* representa a carga de trabalho (em unidades de tempo) necessária para resolver a tarefa. O CFST aborda o problema de alocação de tarefas como um problema de agendamento, onde um conjunto de agentes que decide realizar uma mesma tarefa em um mesmo período de tempo representa uma coalizão. A coalizão reduz a carga de trabalho necessária para resolver determinada tarefa (e.g. extinguir os incêndios no cenário) e as restrições são modeladas como funções de custo que indicam o sucesso na extinção dos incêndios.

O modelo CFST foi otimizado com o algoritmo *max-sum* que, conforme abordado na Seção 3.3, possui complexidade exponencial com relação a computação das mensagens dos fatores para as variáveis.

4.3.2 Modelo para formação de grupos com grafo-fator estruturado em árvore

No modelo proposto por Corrêa (2014), denominado apenas por *factoring*, o problema de formação de grupos em ambientes de desastre urbano foi formalizado com um grafo-fator acíclico utilizando um algoritmo de passagem de mensagens em duas direções. Utilizou-se uma representação em grafo-fator estruturado em árvore, de modo a representar a hierarquia das escolhas dos agentes, e uma variação do algoritmo *sum-product message passing*. A troca de mensagens ocorre dos nós folhas (nós mais baixos na árvore) para o nó raiz (nó mais alto na árvore) e do nó raiz para os nós folhas. O nó raiz representa um agente central que irá assegurar a comunicação dos agentes que estão nos níveis mais baixos na hierarquia. Os agentes são representados como nós variável e os grupos são representados pelos nós fator.

Na criação de um grupo, um dos agentes é eleito líder e irá manter a comunicação com o agente central, os demais agentes se comunicam apenas com o líder do grupo. A seguir, as mensagens são enviadas para os líderes dos grupos até a raiz do grafo-fator. Ao chegar na raiz (agente central), a função objetivo do problema é calculada e uma mensagem de retorno é propagada para os agentes subordinados até os agentes folha (nós mais baixo na hierarquia). O modelo proposto em Corrêa (2014) foi avaliado em um ambiente dinâmico e genérico de

simulação de desastre. A passagem de mensagens ocorre a cada *timestep* de simulação e não a cada iteração do algoritmo, como ocorre na abordagem proposta nessa dissertação. O algoritmo foi comparado com o *bee clustering*, apresentando melhor desempenho com relação à resolução das tarefas no cenário.

De certa maneira, a proposta apresentada nessa dissertação pode ser vista como uma extensão do modelo apresentado em Corrêa (2014), sendo que o procedimento de passagem de mensagens é assíncrono. Optou-se por não comparar o FGMD com o modelo *factoring* pois seria necessário alterações na estrutura do simulador *RMASBench*.

4.3.3 Modelo para alocação de tarefas multi-times

Pujol-Gonzalez et al. (2014) apresentaram o modelo de alocação de tarefas multi-times, do inglês *multi-team task allocation* (MTTA), um dos primeiros trabalhos no domínio de desastre urbano que aborda o problema de coordenação entre times diferentes de agentes (e.g.: um agente bombeiro pode solicitar apoio ao policial para desbloquear uma rua específica, permitindo que o bombeiro possa alcançar o prédio em chamas que ele considerar mais adequado). Os autores representaram o problema como um grafo-fator com variáveis binárias e usaram o algoritmo de passagem de mensagens *binary max-sum* para otimizar uma função objetivo, que representa a utilidade de times de agentes realizando tarefas de desastre.

Seja $i \in I$ um agente bombeiro e $j \in J$ uma tarefa de incêndio, cada agente controla uma variável de decisão $y_i \in Y$ e deve escolher um valor $d_j \in D_i$. Assim, $y_i = d_j$ se o agente i estiver realizando a tarefa j. O objetivo do MTTA é encontrar uma atribuição completa $\mathbf{Y}^\star$ de valores para as variáveis de decisão, que maximiza a utilidade do sistema, representada por $u(\mathbf{Y}) = \sum_{j \in J} u_j(\mathbf{Y})$, onde $u_j(\mathbf{Y})$ é a função de utilidade do time de agentes. A Equação 4.1 apresenta a utilidade do time, que é dividida em dois termos que representam a eficiência na solução da tarefa ($e_j(\mathbf{Y})$) e o custo da atribuição ($r_j(\mathbf{Y})$).

$$u_j(\mathbf{Y}) = e_j(\mathbf{Y}) - r_j(\mathbf{Y}) \tag{4.1}$$

Cada valor $d_j \in D_i$ possui um valor de utilidade v_j, obtido quando $x_i = d_j$. Para que todos os agentes não decidam trabalhar em uma mesma tarefa, o modelo impõe uma quantia limite de agentes que devem escolher a tarefa. O limite depende de parâmetros que incluem o *fieryness* de um incêndio, a área da construção atingida pelo incêndio ($area_j$) e uma constante que representa o alcance da visão dos agentes ($radius$). O *fieryness* é uma característica do

RoboCup Rescue Simulator, que representa a dificuldade na execução da tarefa. O limite de agentes é representado por t_j cujo cálculo é apresentado na Equação 4.2.

$$t_j = \lceil \frac{area_j}{radius} \rceil \times fieryness_j \tag{4.2}$$

Assim, quando uma única tarefa possuir um número de agentes maior que t_j a eficiência dos agentes é penalizada conforme a Equação 4.3, onde $\kappa \geq 0$ e $\gamma \geq 1$ são coeficientes arbitrários para controlar o aumento da penalidade e $n_j(\mathbf{Y})$ é uma função de contagem dos agentes que escolheram a tarefa j.

$$e_j(\mathbf{Y}) = v_j \times n_j(\mathbf{Y}) - \kappa \times [\max(0, n_j(\mathbf{Y}) - t_j)]^\gamma \tag{4.3}$$

A função $r_j(\mathbf{Y})$ representa o custo da atribuição do agente. Como na abordagem os agentes são equivalentes, exceto na sua localização, o custo será proporcional ao quadrado da distância entre os agentes. Na Equação 4.4 a distância espacial entre um agente i e uma tarefa j, representada pela função $dist(i, j)$, é normalizada e um coeficiente $\nu \geq 0$ é usado para ponderar o *fieryness* e a distância espacial de uma tarefa.

$$\begin{aligned} r_j(\mathbf{Y}) &= \sum_{i \in I} r_{ij}(y_i) \\ r_{ij}(y_i) &= \begin{cases} \nu \times dist(i,j)^2 & \textit{se } y_i = d_j \\ 0 & \textit{caso contrário} \end{cases} \end{aligned} \tag{4.4}$$

No *RoboCup Rescue Simulator* (RCRS), agentes policiais devem se coordenar para remover bloqueios das ruas e liberar o caminho para assegurar a mobilidade no cenário. Seja $p \in P$ um agente policial e $b \in B$ uma tarefa de bloqueio no seu domínio. No modelo MTTA, os policiais são representados por um conjunto de variáveis binárias $X = \{x_{pb} | p \in P, b \in B\}$, onde x_{pb} estará ativa se um policial está atribuído a um bloqueio, inativo caso contrário. Duas restrições devem ser observadas com relação aos policiais:

a) Um policial não poderá remover mais de um bloqueio por vez;
b) No *RMASBench*, dois policiais trabalhando juntos em um bloqueio não apresentarão resultado diferente de um único policial sozinho. Assim, cada bloqueio deverá ser realizado por apenas um agente policial.

Para os policiais, a utilidade do sistema é representada por $u(\mathbf{X})$, sendo decomposta em

funções menores para cada bloqueio, tal que $u(\mathbf{X}) = \sum_{b \in B} u_b(\mathbf{X})$, representada na Equação 4.5.

$$u_b(\mathbf{X}) = e_b(\mathbf{X}) - r_b(\mathbf{X}) \tag{4.5}$$

Considera-se que todos os policiais são equivalentes em suas habilidades e os bloqueios não tem características que os distinguem. No modelo, atribui-se utilidade $v_B > 0$ para qualquer bloqueio que estiver sendo atendido. Na Equação 4.6, $n_b(\mathbf{X})$ é o número de policiais atribuídos ao bloqueio b.

$$e_b(\mathbf{X}) = \begin{cases} v_B & \textit{se } n_b(\mathbf{X}) \geq 1 \\ 0 & \textit{caso contrário} \end{cases} \tag{4.6}$$

Para avaliar o custo de um policial p atendendo um bloqueio b, devem ser observadas as seguintes condições: se o bloqueio está diretamente acessível ao policial, isto é, entre um policial p e um bloqueio b não há outro bloqueio b'. Neste caso, o custo é proporcional ao quadrado da distância entre eles. Se o bloqueio não é acessível diretamente é acrescida uma penalidade Q à distância. A Equação 4.7 apresenta a função de custo de um bloqueio, onde η é uma constante de proporcionalidade que deve ser ajustada para o caso de coordenação entre bombeiros e policiais.

$$\begin{aligned} r_b(\mathbf{X}) &= \sum_{p \in P} r_{pb}(x_{pb}) \\ r_{pb}(x_{pb}) &= \begin{cases} 0 & \textit{se } x_{pb} \textit{ está inativo} \\ \eta \times dist(p,b)^2 & \textit{se } x_{pb} \textit{ está ativo e } b \textit{ é acessível} \\ \eta \times dist(p,b)^2 + Q & \textit{caso contrário} \end{cases} \end{aligned} \tag{4.7}$$

Pujol-Gonzalez et al. (2014) apresentam como o modelo MTTA deve ser transformado em uma versão binária, para que seja otimizado utilizando o algoritmo *binary max-sum*. O MTTA foi avaliado através da plataforma *RMASBench*, porém o algoritmo é sensível aos parâmetros de inicialização. Os resultados apresentados no Capítulo 6 demonstram que em certas condições o MTTA requer mais uso de comunicação do que o modelo FGMD.

4.4 Busca estocástica distribuída

O *Distributed Stochastic Algorithm* (DSA) (ZHANG et al., 2003) é um algoritmo de busca distribuída, aplicado a modelos formalizados como problemas de otimização de restrições (COP ou DCOP). Na plataforma *RMASBench* (KLEINER et al., 2013), o DSA é aplicado ao problema de alocação de tarefas de desastre para agentes bombeiros e policiais.

No DSA cada agente escolhe uma tarefa aleatoriamente, podendo alterar sua escolha com probabilidade DSA_p. Caso o agente alterou a tarefa escolhida, é enviado uma mensagem para cada agente vizinho. Quando as mensagens são recebidas, os agentes podem alterar a sua escolha com probabilidade DSA_p de modo a satisfazer as restrições do problema. No *RMAS-Bench*, os agentes alteram sua escolha de modo a maximizar a utilidade da tarefa escolhida e minimizar a penalidade sobre as alocações, considerando que cada tarefa deve ser feita por uma quantidade limitada de agentes.

Uma das vantagens do DSA é o pouco uso de comunicação, uma vez que se trata de um método de busca local. No Capítulo 6, o DSA é comparado com a abordagem proposta no simulador *RMASBench*. Observou-se que, na maioria dos casos, os resultados obtidos com o DSA são semelhantes às demais abordagens, no que tange à resolução das tarefas de incêndio. Com relação à comunicação e verificação de restrições, o DSA apresentou os melhores resultados entre todos os algoritmos.

4.5 Resumo

Os modelos inspirados no problema de atribuição generalizada (GAP) lidam com restrições de recursos, onde os agentes devem escolher as tarefas que irão realizar de modo a maximizar a recompensa do sistema. Esse objetivo é alcançado quando os agentes escolhem tarefas de modo a maximizar as suas competências. Discutiu-se sobre os algoritmos LA-DCOP e *eXtreme-ants*, ambos aplicados ao problema de alocação de tarefas para agentes formalizado como um GAP. Ambos algoritmos se propõem a resolver o problema com uso de pouca comunicação, onde o LA-DCOP utiliza um protocolo baseado em *token* para reduzir a comunicação e o *eXtreme-ants* utiliza a metáfora da divisão do trabalho dos insetos sociais. Um ponto negativo dos modelos baseados em GAP é que possuem uma restrição de exclusão mútua, a qual não permite modelar explicitamente as interações de grupos de agentes.

No modelo de alocação de tarefas baseado em formação de agrupamentos foi apresentado o algoritmo *bee clustering*. Esse algoritmo se propõe a resolver o problema de alocação

de tarefas de uma forma mais genérica e é inspirado na inteligência de colônias de abelhas, para realizar a formação de grupos através da minimização da distância entre os membros do grupo. O algoritmo utiliza apenas informação local para a tomada de decisão dos agentes e o desempenho depende da métrica de distância utilizada. O *bee clustering* foi avaliado no cenário do *RoboCup Rescue Simulator* (RCRS) obtendo resultado superior ao algoritmo de busca gulosa pela melhor escolha. Contudo, por não ser um modelo formalizado como um DCOP sua avaliação no *RMASBench* é inviável. Devido a tomada decisão baseada em informação local, o *bee clustering* obtém vantagem sobre os demais modelos em cenários com restrições de comunicação, sendo que o desempenho dos agentes no cenário depende também da exploração em busca de novas tarefas.

Nos modelos representados como grafo-fator, destacam-se o modelo de formação de coalizões com restrições espaciais e temporais (CFST) e o modelo de alocação de tarefas multi-time (MTTA). No CFST, a decisão dos agentes é baseada em informações sobre o *deadline* e a carga de trabalho da tarefa, o que em cenários realísticos não são valores conhecidos. A solução ótima do CFST consiste em agendar as tarefas que irão ser realizadas ao longo do tempo de simulação, formando uma lista de tarefas para cada agente, baseado nas características de dificuldade e distância de cada tarefa observada. Porém, os autores alegam que encontrar a solução ótima envolve computar todos os possíveis planos para todas as possíveis coalizões de agentes, o que é intratável, como visto na Subseção 4.3.1. Outra abordagem apresentada é o modelo de formação de grupos proposto por Corrêa (2014), que utiliza uma variação do algoritmo *sum-product message passing*. O modelo apresentado em Corrêa (2014) utiliza comunicação síncrona e direcionada em um grafo-fator estruturado em árvore. Por fim, o modelo MTTA utiliza a representação de grafo-fator com variáveis binárias e THOPs para calcular de forma eficiente uma função objetivo que representa a maximização da utilidade de tarefas de incêndio e minimização da penalidade de agentes trabalhando em uma mesma tarefa (PUJOL-GONZALEZ et al., 2014). O MTTA será comparado com o modelo proposto nessa dissertação.

O algoritmo mais simples apresentado, *Distributed Stochastic Algorithm* (DSA), é apresentado na Seção 4.4. Basicamente, é um algoritmo distribuído de busca local gulosa, que utiliza pouca comunicação e consegue obter resultados satisfatórios em ambientes de desastre.

5 FORMAÇÃO DE GRUPOS PELA MINIMIZAÇÃO DA DISTÂNCIA

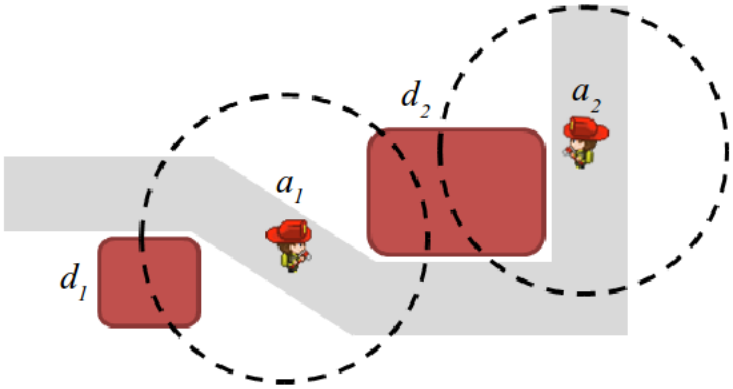
Este capítulo introduz o modelo Formação de Grupos pela Minimização da Distância (FGMD), que se propõe a resolver o problema de alocação de tarefas baseado em formação de grupos, aplicado a ambientes de desastre urbano. O FGMD é inspirado em abordagens como o algoritmo *bee clustering*, apresentado em Santos (2009), Santos e Bazzan (2010) e Santos e Bazzan (2012), no modelo de grafo-fator com variáveis binárias descrito por Pujol-Gonzalez et al. (2014) e no algoritmo de aprendizado não supervisionado para seleção de k-vizinhos proposto por Tarlow et al. (2013). Os agentes no FGMD são autônomos, podendo reagir rapidamente às mudanças no ambiente e interagindo com outros agentes através da troca de mensagens em grupos. Uma mensagem de um agente para um grupo representa o custo da sua escolha dada a sua observação do ambiente em uma unidade de tempo.

5.1 Descrição do problema

Na Subseção 2.3.1 foi apresentado o *RoboCup Rescue Simulation* (RCRS), um simulador para situações de busca e resgate em ambientes de desastre urbano, onde agentes policiais, bombeiros e paramédicos devem coordenar suas ações para resolver as tarefas existentes e minimizar os danos no cenário. O RCRS provê um ambiente dinâmico e incerto, onde tarefas podem surgir ou desaparecer dadas as escolhas dos agentes. O *RMASBench*, apresentado na Subseção 2.3.2, é uma plataforma de avaliação de algoritmos para coordenação multiagente, aplicado em problemas formalizados como um DCOP. O *RMASBench* é executado sobre o RCRS e utiliza um agente central para viabilizar um cenário com comunicação completa e distribuir as tarefas para múltiplos agentes DCOP, que deverão interagir e escolher quais tarefas de desastre deverão ser realizadas em cada *timestep* de simulação.

A Figura 5.1 ilustra um exemplo de cenário, que será utilizado ao longo dessa seção para exemplificar o modelo proposto. Neste cenário, existem dois agentes bombeiros, representados pelo conjunto $A = \{a_1, a_2\}$, e duas tarefas de incêndio, representadas pelo conjunto $D = \{d_1, d_2\}$. Cada agente $a_i \in A$ é responsável por uma variável $x_i \in X$ e possui um domínio finito e discreto D_i com valores definidos de acordo com o alcance da visão dos agentes, representado na Figura 5.1 pelo círculo tracejado. Por facilidade, será utilizado x_i para representar tanto o agente e sua variável, tal que $x_i = d_j$, significa que x_i atribui um valor $d_j \in D_i$.

Figura 5.1: Exemplo de cenário com dois agentes e duas tarefas



Cada agente deverá escolher tarefas de modo a minimizar os danos no ambiente. Por exemplo, agentes bombeiros devem apagar incêndios em prédios e policiais devem remover bloqueios de ruas. Um ponto observado em Pujol-Gonzalez et al. (2014) é que tarefas de incêndio mais recentes são mais propensas a atingir outros prédios do que as tarefas de incêndio mais antigas. Na Equação 5.1, a função de utilidade δ_{ij} calcula o valor de uma tarefa d_j para um agente x_i, onde I_j representa a característica *fieryness* da tarefa de incêndio d_j. Essa característica representa a dificuldade da tarefa, onde I_{max} é o valor máximo de *fieryness*, tal que $I_{max} = 4$ significa que o prédio em chamas não pode mais ser recuperado. A distância espacial entre um agente e uma tarefa é representada por $dist(i, j)$, tal que $0 \leq dist(i, j) \leq 1$. Uma vez que a distância é normalizada, usa-se o parâmetro $\nu \geq 0$ para ponderar o valor da distância espacial em relação a dificuldade da tarefa. Quanto maior o valor de δ_{ij}, mais fácil será a tarefa d_j com relação ao agente x_i.

$$\delta_{ij} = (I_{max} - I_j) - (\nu \times dist(i, j)^2) \tag{5.1}$$

Seja $\max_{\delta_{ij} \in D_i} \delta_{ij}$ a tarefa de maior utilidade no domínio de x_i, a distância entre uma dada tarefa d_j e a tarefa de maior utilidade é representada por uma função exponencial $q_{ij} : x_i \rightarrow \mathbb{R}$, apresentada na Equação 5.2. A função $q_{ij}(x_i)$ é uma métrica de distância baseada no domínio do agente.

$$q_{ij}(x_i) = -\delta_{ij} e^{-\frac{\delta_{ij}}{\max_{\delta_{ij} \in D_i} \delta_{ij}}} \tag{5.2}$$

Os agentes que decidem realizar a mesma tarefa irão formar grupos e trocar mensagens, com o objetivo de distribuir os agentes adequadamente para minimizar os danos no cenário. Uma função de partição, $p_j : \mathbf{x} \rightarrow \mathbb{R}$, formaliza a distribuição da carga de trabalho das tarefas entre os agentes, de modo que sejam encontrados grupos com quantidades adequadas de

agentes, considerando as tarefas percebidas por eles.

$$p_j(\mathbf{x}) = \begin{cases} \min\limits_{\mathbf{x}} \sum\limits_{x_i \in \mathbf{x}} \min\limits_{d_{k \setminus d_j} \in D_i} q_{ik}(x_i) & \forall x_i \in \mathbf{x},\ \textit{se } n < W_j \\ \min\limits_{\mathbf{x}} \dfrac{W_j}{n[n-(W_j-1)]} + \sum\limits_{x_i \in \mathbf{x}} \min\limits_{d_{k \setminus d_j} \in D_i} q_{ik}(x_i) & \textit{caso contrário} \end{cases} \tag{5.3}$$

O primeiro termo da Equação 5.3 representa a distribuição da carga de trabalho (W_j), entre todos os agentes que decidiram realizar a mesma tarefa (n). Se $n < W_j$ então seu valor resultará em 0. O segundo termo representa as distâncias entre as tarefas observadas pelos agentes que estão no mesmo grupo. Os agentes devem escolher tarefas visando minimizar o valor da função $p_j(\mathbf{x})$, e usa-se passagem de mensagens entre os agentes para reorganizar os grupos, de modo que a carga de trabalho dos agentes no grupo seja a menor possível.

O comportamento da função de partição, $p_j(\mathbf{x})$, varia devido à condição que é utilizada. Na Equação 5.3 o valor de p_j é condicionado à $n < W_j$. Assim, observou-se que o valor da carga de trabalho será 1 quando $n = W_j$. Por exemplo, considerando $W_j = 2$ e um dado grupo composto dos agentes x_1 e x_2, a distribuição de carga de trabalho que minimiza a distância será resultante da Equação 5.4.

$$\begin{aligned} p_j(x_1, x_2) = \\ \min\{(\min_{d_k \in D_1} q_{1k}(x_1)), \\ (\min_{d'_k \in D_2} q_{2k'}(x_2)), \\ [1 + (\min_{d_k \in D_1} q_{1k}(x_1) + \min_{d_{k'} \in D_2} q_{2k'}(x_2))]\} \end{aligned} \tag{5.4}$$

O problema descrito será formalizado como um problema de otimização de restrições distribuídas (DCOP) representado como um grafo-fator com variáveis binárias e nós tratáveis de alta ordem (THOPs). Assumindo independência entre as variáveis, cada agente será representado por um grafo-fator com elementos que representam seu domínio, a função de distância entre as tarefas no domínio e a função de partição dos grupos. A computação realizada pelos agentes é local e eles trocam mensagens para encontrar o valor que minimiza a função de partição.

5.2 Representação do Problema

O algoritmo *binary max-sum* (BMS) é utilizado para otimizar de forma eficiente o modelo de Formação de Grupos pela Minimização da Distância (FGMD). Para isso, o problema precisa ser representado como um grafo-fator com variáveis binárias e THOPs. A Seção 3.5 apresentou os THOPs que serão utilizados ao longo dessa seção. Como o modelo FGMD baseia-se na minimização de uma função de partição sobre as distâncias dos agentes e seus domínios, o algoritmo BMS foi adaptado para a operação de minimização. A Equação 5.5 apresenta uma adaptação do cálculo das mensagens de um fator f para um fator g, que como visto na Seção 3.4, pode ser simplificada para tornar a computação mais eficiente.

$$\mu_{f\to g}(\mathbf{z}_{f\cap g}) = \max_{\mathbf{z}_{f-g}} \left[f(\mathbf{z}_{f\cap g}, \mathbf{z}_{f-g}) + \sum_{g'\in\eta(f)-g} \mu_{g'\to f(\mathbf{z}_{g'\cap f})} \right] \tag{5.5}$$

Para a representação do problema, usam-se variáveis binárias z_{ij}, tal que uma variável binária z_{ij} está ativa ($z_{ij} = 1$) se uma tarefa d_j estiver atribuída a um agente x_i, ou inativa ($z_{ij} = 0$) caso contrário. Uma vez que cada agente só pode fazer uma tarefa por vez, uma restrição é imposta para assegurar que cada agente tenha apenas uma variável binária ativa. Para isso utiliza-se um fator de seleção, que representa uma *hard constraint* sobre o conjunto de variáveis binárias de um agente x_i (ver Equação 5.6). No problema representado na Figura 5.1, o domínio dos agentes x_1 e x_2 serão representados pelas variáveis binárias z_{11}, z_{12} e z_{22}.

$$\forall x_i \in X, \sum_{j=1}^{k} z_{ij} = 1 \tag{5.6}$$

Um fator de seleção, s_i, é conectado ao conjunto de variáveis binárias de um agente. O fator de seleção deve garantir que seja satisfeita a restrição formalizada na Equação 5.6 e selecionar o melhor valor para a variável do agente. Para isso, uma função $s_i(\mathbf{z})$, apresentada na Equação 5.7, é utilizada para assegurar que apenas uma variável binária esteja ativa, retornando o pior valor se a restrição não for satisfeita.

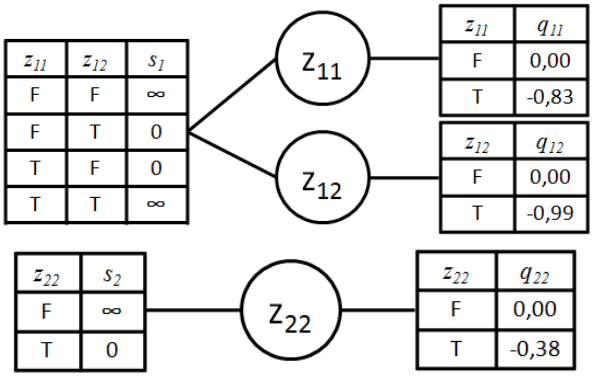
$$s_i(\mathbf{z}) = \begin{cases} 0 & \textit{se } \sum_{j=1}^{k} z_{ij} = 1 \\ \infty & \textit{caso contrário} \end{cases} \tag{5.7}$$

Na Seção 5.1 foi mencionado que cada tarefa no domínio de um agente possui um valor de utilidade, δ_{ij}, e uma métrica de distância denotada pela função $q_{ij}(x_i)$, que representa a distância entre duas tarefas distintas no domínio de x_i. A escolha do agente, sobre qual tarefa realizar, deve ser feita com o objetivo de diminuir a distância entre as tarefas, resultando na resolução de todas as tarefas do seu domínio. Usa-se um fator de custo, q_{ij}, para representar a distância de cada tarefa no domínio do agente. Um fator de custo é conectado a uma variável binária e a mensagem enviada por esse fator é a distância da tarefa representada pela variável binária, quando ativa. A Equação 5.8 apresenta o cálculo da mensagem do nó fator, onde $q_{ij}(x_i)$ é a função representada na Equação 5.2.

$$q_{ij}(z_{ij}) = \begin{cases} q_{ij}(x_i) & \textit{se } z_{ij} = 1 \\ 0 & \textit{caso contrário} \end{cases} \tag{5.8}$$

Tarlow, Givoni e Zemel (2010) abordam o uso combinado de THOPs, sendo possível acessar as mensagens do fator de custo diretamente pelo fator de seleção, reduzindo a quantidade de mensagens trocadas no grafo-fator. A Figura 5.2 mostra uma representação tabular do problema ilustrado pela Figura 5.1.

Figura 5.2: Representação tabular da composição do fator de seleção e do fator de custo para o problema representado na Figura 5.1.



Como visto na Seção 3.4, no *binary max-sum* considera-se que as mensagens fluem de um nó fator para outro através de suas variáveis binárias. Com isso, usa-se o procedimento de simplificação da mensagem descrito em Pujol-Gonzalez et al. (2013), o qual a mensagem trocada entre os fatores representa a diferença entre os dois estados das variáveis binárias.

Dada a representação tabular, uma mensagem do fator de custo q_{ij} para o fator de seleção s_i pode ser acessada em $O(1)$, conforme a Equação 5.9.

$$\begin{aligned} v_{q_{ij}\to s_i} &= \mu_{q_{ij}\to s_i}(1) - \mu_{q_{ij}\to s_i}(0) \\ v_{q_{ij}\to s_i} &= q_{ij}(x_i) - 0 \\ v_{q_{ij}\to s_i} &= q_{ij}(x_i) \end{aligned} \tag{5.9}$$

A mensagem do fator de seleção para o fator de custo pode ser obtida através da Equação 5.5, conforme os passos apresentados na Equação 5.10. A complexidade do cálculo da mensagem do fator de seleção é $O(n)$, onde n é o número de variáveis binárias no seu escopo.

$$\begin{aligned} \mu_{s_i\to q_{ij}}(1) &= 0 \\ \mu_{s_i\to q_{ij}}(0) &= \min_{k\in\eta(s_i)\setminus j} q_{ik}(x_i) \\ v_{s_i\to q_{ij}} &= \mu_{s_i\to q_{ij}}(1) - \mu_{s_i\to q_{ij}}(0) \\ v_{s_i\to q_{ij}} &= 0 - \min_{k\in\eta(s_i)\setminus j} q_{ik}(x_i) \\ v_{s_i\to q_{ij}} &= -\min_{k\in\eta(s_i)\setminus j} q_{ik}(x_i) \end{aligned} \tag{5.10}$$

Para calcular a mensagem do fator de seleção de forma eficiente, são armazenadas as duas melhores entradas recebidas, representadas por uma tupla $\langle v^\star, v^{\star\star}\rangle$. Resultando na simplificação da mensagem do fator de seleção conforme a Equação 5.11.

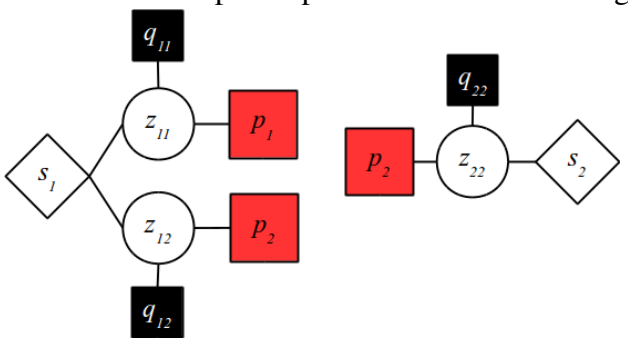
$$v_{s_i\to q_{ij}} = \begin{cases} -v^\star & \textit{se } v_{q_{ij}\to s_i} \neq v^\star \\ -v^{\star\star} & \textit{se } v_{q_{ij}\to s_i} = v^\star \end{cases} \tag{5.11}$$

Para reorganizar as escolhas dos agentes e distribuí-los, visando minimizar a distância entre as tarefas, usa-se uma função de partição $p_j(\mathbf{x})$ (Equação 5.3). A função $p_j(\mathbf{x})$ é um caso particular de fator de cardinalidade, que representa uma função sobre o número de variáveis binárias ativas no seu escopo (TARLOW et al., 2012). Utilizando o Algoritmo 2, apresentado na Seção 3.4, é possível calcular a mensagem do fator de cardinalidade em $O(n \log n)$. No modelo apresentado, utiliza-se o fator de cardinalidade, p_j, para representar o grupo de agentes que está

realizando uma tarefa d_j, tal que cada grupo está conectado às variáveis binárias z_{ij} dos seus membros. A mensagem do agente para o grupo representa a carga de trabalho presumida por ele para uma tarefa que não será realizada. A mensagem do grupo para cada agente representa a distribuição da carga de trabalho total, considerando as mensagens dos agentes.

A Figura 5.3 apresenta a representação em grafo-fator para o problema tratado nesse capítulo. Os fatores de seleção são representados pelo losangos, s_1 e s_2. As variáveis binárias são representadas por círculos, sendo um para cada combinação de agente e tarefa, denotados por z_{11}, z_{12} e z_{22}. Os fatores de custo são representados pelos quadrados na cor preta, denotados por q_{11}, q_{12} e q_{22}, e o fator de cardinalidade representa os grupos possíveis de agentes, de modo que o índice j representa uma tarefa d_j (p_1 e p_2).

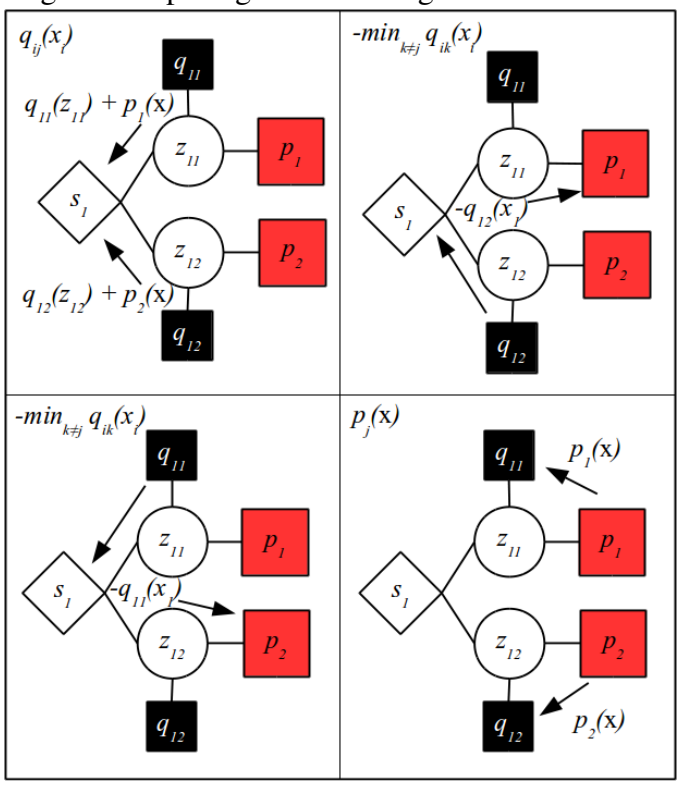
Figura 5.3: Grafo-fator para o problema ilustrado na Figura 5.1



Cada agente cria as estruturas necessárias para a tomada de decisão e computa as mensagens localmente, escolhendo o valor que minimiza a distância e enviando mensagens para os agentes vizinhos através do fator de cardinalidade. Ao receber as mensagens, os agentes calculam novamente as funções locais, podendo escolher novos valores, enviando mensagens novamente. Esse procedimento ocorre até que os agentes não alterem o valor de suas variáveis.

Para garantir o envio e recebimento de mensagens entre os agentes, uma interface de comunicação é requerida. A Figura 5.4 ilustra o procedimento de passagem de mensagens entre os fatores de um agente. O fator de cardinalidade p_j representa a interação entre os agentes no grupo, sendo que sua mensagem pode ser calculada localmente devido as variáveis binárias e a troca de mensagens entre os agentes no grupo. O Algoritmo 3 descreve o procedimento de criação do grafo-fator e passagem de mensagens que é utilizado em cada *timestep* de simulação. No início do *timestep*, cada agente cria as estruturas do grafo-fator necessárias para a passagem de mensagens (Linha 4). Após a criação do grafo-fator de cada agente, tem início o procedimento de passagem de mensagens, que termina quando o algoritmo converge ou quando é alcançado o limite de iterações (Linha 11). Por fim, cada agente poderá acessar diretamente o valor que minimiza a distância, acessando o valor $v^\star$ no fator de seleção s_i (Linha 19).

Figura 5.4: Diagrama da passagem de mensagens entre os fatores do agente x_i



5.3 Resumo

Esse capítulo apresentou o modelo Formação de Grupos pela Minimização da Distância (FGMD), que será aplicado ao problema de alocação de tarefas baseado em formação de grupos de agentes. O modelo propõe-se a organizar grupos de agentes para minimizar uma métrica de distância que representa a diferença de utillidade entre tarefas de incêndio em uma cidade simulada na plataforma *RMASBench*.

A Seção 5.1 apresentou sua definição e formalização como um problema de otimização de restrições distribuídas representado como um grafo-fator, com estruturas que representam as variáveis do problema, o domínio e as restrições. Na Seção 5.2 foi apresentada a representação do problema como um grafo-fator com variáveis binárias e nós tratáveis de alta ordem (THOPs) para representar as restrições do problema como funções de custo sobre variáveis binárias.

O FGMD é inspirado em algoritmos para minimização de distância, como o *kNCA* (*k Neighbourhood Component Analysis*), *SNE* (Stochastic Neighbour Embedding), propostos por Tarlow et al. (2013) e o algoritmo distribuído para formação de agrupamentos, *bee clustering*,

Algoritmo 3 Procedimento de criação do grafo-fator e passagem de mensagens

1: $X = \{x_1, \cdots, x_i, \cdots, x_n\}$ # conjunto de agentes
2: $D = \{D_1, \cdots, D_i, \cdots, D_n\}$ # domínio dos agentes
conjunto de todos os fatores
3: $F = \{\{\{q_{ij}\} \cup \{s_i\} \cup \{p_j\}\}, \forall x_i \in X, \forall d_j \in D_i\}$
Inicialização do grafo-fator
4: **for** $x_i \in X$ **do**
5: CRIAR(s_i)
6: **for** $d_j \in D_i$ **do**
7: CRIAR(q_{ij})
8: CONECTAR($q_{ij} \rightarrow s_i$)
9: CRIAR(p_j)
10: CONECTAR($p_j \rightarrow q_{ij}$)
11: **for** $itr = 0$ até $itr < itr_{max}$ ou seja alcançada condição de término **do**
12: **for** $x_i \in X$ **do**
13: **for** $q_{ij} \in F$ **do**
14: ENVIAR($v_{q_{ij} \rightarrow s_i}$)
15: **for** $p_j \in F$ **do**
16: ENVIAR($v_{s_i \rightarrow p_j}$)
17: **for** $p_j \in F$ **do**
18: ENVIAR($v_{p_j \rightarrow q_{ij}}$)
19: OTIMIZARESCOLHA(x_i)

apresentado em Santos (2009), Santos e Bazzan (2010) e Santos e Bazzan (2012), que foi uma das primeiras propostas de aplicação de modelos baseado em formação de grupos para o ambiente de desastre urbano. O Capítulo 6 apresentará a avaliação empírica do FGMD na plataforma *RMASBench*, comparando com abordagens do estado-da-arte.

6 AVALIAÇÃO EMPÍRICA E RESULTADOS

O modelo de Formação de Grupos pela Minimização da Distância (FGMD) foi avaliado através de uma série de experimentos realizados na plataforma *RMASBench* apresentada na Subseção 2.3.2. Essa plataforma utiliza uma camada central sobre o *RoboCup Rescue Simulator* (RCRS), possibilitando a avaliação empírica de algoritmos de otimização para problemas formalizados como um DCOP (Seção 3.1).

A plataforma de avaliação *RMASBench* possibilita avaliar o desempenho de algoritmos comparando métricas, como: quantidade e tamanho das mensagens enviadas ao longo da simulação, verificação de restrições não concorrentes (NCCCs), pontuação no simulador, número de prédios destruídos, tempo total de extinção das chamas, ganho de utilidade, entre outros. A pontuação no simulador representa o percentual de dano sofrido no ambiente, de modo que 0% significa que o ambiente não sofreu nenhum dano e 100% significa que o cenário foi completamente destruído.

Nesse capítulo serão comparados os modelos FGMD e *multi-team task allocation* (MTTA), ambos otimizados com o algoritmo *binary max-sum*, sendo que no modelo FGMD é utilizada a operação de minimização da função objetivo. Foram realizados experimentos com agentes bombeiros e policiais, que devem alocar, respectivamente, tarefas de incêndio e remoção de bloqueios em ruas, de modo a minimizar os danos no cenário. Avaliou-se também o desempenho do algoritmo *max-sum* (MS) e *distributed stochastic algorithm* (DSA) em cenários apenas com agentes bombeiros.

O objetivo dos experimentos é confrontar, prioritariamente, as métricas de verificação de restrições, troca de mensagens e o percentual de dano sofrido no final da simulação comparando abordagens baseadas em passagem de mensagens e busca estocástica distribuída. Avaliou-se também a relação entre o coeficiente de *damping* e uso da comunicação, nos algoritmos de passagem de mensagens.

Com relação aos parâmetros dos algoritmos comparados, no algoritmo DSA (ver Seção 4.4) cada agente inicia com uma tarefa aleatória e otimiza a escolha da tarefa com probabilidade $DSA_p = 0.6$, valor escolhido por existirem evidências de que é o mais adequado para obter bons resultados para o DSA, conforme Pujol-Gonzalez et al. (2014). O algoritmo *max-sum* foi usado para otimizar o modelo *multi-team task allocation*, identificado nos resultados pela sigla MTTA-MS. Avaliou-se ainda, o modelo *multi-team task allocation* com o algoritmo *binary max-sum*, identificado nos resultados pela sigla MTTA-BMS. O modelo de formação de grupos, proposto nesse trabalho, também é avaliado com o algoritmo *binary max-sum*, identificado

pela sigla FGMD-BMS. Os parâmetros do modelo *multi-team task allocation* (MTTA) foram escolhidos de modo a reproduzir os resultados apresentados em Pujol-Gonzalez et al. (2014). Para isso, usou-se $\kappa = 2$ e $\gamma = 2$ como parâmetros de ponderação para a penalidade dos agentes (ver Equação 4.3). Nos modelos MTTA e FGMD, usou-se $\nu = 10$ para balancear o valor das características *fieryness* e distância espacial, na utilidade. No modelo FGMD, a função de partição $p_j(\mathbf{x})$ (Equação 5.3) representa uma função de partição baseada na distribuição da carga de trabalho entre os agentes. O parâmetro de carga de trabalho de uma tarefa foi ajustado para $W_j = 2$. Esse valor foi, inicialmente, escolhido de modo a representar a função de partição para a minimização do risco empírico, apresentada por Bian e Tao (2011). Preliminarmente, para verificar o comportamento do modelo, foram realizados experimentos com diferentes valores de carga de trabalho (W_j) e com o mudanças no parâmetro de poda do domínio no *RMASBench* (k). Não tendo sido constatada diferenças significantes com relação a variação de W_j, os resultados foram omitidos. Com relação a poda do domínio, constatou-se o aumento da comunicação quando o ambiente é completamente observável, isso é, quando $k = |D|$.

Os resultados apresentados ao longo desse capítulo são oriundos de experimentos realizados no mapa da cidade Paris, com tempo total de simulação configurado para 300 *timesteps*, tal que em cada *timestep*, uma instância do algoritmo analisado é executada por até 100 iterações ou acaba quando os agentes não alterarem a tarefa escolhida de uma iteração para outra. Ainda, os experimentos foram realizados com tempos de início da simulação diferenciados, para calibrar a dificuldade do cenário, quantidades variadas de agentes e diferentes valores de coeficiente de *damping*. Após a escolha das tarefas que serão realizadas pelos agentes, a camada central do *RMASBench* envia para o RCRS as ações que deverão ser executadas naquele *timestep*. No *timestep* seguinte, os agentes são reinicializados com novas informações sobre o ambiente e uma nova instância do algoritmo é executada.

Para realizar a análise dos dados obtidos ao longo das simulações, usou-se o teste *Shapiro-Wilk*, com o objetivo de analisar a normalidade dos dados. Uma vez que os dados não pertencem a uma distribuição normal, utilizou-se o teste *Kruskal-Wallis* para verificar a significância estatística dos resultados, com intervalo de confiança de 95%.

6.1 Experimento A: Variação no Fator de *Damping*

Os resultados dessa seção comparam o desempenho dos modelos MTTA e FGMD, otimizados com algoritmo *binary max-sum* com diferentes valores para o coeficiente de *damping*, tendo sido usado os valores 0.0, 0.5 e 0.9. Com valor 0.0, o algoritmo não usa o coeficiente

de *damping*, e uma mensagem recebida por um agente é aproveitada completamente na busca pela solução. Com 0.5 aproveita-se 50% da mensagem e com 0.9 é utilizado apenas 10% da mensagem nova recebida.

Utilizou-se tempo de início de simulação de 35 *timesteps* em um ambiente com 27 agentes bombeiros e 15 agentes policiais. As Figuras 6.1 e 6.2 ilustram os resultados dos modelos para as métricas de quantidade de comunicação entre os agentes e verificação de restrições, sobre 10 simulações para cada modelo.

Nos experimentos com diferentes valores para o coeficiente de *damping*, constatou-se diferenças significantes quanto a quantidade de mensagens trocadas e verificação de restrições (NCCCs). O modelo MTTA usa menos comunicação e NCCCs com coeficiente de *damping* $= 0.9$. Para os valores de *damping* $= 0.0$ e *damping* $= 0.5$, o modelo FGMD usa menos comunicação e NCCCs.

Figura 6.1: Experimento A: Comparação entre os modelos otimizados com o *binary max-sum* - Quantidade de Mensagens x Fator de *damping*

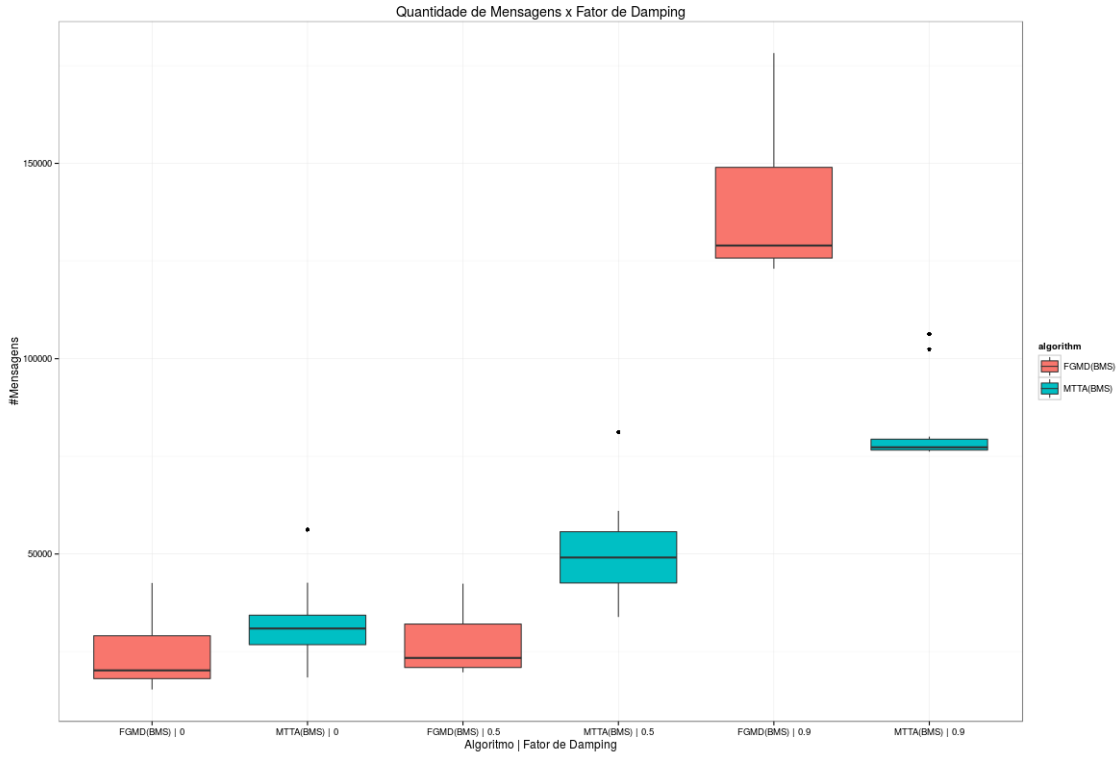
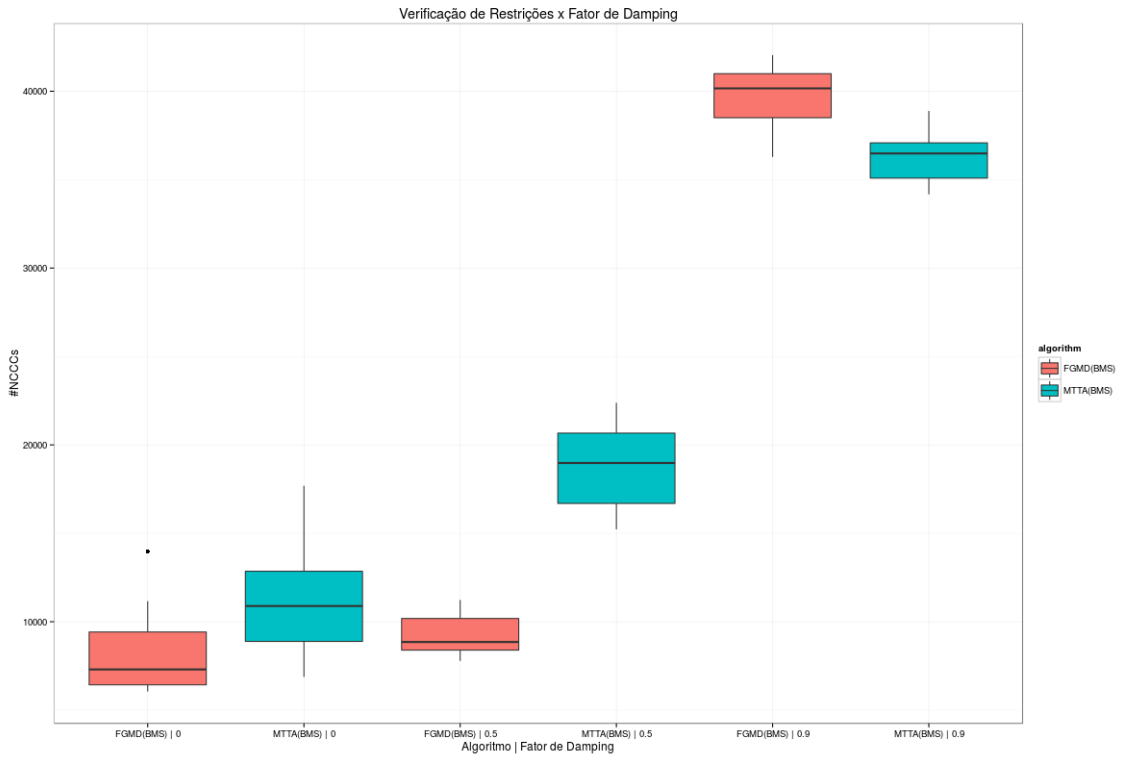


Figura 6.2: Experimento A: Comparação entre os modelos otimizados com o *binary max-sum* - Verificação de restrções x Fator de *damping*



6.2 Experimento B: Extinção de tarefas de incêndio

Nesse conjunto de experimentos, fez-se uma avaliação quantitativa sobre métricas relacionadas ao desempenho dos agentes, uso de comunicação e exigência de recursos computacionais. Comparou-se: o modelo proposto, otimizado com o algoritmo *binary max-sum*; o modelo *multi-team task allocation*, otimizado com os algoritmos *binary max-sum* e *max-sum*; e o algoritmo de busca local *distributed stochastic algorithm* (DSA), apresentado na Seção 4.4. O parâmetro de poda do domínio, da plataforma *RMASBench*, foi ajustado para $k = 4$, indicando o máximo de tarefas que cada agente terá no domínio. Esse valor foi escolhido por ser o mais alto que possibilitou obter resultados com o algoritmo *max-sum*. O cenário foi configurado para ter 19 agentes bombeiros e 5 pontos de ignição de incêndio, com início das atividades dos agentes a partir do *timestep* 35, de modo a calibrar sua dificuldade.

Os resultados são agrupados conforme o uso do coeficiente de *damping* dos algoritmos *max-sum* e *binary max-sum*, onde no conjunto de experimentos identificado por B-1 foi usado o coeficiente de *damping* $= 0.9$ e no conjunto identificado por B-2 usou-se *damping* $= 0.0$.

A Tabela 6.1 resume os parâmetros utilizados para os algoritmos comparados no Experimento B-1. Os resultados são apresentados na Tabela 6.2, com os melhores resultados

destacados em negrito, com desvio padrão em colchetes. Os resultados são a média sobre 30 simulações para cada algoritmo. As métricas analisadas são:

Tamanho das mensagens: cada mensagem trocada entre os agentes possui tamanho de 8 *bytes*. Essa métrica apresenta a quantidade média de *bytes* enviados entre cada agente;

Número de mensagens: média da quantidade de mensagens trocadas entre os agentes;

NCCCs: média de computação não-paralelizada realizada pelos agentes;

Destruição do cenário: é o *score* do simulador *RMASBench*, representado por um valor percentual. Quanto maior a porcentagem, maior a destruição do cenário. Essa métrica apresenta a média da pontuação de cada algoritmo no simulador;

Iterações: média de iterações necessárias para convergência do algoritmo;

Tempo para extinção das chamas: média da quantidade de tempo requerida, em *timesteps*, para extinção das chamas.

Tabela 6.1 Parâmetros do Experimento B-1

	Damping Factor	κ	γ	ν	k	W_j
DSA	-	-	-	10	4	-
MTTA(MS)	0.9	2	2	10	4	-
MTTA(BMS)	0.9	2	2	10	4	-
FGMD(BMS)	0.9	-	-	10	4	2

No Experimento B-1, os algoritmos comparados são estatisticamente semelhantes com relação ao percentual de dano no final da simulação (Figura 6.3). Nas Figuras 6.4 e 6.5 são ilustrados os resultados para as métricas relacionadas, respectivamente, a quantidade de mensagens e verificação de restrições, onde observou-se que o DSA requer menos comunicação e NCCCs, e os modelos FGMD-BMS e MTTA-BMS são semelhantes. A quantidade de mensagens trocadas entre os agentes, no modelo MTTA-MS, é aproximadamente quatro vezes maior, comparado com o *binary max-sum*.

O Experimento B-2 foi realizado conforme os parâmetros apresentados na Tabela 6.3. O objetivo deste conjunto de experimentos foi comparar o desempenho dos algoritmos de passagem de mensagem, usando coeficiente de *damping* = 0.0. Os resultados apresentados na Tabela

Tabela 6.2: Experimento B-1: Resultados para DSA, MTTA (MS), MTTA (BMS), FGMD (BMS), média sobre 30 simulações no mapa Paris. Os melhores resultados são destacados em negrito e o desvio padrão entre colchetes.

Métrica	DSA	MTTA (MS)	MTTA (BMS)	FGMD (BMS)
Tamanho das mensagens (em KB)	**6.52KB [± 0.64KB]**	484.07KB [± 8.76KB]	102.41KB [± 2.66KB]	87.18KB [± 10.14KB]
Número de Mensagens	**815.10 [± 80.48]**	13697.03 [± 178.83]	12802.39 [± 333.58]	10898.70 [± 1267.52]
NCCCs	**167.25 [± 11.50]**	49711.90 [± 4334.53]	11150.59 [± 708.98]	9638.16 [± 1444.53]
Destruição do cenário	**8.64% [± 13.57%]**	**8.12% [± 12.92%]**	**8.79% [± 13.39%]**	**6.09% [± 2.14%]**
Iterações	**6.58 [± 0.38]**	98.24 [± 1.12]	92.69 [± 3.61]	77.15 [± 9.92]
Tempo para extinção das chamas	**116.43 [± 39.66]**	**129.60 [± 44.39]**	**132.96 [± 50.58]**	**135.50 [± 32.52]**

Figura 6.3: Experimento B-1: Média do percentual de dano no cenário

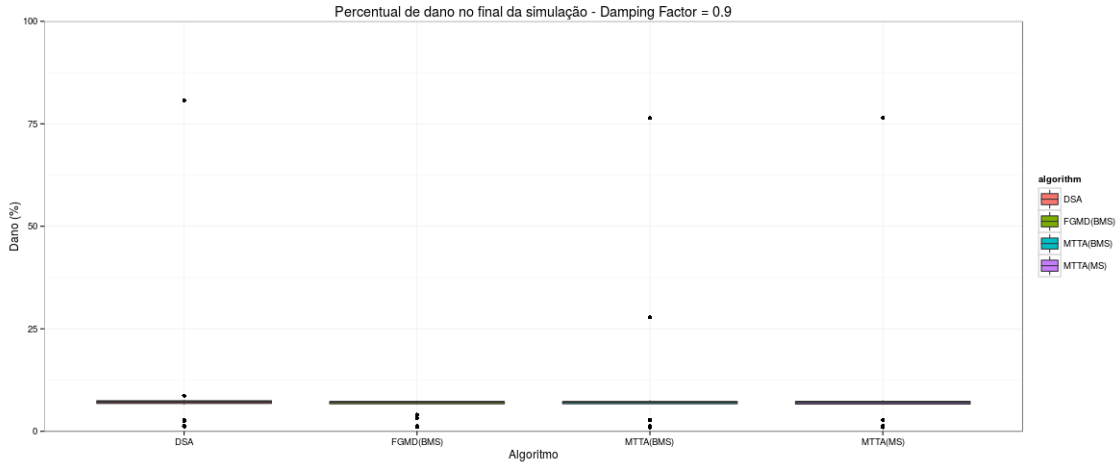


Figura 6.4: Experimento B-1: Média de troca de mensagens durante cada iteração

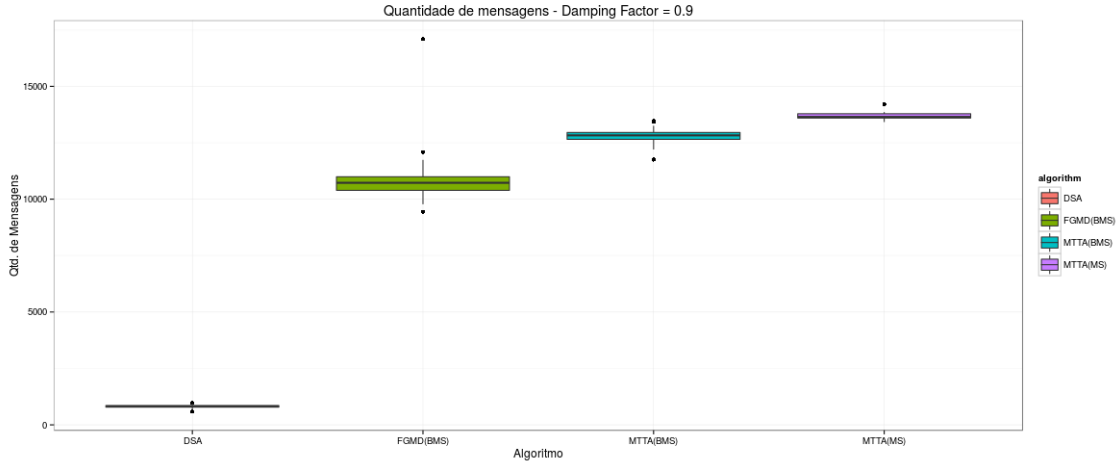
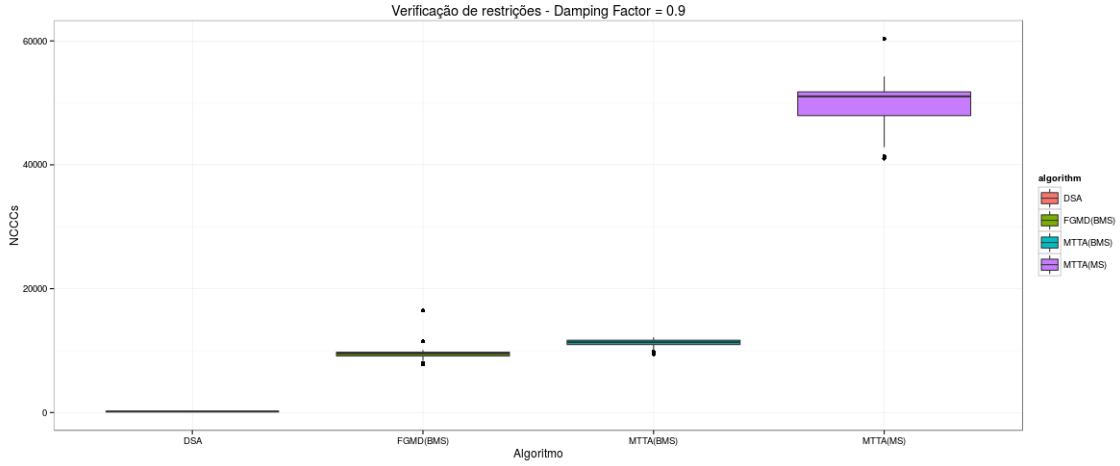


Figura 6.5: Experimento B-1: Média da verificação de restrições dos agentes



6.4 apresentam a média sobre 30 simulações para cada algoritmo. Os melhores valores são destacados em negrito, com desvio-padrão entre colchetes.

Tabela 6.3 Parâmetros do Experimento B-2

	Damping Factor	κ	γ	ν	k	W_j
DSA	-	-	-	10	4	-
MTTA(MS)	0.0	2	2	10	4	-
MTTA(BMS)	0.0	2	2	10	4	-
FGMD(BMS)	0.0	-	-	10	4	2

Com relação ao percentual de dano, constatou-se que os resultados são semelhantes para os algoritmos de passagem de mensagens. Entretanto, para o FGMD-BMS e o DSA as diferenças são significativas (ver Tabela 6.4). A quantidade de mensagens trocadas com o modelo FGMD-BMS é maior que no modelo MTTA-BMS e MTTA-MS (Figura 6.7). Com relação a quantidade de verificação de restrições (NCCCs), o FGMD-BMS tem resultado melhor que o MTTA-MS e possui um aumento significativo com relação ao MTTA-BMS (Figura 6.8).

Figura 6.6: Experimento B-2: Média do percentual de dano no cenário

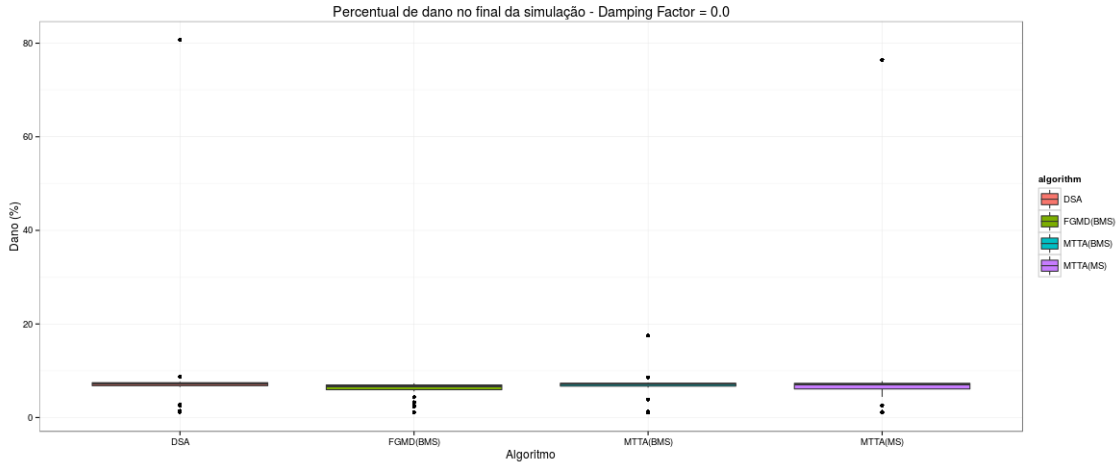


Tabela 6.4: Experimento B-2: Resultados para DSA, MTTA (MS), MTTA (BMS), FGMD (BMS), média sobre 30 simulações no mapa Paris. Os melhores resultados são destacados em negrito e o desvio padrão entre colchetes.

Métrica	DSA	MTTA (MS)	MTTA (BMS)	FGMD (BMS)
Tamanho das mensagens (em KB)	**6.52KB [± 0.63KB]**	73.80KB [± 24.29KB]	15.43KB [± 4.52KBKB]	22.32KB [± 7.23KB]
Número de Mensagens	**815.96 [± 79.25]**	2128.96 [± 690.18]	1929.99 [± 565.36]	2790.77 [± 904.12]
NCCCs	**167.35 [± 11.36]**	7866.74 [± 2187.98]	1750.63 [± 402.96]	2773.02 [± 705.52]
Destruição do cenário	8.64% [± 13.57%]	8.21% [± 12.88%]	6.43% [± 3.08%]	**5.88% [± 1.70%]**
Iterações	**6.58 [± 0.38]**	17.22 [± 5.08]	15.19 [± 4.18]	8.83 [± 2.54]
Tempo para extinção das chamas	**116.46 [± 39.67]**	132.96 [± 43.50]	135.66 [± 43.70]	128.63 [± 21.48]

Figura 6.7: Experimento B-2: Média de troca de mensagens durante cada iteração

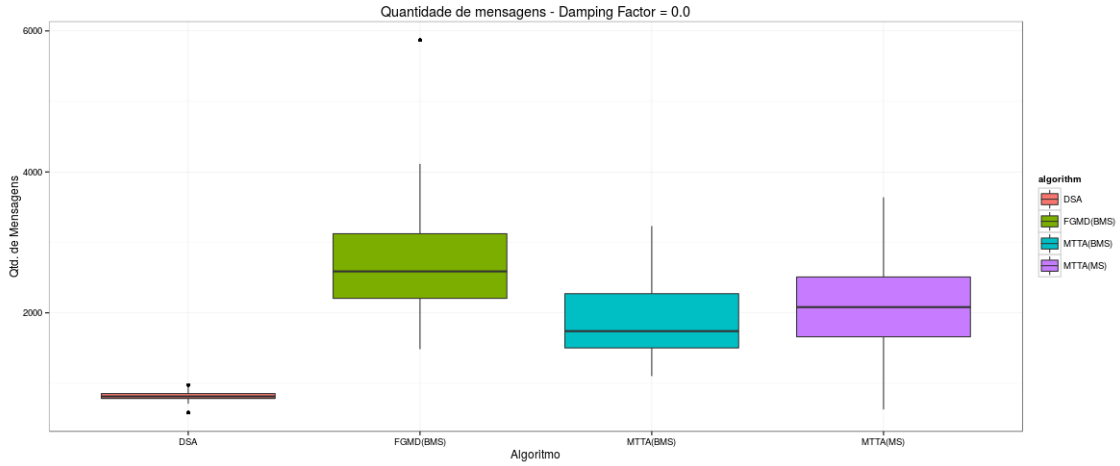
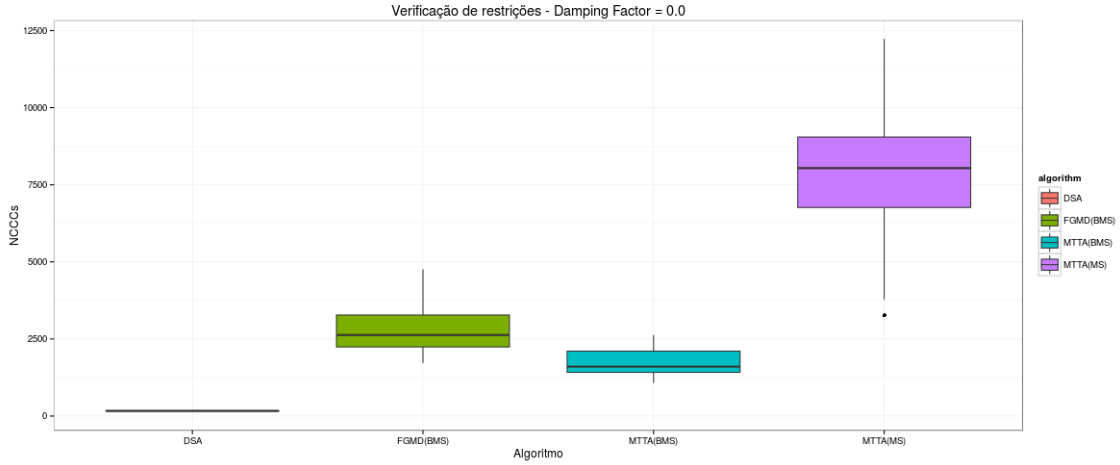


Figura 6.8: Experimento B-2: Média da verificação de restrições dos agentes



6.3 Experimento C: Extinção de tarefas de incêndio em cenários com bloqueios

Nesse conjunto de experimentos foram avaliados os modelos *multi-task team allocation* (MTTA) e o modelo de formação de grupos pela minimização da distância (FGMD) aplicados ao problema de coordenação de agentes bombeiros e policiais, com o objetivo de verificar o desempenho dos agentes em cenários com restrições de mobilidade. No cenário com bloqueios, os agentes bombeiros tem a mobilidade limitada e dependem da ação dos agentes policiais para desbloquear as ruas. Uma vez que não há diferenças entre o esforço realizado por um agente ou um grupo de agentes, em uma tarefa de remoção de bloqueio, usou-se o modelo apresentado em Pujol-Gonzalez et al. (2014) e descrito no Capítulo 4, adaptado para minimização. Assim, os policiais agem de forma gulosa escolhendo o bloqueio mais próximo para remover. Se o bloqueio escolhido está inacessível devido a outro bloqueio, é inserida uma penalidade Q. Nos experimentos realizados utilizou-se $Q = 50$ por ser o valor utilizado nos experimentos

apresentados por Pujol-Gonzalez et al. (2014).

O modelo MTTA permite a coordenação entre agentes heterogêneos, porém optou-se por não utilizar essa forma de coordenação em uma primeira análise. Nos experimentos realizados, os time de agentes bombeiros e policiais atuaram independentemente, onde foi analisado o desempenho dos modelos com relação ao aumento na dificuldade e ao aumento de recursos. Para ajustar a dificuldade, durante cada experimento o *timestep* inicial de simulação foi modificado e os valores usados para os parâmetros dos modelos MTTA e FGMD são apresentados na Tabela 6.5.

Usou-se o valor do coeficiente de *damping* = 0.0, pois ao contrário dos resultados apresentados na Seção 6.2, o FGMD apresentou diferenças significantes quanto ao uso de comunicação e verificação de restrições. Foi usado o valor de poda do domínio, $k = 10$, de modo que os agentes terão no máximo 10 tarefas. Os experimentos foram realizados no mapa Paris com 27 agentes bombeiros e 15 agentes policiais. Os resultados apresentados representam a média sobre 10 simulações, tendo sido verificada a significância dos resultados com o teste de *Kruskal-Wallis*, com 95% de confiança.

Tabela 6.5 Parâmetros do Experimento C-1 e C-2

	Damping Factor	κ	γ	ν	k	W_j
MTTA(BMS)	0.0	2	2	10	10	-
FGMD(BMS)	0.0	-	-	10	10	2

No Experimento C-1 avaliou-se o desempenho dos agentes nos seguintes níveis de dificuldade: 15, 20, 25, 30 e 35. Esses valores representam o início das atividades dos agentes na simulação, assim, quão menor é o valor de dificuldade, mais fácil é o cenário. A Figura 6.9 apresenta os resultados para a métrica de percentual de dano no ambiente, agrupados pelo nível de dificuldade em cada modelo. Com dificuldades 15 e 20, as tarefas são resolvidas rapidamente pois não há grande propagação das chamas, a diferença observada não é estatisticamente significante. Com nível de dificuldade 25, o FGMD possui diferenças significativas, apresentando melhor resultado com relação ao percentual de dano no final da simulação.

A quantidade média de mensagens trocadas entre os agentes, durante cada *timestep*, é apresentada na Figura 6.10, onde veficou-se que o FGMD usa menos comunicação nos tempos de início da simulação 15 e 20.

Figura 6.9: Experimento C-1: Percentual de dano X Aumento da Dificuldade

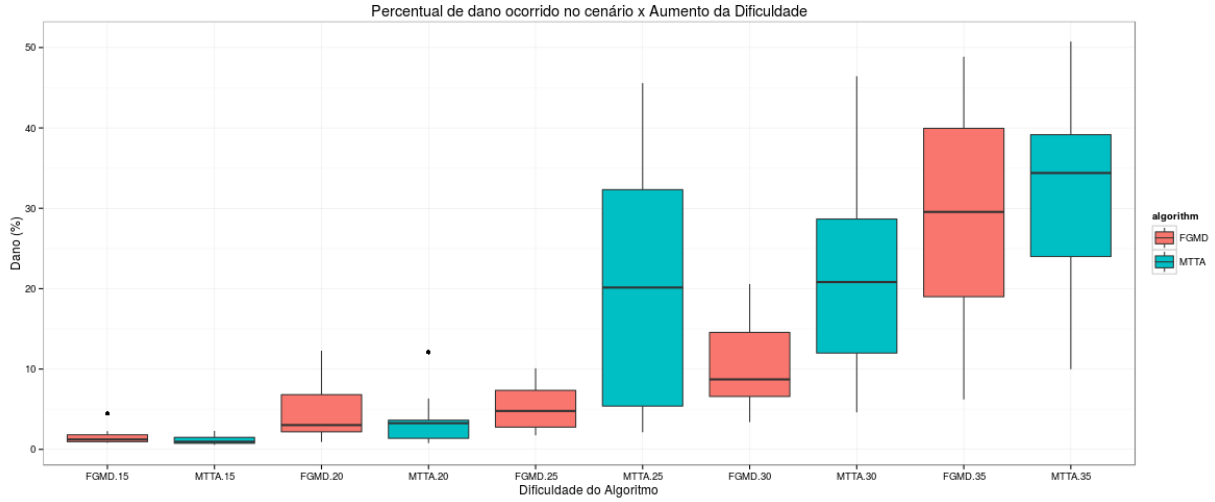
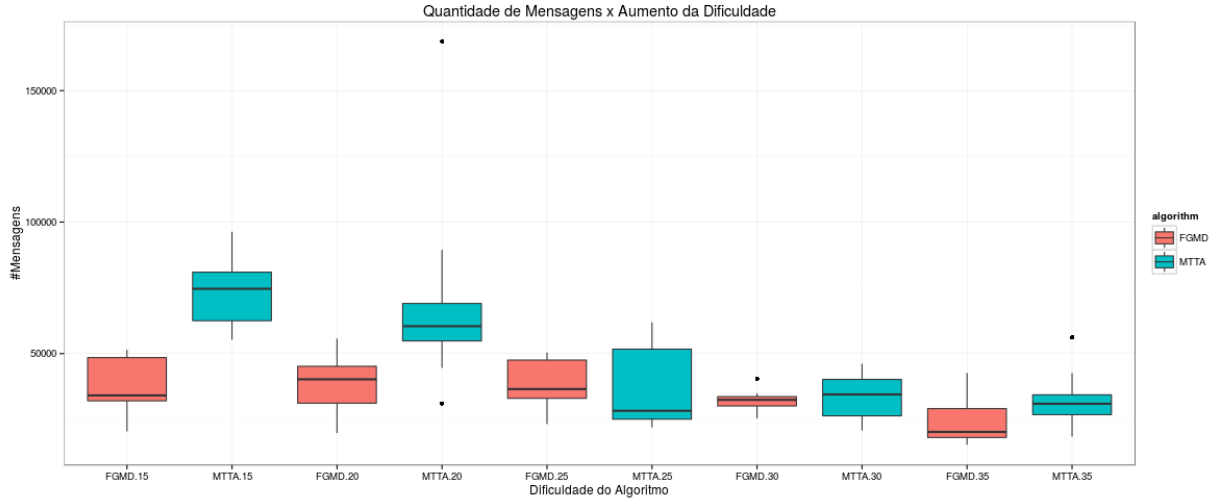


Figura 6.10: Experiento C-1: Quantidade de Mensagens X Aumento da Dificuldade

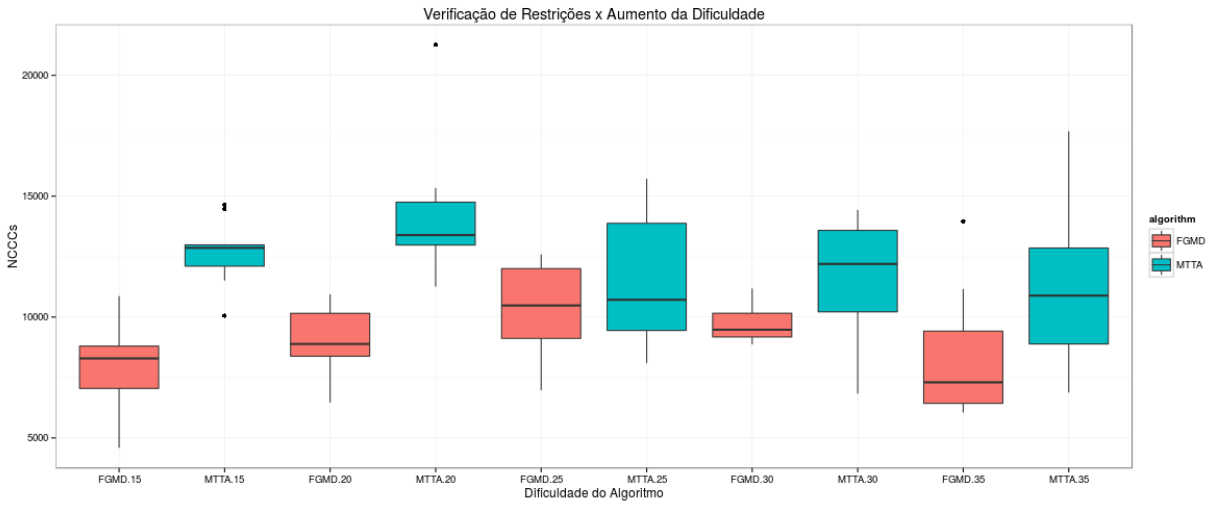


A Figura 6.11 apresenta os resultados da quantidade média de verificação de restrições, agrupados pelo tempo de início dos agentes em cada modelo. No modelo FGMD, os agentes realizam menos verificação de restrições (NCCCs), nos tempos de início de simulação 15, 20, 30 e 35.

O conjunto de experimentos C-2 avaliou o aumento de recursos, aumentando o número de agentes na simulação e usando tempo de início da simulação 35 *timesteps*. Demais valores de parâmetros para os modelos utilizados na avaliação são os mesmos informados na Tabela 6.5.

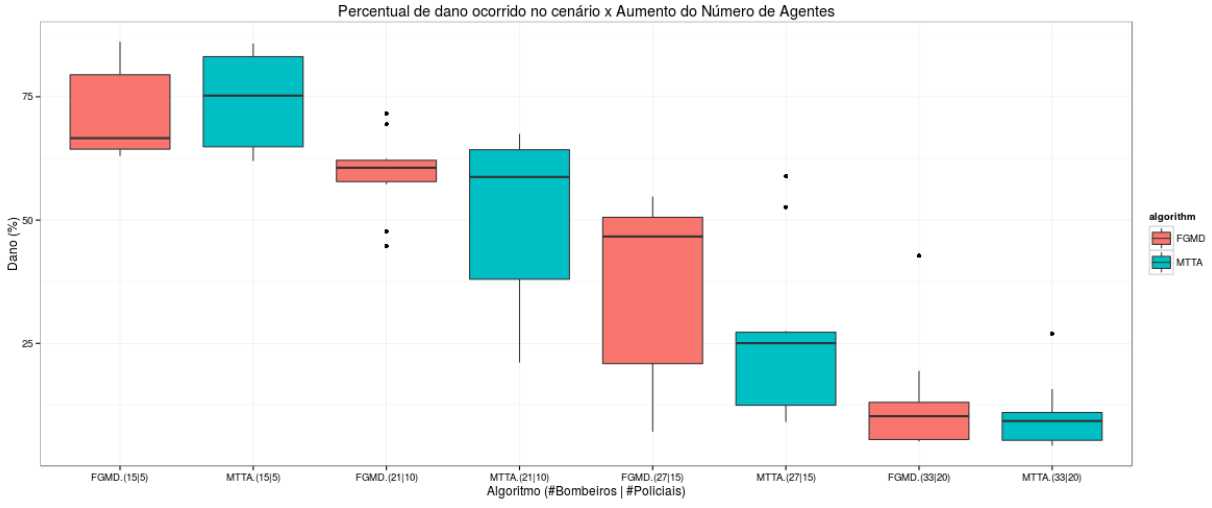
Os resultados do Experimento C-2 correspondem a média sobre 10 simulações, agrupados por modelo e quantidade de agentes, respectivamente, $FGMD(\#Bombeiros|\#Policiais)$

Figura 6.11: Experimento C-1: NCCCs X Aumento da Dificuldade



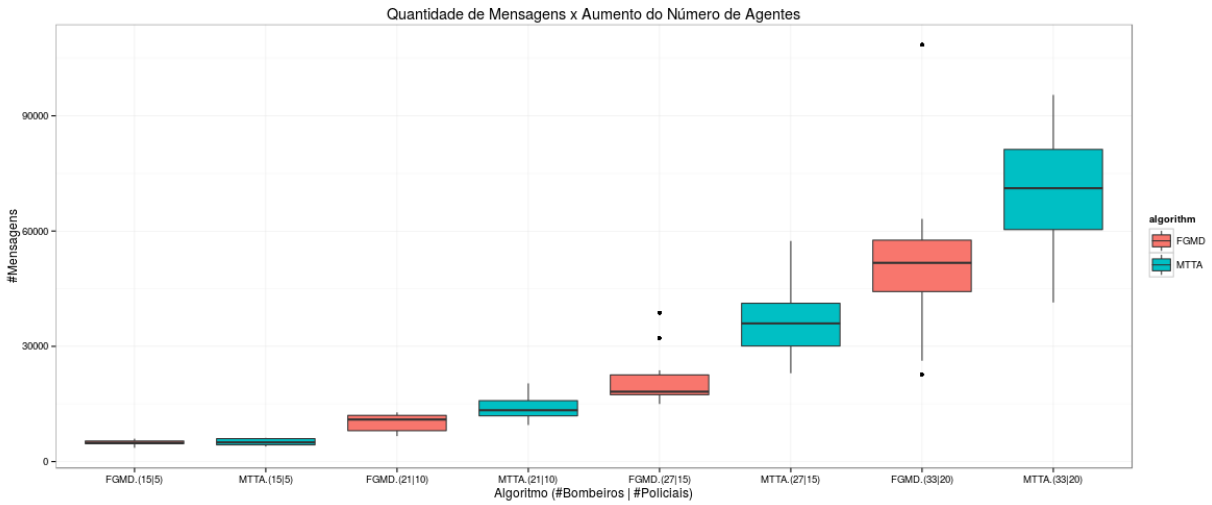
para o modelo de formação de grupos proposto e $MTTA(\#Bombeiros|\#Policiais)$ para o modelo de Pujol-Gonzalez et al. (2014). A Figura 6.12 apresenta os resultados para a métrica de percentual de dano no cenário, agrupado pela quantidade de agentes para cada modelo, onde não foram observadas diferenças significantes entre os modelos

Figura 6.12: Experimento C-2: Percentual de Dano X Aumento de Recursos



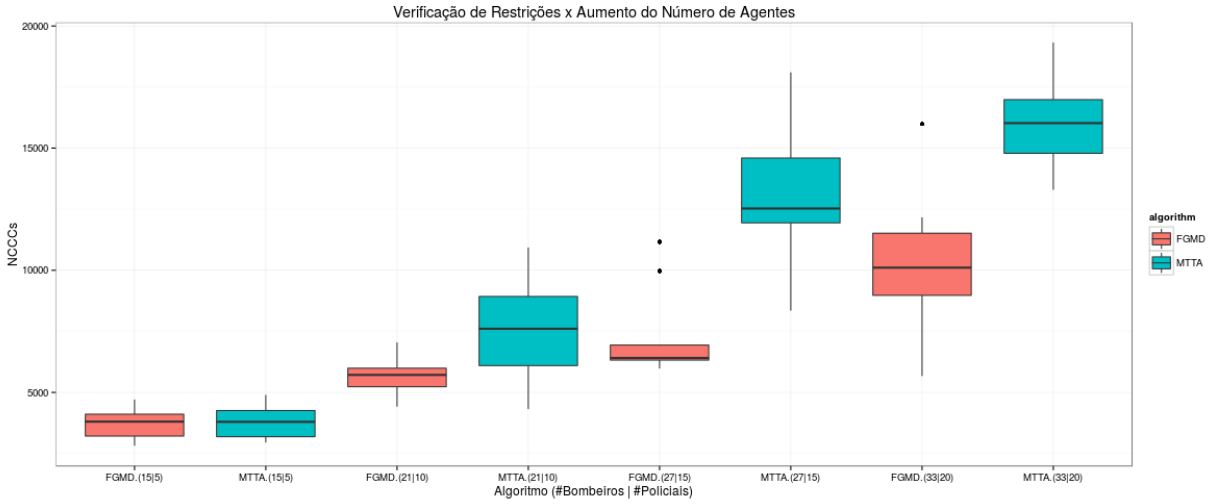
Com respeito as métricas de comunicação, os resultados ilustrados na Figura 6.13 apresentam diferenças entre os modelos FGMD e MTTA para quantidades de agentes $(21|15)$, $(27|15)$ e $(33|20)$. Nesses resultados, observou-se que o FGMD usa menos mensagens com maior número de agentes.

Figura 6.13: Experimento C-2: Quantidade de Mensagens X Aumento de Recursos



A Figura 6.14 apresenta os resultados dos experimentos com relação a métrica de verificação de restrições (NCCCs). Os resultados apresentados indicam que o modelo FGMD faz menos verificações de restrições nos experimentos com quantidades de agentes $(21|10)$, $(27|15)$ e $(33|20)$. Com quantidade de agentes $(15|5)$ não foram observadas diferenças entre os dois modelos.

Figura 6.14: Experimento C-2: NCCCs x Aumento de Recursos



Os resultados apresentados nessa seção mostraram a avaliação dos modelos FGMD e MTTA otimizados com o algoritmo *binary max-sum* em experimentos que simulam variações na dificuldade do cenário e na quantidade de recursos. Observou-se que a recuperação do cenário é mais eficiente no modelo FGMD, em condições de dificuldade intermediária, ou seja,

tempo de início de simulação 25. Os resultados sugerem que a diferença na recuperação do cenário pode ser balanceada com o uso de comunicação. Na maioria dos experimentos, os resultados indicaram que o FGMD faz menos verificações de restrições e usa menos mensagens. Porém, nesses experimentos o desempenho dos agentes com relação a mitigação dos desastres é semelhantes as outras abordagens comparadas.

Os resultados apresentados na Seção 6.2 mostraram que o algoritmo *max-sum* (MS) realiza mais verificações de restrições que as outras abordagens comparadas. Por esse motivo, optou-se por não utilizar o MS para otimizar o modelo FGMD. O *distributed stochastic algorithm* (DSA) apresentou os melhores resultados com relação ao uso de comunicação e verificação de restrições. Como o DSA é um algoritmo de busca local estocástica, requer menos esforço computacional para encontrar uma solução. Sem o uso do coeficiente de *damping* no cenário com agentes bombeiros, o FGMD apresentou desempenho superior ao DSA com relação a mitigação do desastre, porém apresentando aumento no uso de comunicação e verificação de restrições.

7 CONCLUSÃO

Nesse trabalho foi apresentado o modelo de Formação de Grupos pela Minimização da Distância (FGMD), uma abordagem para o problema de alocação de tarefas baseado em formação de grupos, formalizado como um problema de otimização de restrições distribuídas para agentes e otimizado com o algoritmo de passagem de mensagens *binary max-sum*. O objetivo do FGMD é formar grupos de agentes, de modo que uma métrica de distância sobre as tarefas nos seus domínios, seja minimizada. Uma função de partição baseada no parâmetro de carga de trabalho da tarefa e na quantidade de agentes que decidem realizá-la é usada para distribuir os agentes nos grupos, até que todas as tarefas sejam resolvidas.

Os algoritmos baseados em passagem de mensagens entre os agentes, como *max-sum* e *binary max-sum*, têm sido aplicados em problemas de desastre urbano na plataforma *RMAS-Bench*, pois tais algoritmos são distribuídos e possuem garantia de convergência para uma solução ótima em grafos acíclicos. Essa plataforma é executada sobre o *RoboCup Simulation* (RCRS), garantindo um canal de comunicação aberta entre os agentes. Apesar do RCRS prover um ambiente mais próximo do mundo real, com comumicação incerta, o *RMASBench* possibilitou validar o modelo antes de sua aplicação em um domínio dinâmico. Nesse trabalho, o FGMD é formalizado como um DCOP e representado com um grafo-fator com variáveis binárias e THOPs que representam os processos de decisão do agente.

As principais métricas utilizadas para análise dos resultados foram o percentual de dano ocorrido na cidade no final da simulação, quantidade de mensagens trocadas entre os agentes e a verificação de restrições (do inglês, *non-concurrent constraint checks* ou NCCCs), onde foram comparados os modelos FGMD e MTTA com o algoritmo *binary max-sum* (BMS). O modelo MTTA foi otimizado, também, com o algoritmo *max-sum* com o intuito de analisar as diferenças entre os algoritmos de passagem de mensagem. Foi realizado um conjunto de experimentos com o algoritmo *distributed stochastic algorithm* (DSA), com o objetivo de comparar os modelos com um algoritmo de busca local estocástica.

Os resultados apresentados indicaram que o FGMD apresenta melhor resultado na mitigação do desastre, somente em situações onde o nível de dificuldade é intermediária (tempo de início de simulação 25). Os resultados sugerem que o aumento no uso da comunicação e verificação de restrições aprimora o resultado com relação ao percentual de dano, pois na maioria dos experimentos o FGMD usa menos comunicação que o modelo MTTA. Porém, quando o percentual de dano possui diferenças significantes, o uso da comunicação e NCCCs é semelhante entre ambos os modelos. Diferenças significantes são percebidas também em relação ao

valor do coeficiente de *damping*, onde os valores 0.0 e 0.5 apresentam melhores resultados para a métrica de quantidade de mensagens e NCCCs.

Os resultados das avaliações com o algoritmo DSA apresentaram o melhor desempenho com relação ao uso de comunicação e verificação de restrições. Porém mostrou-se que o FGMD obteve melhor desempenho com relação a mitigação dos danos, em relação ao DSA, em cenário com agentes bombeiros e sem o uso do coeficiente de *damping*. No modelo MTTA, observou-se que o algoritmo *max-sum* utiliza mais verificação de restrições e mais troca de mensagens que o algoritmo *binary max-sum* e não apresenta diferenças com relação ao percentual de dano no final da simulação.

Em resumo, os resultados do FGMD, quando comparados com o MTTA, indicam menos uso de comunicação e NCCCs, nos experimentos sem coeficiente de *damping* e em cenários com grande número de agentes. Os experimentos realizados com um número baixo de agentes bombeiros (Seção 6.2) usam menos comunicação e NCCCs com o coeficiente de *damping*. Com relação ao desempenho dos agentes na mitigação dos desastres, no modelo FGMD observou-se que os resultados que apresentaram diferenças significativas com relação aos algoritmos comparados usaram mais troca de mensagens entre os agentes.

Como proposta de trabalhos futuros, surgem duas linhas de interesse: a primeira consiste na aplicação do modelo em cenários dinâmicos e com comunicação incerta. Para isso, o grafo-fator do problema deverá tratar problemas decorrentes do não recebimento das mensagens, onde sugere-se o uso do coeficiente de *damping* para ponderar a escolha do agente. A exemplo de abordagens como *bee clustering* e *extreme-ants*, pode ser necessário alterar o modelo para a inclusão de parâmetros de estímulo e resposta, de modo a garantir a permanência dos agentes nos grupos até que as tarefas sejam realizadas. A segunda linha de interesse consiste na aplicação do modelo FGMD em problemas de classificação e formação de agrupamentos em conjuntos de dados reais. Para isso, deve ser modelada uma função de utilidade que represente a distância entre os objetos a serem agrupados.

REFERÊNCIAS

AKIN, H. L. et al. Robocup Rescue Robot and Simulation Leagues. **AI Magazine**, v. 34, n. 1, p. 78–86, 2013.

BIAN, W.; TAO, D. Learning a distance metric by empirical loss minimization. In: WALSH, T. (Ed.). **IJCAI**. [S.l.]: IJCAI/AAAI, 2011. p. 1186–1191.

CORRÊA, A. Distributed team formation in urban disaster environments. In: **Proceedings of Intelligent Agents (IA) on SSCI 2014**. Orlando, FL, USA: IEEE, 2014. p. 57–64. Disponível em: <http://dx.doi.org/10.1109/IA.2014.7009459>.

FARINELLI, A.; ROGERS, A.; JENNINGS, N. R. Agent-based decentralised coordination for sensor networks using the max-sum algorithm. **Autonomous Agents and Multi-Agent Systems**, Springer US, v. 28, n. 3, p. 337–380, 2014.

FRANKLIN, S.; GRAESSER, A. It is an agent, or just a program? a taxonomy for autonomous agents. In: SPRINGER (Ed.). **Intelligent Agents**. [S.l.]: Jennings, N. and Wooldridge, M., 1997. III.

JENNINGS, N.; SYCARA, K.; WOOLDRIGE, M. A roadmap of agent research and development. **Autonmous Agents and Multi-Agent Systems**, n. 1, p. 7–38, 1998.

KITANO, H.; TADOKORO, S. Robocup rescue: A grand challenge for multiagent and intelligent systems. **AI Magazine**, v. 22, n. 1, p. 39–52, 2001.

KLEINER, A. et al. Rmasbench: Benchmarking dynamic multi-agent coordination in urban search and rescue. In: **Proceedings of the 2013 International Conference on Autonomous Agents and Multi-agent systems**. [S.l.: s.n.], 2013. p. 1195–1196.

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

MEISELS, A. **Distributed Search By Constrained Agents**. London: Springer, 2008. ISBN 978-1-84800-040-7.

MODI, P. J. et al. An asynchronous complete method for distributed constraint optimization. In: **Proc. of the Second International Joint Conference on Autonomous Agents and Multiagent Systems**. New York, USA: ACM Press, 2003. p. 161–168.

PETCU, A. **A Class of Algorithms for Distributed Constraint Optimization**. Tese (PhD. Thesis No. 3942) — Swiss Federal Institute of Technology (EPFL), Lausanne (Switzerland), 2007. Disponível em: <http://liawww.epfl.ch/Publications/Archive/Petcu2007thesis.pdf>.

PUJOL-GONZALEZ, M. et al. On binary max-sum and tractable hops. **11th European Workshop on Multi-agent Systems (EUMAS 2013)**, Toulouse, France, v. 1113, 2013.

PUJOL-GONZALEZ, M. et al. Binary max-sum for multi-team task allocation in robocup rescue. **Optimisation in Multi-Agent Systems and Distributed Constraint Reasoning (OptMAS-DCR)**, Paris, France, 2014.

PUJOL-GONZALEZ, M. et al. Engineering the decentralized coordination of uavs with limited communication range. **CAEPIA 2013**, Spriger-Verlag, Madrid, 2013.

RAMCHURN, S. et al. Decentralised coordination in robocup rescue. **The Computer Journal**, Oxford Journals, v. 53, n. 9, p. 1–15, 2010. Disponível em: <http://eprints.soton.ac.uk/268499/>.

RUSSEL, S.; NORVIG, P. **Artificial Intelligence: A Modern Approach**. [S.l.]: Prentice-Hall, 1995. 932 p.

SANTOS, D. S. dos. **Bee clustering: um Algoritmo para Agrupamento de Dados Inspirado em Inteligência de Enxames**. Dissertação (Mestrado) — PPGC-UFRGS, Porto alegre, Abril 2009. Disponível em: <http://www.lume.ufrgs.br/handle/10183/18249>.

SANTOS, D. S. dos; BAZZAN, A. L. C. Distributed clustering for group formation and task allocation. In: HOEK, W. van der et al. (Ed.). **Proceedings of the Ninth International Conference on Autonomous Agents and Multi-Agent Systems**. Toronto: IFAAMAS: International Foundation for Autonomous Agents and Multiagent Systems, 2010. p. 1429–1430. Disponível em: <http://www.ifaamas.org/Proceedings/aamas2010/pdf/02%20Extended%20Abstracts/Red/R-31.pdf>.

SANTOS, D. S. dos; BAZZAN, A. L. C. Distributed clustering for group formation and task allocation in multiagent systems: A swarm intelligence approach. **Appl. Soft Comput.**, v. 12, n. 8, p. 2123–2131, 2012. Disponível em: <http://dx.doi.org/10.1016/j.asoc.2012.03.016>.

SANTOS, F. dos. **eXtreme-Ants: Algoritmo Inspirado em Formigas para Alocação de Tarefas em Extreme Teams**. Dissertação (Mestrado) — Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre, Maio 2009. Disponível em: <http://hdl.handle.net/10183/17041>.

SANTOS, F. dos; BAZZAN, A. L. C. eXtreme-ants: Ant based algorithm for task allocation in extreme teams. In: JENNINGS, N. R. et al. (Ed.). **Proceedings of the Second International Workshop on Optimisation in Multi-Agent Systems**. Budapest, Hungary: [s.n.], 2009. p. 1–8. Disponível em: <www.inf.ufrgs.br/maslab/pergamus/pubs/Santos&Bazzan2009optmas.pdf>.

SCERRI, P. et al. Allocating tasks in extreme teams. In: DIGNUM, F. et al. (Ed.). **Proc. of the Fourth International Joint Conference on Autonomous Agents and Multiagent Systems**. New York, USA: ACM Press, 2005. p. 727–734. ISBN 1-59593-093-0.

TARLOW, D.; GIVONI, I. E.; ZEMEL, R. S. Hop-map: Efficient message passing with high order potentials. In: **Proceedings of 13th Conference on Artificial Intelligence and Statistics**. [S.l.: s.n.], 2010.

TARLOW, D. et al. Stochastic k-neighborhood selection for supervised and unsupervised learning. In: **Procs. of the Thirtieth International Conference on Machine Learning**. [S.l.: s.n.], 2013.

TARLOW, D. et al. Fast exact inference for recursive cardinality models. In: **In Uncertainty in Artificial Intelligence**. [S.l.: s.n.], 2012.

WOOLDRIDGE, M. J. **An Introduction to MultiAgent Systems**. Chichester: John Wiley & Sons, 2009. 461 p. 2nd edition.

WOOLDRIGE, M.; JENNINGS, N. Intelligent agents: Theory and practice. **Knowledge Engineering Review**, v. 10, n. 2, p. 115–152, 1995.

ZHANG, W. et al. An analysis and application of distributed constraint satisfaction and optimization algorithms in sensor networks abstract. In: **Proceedings Second International Joint Conference on Autonomous Agents and Multi Agent Systems (AAMAS03)**. [S.l.: s.n.], 2003.

Apêndice A — ALGORITMO *DCOPSOLVER* DO *RMASBENCH*

1: $X = \{x_1, \cdots, x_i, \cdots, x_n\}$
2: $D = \{D_1, \cdots, D_i, \cdots, D_n\}$
3: $V = \{d_1, \cdots, d_j, \cdots, d_m\}$
4: utilityMatrix $=$ $\underbrace{\left.\begin{array}{|c|c|c|}\hline \delta_{11} & \cdots & \delta_{n1} \\ \hline \vdots & \delta_{ij} & \vdots \\ \hline \delta_{1m} & \cdots & \delta_{nm} \\ \hline \end{array}\right\}|V|}_{|A|}$
5: $itrMax = 100$
6: $itr = 0$
7: $ok = false$
8: MONTAGRAFODERESTRICOES(X,D,V)
9: **while** $!ok \wedge itr < itrMax$ **do**
10: **for** $x_i \in X$ **do**
11: ENVIARMENSAGENS(x_i)
12: **for** $x_i \in X$ **do**
13: RECEBERMENSAGENS(x_i)
14: $ok = true$
15: **for** $x_i \in X$ **do**
16: $update =$OTIMIZARESCOLHA(x_i)
17: $ok = ok \wedge !update$

O *DCOPSolver* consiste dos procedimentos básicos de busca pela solução ótima em um DCOP. Considerando um DCOP onde cada agente é responsável por uma única variável $x_i \in X$, os agentes devem atribuir um valor d_j para suas variáveis e enviar mensagens para os agentes vizinhos, de modo a verificar a consistência das novas atribuições. A troca de mensagens entre os agentes termina quando não houver mais alterações nos valores das variáveis dos agentes, ou termina ao alcançar o total de iterações. Cada $D_i \in D$ é um conjunto finito e discreto de valores para a variável de um agente x_i, tal que $D_i = \{d_{i1}, \cdots, d_{ij}, \cdots, d_{ik}\}$ e k é a cardinalidade do domínio de x_i. O conjunto $V = \{d_1, \cdots, d_j, \cdots, d_m\}$ representa todos os valores possíveis para todas as variáveis. Se $k = m$ para todo e qualquer agente, então pode ser assumido que o ambiente é completamente observável.

Cada atribuição possível de valor d_j para uma variável x_i, denotado por d_{ij}, tem um valor de utilidade δ_{ij}, representado em uma matriz de utilidade apresentada na Linha 4 do algoritmo. Os valores da matriz de utilidade são atualizados através da troca de mensagens entre os agentes. A função *MontaGrafodeRestricoes(X,D,V)* (Linha 8) cria a representação gráfica do problema,

que será utilizada para direcionar a troca de mensagens entre os agentes. No *RMASBench* um agente está conectado a outro, por uma aresta, se possuem valores em comum no domínio.

Após a montagem do grafo de restrições, os agentes enviam e recebem mensagens, e alteram o valor da variável. A primeira mensagem dos agentes diz respeito ao valor escolhido na inicialização do algoritmo DCOP que será utilizado. A função *Booleana OtimizarEscolha(x_i)* (Linha 16) verifica se existe algum outro valor que minimize (ou maximize) a função objetivo do problema. Caso exista, o agente altera o valor da variável, envia mensagens para os agentes vizinhos.