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

PAULO ROBERTO FERREIRA JÚNIOR

Coordenação de Sistemas Multiagente Atuando em Cenários Complexos: uma Abordagem Baseada na Divisão de Trabalho dos Insetos Sociais

Tese apresentada como requisito parcial
para a obtenção do grau de
Doutor em Ciência da Computação

Profa. Dra. Ana Lúcia Cetertich Bazzan
Orientador

Porto Alegre, maio de 2008

CIP – CATALOGAÇÃO NA PUBLICAÇÃO

Ferreira Júnior, Paulo Roberto

Coordenação de Sistemas Multiagente Atuando em Cenários Complexos: uma Abordagem Baseada na Divisão de Trabalho dos Insetos Sociais / Paulo Roberto Ferreira Júnior. – Porto Alegre: PPGC da UFRGS, 2008.

123 f.: il.

Tese (doutorado) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR–RS, 2008. Orientador: Ana Lúcia Cetertich Bazzan.

1. Organização em Sistemas Multiagente. 2. Comportamento Coletivo e Emergente em Agentes. 3. Alocação de Recursos e Tarefas em Sistemas Multiagente. 4. Adaptação e Aprendizado. I. Bazzan, Ana Lúcia Cetertich. II. Título.

AGRADECIMENTOS

Primeiramente, gostaria de agradecer aos órgãos de fomento que deram suporte a execução desta tese: ao CNPq, que proveu minha bolsa de doutorado; à Capes, que financiou o projeto intitulado "Modelagem de Organizações Descentralizadas em Coordenação de Agentes: Comportamento Emergente em Superorganismos e Aplicações", apoiado pelo acordo de cooperação entre Brasil e Alemanha (Capes-Baviera), do qual esta tese fez parte; e ao Air Force Office of Scientific Research (AFOSR), que financia o projeto intitulado "*Coordination and Task Allocation in Disaster Scenarios Using Swarm Intelligence and Multiagent Techniques*" (*grant number* FA9550-06-1-0517), do qual esta tese faz parte atualmente.

Fico especialmente grato à minha orientadora, Professora Ana Bazzan, cuja competência como pesquisadora é notória e inspiradora. Sua contribuição para minha formação como professor e pesquisador é indelével. Tenho tentado expressar minha gratidão com trabalho e dedicação. Espero sinceramente ter atendido as expectativas que ela depositou em mim confiando-me parte de suas idéias.

Tive a sorte de trabalhar em um laboratório (o MASLAB) com um grupo de pesquisa especialmente talentoso. Obrigado aos colegas Licurgo, Denise, Leonardo, Toyama, Bruno, Alexandre, Daniela, Fabiana e Fernando pelo auxílio, companheirismo e amizade. Agradeço, em especial, ao colega Felipe Boffo que em seu trabalho de diplomação contribuiu direta e significativamente para a realização desta tese.

Gostaria de registrar também minha gratidão às pessoas que despenderam algum tempo comigo, discutindo as idéias principais e levantando questões importantes sobre meu trabalho. Obrigado à Franziska Klügl, Martin Middendorf, Jomi Hübner, Vitor Lesser e Roberto da Silva. Agradeço em especial à Luiz Chaimowicz, Graçaliz Demuro e Luciana Buriol, membros da banca examinadora, cujas contribuições foram muito importantes para a conclusão deste trabalho.

Agradeço também o apoio dos amigos Eurico, Juliano e Roger. Estes amigos, que considero como irmãos, tornaram a computação muito mais interessante e são responsáveis por praticamente todo o conhecimento e experiência que tenho na área. Obrigado também a todos os amigos da CaMbAdA (grupo de colegas da graduação) que tornam a vida ainda mais divertida. Aos amigos Marcelo e Flávia, obrigado pelo companherismo e o carinho com as crianças.

Finalmente, obrigado aos meus pais, Paulo e Telma, pelo carinho e suporte incondicional. Sou muito grato também pela compreensão de minha esposa, Débora, a qual fez inúmeros sacrifícios para que eu pudesse me dedicar a esta tese. Aos meus filhos, Lucas e Marina, devo o tempo que não dediquei a eles durante o doutorado, que vou recuperar com juros e correção em breve.

SUMÁRIO

LISTA DE ABREVIATURAS E SIGLAS

ACO	*Ant Colony Optimization*
B&B	*Branch-and-Bound*
COP	*Constraint Optimization Problem*
CSP	*Constraint Satisfaction Problem*
DCOP	*Distributed Constraint Optimization Problem*
DisCSP	*Distributed Constraint Satisfaction Problem*
DRCPSP	*Distributed RCPSP*
DiMES	*Distribute Multi-Event Scheduling*
DTC	*Design-to-Criteria*
E-GAP	*Extended GAP*
GAP	*Generalized Assignment Problem*
GPGP	*General Partial Global Planning*
JH	*Juvenile Hormone*
LA-DCOP	*Low-communication Aproximation DCOP*
NLE	*Non-Local Effect*
MSP	*Machine Sequencing Problem*
OptAPO	*Optimal Asynchronous Partial Overlay*
PSPLIB	*Project Scheduling Problem Library*
QAF	*Quality Accumulation Function*
RCPSP	*Resource-Constrained Project Scheduling Problem*
SMA	Sistemas Multiagente
TG	*Task Group*
TS	*Task Structure*

LISTA DE SÍMBOLOS

A	matriz de alocação do GAP
a_{ij}	valor da i-ésima linha e a j-ésima coluna da matriz de alocação A
α	taxa de desconto para diminuir o impacto do viés de atualização sobre o estímulo
$\mathcal{B}_j$	conjunto de subtarefas da tarefa j
$\|\mathcal{B}_j\|$	tamanho do conjunto de subtarefas da tarefa j
c_j	custo do método j, definido na TS
$CA_{TG}(\mathcal{S}_n)$	custo total do escalonamento $\mathcal{S}_n$
χ	número de iterações para a obtensão de um escalonamento válido no Swarm-RCPSP
d_j	duração da tarefa j
δ	constante para diminuir o valor do estímulo com o passar do tempo
$D(\mathcal{S}_n)$	desempenho dos agentes no escalonamento $\mathcal{S}_n$
$distancia(j, i)$	distância euclidiana no mapa entre o agente i e a tarefa j
η	número de tentativas para se obter o melhor escalonamento válido possível no Swarm-RCPSP
$\mathcal{F}_j$	conjuntos que contém as tarefas que se relacionam com a tarefa j através dos NLEs restritivos *hinder*
ϕ	crescimento da intensidade em função do tempo
$\Gamma(j)$	viés de atualização do estímulo da tarefa j
$\mathcal{H}_j$	conjuntos que contém as tarefas que se relacionam com a tarefa j através dos NLEs não restritivos *facilitate*
i	agente
$\mathcal{I}$	conjunto de agentes
j	tarefa ou método
$\mathcal{J}$	conjunto de tarefas
k_{ij}	competência do agente i executar a tarefa j
λ	constante que captura as diferentes maneiras na percepção dos agentes

$\mathcal{M}_n$	conjunto de tarefas escalonadas no escalonamento n
$\|\mathcal{M}_n\|$	número de tarefas escalonadas no escalonamento n
μ	constante que define o número de escalonamentos entre as atualizações
N_{act}	número de indivíduos ativos na colônia
N	número de indivíduos que podem se tornar ativos na colônia
ψ	fator de escala associado à eficiência da realização da tarefa
q_ij	qualidade da tarefa j quando executada pelo agente i
$QAF_{TG}(\mathcal{S}_n)$	qualidade acumulada total segundo as QAFs para o escalonamento $\mathcal{S}_n$
Q_r	quantidade total disponível associada a cada recurso $r \in \mathcal{R}$
$P(j)$	conjunto de predecessoras diretas de j
r_i	quantidade de recursos do agente i
$\mathcal{R}$	conjunto de recursos disponíveis
ρ	coeficiente de esquecimento
s_{ij}	estímulo da tarefa j para o agente i
$s_{ij}(t)$	estímulo local da tarefa j na rodada t
$s_{ij}(t-\mu)$	estímulo local da tarefa j em uma rodada anterior $t-\mu$
$\|\mathcal{S}_n\|$	duração total do escalonamento $\mathcal{S}_n$
$\|\mathcal{S}_{n^*}\|$	duração total do melhor escalonamento $\mathcal{S}_{n^*}$
$start_n(j)$	tempo onde a tarefa j teve sua alocação iniciada no escalonamento n
$start_{n^*}(j)$	tempo em que a tarefa j teve sua alocação iniciada no melhor escalonamento $n \in \mathcal{S}$
θ_{ij}	limiar interno do agente i em relação a tarefa j
u_{ij}	unidades de recurso do agente i consumidos pela tarefa j
w_{ij}	recompensa local para a alocação da tarefa j pelo agente i
ξ	coeficiente de aprendizado

LISTA DE FIGURAS

LISTA DE TABELAS

RESUMO

Agentes atuando em sociedade devem agir de maneira coerente para atingir um objetivo comum. A coordenação nos sistemas multiagente previne o comportamento caótico dos agentes, permite que o sistema lide com restrições globais e a interdependência entre os agentes, e faz com que o sistema possa ser composto por agentes com diferentes competências. A coordenação pode ser baseada na estrutura organizacional, onde a comunidade de agentes atua a favor de um objetivo comum através da forma como estão organizados. Em ambientes dinâmicos a organização dos agentes deve se adaptar a mudanças nos objetivos do sistema, na disponibilidade de recursos, nos relacionamentos entre os agentes, e assim por diante. Esta flexibilidade é um problema chave nos sistemas multiagente e está relacionada a modelos de adaptação como os observados nos insetos sociais. O presente trabalho propõe uma abordagem para a geração e adaptação da organização de um sistema multiagente, em tempo de execução, utilizando como base os modelos teóricos de organização das colônias de insetos sociais. Esta abordagem enfoca a alocação e o escalonamento dinâmicos de tarefas distribuídos entre agentes com diferentes competências e em ambientes de larga escala. Dois cenários principais são utilizados para experimentar e validar a abordagem proposta. Estes cenários estão baseados em problemas de pesquisa operacional denominados *Resource-Constrained Project Scheduling Problem* (RCPSP) e o *Generalized Assignment Problem* (GAP). Este trabalho contribui para o avanço do estado-da-arte no estudo e desenvolvimento de sistemas multiagente e na modelagem e aplicação de técnicas de inteligência de enxames em problemas computacionais. A abordagem proposta para coordenação de agentes em cenários complexos é nova, eficaz e robusta. De maneira geral, esta abordagem contribui para busca da solução de problemas de coordenação de sistemas multiagente aplicados a problemas reais.

Palavras-chave: Organização em Sistemas Multiagente, Comportamento Coletivo e Emergente em Agentes, Alocação de Recursos e Tarefas em Sistemas Multiagente, Adaptação e Aprendizado.

Coordination in Multiagent Systems Applied to Complex Scenarios Based on the Theoretical Models of Division of Labor in Social Insects

ABSTRACT

A community of individual agents must work in a coherent manner to reach some common goal. The coordination process in multiagent systems prevents chaotic behavior of agents, makes the system able to deal with global constraints and inter-agents dependencies, and allows the system to be composed of agents with different capabilities. This process is normally based on the organizational structure, where the community of agents works towards the system goal through the manner they are organized. However, in dynamic environments, agents must be able to adapt to the changing goals of the system, to the resources available, to their relationships with another agents, to changes on the environment and so on. This problem is a key one in multiagent systems and relates to models of adaptation, such as those observed among social insects. This work proposes a new approach to generate and adapt the multiagent organization on the fly based on the theoretical models of social insects colonies organization. This approach focuses on distributed dynamic scheduling and task allocation using agents with different capabilities in large scale environments. Two main scenarios have been used to experiment and validate the proposed approach: the Resource-Constrained Project Scheduling Problem (RCPSP) and the Generalized Assignment Problem (GAP). This work contributes to advancing the state-of-the-art in the study and development of multiagent systems and in the modeling and application of swarm intelligence techniques. The proposed approach to coordinate agents in complex scenarios is novel, effective and robust. This approach contributes to the search of coordination solutions to multiagent systems real applications.

Keywords: Organization in Multiagent Systems, Collective and Emergent Agent Behavior, Task and Resource Allocation in Agent Systems, Adaptation and Learning.

1 INTRODUÇÃO

O paradigma baseado em sistemas multiagente tem a coordenação como ponto central. A coordenação é o processo no qual os agentes se engajam para garantir que o conjunto deles que compõe o sistema irá atuar de uma maneira coerente (NWANA; LEE; JENNINGS, 1996). Quando agentes distribuídos trabalham para cumprir determinados objetivos, eles devem agir como uma unidade, coordenando suas ações, minimizando esforços redundantes, compartilhando recursos, dentre outros aspectos.

Em muitas aplicações reais de um sistema multiagente (SMA), um grande número de agentes deve executar um igualmente grande número de tarefas. Tais tarefas e suas características podem mudar com o tempo. Além disso, neste tipo de ambiente, as informações podem ser incompletas dada a observação parcial e as restrições de comunicação que podem ser impostas aos agentes por diversos fatores, inclusive pela dinamicidade do ambiente.

Em outras palavras, cenários do mundo real são: de larga escala, onde existem centenas de agentes e tarefas e as decisões precisam ser tomadas o mais rápido possível; dinâmicos, onde novas tarefas podem ingressar no sistema a qualquer tempo e os requisitos das tarefas podem mudar; e parcialmente observáveis, onde os agentes podem ter uma visão local e incompleta sobre os objetivos do sistema. Este tipo de cenário, o qual optou-se por denominar resumidamente de **cenário complexo**, está se tornando cada vez mais comum nas indústrias, na robótica e na própria computação.

A principal questão abordada por este trabalho é:

Como coordenar um grande número de agentes em um sistema multiagente para atuar em cenários complexos na busca de um objetivo comum?

Assume-se que:

- O agentes são cooperativos, todos desejam realizar o maior número possível de tarefas com a máxima eficiência, para que os objetivos do sistema sejam alcançados;
- Os agentes possuem arquitetura reativa, onde suas decisões são tomadas em reação as percepções individuais a cerca do ambiente;
- A coordenação entre os agentes deve ocorrer de forma distribuída, utilizando um canal de comunicação limitado.
- Os agentes não lidam com a resolução distribuída de conflitos na percepção das tarefas. Assume-se que a tarefa pode ser identificada como alocada ou simples-

mente que as tarefas não são percebidas simultaneamente pelos agentes. Este tipo de resolução de conflito é um problema complexo e foge do escopo deste trabalho.

A coordenação de um SMA atuando em ambientes com as características descritas acima pode ser obtida através de sua estrutura organizacional. A descrição desta estrutura organizacional define os objetivos do sistema, as tarefas necessárias para atingir estes objetivos, como estas tarefas se dividem em subtarefas e como estas se relacionam entre si. Os agentes precisam então deliberar sobre em quais tarefas eles irão se engajar e quando cada tarefa será realizada.

Desta forma, a organização de um SMA descreve o cenário em que o sistema está inserido e quais os passos necessários para que os objetivos deste sistema sejam atingidos. Tal descrição pode ser especificada utilizando uma linguagem independente de domínio como o TÆMS (DECKER; LESSER, 1993). A principal dificuldade com relação à coordenação dos agentes segundo uma determinada estrutura organizacional está na definição das ações dos agentes em relação às tarefas nas quais o objetivo do sistema foi decomposto.

Este trabalho parte da idéia de que é possível obter a emergência da organização dos agentes em relação a sua atuação nas tarefas através da utilização de modelos teóricos inspirados na divisão de trabalho em colônias de insetos sociais.

As colônias de insetos sociais têm uma das estratégias de sobrevivência mais bem sucedidas da natureza. Esta estratégia é centrada na organização das colônias, cujas principais características são a divisão do trabalho e a especialização. As necessidades das colônias se modificam com o passar do tempo. Estas modificações estão associadas com a fase de desenvolvimento da colônia, a época do ano, a disponibilidade de alimentos, a pressão de predadores e as condições climáticas. A despeito dessas drásticas variações nas condições da colônia, os insetos sociais têm notório sucesso ecológico.

Uma colônia de insetos sociais opera sem coordenação explícita. Um inseto pertencente à casta operária individualmente não conhece as necessidades da colônia como um todo; este tem apenas informações locais simples, capacidade de comunicação extremamente limitada e ninguém é encarregado de coordenar suas ações. Através da agregação destes trabalhadores, o comportamento da colônia emerge sem qualquer tipo de planejamento ou coordenação explícita. A chave para este comportamento emergente está na plasticidade da divisão de trabalho nas colônias (ROBISON, 1992). Estas colônias respondem à variação nas condições do ambiente através do ajuste das taxas de trabalhadores engajados nas diversas tarefas que precisam ser executadas.

Este trabalho propõe uma abordagem para gerar e adaptar a organização de sistemas multiagente atuando em cenários complexos, através da alocação e re-alocação de tarefas entre os agentes, utilizando como base os modelos teóricos de divisão do trabalho dos insetos sociais.

O problema em questão pode ser redefinido como um problema de alocação e (ou) de escalonamento dinâmico e distribuído de tarefas em larga escala entre agentes com diferentes competências e percepções. Por alocação entende-se o processo de atribuir agentes a tarefas imediatamente, onde não é possível despender tempo planejando, ou não se tem

informação suficiente, e os agentes podem apenas adaptar-se utilizando experiências passadas. Por escalonamento entende-se o processo de determinar que agentes realizaram quais tarefas e organizar a execução dessas tarefas em relação ao tempo. Neste último, os agentes dispõem de tempo e informação suficientes para planejar a alocação das tarefas e sua disposição no tempo deve ser feita de maneira a minimizar o tempo total necessário para a realização das mesmas.

Dois problemas da área de pesquisa operacional modelam partes diferentes, porém complementares, do problema descrito acima. O *Generalized Assignment Problem* (GAP) (MARTELLO; TOTH, 1990) e o *Resource-Constrained Project Scheduling Problem* (RCPSP) (BRUCKER et al., 1999). Estes problemas não são originalmente dinâmicos nem tão pouco distribuídos. O número de tarefas e agentes (ou recursos) são fixos. Contudo, podem ser facilmente estendidos para lidar com essas características.

No GAP tarefas são associadas a agentes de acordo com restrições de competência e disponibilidade de recurso destes agentes. O objetivo é buscar uma associação dos agentes as tarefas nas quais estes são mais competentes. O GAP é formalmente definido na seção 5.1. No RCPSP, tarefas devem ser escalonadas considerando um conjunto limitado de recursos por unidade de tempo. As tarefas são relacionadas por restrições de interdependência de execução, ou seja, determinadas tarefas devem ser executadas antes de outras. O objetivo é buscar um escalonamento de todas as tarefas utilizando o menor intervalo de tempo possível. O RCPSP é formalmente definido na seção 6.1.

Estes problemas compõem dois cenários onde a abordagem proposta neste trabalho foi experimentada e validada. Além de modelar vários aspectos de aplicações reais, estes problemas foram largamente estudados, existindo algoritmos e conjuntos de dados disponíveis que podem ser usados para comparação. A abordagem proposta foi aplicada primeiramente na versão estendida do GAP (E-GAP) e, em seguida, na versão distribuída do RCPSP (DRCPSP).

Este trabalho contribui para o avanço do estado-da-arte no estudo e desenvolvimento de SMAs e na modelagem e aplicação de técnicas de inteligência de enxames em problemas computacionais, onde deve-se destacar:

- A abordagem proposta é nova, eficaz e robusta, podendo ser utilizada para coordenação distribuída de agentes em cenários complexos. Esta abordagem oferece simultaneamente os mecanismos necessários para a alocação (imediata) e o escalonamento (com planejamento) de tarefas, podendo ser vista como uma técnica geral para a coordenação de SMAs aplicados a problemas reais.

- O algoritmo Swarm-GAP, concebido a partir da abordagem proposta, resolve de forma aproximada o E-GAP. O E-GAP vem sendo utilizado para modelar muitos problemas de alocação de tarefas, incluindo cenários de coordenação de equipes de resgate e salvamento, onde um número muito restrito de soluções distribuídas foram propostas;

- O algoritmo Swarm-RCPSP, também concebido a partir da abordagem proposta, resolve de forma aproximada o DRCPSP. Este problema modela inúmeras aplicações reais, principalmente relacionadas ao escalonamento de tarefas em manufaturas. Uma abordagem distribuída, como a que está sendo proposta, junta-se ao esforço de outras soluções para tentar resolver problemas de escalonamento, que são críticos no estilo moderno de produção;

- A abordagem proposta é fortemente inspirada no comportamento dos insetos sociais. As extensões e modificações nos modelos teóricos de organização dos insetos sociais para resolver problemas de forma distribuída foram aqui propostas, avaliadas e validadas. Estas modificações e extensões contribuem para o desenvolvimento da pesquisa na área de inteligência de enxames.

Todos os mecanismos da abordagem proposta foram experimentados e validados. Os resultados empíricos destes experimentos mostraram que a abordagem proposta, baseada em modelos teóricos da divisão do trabalho pelos insetos sociais, pode ser utilizada com sucesso para a alocação e (ou) escalonamento distribuído de tarefas. Utilizando o algoritmo Swarm-GAP, os agentes foram capazes de obter alocações de qualidade comparáveis a um outro algoritmo recentemente proposto que apresentou resultados proeminentes, sendo superior a este em alguns casos. O algoritmo Swarm-RCPSP permitiu aos agentes obter escalonamentos eficientes e significativamente melhores que os obtidos por uma abordagem distribuída gulosa.

Este trabalho está organizado da seguinte forma:

Capítulo 2 Neste capítulo são abordados alguns fundamentos necessários para a compreensão deste trabalho a respeito da coordenação e a organização dos sistemas multiagente. Além disso, serão apresentados os trabalhos relacionados à presente tese na área de sistemas multiagente;

Capítulo 3 Neste capítulo serão discutidos os aspectos acerca dos insetos sociais relevantes a este trabalho, onde o principal foco é a forma como a divisão do trabalho emerge nas colônias a partir da interação entre os elementos que a compõe. Além disso, serão apresentados os trabalhos relacionados a presente tese na área de inteligência de enxames;

Capítulo 4 Neste capítulo será detalhada a abordagem proposta;

Capítulo 5 Neste capítulo são apresentados e discutidos os experimentos utilizados na validação da abordagem proposta em termos de uma versão distribuída e dinâmica do *Generalized Assignment Problem* (E-GAP). Os resultados gerais da abordagem proposta são essencialmente comparados com os resultados obtidos por um proeminente algoritmo equivalente que foi recentemente proposto.

Capítulo 6 Neste capítulo são apresentados e discutidos os experimentos utilizados na validação da abordagem proposta através de sua aplicação em uma versão distribuída do *Resource-Constrained Project Scheduling Problem* (DRCPSP). Os resultados gerais da abordagem proposta são comparados com dados de *benchmark* para este problema.

Capítulo 7 Uma vez que os experimentos conduzidos com a abordagem proposta são aplicados de maneira que alguns dos seus mecanismos são testados e validados isoladamente nos capítulos 5 e 6, neste capítulo são discutidos e analisados como os mecanismos da abordagem proposta podem ser integrados e são analisados os impactos disto sobre a validação apresentada nos referidos capítulos.

Capítulo 8 Neste capítulo serão apresentadas as conclusões e os aspectos não cobertos por esse trabalho que podem levar a trabalhos futuros.

2 COORDENAÇÃO EM SISTEMAS MULTIAGENTE

Este capítulo apresenta a fundamentação teórica deste trabalho no que diz respeito aos sistemas multiagente (SMAs) e está organizado da seguinte maneira: na seção 2.1 são discutidos alguns conceitos de SMAs, que são fundamentais para a compreensão deste trabalho; na seção 2.2 são discutidos os aspectos que envolvem a coordenação de agentes, a maneira como se obtém coordenação a partir da organização e as dificuldade de se obter uma organização eficiente em cenários complexos; na seção 2.3 o modelo para a descrição da estrutura organizacional que este trabalho utiliza é detalhado; e, finalmente, na seção 2.4 são abordados os trabalhos relacionados na área.

2.1 Agentes Inteligentes e Sistemas Multiagente

Um agente inteligente pode ser definido como uma entidade de software capaz de realizar ações de forma autônoma no ambiente em que está situado, visando atingir os objetivos para o qual foi projetado (WOOLDRIDGE, 2002). As ações que um agente pode realizar são normalmente restritas e representam a capacidade do agente de modificar o ambiente em que está atuando. A principal questão enfrentada pelos agentes inteligentes é decidir, dentre todas as ações de seu repertório, quais tomar para melhor atingir seus objetivos.

RUSSEL; NORVIG (2003) classificaram as propriedades dos ambientes em que os agentes atuam da seguinte forma:

Completamente Observável ou Parcialmente Observável. os ambientes completamente observáveis são aqueles em que os agentes obtém informações completas, corretas e atualizadas sobre seu estado. Nos ambientes em questão neste trabalho, as informações que os agentes possuem, acerca dos estados do ambiente no qual estes estão inseridos, podem ser consideradas corretas e atualizadas. Porém, dada a limitação da percepção dos agentes, que é estritamente local, os estados do ambiente são observados parcialmente. Os agentes, com isso, possuem informações incompletas sobre o ambiente como um todo.

Determinístico ou Estocástico. os ambientes determinísticos são aqueles em que não existem incertezas sobre os impactos e resultados de cada uma das ações dos agentes sobre ele. Os ambientes em questão nesse trabalho são determinísticos, nenhuma variação nos efeitos das ações dos agentes foi considerada. Essa questão é abordada no seção 8.2 com um tópico em aberto para trabalhos futuros.

Episódico ou Seqüencial. os ambientes episódicos são aqueles em que a experiência dos agentes é dividida em episódios. As ações realizadas em um episódio não interfe-

rem nas ações que são realizadas em outro. Neste trabalho ambos foram cosiderados. Como comentado na seção 1, o problema em questão neste trabalho pode ser definido como um problema de alocação e (ou) escalonamento de tarefas. No caso da alocação, o ambiente considerado é episódico pois as decisões acerca de quais tarefas alocar em cada momento são independentes. Já no caso do escalonamento tem-se um ambiente seqüencial, pois o escalonamento de uma tarefa pode depender do escalonamento prévio de outra.

Estático ou Dinâmico. os ambientes estáticos são aqueles que apenas mudam para refletir a atuação dos agentes sobre eles. Contrariamente, nos ambientes dinâmicos ocorrem mudanças produzidas por outros fatores que fogem completamente do controle dos agentes que neles atuam. Este trabalho foca ambientes que se caracterizam essencialmente pelo seu dinamismo.

Discreto ou Contínuo. os ambientes discretos são caracterizados pela conjunto fixo e finito de ações que podem ser executadas sobre eles. Os ambientes considerados nesse trabalho são essencialmente discretos.

Agente Único ou Multiagente. os ambientes multiagente são caracterizados por envolverem as ações coordenadas de mais de um agente. Este trabalho enfoca os ambientes multiagente, cujas características serão melhor discutidas mais adiante no texto.

Além das propriedades descritas acima, pode-se incorporar a **escala do ambiente (larga ou restrita)** como mais um aspecto que se tornou relevante para o projeto e construção de agentes. Agentes inteligentes atuando em ambientes de larga escala têm seu processo de deliberação sobre quais ações tomar ainda mais dificultado. Nesses ambientes, o número de ações possíveis é muito grande e o processo de deliberação precisa ser computacionalmente eficiente para garantir a aplicabilidade do agente. Este trabalho considera agentes atuando em ambientes de larga escala.

Os agentes que atuam em ambientes com as propriedades em questão apresentam algumas características fundamentais que, na prática, é o que os permite atuar com eficiência nesses ambientes (WOOLDRIDGE, 2002), as quais são: reatividade, os agentes inteligentes percebem seu ambiente e agem em resposta a mudanças nesse ambiente; próatividade, os agentes inteligentes realizam ações, por iniciativa própria, orientadas por seus objetivos; e sociabilidade, os agentes inteligentes são capazes de interagir com outros agentes na busca por seus objetivos.

A metodologia seguida na construção de um agente determina sua **arquitetura** (RUSSEL; NORVIG, 2003). Uma classificação geral para as diferentes arquiteturas dos agentes foi definida em WOOLDRIDGE; JENNINGS (1994):

Arquiteturas Deliberativas. onde as decisões dos agentes são tomadas através de processos sofisticados de deliberação. O agente possui uma representação interna do mundo e um estado mental explícito que pode ser modificado por alguma forma de raciocínio simbólico.

Arquiteturas Reativas. onde as decisões são tomadas com base em regras de reação ao estado atual do ambiente. Como por exemplo, a arquitetura de subsunção (BROOKS, 1991), na qual regras simples indicam que uma determinada situação leva a uma determinada ação;

Arquiteturas Híbridas. que combinam as características das duas arquiteturas anteriormente citadas.

A abordagem proposta nesse trabalho foi concebida para compor o processo de deliberação de agentes com arquitetura reativa. Tal arquitetura tem vantagens quando aplicada a cenários complexos, dentre elas a simplicidade, o baixo custo computacional e a robustez contra falhas. De forma resumida, através da abordagem que está sendo proposta aqui, os agentes decidem reativamente que ações irão realizar seguindo os modelos teóricos de como os insetos sociais tomam essa decisão.

Agentes inteligentes atuando em grupos, cada um com diferentes capacidades de percepção e ação no ambiente, compõem os sistemas multiagente. Nestes sistemas, dois ou mais agentes interagem ou trabalham em conjunto na busca de um objetivo comum (WOOLDRIDGE, 2002). Para que um agente possa atuar nessa busca, é necessária a existência de uma infra-estrutura que permita a comunicação entre os agentes que compõem o sistema. Os agentes em um SMA podem ser homogêneos ou heterogêneos. No trabalho discutido aqui são considerados os SMAs compostos por agentes com diferentes características, portanto heterogêneos.

Alguns problemas, como os abordados nesse trabalho, são inerentemente multiagente, ou seja, existe a necessidade de que vários agentes estejam engajados na busca de sua solução. A próxima seção detalha como os agentes devem se comportar em um SMA para que sua ações autônomas façam sentido quando estes buscam um objetivo coletivo.

2.2 Coordenação de Agentes

Como garantir que uma comunidade de agentes individuais trabalhem juntos de uma maneira coerente para atingir um objetivo comum? O processo de coordenação responde esta questão (NWANA; LEE; JENNINGS, 1996). A coordenação impede o comportamento caótico do sistema, permite que o sistema lide com restrições globais e a interdependência entre os agentes, e permite que o sistema seja composto por agentes diferentes com diferentes competências. De uma maneira mais geral, a coordenação aumenta o desempenho dos sistemas multiagente. Existem muitas maneiras de coordenar agentes em SMAs, classificadas em quatro categorias por NWANA; LEE; JENNINGS (1996):

Estrutura Organizacional. essa técnica explora uma descrição prévia da estrutura organizacional do SMA. Esta estrutura define implicitamente as responsabilidades dos agentes, suas capacidades, suas relações, etc. Mais adiante neste texto serão discutidos mais detalhes dessa técnica.

Contratos. nesse caso a coordenação se dá por um processo descentralizado de relações de mercado. Os agentes podem tornar-se gerentes, partir um problema em sub-problemas e contratar outros agentes para realizá-los e (ou) monitorar a qualidade geral da solução. Os agentes contratados podem, recursivamente, fazer novas contratações. Usualmente o processo de contratação se dá através de leilões que consideram a capacidade dos agentes para realizar as tarefas.

Planejamento Multiagente. esta técnica se baseia no planejamento conjunto entre os agentes para detalhar suas ações futuras e as interações necessárias para que o objetivo do sistemas seja atingido. Este planejamento, que pode ser distribuído ou centralizado, evita inconsistências e ações conflitantes.

Negociação. esta técnica explora o processo de comunicação entre os agentes para que estes componham acordos mútuos sobre diversos aspectos de sua atuação conjunta no ambiente. Existem várias técnicas empregadas nessa negociação que são essencialmente caracterizadas como: baseadas na teoria dos jogos; baseadas em planejamento; e inspiradas na interação entre humanos.

Este trabalho está interessado na coordenação baseada na estrutura organizacional, onde a comunidade de agentes age como uma unidade. Os agentes atuam a favor de um objetivo comum através da forma como estão organizados.

Existem muitos trabalhos na literatura que apresentam diferentes abordagens para a definição da organização de um SMA. LEMAITRE-LEÓN; EXCELENTE-TOLEDO (1998) classificaram estas abordagens em dois grupos: centrada nos agentes e centrada na organização. Nas abordagens centradas em agentes, o SMA não tem uma representação explícita da organização. Cada agente constrói sua própria representação de acordo com sua visão do comportamento dos outros agentes. Nas abordagens centradas na organização, esta é explicitamente definida e um modelo de organização deve ser usado para representar os aspectos envolvidos na interação entre os agentes.

KIRN; GASSER (1998) formaliza o processo de modelar organizações em sistemas multiagente, discutindo as atividades e elementos relacionados com o projeto formal da estrutura organizacional. Muitos modelos foram propostos na literatura. Na abordagem proposta neste trabalho será utilizado o modelo TÆMS (DECKER; LESSER, 1993) como a base da organização do SMA. A linguagem TÆMS é utilizada aqui para modelar a estrutura de tarefas e as atividades necessárias para atingir o objetivo do sistema. Este modelo será detalhado na seção 2.3.

Segundo LESSER (1998), a estrutura organizacional de um sistema multiagente é um dos aspectos mais significativos para seu sucesso. A organização dos agentes depende das características do sistema quanto a forma de atingir seus objetivos, perceber o ambiente e determinar as atividades e interações entre os agentes. O problema está em definir que tipo de organização se adequa melhor a estas necessidades.

Uma maneira simples de resolver este problema é definir a organização de forma estática, o que significa encontrar os requisitos e projetar a organização mais apropriada. Uma vez que isto é feito *off-line*, as vantagens de uma organização bem definida se tornam desvantagens em ambientes dinâmicos.

Agentes atuando em ambientes dinâmicos devem estar habilitados a lidar com requisições de tarefas aparecendo a todo momento, mudanças nos recursos disponíveis, entre outros. Estas questões tornam difícil o projeto de uma organização que considere todas as futuras situações possíveis. A medida que os sistemas multiagente estão sendo utilizados em problemas dinâmicos, estruturas organizacionais estáticas com definições rígidas se tornam ineficientes.

Desta forma, SMAs precisam lidar com a dinâmica dos ambientes como a variação no número de agentes, mudanças no ambiente e nos objetivos do sistema. A questão é como produzir uma estrutura organizacional dada uma situação em particular. HORLING; LESSER (2004) abordam estratégias para o projeto de organizações em sistemas multiagente, onde vários paradigmas são considerados. A adaptação da organização é tratada como um tópico importante acerca do projeto de organizações de sistemas multiagente que atuam em ambientes dinâmicos.

O processo de gerar, adaptar e modificar a organização dinamicamente em SMAs é usualmente chamada de "auto-organização". Alguns autores preferem denominar esse processo de "adaptação organizacional" (HORLING; BENYO; LESSER, 2001) ou ainda

"organização auto-projetada". SCHILLO et al. (2002) definem que um sistema multiagente é auto-organizado se os agentes: estão habilitados a determinar a estrutura organizacional mais apropriada para o sistema em tempo de execução; e são capazes de mudar esta estrutura conforme o ambiente muda. No presente trabalho, o termo "adaptação da organização" será utilizado para representar essa característica. Na seção 2.4.1 serão discutidas abordagens que utilizam aspectos de auto-organização para adaptar a estrutura organizacional do sistema.

Além das dificuldades impostas pelos ambientes dinâmicos uma nova característica mencionada anteriormente, também do ambiente, tem sido foco da atenção dos pesquisadores. Cada vez mais os sistemas multiagente vêm sendo aplicados em ambientes de larga escala, onde centenas de agentes devem realizar centenas de tarefas. Quando os ambientes dinâmicos tornam-se de larga escala, os problemas relacionados a estes crescem exponencialmente. Não se conhecem abordagens de (auto-) organização eficientes para este tipo de ambiente.

Os demais aspectos que definem os cenários complexos tratados pela abordagem proposta neste trabalho são fatores adicionais na dificuldade de coordenação dos agentes. Por exemplo, a tomada de decisão em um curto espaço de tempo limita a utilização de abordagens cuja complexidade computacional ultrapasse um certo limite de viabilidade.

2.3 TÆMS

A plataforma TÆMS (DECKER; LESSER, 1993), *Design-to-Criteria* (DTC) (WAGNER; LESSER, 2000) e *General Partial Global Planning* (GPGP) (DECKER; LESSER, 1995) é utilizada como uma ferramenta independente de domínio para a descrição da estrutura de tarefas associadas a agentes e a determinação de como estes agentes serão coordenados para que tais tarefas sejam escalonadas.

O TÆMS é uma linguagem concebida para descrever a estrutura de tarefas, ou *task structure* (TS), dos agentes (DECKER; LESSER, 1993). Essencialmente, uma TS é uma forma de representação da decomposição de tarefas em uma árvore. O nível mais alto de uma árvore (raiz) é denominado TG (*task group*) e representa o objetivo do agente. Uma TS então é uma seqüência de métodos e tarefas que descrevem como aquele conjunto de tarefas será executado para que o objetivo do sistema seja atingido. Os métodos são terminais (folhas na árvore) e representam as ações primitivas que os agentes podem executar. Todos os métodos em TÆMS possuem distribuições de probabilidade associadas a qualidade, custo e duração. Assim, o método j tem qualidade q_j, custo c_j, e duração d_j. As subdivisões de uma tarefa são denominadas "subtarefas" e a tarefa pai de uma tarefa na árvore é denominada "supertarefa".

A figura 2.1 traz uma TS com várias tarefas relacionadas a um TG, representadas por elipses. As anotações imediatamente abaixo das elipses, são intituladas *Quality Accumulation Function* - QAF, e representam a forma como a qualidade da resolução das subtarefas ou métodos é usada para compor a qualidade ou o custo da tarefa. Pode-se dizer que as QAFs determinam a semântica correta para as diferentes combinações e ordenações possíveis de resolução das subtarefas, considerando seus efeitos locais. A tabela 2.1 traz as diferentes QAFs que podem ser utilizadas em TÆMS.

A QAF *sync sum* foi recentemente incorporada a uma versão do TÆMS denominada C_TÆMS (SULTANIK; MODI; REGLI, 2007). O C_TÆMS introduz algumas simplificações e modificações que vêm sendo usadas pela comunidade de SMA e foi concebida especificamente para lidar com problemas de coordenação. A abordagem que está sendo

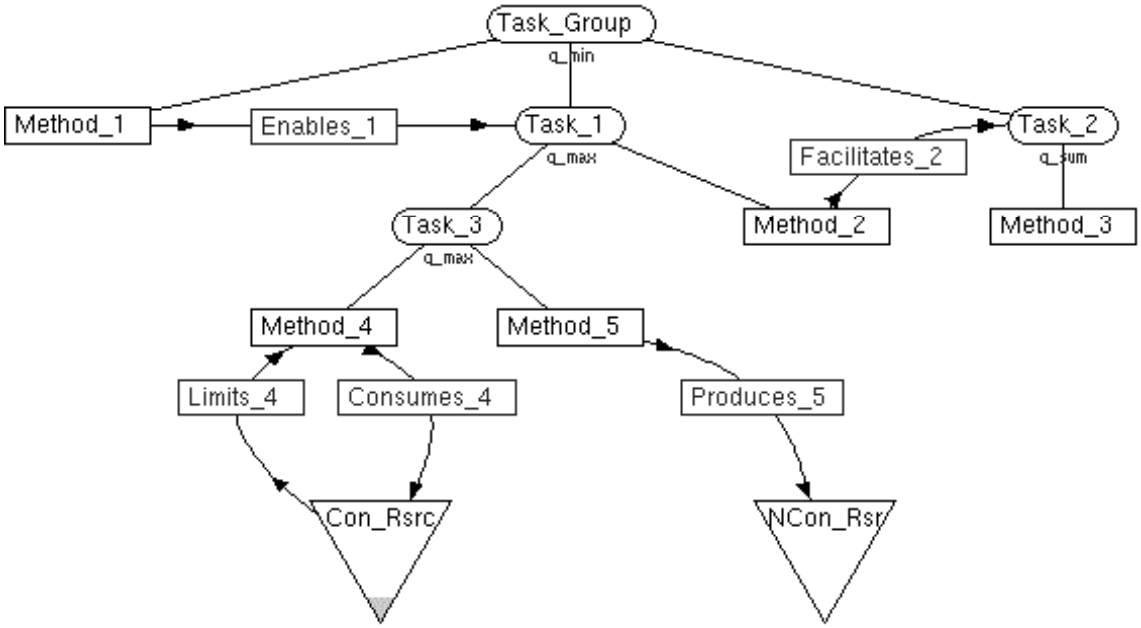


Figura 2.1: Exemplo de uma TS (*task structure*) em TÆMS, figura originalmente publicada em (HORLING et al., 1999).

proposta aqui é capaz de lidar com o C_TÆMS da mesma maneira que lida com sua versão original pois a linguagem de descrição que é adotada aqui é, salvo por esta QAF especificamente, absolutamente equivalente. Esta QAF evita uma série de recursos que têm de ser usados na versão original para produzir esse efeito de sincronização na execução de tarefas, como discutido por LESSER (2002).

A qualidade da execução dos métodos é combinada segundo uma QAF que determina a qualidade da supertarefa destes métodos. Este procedimento de calcular a qualidade da supertarefa em função de suas subtarefas ou métodos segue recursivamente até o TG (das folhas até a raiz), onde então se determina a qualidade da TS como um todo.

Os arcos entre métodos ou tarefas, na figura 2.1, indicam relacionamentos entre tarefas que não pertencem a mesma supertarefa, onde a execução de um método terá um efeito positivo ou negativo, dependendo da qualidade e duração de sua execução, no outro método. Este relacionamento é denominado *non-local effect* (NLE). A tabela 2.2 traz os diferentes relacionamentos que podem existir entre métodos e tarefas, destacando os dois tipos em que podem se enquadrar: restritivo e não restritivo.

Além dos recursos de modelagem mostrados acima, o TÆMS permite modelar aspectos do ambiente através da modelagem de recursos, triângulo na figura 2.1. Um recurso é um conceito abstrato que possui um determinado nível e pode ser descrito como consumível ou não consumível. Relacionamentos entre métodos e recursos descritos como *produces* indicam que, quando um método for realizado com sucesso, uma determinada quantidade do recurso será produzida. Por outro lado, os relacionamentos denominados *consumes* são usados para indicar que o método relacionado precisa consumir uma determinada quantidade do recurso para que possa ser realizado. Existem também relacionamentos que impõem restrições aos métodos caso não existam recursos suficientes, denominados *limits*.

Através destas representações é possível construir a estrutura de tarefas para a resolução de qualquer problema. Esta estrutura é chamada de modelo objetivo do ambiente, e

Tabela 2.1: Funções que determinam o cálculo da qualidade da tarefa - QAF.

QAF	Descrição
min	A qualidade da tarefa é a menor qualidade das subtarefas ou métodos, e todas as subtarefas ou métodos devem ser escalonados obrigatoriamente (AND lógico entre elas).
max	A qualidade da tarefa é a maior qualidade das subtarefas ou métodos, e as subtarefas ou métodos podem ser escalonados em qualquer uma de suas combinações (OR lógico entre elas).
sum	A qualidade da tarefa é a soma da qualidade das subtarefas ou métodos, e as subtarefas ou métodos podem ser escalonados em qualquer uma de suas combinações, quanto mais métodos ou subtarefas escalonados maior será a qualidade da tarefa.
all	A qualidade da tarefa é a soma da qualidade das subtarefas ou métodos, e todas as subtarefas ou métodos devem ser escalonados.
seq min	A qualidade da tarefa é a menor qualidade das subtarefas ou métodos, e todas as subtarefas ou métodos devem ser escalonados obrigatoriamente (AND lógico entre elas) em uma determinada ordem.
last	A qualidade da tarefa é a qualidade da subtarefa ou método terminado por último, e as subtarefas ou métodos devem ser todos escalonados.
seq max	A qualidade da tarefa é a maior qualidade das subtarefas ou métodos, e as subtarefas ou métodos podem ser escalonados em qualquer uma de suas combinações (OR lógico entre elas) que respeitem uma determinada ordem.
seq sum	A qualidade da tarefa é a soma da qualidade das subtarefas ou métodos, e as subtarefas ou métodos podem ser escalonados em qualquer uma de suas combinações que respeitem uma determinada ordem, quanto mais métodos ou subtarefas escalonados maior será a qualidade da tarefa.
seq last	A qualidade da tarefa é a qualidade da subtarefa ou método terminado por último, e as subtarefas ou métodos devem ser todos escalonados e respeitarem uma determinada ordem.
exactly one	A qualidade da tarefa é a maior qualidade das subtarefas ou métodos, e somente uma dessas subtarefas ou métodos podem ser escalonados (XOR lógico entre elas).
sync sum	A qualidade da tarefa é a soma da qualidade das subtarefas ou métodos, e as subtarefas ou métodos devem ser todos escalonados para terem início simultaneamente.

Tabela 2.2: Relacionamentos entre tarefas e/ou métodos.

Relacionamento	Descrição
enable	Relacionamento com comportamento restritivo. O método ou tarefa só poderá ser escalonado se o método ou tarefa relacionada for escalonado.
disable	Relacionamento com comportamento restritivo. O método ou tarefa não poderá ser escalonado se o método ou tarefa relacionada for escalonado.
facilitate	Relacionamento com comportamento não restritivo. O método ou tarefa será escalonado com uma qualidade maior se o método ou tarefa relacionada for escalonado.
hinder	Relacionamento com comportamento não restritivo. O método ou tarefa será escalonado com uma qualidade menor se o método ou tarefa relacionada for escalonado.

não é observado pelos agentes. Cada agente tem uma visão subjetiva desta estrutura de tarefas. A visão subjetiva contém as tarefas e relacionamento que o agente acredita ser o modelo completo de suas alternativas.

O TÆMS oferece um ferramental completo para representar os meios necessários para os agentes atingirem seus objetivos. Realizar a análise deste processo e determinar um curso apropriado para as ações locais de cada um dos agentes, dadas as restrições temporais, é o papel do escalonador. O escalonador utilizado pela plataforma de desenvolvimento e simulação de agentes que incorpora o TÆMS é denominado *Design-to-Criteria* (DTC) (WAGNER; LESSER, 2000).

Em geral o número de escalonamentos possíveis para qualquer TS é significativamente grande e determinar cada um deles é intratável computacionalmente. O conjunto de restrições determinado pelas QAFs e pelas relações entre as tarefas impede a adoção de uma abordagem convencional de determinação da melhor solução. O DTC utiliza uma conjunto de técnicas para lidar com estas características.

O GPGP é um mecanismo de coordenação de agentes que permite a integração das soluções locais dos agentes, evita que tarefas sejam executadas de forma redundante e indica a compatibilidade entre os objetivos individuais. LESSER (2002) discute que, apesar da ampla utilização de sua abordagem em diversos tipos de problemas de coordenação, essa ainda apresenta algumas limitações pela necessidade de recursos computacionais e de comunicação na busca por soluções ótimas. A busca pela solução ótima em problemas de larga escala não pode ser realizada através do GPGP e o DTC. Algoritmos aproximativos distribuídos para a alocação de tarefas para a plataforma TÆMS são incipientes.

Algumas abordagens foram propostas para representar um modelo em TÆMS como um Processo de Decisão de Markov (PDM) (MUSLINER et al., 2006; WAGNER; RAJA; LESSER, 2006). A idéia por traz desta representação é substituir o DTC por algoritmos de formação de políticas para PDM (PUTERMAN, 2005). Contudo, este tipo de abordagem rapidamente torna-se impraticável a medida que o tamanho do problema cresce. Recentemente foi proposto um mecanismo bem sucedido para ajudar os agentes a determinar a melhor política parcial possível respeitando restrições de tempo (PUTERMAN, 2005). Os autores deste mecanismo sugerem o uso dos mesmos mecanismos do GPGP para lidar com a coordenação para a solução de interdependências entre agentes. As abordagens baseadas em PDM lidam apenas com o escalonamento local de cada agente (exatamente

como o DTC).

2.4 Trabalhos Relacionados

Nesta seção serão discutidos os trabalhos relacionados a este que foram apresentados pela comunidade de sistemas multiagente. Primeiramente, na seção 2.4.1, serão discutidas algumas abordagens que utilizam a adaptação para obter uma organização mais eficiente quando o ambiente é dinâmico. Em seguida, na seção 2.4.2, serão apresentadas as abordagens recentemente propostas para a alocação distribuída de tarefas entre agentes.

2.4.1 Auto-Organização

Os primeiros trabalhos em auto-organização de sistemas multiagente enfocam o balanço de carga adaptativo. ISHIDA; YOKOO; GASSER (1990) apresentam uma abordagem para melhorar o desempenho de sistemas de produção. Nesta abordagem, um agente em particular (denominado *problem solver*) compartilha uma coleção de recursos com outros agentes. Requisições para resolver problemas chegam continuamente em taxas variáveis. A abordagem então faz a reorganização dos agentes para que estes possam atender a estas requisições, compondo e decompondo os agentes e re-alocando conhecimento para resolver os problemas.

SHEHORY et al. (1998) apresentam uma abordagem abstrata seguindo as mesmas idéias. Nesta abordagem agentes sobrecarregados podem migrar, passar tarefas para outros agentes, se combinar com outros agentes e se clonar. A clonagem dos agentes aumenta o número de tarefas realizadas pelo sistema, aumentando seu desempenho.

SO; DURFEE (1993) apresentam uma abordagem de auto-organização baseada na alocação de tarefas. Nesta abordagem um conjunto de agentes autônomos está engajado na resolução de problemas através da cooperação. Os agentes percebem sua eficiência na realização de cada tarefa de maneira local e global. Quando um limiar específico de resposta é ultrapassado inicia-se um processo de reorganização. A estrutura organizacional nesta abordagem inclui a questão sobre como as tarefas são decompostas em subtarefas, como as subtarefas são alocadas entre os agentes, quantos agentes estão envolvidos com cada tarefa, etc. Esta abordagem é muito próxima da que está sendo proposta neste trabalho em relação a idéia geral. Contudo, ela apresenta muitas limitações: as tarefas são decompostas em subtarefas independentes, não sendo consideradas a interdependência entre tarefas; os agentes não interagem entre si com o objetivo de se comprometer com a realização de tarefas; não existe o compartilhamento de recursos pelos agentes; etc.

Uma abordagem que pretende ser mais genérica, baseada em TÆMS e composta com um método para auto-diagnóstico foi proposta por HORLING; BENYO; LESSER (2001). Esta abordagem envolve coordenação de alto nível baseada na especificação da organização, o planejamento e a alocação das tarefas entre os agentes e seu escalonamento no tempo. Esta abordagem mostrou bons resultados, mas algumas questões sobre sua eficiência em termos mais gerais continuam em aberto: com relação à comunicação, especialmente com um grande número de agentes, quão eficiente é a organização resultante?

2.4.2 Alocação Distribuída de Tarefas

O *Distributed Constraint Optimization Problem* (DCOP) (MODI et al., 2003) é um formalismo para modelar um grande conjunto de problemas que envolvem a alocação de tarefas e tem sido tema de muitos avanços nos últimos anos. Os DCOPs são uma versão distribuída do *Constraint Optimization Problem* (COP), que por sua vez é derivado do

Constraint Satisfaction Problem (CSP). Nos DCOPs agentes colaboram para obter soluções sob um conjunto distribuído de restrições. A seguir, tais problemas serão discutidos na ordem em que foram concebidos, ou seja, CSP, COP e DCOP.

Os CSPs consistem em problemas de atribuição de valores a variáveis, respeitando restrições de valores entre elas. Cada restrição neste caso é proposicional (falso ou verdadeiro) e o objetivo final é obter uma solução que respeite todas as restrições. Ao longo dos anos foram propostas muitas soluções para os CSPs, onde as mais recentes e aplicáveis são baseadas em técnicas de busca exaustiva como a busca em profundidade (*Depth First Search*) e em largura (*Breadth First Search*). Este tipo de representação, baseada na satisfação de restrições, não é adequada para a grande maioria dos problemas reais onde a solução pode ter graus de qualidade ou custo. Outro aspecto dos problemas reais é que estes normalmente têm muitas restrições, sendo difícil satisfazer todas elas. Para problemas deste tipo se deseja minimizar o número de restrições violadas e ainda otimizar seus valores associados.

Com esse intuito, uma classe de problemas foi derivada dos CSPs e denominada *Constraint Optimization Problems* (COPs). Neste tipo de problema, as restrições não são consideradas proposicionais e valores são associados a cada restrição de acordo com os valores que podem assumir as variáveis relacionadas por esta. Uma função objetivo global está associada ao problema e o objetivo é maximizá-la ou minimizá-la. Esta função global utiliza os valores associados a cada restrição. Assim, cada valor atribuído às variáveis relacionadas por uma restrição gera um valor diferente para a restrição, que é determinado por esta, e que é utilizado pela função objetivo global. O método *Branch-and-Bound* (B&B) é o mais popular para a solução deste tipo de problema.

Abordagens utilizadas para solucionar CSPs também são utilizadas em COPs que, adicionalmente, oferecem recursos para que heurísticas de aproximação possam ser utilizadas com eficiência uma vez que se pode prever se o caminho que está sendo utilizado na busca por uma solução é ou não promissor pela aproximação da função objetivo.

Os DCOPs diferem dos COPs pois cada variável é associada a um agente e somente este tem controle sobre seu valor. As restrições são distribuídas entre os agentes que compõem o sistema e um processo de interação entre eles deve ser utilizado para se obter uma solução.

Formalmente um DCOP consiste de n variáveis $V = \{x_1, x_2, ..., x_n\}$ que podem assumir valores de um domínio discreto $D_1, D_2, ..., D_n$, respectivamente. Cada variável é associada a um agente que tem controle sobre seu valor. O objetivo do agente é escolher um valor para sua variável que otimize uma função objetivo global. Esta função é descrita como o somatório sobre o conjunto de restrições valoradas relacionadas a pares de variáveis. Assim, para um par de variáveis (x_i, x_j) existe uma função de custo definida como $f_{ij} : D_i \times D_j \rightarrow N$.

Uma solução para COPs baseada no algoritmo Branch-and-Bound (B&B) é uma das mais conhecidas soluções também para DCOP. Contudo, neste caso a comunicação entre os agentes precisa ser síncrona o que aumenta significativamente o tempo de execução do algoritmo. O principal problema dos métodos para a solução de DCOPs é que estes ou são soluções que precisam de sincronismo na comunicação, deixando o processo lento em demasia, ou não garantem a qualidade da solução, podendo não obter a solução ótima.

Um método assíncrono completo para a solução de DCOPs denominado Adopt foi proposto por MODI et al. (2003). Com isso, o Adopt é considerado um método eficiente para a solução de DCOPs (MAHESWARAN et al., 2004), garantindo a qualidade da solução e permitindo a comunicação assíncrona. A idéia principal do Adopt é fazer

com que os agentes troquem os valores das variáveis enquanto detectam que existe uma solução melhor que a que está sendo investigada por eles. Esta estratégia permite que o método seja assíncrono, pois os agentes podem tomar decisões apenas com informação local. Soluções parciais podem ser abandonadas e reconstruídas até que a solução ótima seja encontrada.

Uma outra abordagem completa para a solução de DCOPs, baseada na mediação cooperativa, foi proposta por MAILLER; LESSER (2004). A mediação cooperativa é uma técnica parcialmente centralizada de mediação que é a base para o algoritmo chamado *Optimal Asynchronous Partial Overlay* (OptAPO). O OptAPO é um algoritmo distribuído e completo para a solução de DCOPs. Os autores desta abordagem mostraram que o OptAPO tem um desempenho melhor que outros algoritmos para DCOP em problemas abstratos como o problema de coloração de grafos.

Uma questão importante sobre as abordagens para a solução de DCOPs é se estas têm desempenho suficiente para serem aplicadas em problemas reais. É importante identificar se o número e o tamanho das mensagens trocadas entre os agentes permitem a aplicação da abordagem na prática. Em abordagens distribuídas, a comunicação normalmente demanda bastante esforço e pode causar uma sobrecarga na rede. O tempo total consumido pelos agentes é aceitável para aplicações reais? Os problemas reais normalmente demandam planejamento e ação no menor tempo possível. A grande maioria das abordagens para DCOP apresentam bons resultados em cenários simples mas, como será discutido a seguir, seu desempenho em cenários mais complexos não tem sido satisfatório.

MAHESWARAN et al. (2004) mostraram como mapear problemas de escalonamento para DCOP utilizando uma plataforma re-utilizável denominada *Distribute Multi-Event Scheduling* (DiMES). Além de realizar este mapeamento, os autores testaram a eficiência do Adopt em alguns cenários reais mapeados para DCOP através do DiMES e duas heurísticas foram propostas para aumentar o desempenho do Adopt nestes cenários. Foi mostrado que o tempo de convergência do Adopt para cenários reais é cem vezes maior que o esperado, destacando a diferença considerável existente entre o desempenho do algoritmo em cenários abstratos simples em relação a cenários mais complexos e próximos da realidade. Segundo os autores, as heurísticas apresentadas reduziram o tempo de convergência aos valores esperados e habilitaram o Adopt a lidar com problemas complexos. Mesmo assim, o número de mensagens trocadas pelos agentes permaneceu bastante alto.

Uma vez que o OptAPO, como mencionado acima, mostrou um desempenho melhor que o Adopt no cenário de coloração de grafos, era de se esperar um comportamento similar no cenário de escalonamento. FERREIRA JÚNIOR; BAZZAN (2005) discutem que essa premissa não é verdadeira, apresentado resultados que mostram a ineficiência do OptAPO nos mesmos cenários utilizados por MAHESWARAN et al. (2004).

Mais recentemente, um algoritmo para DCOP denominado DPOP, baseado em programação dinâmica, foi proposto (PETCU; FALTINGS, 2005). Neste algoritmo, conjuntos com todas as soluções parciais possíveis para todas as variáveis são incrementalmente computados pelos agentes de acordo com uma ordem pré-determinada. Os autores argumentam que o número de mensagens trocadas entre os agentes, com estes conjuntos com todas as soluções possíveis, cresce linearmente em relação ao tamanho do problema. Contudo, o tamanho das mensagens trocadas pelos agentes executando o algoritmo DPOP cresce exponencialmente. O DPOP é muito superior ao Adopt quanto ao número de mensagens trocadas, mas estas mensagens são bem maiores que as utilizadas no Adopt. Além disso, o DPOP necessita de bem mais tempo para a computação de cada da solução entre cada mensagem enviada que o utilizado pelo Adopt. Resultados recentes mostraram que o

DPOP foi capaz de resolver adequadamente os problemas apresentados por MAHESWARAN et al. (2004).

Os algoritmos para DCOP mencionados até aqui nesta seção são amplamente discutidos e detalhadamente comparados por SANTOS (2007). Além disso, no referido trabalho, o autor analisa extensões para estes algoritmos propostas recentemente e outros algoritmos menos expressivos que não foram abordados nesta seção.

Como problemas complexos, se considerou até agora cenários de escalonamento que envolvem algumas dezenas de agentes e tarefas. Na seção 2.2 foi salientado que os sistemas multiagente têm sido aplicados em cenários de larga escala. Como visto anteriormente, a maioria dos algoritmos ótimos para DCOP não são eficientes em cenários complexos. O Adopt precisou de heurísticas adicionais para lidar com este tipo de problema enquanto que o OptAPO não foi minimamente capaz.

A maioria dos algoritmos para DCOP mencionados acima também possuem mecanismos para obter soluções aproximadas. É possível, por exemplo, definir um limite superior para o número de mensagens que os agentes trocam no Adopt, bem como é possível limitar o tamanho dessas mensagens no DPOP. Porém, estes limites não são suficientes para lidar com cenários complexos. Estes mecanismos usados pelos algoritmos de DCOP se tornam muito ineficientes a medida que a escala do problema aumenta. Visto que os algoritmos ótimos, nem tão pouco suas variações aproximadas, são capazes de lidar com os cenários em questão neste trabalho, abordagens heurísticas baseadas em outros mecanismos vêm sendo estudadas.

Neste sentido, SCERRI et al. (2005) propõem um algoritmo para DCOP especificamente para ambientes dinâmicos e em larga escala, denominado LA-DCOP (*Low-communication Aproximation* DCOP). O LA-DCOP utiliza um protocolo baseado em *tokens* para aumentar seu desempenho quanto à comunicação. Neste algoritmo os agentes podem perceber tarefas no ambiente ou receber *tokens* de outros agentes. Cada um dos *tokens* recebidos também contém tarefas, uma cada um. Os agentes decidem ou não alocar uma tarefa baseados em um limiar global, tentando maximizar o uso de seus recursos. Depois de decidir quais tarefas alocar, os agentes enviam *tokens* com tarefas não alocadas para outros agentes. O limiar mencionado representa a capacidade dos agentes para alocar cada tarefa e pode ser fixo ou dinamicamente calculado. *Tokens* adcionais, chamados "*tokens* potenciais", são utilizados pelos agentes para estabelecer compromissos com relação à alocação de algumas tarefas para lidar com a interrelação de alocação simultânea (AND) que podem existir entre elas.

Os autores do LA-DCOP argumentam que o seu algoritmo é melhor que abordagens anteriormente propostas, comparando-o diretamente com o DSA (ZHANG; WITTENBURG, 2002). O DSA utiliza uma estratégia baseada em *hill-climbing* para permitir que os agentes, com base nas informações de outros agentes próximos, aloquem as tarefas de modo a maximizar a solução do problema.

Agentes executando o LA-DCOP, ainda assim, enfrentam algumas limitações para lidar com os problemas de larga escala, as quais são:

- Os agentes escolhem alocar as tarefas que maximizam o somatório de suas capacidades, respeitando suas limitações de recursos. Este problema de maximização pode ser reduzido ao Problema da Mochila Binário (PMB), que é provado ser NP-Completo. A complexidade computacional do LA-DCOP depende da complexidade de sua função para resolver esse problema. Uma vez que o número de tarefas percebidas pelo agentes aumenta, o tempo consumido pelos agentes em sua deli-

beração aumenta exponencialmente. Cada agente resolve inúmeras instâncias do PMB durante um processo de alocação completa.

- Os agentes precisam trocar informações sobre tarefas e limiares para aumentar o desempenho do sistema. Se os canais de comunicação se tornarem limitados ou ineficientes, o desempenho do algoritmo cai significativamente.

- Os agentes não possuem mecanismos para adaptar seu comportamento para deliberar sobre tarefas desconhecidas, ou seja, em relação aquelas que estes não sabem sua capacidade para executá-las.

O cenário utilizado em SCERRI et al. (2005) para experimentar o LA-DCOP baseia-se na alocação de tarefas a agentes que têm competências diferentes para atuar em cada tarefa. Este cenário foi modelado como uma versão do GAP estendida pelos autores, denominada Extended GAP (E-GAP). O objetivo nesse problema é realizar alocações que maximizem o somatório da eficiência dos agentes em cada instante. A eficiência é a medida da competência do agente de realizar a tarefa que ele está realizando no momento. Centenas de tarefas aparecem e desaparecem com o passar do tempo e centenas de agentes devem realizá-las. Os resultados da abordagem proposta neste trabalho foram comparados com os resultados obtidos pelo LA-DCOP, ambos aplicados na solução do E-GAP.

2.5 Conclusão

Este capítulo apresentou os conceitos relacionados aos SMAs que são importantes para a compreensão deste trabalho. A abordagem proposta é utilizada no processo de decisão de agentes inteligentes de arquitetura reativa atuando em ambientes que são, segundo as definições apresentadas neste capítulo: acessíveis, determinísticos, dinâmicos, discretos e de larga escala. Tais agentes devem atuar em comunidade na busca de um objetivo comum. Neste trabalho, agentes inteligentes compõem um sistema multiagente heterogêneo. Este sistema tem suas ações coordenadas com base na sua estrutura organizacional, onde: a linguagem TÆMS é utilizada para descrever como o objetivo do sistema pode ser decomposto em tarefas; e a abordagem proposta permite a (auto-)organização dos agentes acerca de quando e quais destas tarefas executar.

Além disso, neste capítulo foram apresentados os trabalhos relacionados a este na área de sistemas multiagente. Diversas abordagens para a coordenação de SMAs, previamente propostas, foram discutidas. Em geral, tais abordagens apresentam muitas limitações para lidar com problemas gerais ou não podem ser aplicadas em problemas complexos. O algoritmo LA-DCOP, capaz de lidar com problemas dinâmicos de escala equivalente aos da abordagem proposta, foi mencionado. Este algoritimo, apesar de não tratar todas as características dos problemas que podem ser modelados em TÆMS, pode ser aplicado em problemas como o E-GAP, para os quais a abordagem proposta neste trabalho tem mecanismos de aplicação equivalente. Desta forma, a abordagem proposta pode ter sua eficiência comparada ao LA-DCOP quando aplicada no E-GAP, contribuindo para sua validação. Esta comparação será apresentada mais adiante neste trabalho.

3 INTELIGÊNCIA DE ENXAMES

Este capítulo apresenta a fundamentação teórica deste trabalho com relação a inteligência de enxames e está organizado da seguinte forma: a seção 3.1 apresenta um resumo sobre o conhecimento dos pesquisadores sobre a organização das colônias de insetos sociais do ponto de vista biológico e tem como objetivo mostrar as características destes seres que motivaram o desenvolvimento dos modelos teóricos discutidos em seguida; a seção 3.2 aborda os modelos teóricos para a organização dos insetos sociais que são a base da abordagem proposta neste trabalho; a seção 3.3 discute como a atividade de forragear das formigas deu origem a algoritmos aplicados a problemas de otimização; e a seção 3.4 traz os trabalhos relacionados a este na área de inteligência de enxames.

3.1 Organização das Sociedades de Insetos

As formigas, as abelhas e vespas (algumas), e os cupins são considerados insetos sociais. Estes insetos possuem uma das estratégias de sobrevivência mais bem sucedidas da natureza, por isso, há uma imensa quantidade e variedade destes (WILSON, 2000). A organização de suas sociedades se apresenta aos biólogos como um tópico instigante de estudo.

Como os insetos regulam a divisão de trabalho para adaptarem-se às mudanças no ambiente é uma questão que está sendo amplamente estudada. Modelos que descrevem os mecanismos que controlam o comportamento de cada indivíduo membro da colônia são os mais usualmente utilizados. Destes modelos, destaca-se o modelo de divisão do trabalho baseada em diferenças de limiares de resposta (ROBISON, 1992).

Insetos altamente sociais vivem em colônias com um grande número de indivíduos e diversos estímulos diferentes podem ser associados às diferentes tarefas que podem ser executadas por eles. As atividades que os indivíduos executam durante um determinado momento estão ligadas à quantidade de estímulo associado a esta tarefa específica e a sensibilidade a este estímulo é associada a casta específica que o indivíduo pertence. Diferentes respostas a estímulo foram observadas em diferentes castas físicas em formigas e cupins. Diferenças nos limiares de resposta de acordo com a idade foram observadas em abelhas, as abelhas mais velhas tendem a ter um comportamento mais defensivo do que trabalhadoras mais jovens. A probabilidade de um indivíduo executar uma tarefa está relacionada com dois fatores:

- Magnitude do estímulo associado à tarefa;
- Limiar de execução da tarefa pelo indivíduo.

A flexibilidade na execução de tarefas é uma conseqüência dos fatores ambientais atuando no indivíduo. Os limiares de resposta a uma determinada tarefa podem mudar de acordo com o tempo que o indivíduo executa a tarefa ou de acordo com as mudanças no ambiente. Em abelhas, o *juvenile hormone* (JH) tem sido considerado um dos fatores para as mudanças nos limiares de execução de tarefas específicas.

De acordo com WILSON (2000), assim que uma espécie chega até o nível social a organização da colônia pode avançar em dois sentidos: no aumento do grau de especialização das castas operárias, de forma física e/ou temporal (o polimorfismo e o polietismo temporal, respectivamente) e no alargamento do código de comunicação por onde os membros da colônia coordenam suas atividades. Ambos serão discutidos nas seções a seguir.

3.1.1 Especialização das Castas Operárias

As castas, nos insetos sociais, podem ser de dois tipos essencialmente: físicas, determinadas pelo polietismo morfológico ou, simplesmente, polimorfismo; e temporais, determinadas pelo polietismo temporal.

As formigas fêmeas se dividem em três castas físicas: operárias, soldados e rainha. Já os machos não se dividem em castas, mas podem, num sentido mais abstrato, serem considerados como uma quarta casta. Em algumas espécies de formigas existe uma clara definição dessas três castas de fêmeas (GORDON, 1996). As formigas soldado normalmente são as maiores, possuindo uma cabeça grande com mandíbulas em forma de espada.

Em alguns casos não existe essa definição clara, as mudanças fisiológicas e de comportamento ocorrem com a idade do indivíduo. Essas mudanças representam saltos de uma casta temporal para outra. De acordo com GORDON (1996), operárias jovens primeiro desempenham tarefas dentro da toca, como armazenar alimento e cuidar da prole, enquanto operárias mais velhas desempenham tarefas externas, como forragear. As formigas que cuidam da manutenção do formigueiro estão em transição entre o interior e o exterior, porque esse trabalho é feito em parte dentro do formigueiro, quando as formigas amontoam areia proveniente de túneis escavados, e em parte no exterior, quando a areia é levada para fora da boca do formigueiro. Até as larvas podem ser consideradas uma sub-casta temporal. Em muitas espécies as larvas possuem uma glândula capaz de liberar um material líquido rico em nutrientes que é importante para a alimentação da colônia.

Com abelhas e vespas, a divisão de castas é definida totalmente pelo polietismo temporal. Isto ocorre, segundo WILSON (2000), por que abelhas e vespas tomaram um caminho evolucionário diferente, por isso não há diferenças físicas entre vespas e abelhas, exceto entre a rainha e as operárias. A principal diferença morfológica existente nessas espécies é o tamanho das operárias. Operárias grandes tendem a forragear enquanto as pequenas tendem a trabalhar dentro da colméia. Dentro dessas espécies, a rainha pode ter um comportamento passivo (máquina de botar ovos) ou um comportamento agressivo com suas irmãs operárias.

3.1.2 Comunicação

Muitas formigas operárias buscam presas ou itens de comida maiores e mais pesadas que a capacidade de um único indivíduo. Estas formigas utilizam o transporte cooperativo para lidar com grandes cargas. Uma operária pode recrutar suas colegas de colônia, através da comunicação, se não for capaz de carregar determinados objetos (KUBE; BONABEAU, 2000). Colegas de ninho na vizinhança são atraídas por este chamado simples

e rapidamente se juntam aos esforços de quem pediu ajuda em um ato de cooperação.

As formas de comunicação entre os insetos sociais são variadas, incluindo pequenas pancadas, sons, toques de antenas, liberação de substâncias químicas, entre muitos outros. Devido a grande variedade de formas de comunicação torna-se muito complicado estudar todos os detalhes das interações entre os indivíduos de uma colônia, por isso, a comunicação dos insetos sociais respeita três generalizações:

- A maior parte da comunicação é química, através de feromônios. O uso de sinais visuais é esparso e muito simples. Em alguns grupos, particularmente os cupins e formigas subterrâneas, esta forma de comunicação não possui nenhum papel no dia-a-dia da colônia. Os sons também são pouco identificáveis pelos insetos sociais e não são definidos em nenhum sistema de comunicação importante. Os toques são amplamente utilizados nas colônias de insetos, mas também não foi identificada nenhuma espécie de código capaz de carregar grandes quantidades de informação. Por outro lado, os sinais químicos se apresentam em praticamente todas as categorias de comunicação, principalmente através de feromônios, capazes de estimular ou inibir diversas características fisiológicas nos membros de uma colônia.

- Qualquer forma de comunicação sempre encontra paralelo em espécies pré-sociais. Por exemplo, as hierarquias de dominância, que possuem papel chave nas sociedades de abelhas e vespas, possuem um precedente no comportamento territorial de muitas espécies solitárias. Outro exemplo são as substâncias de alarme que, em muitos casos, são simples modificações de secreções defensivas.

- O comportamento complexo da colônia emerge da integração de simples padrões individuais através da comunicação. WILSON (2000) apresenta nove categorias de resposta nos insetos sociais. Entre elas: alarme; atração simples ou múltipla; recrutamento (para um novo ponto de coleta de comida ou um ponto específico do ninho); assistência; determinação de casta, por inibição ou estimulação; etc.

O mecanismo de transporte cooperativo das formigas, comentado no início dessa seção, será utilizado pela abordagem proposta nesse trabalho para lidar com a interrelação entre as tarefas, como será visto mais adiante. Os agentes utilizam mensagens simples para compartilhar a necessidade de uma tarefa ser alocada em conjunto com outros agentes, por exemplo.

3.2 Modelos Teóricos para Divisão de Trabalho

Modelos teóricos e matemáticos sobre a organização das colônias de insetos sociais foram concebidos baseados nos estudos e observações realizadas pelos entomologistas. Tais modelos foram construídos em sua maioria com o objetivo de simular o funcionamento das colônias para tentar compreender sua organização. Em (GORDON, 2002, capítulo 8), são apresentados alguns destes modelos que, em sua maioria, consideram a alocação de tarefas orientada pelo padrão de interações que cada indivíduo experimenta.

Os modelos apresentados até então produzem resultados que se assemelham ao comportamento observado de colônias reais, mas não demonstram o modo como os insetos operam. Assim, estes modelos ainda não puderam ser utilizados para o objetivo a que foram concebidos, pois não se sabe se os detalhes empíricos modelados são suficientes.

O modelo apresentado em (BONABEAU; THERAULAZ; DORIGO, 1999, capítulo 3)[1], já por pesquisadores da computação, pretende ser um modelo genérico que cobre todos os aspectos que envolvem a divisão do trabalho nas colônias. Neste modelo cada indivíduo da colônia tem um limiar de resposta a estímulos para realizar determinada tarefa. Um indivíduo passa a executar uma tarefa quando o estímulo para executar esta tarefa ultrapassa seu limiar associado.

Seja s a intensidade de um estímulo associado a uma atividade em particular, onde s pode ser o número de encontros com outros indivíduos, uma concentração química ou qualquer outro fator quantitativo que possa ser sentido por um indivíduo. O limiar de resposta θ, expresso em unidades de intensidade de estímulo, é uma variável interna que determina a tendência de um indivíduo, respondendo ao estímulo s, realizar a tarefa associada.

A equação 3.1 mostra uma possível função para a probabilidade de um indivíduo atender a resposta de um estímulo. Outras funções podem levar ao mesmo padrão de resultados esperado pelo modelo. Qualquer número inteiro n maior que 1 pode ser usado com expoente de s, que determina o índice de crescimento de θ, porém o autor utiliza em todos seus exemplos $n = 2$.

$$T_\theta(s) = \frac{s^2}{s^2 + \theta^2} \tag{3.1}$$

Desta forma, se o valor de θ for muito pequeno, a probabilidade do indivíduo atender ao estímulo tende a 1. Se o valor de θ for muito grande tal probabilidade irá tender a 0. Com $s = \theta$ a probabilidade é exatamente $1/2$.

A equação 3.2 traz outro exemplo de função para a probabilidade de um indivíduo atender a resposta a um estímulo.

$$T_\theta(s) = 1 - e^{\frac{-s}{\theta}} \tag{3.2}$$

O comportamento das funções apresentadas acima é equivalente, variando apenas a curva de resposta aos estímulos.

O limiar de resposta a um estímulo, valor de θ nas funções acima, pode variar para cada indivíduo para refletir o polimorfismo e o polietismo temporal dos insetos sociais. Indivíduos fisicamente diferentes ou de diferentes idades podem ter tendências diferentes de executar determinadas tarefas.

No modelo apresentado em BONABEAU; THERAULAZ; DORIGO (1999, capítulo 3), um indivíduo torna-se mais sensitivo ao estímulo associado a uma tarefa em que está engajado e menos sensitivo aos estímulos associados às demais tarefas. Cada indivíduo no modelo tem um limiar de resposta para cada tarefa. Este limiar é atualizado (aumentando ou diminuindo) de acordo com dois coeficientes diferentes (ξ e ρ).

O limiar de resposta θ_{ij} de um indivíduo i quando executando a tarefa j durante o intervalo de tempo Δt é dado pela equação 3.3.

$$\theta_{ij} = \theta_{ij} - \xi \Delta t_{ij} \tag{3.3}$$

onde:

ξ coeficiente de aprendizado

[1]THERAULAZ; BONABEAU; DENEUBOURG (1998) apresentam outros modelos nesta linha e são discutidas as diferenças entre tais modelos e o modelo do autor.

O limiar de resposta θ_{ij} de um indivíduo i quando não executando a tarefa j durante o intervalo de tempo Δt é dado pela equação 3.4.

$$\theta_{ij} = \theta_{ij} + \rho \Delta t_{ij} \tag{3.4}$$

onde

ρ coeficiente de esquecimento

O estímulo s associado a uma tarefa deve variar conforme sua execução é satisfatória ou não. Uma tarefa pode ser executada de forma ineficiente, seja por um ou mais indivíduos, o que deve aumentar seu estímulo associado para fazer com que outros indivíduos se engajem na sua execução.

A variação da intensidade de um estímulo associado à execução de uma tarefa, que reduz a intensidade do estímulo, e associado ao crescimento natural da demanda é dada pela seguinte equação.

$$s(t+1) = s(t) + \phi - \frac{\psi N_{act}}{N} \tag{3.5}$$

onde:

$s(t+1)$ estímulo associado a tarefa j no tempo $t+1$.

$s(t)$ estímulo associado a tarefa j no tempo t.

ϕ crescimento da intensidade em função do tempo.

ψ fator de escala associado à eficiência da realização da tarefa.

N_{act} número de indivíduos ativos na colônia.

N número de indivíduos que podem ser tornar ativos na colônia.

Todo indivíduo tem a mesma probabilidade fixa de deixar de executar uma tarefa e pode retomar sua execução imediatamente se o estímulo que este indivíduo tem para executar a tarefa que acabou de abandonar for superior ao seu limiar.

Os modelos apresentados nesta seção serviram de base para a concepção de mecanismos propostos nesse trabalho.

3.3 Otimização Baseada em Colônias de Formigas

Outras características dos insetos sociais também já serviram de inspiração para a concepção de modelos teóricos que estão sendo aplicados em problemas computacionais. Modelos baseados na maneira como a formigas forrageiam deram origem a uma série de algoritmos para a solução de problemas de otimização combinatoria.

Atualmente há uma série de pesquisas sendo realizadas nessa linha, o que levou ao surgimento de uma nova área de pesquisa intitulada *Ant Colony Optimization* (ACO) (DORIGO, 1992). Abordagens de ACO para problemas como a escolha de rotas para veículos, ordenação seqüencial, coloração de grafos, roteamento em redes de comunicação, entre outros, têm sido aplicadas com sucesso.

As abordagens de ACO, como dito acima, foram baseadas na maneira como as formigas forrageiam, mais especificamente, na forma com que elas determinam o menor

caminho entre a fonte de alimento e seu ninho. Enquanto se deslocam do ninho até a fonte de alimento encontrada, e de volta para o ninho, as formigas secretam uma substância química chamada feromônio. Esta substância, considerada como uma forma primitiva de comunicação, é percebida por outras formigas que escolhem seu caminho probabilisticamente de acordo com a concentração de feromônio em cada caminho possível. Uma colônia de formigas seguindo esta trilha química faz com que emerja o caminho mais curto entre o ninho e a fonte de alimento.

O modelo probabilístico que descreve este comportamento será apresentado a seguir. Neste modelo a probabilidade de escolher um caminho específico em um determinado momento é proporcional ao número de formigas que escolheu aquele caminho até tal momento. Assim, considerando que dois caminhos são possíveis até a fonte de alimento e que m formigas ($m = L_m + U_m$) se deslocaram por estes caminhos, o número de formigas que escolheu o primeiro é dado por U_m e o número de formigas que o escolheu o segundo é dado por L_m. A probabilidade $P_U(m)$ da $(m+1)$-ésima formiga escolher o primeiro caminho é dada pela equação 3.6, onde k e h são constantes utilizadas para adequar o modelo a observações reais.

$$P_U(m) = \frac{(U_m + k)^h}{(U_m + k)^h + (L_m + k)^h} \tag{3.6}$$

A probabilidade $P_L(m)$ que a mesma formiga escolha o segundo caminho é dada por $P_L(m) = 1 - P_U(m)$.

Quando um dos caminhos é mais curto a quantidade de formigas que irá selecioná-lo é maior. Isto ocorre porque as primeiras formigas a chegarem na fonte de alimento são as que escolheram o caminho mais curto e, por conseqüência, estas formigas voltam mais cedo para o ninho. Com isso, haverá mais feromônio depositado no caminho mais curto, fazendo com que outras formigas tendam a seguir este caminho.

Estas equações foram obtidas através de experimentos com formigas reais em cenários controlados. Utilizando o modelo descrito acima, as abordagens de ACO oferecem metaheurísticas onde formigas artificiais trabalham cooperativamente para encontrar boas soluções para problemas complexos. As formigas reais serviram de base para tais formigas artificiais. Contudo, foram dadas habilidades especiais para estas últimas para torná-las ainda mais eficientes que suas irmãs reais.

Em DORIGO; DI CARO; GAMBARDELLA (1999) pode-se encontrar mais detalhes sobre ACO, tanto em relação às diferenças entre formigas reais e virtuais, quanto em relação ao modelo discutido acima. Além disso, os autores apresentam uma visão geral da aplicação de ACO em problemas de otimização, como os descritos no início desta seção, e mostram o quão poderosa e eficiente pode ser esta abordagem se comparada com outras anteriormente propostas. Na seção 3.4 serão discutidas abordagens de ACO que têm relação com este trabalho.

3.4 Trabalhos Relacionados

A seguir serão discutidos os trabalhos relacionados a este que foram realizados na área de inteligência coletiva. Primeiramente, na seção 3.4.1, serão discutidas as abordagens que utilizam o modelo de divisão de trabalho nas colônias de insetos sociais, modelos estes que também serviram como base para a abordagem proposta neste trabalho. Em seguida, na seção 3.4.2, serão discutidas abordagens que utilizam a comunicação através dos feromônios para otimização. Tais abordagens estão relacionadas a este trabalho pois

são utilizadas para resolver problemas em cenários próximos aos que são considerados para a experimentação e validação da abordagem proposta.

3.4.1 Divisão de Trabalho e Alocação de Tarefas

CAMPOS et al. (2000) propuseram a utilização dos modelos teóricos de divisão de trabalho discutidos neste capítulo para a coordenação de agentes que atuam em ambientes dinâmicos. Neste artigo os autores apresentam uma abordagem que é aplicada ao *Dynamic Flow Shop Scheduling Problem*.

Neste cenário, máquinas capazes de realizar diferentes tipos de tarefas são representadas por agentes que se comportam como insetos sociais. Novas tarefas chegam em uma linha de produção e existe um custo de configuração das máquinas quando estas trocam de um tipo de tarefa para outro. Os agentes, representando as máquinas, decidem se vão ou não realizar cada tarefa. O objetivo da abordagem é minimizar o tempo total de realização das tarefas especializando as máquinas por tipo de tarefa.

Experimentos foram realizados com 20 tipos de tarefas e utilizando de 6 a 15 máquinas. Cada máquina possui uma fila de 5 tarefas que podem ficar aguardando para serem realizadas. Uma vez que a tarefa é escolhida pelo agente ela entra nesta fila. A cada passo do experimento uma ou mais máquinas podem falhar. A dinamicidade do cenário está exatamente nestas falhas. As tarefas aguardando na fila de uma máquina que falha não são colocadas de volta na linha de produção.

Os resultados obtidos mostram que a abordagem baseada em insetos sociais tem desempenho equivalente a uma abordagem *market-based*, usada no artigo para comparação. Ambas abordagens tiveram desempenho melhor que abordagens que não utilizam agentes. Os autores detalham as diferenças e semelhanças entre as duas abordagens e argumentam que modelos baseados em insetos sociais, que se mostraram poderosos através de ACO para otimização estática e centralizada, podem ser eficientes também para otimização em cenários dinâmicos e distribuídos. No artigo a alocação de tarefas é vista como um problema de escalonamento que é continuamente resolvido pelos agentes em um ambiente variável.

CICIRELLO; SMITH (2004) apresentam uma extensão da abordagem para o mesmo problema de CAMPOS et al. (2000), agora intitulado (*Dynamic Shop Floor Routing problem*). Nesta nova abordagem são feitas algumas modificações nas equações apresentadas por CAMPOS et al. (2000) que melhoram seu desempenho. Além disso é proposto um modelo de competição por dominância, onde agentes que decidem realizar a mesma tarefa disputam pelo direito de realizá-la.

Os aspectos dinâmicos deste cenário são os seguintes: diferentes tipos de tarefas são enviadas simultaneamente para o chão de fábrica; as taxas em que estas tarefas chegam são determinadas pela probabilidade de uma tarefa de um tipo diferente da anterior aparecer. Inicialmente, três casos com diferentes probabilidades para dois tipos de tarefas foram analisados: 50-50, 75-25 e 100-0. A abordagem apresentada foi capaz de especializar as máquinas a estas probabilidades. Mais tarde, uma distribuição variável foi utilizada para simular a dinamicidade do ambiente de uma maneira controlada: a distribuição inicial é alterada depois da primeira metade do experimento.

A adaptação com relação à troca da distribuição necessitou de uma quantidade de tempo considerável. Os autores argumentam que este tempo de adaptação não prejudica o desempenho da abordagem. Apesar desta limitação e do aspecto controlado da dinamicidade do ambiente, os resultados mostraram que a utilização dos modelos teóricos dos insetos sociais é competitivo, e em alguns casos até superior a outros sistemas basea-

dos em agentes apresentados anteriormente (inclusive o de CAMPOS et al. (2000), cujo modelo foi estendido).

Contudo, existem questões em aberto nas referidas abordagens quanto a especificidade das soluções propostas. Quando aplicado a cenários mais complexos, onde mais de uma máquina pode ser requisitada para realizar a mesma tarefa, a simples especialização não é suficiente. Os mecanismos utilizados nestas abordagens foram desenvolvidos para tratar problemas específicos, como os discutidos aqui. De maneira mais geral, acredita-se que estes não são capazes de lidar com outros problemas de escalonamento e alocação de tarefas como os abordados neste trabalho.

3.4.2 Formigas Forrageando e Otimização

MERKLE; MIDDENDORF; SCHMECK (2002) apresentam uma abordagem para o *Resource Constraint Project Scheduling Problem* - RCPSP que, segundo os autores, é um problema geral de escalonamento que engloba problemas de *job-shop*, *flow-shop* e *open-shop*. Por este cenário ter sido largamente estudado existem disponíveis comparações de desempenho entre os mais diversos algoritmos e conjuntos de teste padrão que são usados nestas comparações. Já foram propostos algoritmos para RCPSP baseados em algoritmos genéticos, *simulated annealing*, *tabu search*, entre outros.

Muitas abordagens baseadas em ACO têm sido propostas para problemas de escalonamento que são especializações do RCPSP. Nestas abordagens, muitas gerações de formigas artificiais buscam por uma boa solução. Cada formiga de uma geração constrói uma solução tomando decisões probabilísticas. Em geral, formigas que encontram uma boa solução marcam com feromônios o caminho que as levou até essa solução. As formigas das próximas gerações são atraídas pelos feromônios e tendem a buscar uma solução próxima à solução encontrada pela geração anterior. Além dos feromônios, normalmente estas abordagens adicionam alguma heurística, especificamente relacionada com o problema, ao processo de tomada de decisão.

A abordagem proposta por MERKLE; MIDDENDORF; SCHMECK (2002) difere das outras até então propostas por várias razões. Primeiramente, por combinar duas formas diferentes, anteriormente propostas, de considerar os feromônios para determinar as probabilidades de escalonamento de cada tarefa. Numa delas, denominada direta, os feromônios depositados em determinada unidade de tempo possível para o escalonamento de cada tarefa são considerados diretamente para determinar a probabilidade da tarefa ser escalonada nesta posição. Na outra, denominada somatório, os feromônios de cada posição possível no escalonamento, até a posição que se está considerando, são somadas para compor a probabilidade.

Além disso, os autores propõe que a influência da heurística relacionada com o problema diminua com o passar do tempo, ou seja, as primeiras gerações são mais influenciadas pela heurística que as últimas. Esta modificação faz com que a heurística determinística não impeça que as formigas busquem melhores soluções de geração para geração, mas as orienta no início do processo.

Outra modificação importante proposta por MERKLE; MIDDENDORF; SCHMECK (2002) diz respeito à influência da formiga que encontra a melhor solução e é autorizada a atualizar os feromônios, chamada de elite. Em outras abordagens esta formiga mantém a melhor solução encontrada e atualiza os feromônios de acordo com essa solução em cada geração. Na abordagem proposta esta ação é limitada, permitindo à formiga elite atualizar os feromônios depois de um certo número de gerações.

No referido trabalho foram realizados experimentos comparando esta abordagem com

outras baseadas em diferentes técnicas. Quando foi estabelecido um número máximo de soluções que as abordagens puderam computar, a abordagem em questão obteve resultados equivalentes aos até então conhecidos. Quando esta limitação no número de soluções não foi considerada, esta abordagem encontrou soluções mais eficientes que as até então conhecidas para 30% dos problemas utilizados nos experimentos.

O RCPSP será utilizado como um dos cenários para experimentar a abordagem que está sendo proposta neste trabalho.

Como comentado na seção 2.4.2, alguns problemas de escalonamento e alocação de tarefas também podem ser descritos como Problemas de Satisfação de Restrições - CSPs. Os CSPs são caracterizados por conterem um conjunto de variáveis, cada uma possuindo um domínio específico (conjunto discreto de valores diferentes que a variável pode assumir) e um conjunto de restrições com relação a distribuição dos valores dos domínios entre as variáveis. O problema é considerado resolvido quando o menor número possível de restrições é desrespeitado. Este tipo de problema pode ser mapeado como um grafo onde os nodos representam as variáveis associadas com cada um de seus valores possíveis e os vértices representam as restrições existentes com relação aos pares variável/valor representados pelos nodos interligados pelo vértice em questão.

TARRANT; BRIDGE (2004) propõem a aplicação de ACO em CSPs, apresentando um algoritmo e discutindo as diversas abordagens para se obter a solução do referido problema usando este algoritmo. Para seu funcionamento, uma quantidade inicial de feromônios é distribuída pelo grafo. Em seguida, formigas artificiais são aleatoriamente distribuídas nos nodos do grafo e se deslocam através dos nodos formando uma atribuição completa, isto é, percorrendo um nodo relacionado com cada variável. Ao final deste percurso são distribuídos feromônios de acordo com a qualidade da solução. Simultaneamente, os feromônios depositados evaporam segundo uma taxa pré definida. O algoritmo termina quando uma solução para o problema é encontrada ou quando um determinado número de percursos foi realizado.

O objetivo dos autores foi determinar qual destas abordagens seria a mais eficiente para, no futuro, comparar o algoritmo baseado em insetos sociais com outros já apresentados. Assim, não se tem conhecimento ainda da diferença de desempenho entre ACO e as outras soluções propostas para CSPs, bem como não se sabe a diferença desta solução especificamente se comparada a outras também baseadas em ACO para problemas que possam ser representados como CSPs.

3.5 Conclusão

Este capítulo apresentou a fundamentação deste trabalho na área de inteligência de enxames. A abordagem proposta tem como base os modelos teóricos para divisão de trabalho dos insetos sociais apresentados aqui. Estes modelos foram modificados e estendidos para suportar as características dos problemas que a abordagem proposta pretende tratar. O processo de recrutamento para o tranporte cooperativo dos insetos sociais, detalhado também neste capítulo, inspirou parte da abordagem proposta no que diz respeito à coordenação dos agentes na execução de tarefas interrelacionadas.

Além disso, neste capítulo foram apresentados os trabalhos relacionados a este na área de inteligência de enxames. As abordagens para a alocação distribuída de tarefas que têm como base o modelo teórico de divisão do trabalho que embasou a abordagem proposta foram discutidos. Como mostrado, tais abordagens foram desenvolvidas para tratar problemas específicos e não são capazes de lidar com problemas gerais, como os

que podem ser decritos em TÆMS. Neste capítulo também foram mencionadas algumas abordagens para a solução de alguns problemas de otimização. Uma destas abordagems lida com o RCPSP. Como dito anteriormente, este problema será estendido neste trabalho para contribuir com a validação da abordagem proposta.

4 ABORDAGEM PROPOSTA

Este capítulo apresenta a abordagem proposta nesse trabalho e está organizado da seguinte forma: na seção 4.1 a abordagem proposta é contextualizada; nas seções 4.2, 4.3, 4.4, 4.5 e 4.6, são discutidos os mecanismos que compõem esta abordagem; e, finalmente, na seção 4.7 são introduzidos os problemas que servirão de cenários para a validação da abordagem proposta.

4.1 Idéias Básicas

Como mencionado no capítulo 1, este trabalho enfoca o problema da alocação e (ou) escalonamento distribuído de tarefas em cenários complexos (dinâmicos, parcialmente observáveis e de larga escala).

Neste problema, tarefas devem ser alocadas e (ou) escalonadas por agentes que têm competências diferentes para realizar cada tarefa. Tais tarefas consomem recursos, que os agentes possuem em quantidade limitada, e aparecem e desaparecem dinamicamente.

Para a discussão dos mecanismos da abordagem proposta, feita neste capítulo, considera-se que uma alocação é equivalente a um escalonamento onde os agentes têm o horizonte de tempo limitado a uma, e somente uma, unidade de tempo (a duração das tarefas respeitam essa limitação). Com isso, será utilizado o conceito de escalonamento de maneira ampla, referindo-se também à alocação.

O controle distribuído do processo de escalonamento implica na tomada de decisão dos agentes de modo independente, com estes tendo apenas uma visão local sobre as tarefas percebidas e sobre o desempenho obtido depois da tarefa escalonada.

Como larga escala entende-se que mais de uma centena de agentes estão envolvidos na realização de mais de uma centena de tarefas. Com isso, os agentes precisam deliberar o mais rápido possível para que o SMA possa atingir seu objetivo em um tempo aceitável.

Algumas tarefas podem interferir na execução de outras de muitas maneiras diferentes, sendo interdependentes, mutuamente exclusivas, etc.

De maneira geral, o objetivo do SMA é minimizar o intervalo de tempo necessário no escalonamento dessas tarefas, utilizando os agentes mais capacitados no escalonamento de cada uma delas.

O embasamento da abordagem proposta é a plasticidade na divisão do trabalho nas colônias dos insetos sociais. Como discutido no capítulo 3, esta plasticidade emerge da interação entre os indivíduos e deles com o ambiente, utilizando padrões simples de comunicação e sem coordenação explícita. Centenas, e até milhares, de indivíduos compõe uma colônia que se auto-organiza para atender suas necessidades em um ambiente dinâmico.

A alocação de tarefas entre os indivíduos que compõem uma colônia deve ser tal

que as variações do ambiente sejam absorvidas e que a energia gasta pela colônia seja a menor possível. Além disso, os indivíduos trocam de tarefa para satisfazer as demandas da colônia em relação às mudanças no ambiente e buscam desempenhar as tarefas para as quais têm maior competência, usualmente determinada por sua forma física.

Agentes podem decidir quais tarefas se engajar e quando fazer isso inspirados na forma como os insetos sociais tomam este tipo de decisão, sem utilizar processos explícitos de coordenação e buscando a maior eficiência possível. Este comportamento é adequado para agentes que compõem um SMA.

Para representar as tarefas necessárias para que o SMA atinja seu objetivo, bem como a especificação da organização hierárquica destas tarefas e sua interdependência, a abordagem proposta utiliza a linguagem TÆMS. Esta linguagem, explicada em detalhes no capítulo 2, permite especificar de uma forma bastante rica essa estrutura de tarefas.

Deve-se considerar que os agentes atuam em ambientes dinâmicos, onde a estrutura de tarefas descrita em TÆMS pode ser modificada em tempo de execução: métodos podem aparecer e desaparecer; a relação entre as tarefas pode mudar; e podem variar as dependências entre os métodos.

Contudo, a estrutura de tarefas pode mudar apenas quando o objetivo desta estrutura é atingido. Esta restrição é utilizada para que sempre se obtenha um escalonamento completo da estrutura de tarefas, possibilitando a adaptação dos agentes em função dos resultados obtidos.

Cabe ressaltar que os agentes em TÆMS escalonam de fato os métodos (folhas da árvore) e que as tarefas representadas na estrutura de tarefas (TS) são apenas a organização hierárquica desses métodos para compor o objetivo do agente. Durante este capítulo a palavra "tarefa" é usada inúmeras vezes para fazer referência ao que em TÆMS, especificamente, é denominado "método".

Nas seções a seguir serão apresentados, formalizados e demonstrados cada um dos mecanismos propostos que podem ser usados pelos agentes para satisfazer o objetivo de um SMA atuando conforme discutido aqui.

4.2 Tendência

Nesta abordagem os agentes decidem de forma probabilística quais tarefas escalonar tendo como base o modelo teórico do limiar de resposta apresentado na seção 3.2. As tarefas produzem um estímulo para os agentes que, dados os seus limiares de resposta interno, têm diferentes tendências para escalonar cada uma delas. A medida que as tarefas são percebidas pelos agentes, estes devem analisá-las e decidir quais escalonar utilizando essa tendência. Isto é feito através de um sistema de roleta, tendo esta tendência como a probabilidade que determina se o agente escalonará ou não a tarefa em análise.

A tendência $T_{\theta_{ij}}(s_{ij})$ do agente i se engajar no escalonamento da tarefa j é determinada pela relação entre o estímulo dessa tarefa e o limiar interno do agente, como mostra a equação 4.1. Como está sendo considerado um processo distribuído, o estímulo das tarefas são específicos do agente, podendo uma mesma tarefa estimular de maneira diferente cada um dos agentes.

$$T_{\theta_{ij}}(s_{ij}) = \frac{s_{ij}^2}{s_{ij}^2 + \theta_{ij}^2} \tag{4.1}$$

onde:

s_{ij} estímulo associado com a tarefa j para o agente i

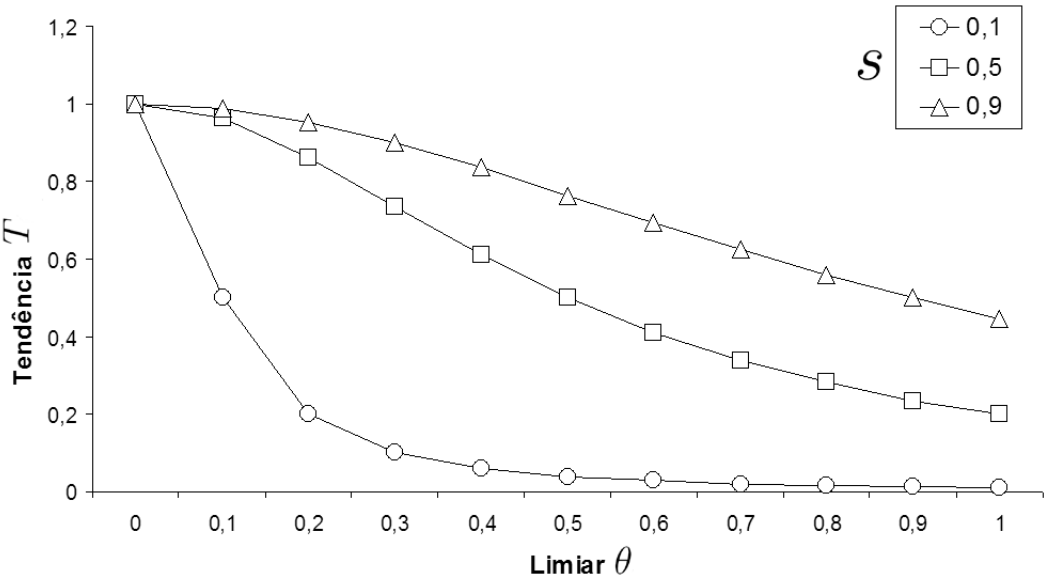


Figura 4.1: Exemplo de valores para a tendência $T_{\theta_{ij}}(s_{ij})$ para três diferentes valores para o estímulo s_{ij} (0.1, 0.5 e 0.9), considerando a variação do limiar interno θ_{ij}

θ_{ij} limiar interno do agente i em relação à tarefa j

Considerando que os valores para o estímulo e o limiar interno dos agentes encontram-se no intervalo $[0, 1]$, os valores para a tendência também variam dentro desse mesmo intervalo. Cabe ressaltar que, considerando a equação apresentada acima, o agente com maior competência para escalonar determinada tarefa é aquele que tem um menor limiar interno para esta tarefa.

A figura 4.1 mostra o comportamento da tendência $T_{\theta_{ij}}(s_{ij})$ para três diferentes valores para o estímulo s_{ij} (0.1, 0.5 e 0.9), considerando a variação do limiar interno θ_{ij}.

Pode-se verificar que, quando o indivíduo tem um alto estímulo para uma determinada tarefa (por exemplo, 0.9), a tendência do indivíduo para se engajar nesta tarefa é sempre maior que 0.5, mesmo quando sua competência para escaloná-la é baixa. Desta forma, uma tarefa com alto estímulo tem uma chance muito maior de ser escolhida, mesmo quando os agentes são pouco competentes para lidar com ela.

No outro caso, quando o estímulo assume um valor intermediário (por exemplo, 0.5), conforme o limiar interno aumenta, a tendência diminui linearmente até assumir um valor próximo a 0.3.

Já no último caso, a tendência é alta apenas enquanto o limiar interno assume valores menores que 0.1 e diminui drasticamente quando o limiar interno atinge valores mais altos. A tendência diminui exponencialmente em relação ao aumento do limiar interno.

4.3 Tomada de Decisão

Como os agentes têm uma quantidade limitada de recursos para empregar em cada escalonamento, o processo de tomada de decisão é fortemente influenciado pela ordem em que as tarefas são analisadas. Os agentes tomam suas decisões para todas as tarefas, analisando-as uma a uma, e despendendo seus recursos com cada uma das que são escolhidas.

Esse processo pode se repetir até que todas as tarefas tenham sido escolhidas ou quando os recursos do agente estiverem acabado. Quando este último ocorre, as tarefas que ainda não foram analisadas ou escolhidas serão desconsideradas. Os agentes compõem o escalonamento das tarefas a medida que cada tarefa é escolhida, optando sempre por acomodá-la o mais cedo possível em sua linha de tempo ou respeitando algum tempo de início específico para essa tarefa, como será discutido mais adiante nesta seção.

Portanto, a ordem das tarefas tem um papel fundamental na tomada de decisão. As tarefas são percebidas pelos agentes organizadas em uma estrutura descrita em TÆMS. Todas as tarefas percebidas devem ser consideradas em uma ordem que respeite as restrições impostas pelos interrelacionamentos (QAFs em TÆMS) e a interdependência (NLEs em TÆMS) entre as tarefas.

As QAFs dão significado aos interrelacionamentos entre as tarefas filhas de uma supertarefa em TÆMS, como discutido na seção 2.3. A execução de uma tarefa pode demandar que outra tarefa também seja executada simultaneamente, pois só assim a supertarefa que abriga as duas será realizada por completo. Ou ainda, uma tarefa que está sendo executada inibe outras tarefas da mesma supertarefa, pois esta só precisa que uma de suas subtarefas seja executada. A tabela 2.1 mostra todas as possibilidades de QAFs.

O mesmo tipo de interferência entre tarefas comentado acima pode ocorrer entre aquelas que não pertencem a mesma supertarefa. Os NLEs, também previamente discutidos na seção 2.3, representam a interdependência no escalonamento entre as tarefas que não são filhas imediatas da mesma supertarefa. Todas as possibilidades para os NLEs são mostradas na tabela 2.2.

A seguir serão discutidos os mecanismos da abordagem proposta relacionados as QAFs e aos NLEs, nessa ordem.

4.3.1 Interrelacionamentos entre Subtarefas (QAFs)

Para que a abordagem aqui proposta lide com as QAFs, estas foram combinadas em três grupos de acordo com seu significado mais geral. Os agentes adotam comportamentos diferentes para cada um dos grupos, cujos nomes já dão uma idéia preliminar de qual será. Tais grupos são:

Priorização. As QAFs *sum*, *seq sum*, *min*, *seq min*, *all* e *sync sum* caracterizam-se por requererem que todas (AND lógico) ou o maior número possível de subtarefas ou métodos sejam escalonados;

Inibição. A QAF *exactly one* caracteriza-se por requerer que uma, e somente uma, das subtarefas ou métodos sejam escalonados (XOR lógico);

Indiferentes. As QAFs *max*, *seq max*, *last* e *seq last* caracterizam-se por considerar que mais de uma das subtarefas ou dos métodos possam ser escalonados, mas somente levando em conta a qualidade da execução de apenas um deles.

Quando um agente percebe uma tarefa ligada a outras por uma QAF do grupo denominado "Priorização" ele imediatamente delibera sobre as demais tarefas interrelacionadas a esta. Tais tarefas interrelacionadas serão então analisadas antes que as demais tarefas percebidas, atribuindo-as prioridade na utilização dos recursos do agente naquele momento. Assim, os agentes tomam sua decisão com base na tendência de escolha para cada tarefa, respeitando a priorização que os incentiva a cumprir as QAFs deste grupo.

Caso o agente decida escalonar uma tarefa que é interrelacionada por uma QAF do grupo Priorização, mas não possua recursos suficientes para escalonar todas as demais

tarefas interrelacionadas que percebeu, este agente utiliza o mesmo mecanismo de "pedir ajuda" presente no transporte cooperativo dos insetos sociais discutido no capítulo 3. Através de um processo de comunicação simples com os agentes mais próximos, seja fisicamente ou por algum outro critério de acessibilidade, o agente solicita aos seus vizinhos que priorizem em sua análise a escolha das tarefas interrelacionadas que não foram escalonadas por ele. Os agentes que recebem a mensagem adicionam essa tarefa na estrutura de tarefas (TS em TÆMS) percebida, a esquerda e imediatamente abaixo do nodo raiz (TG em TÆMS) da árvore com a estrutura de tarefas. Além disso, essa nova tarefa é interrelacionada com as demais do mesmo nível na árvore por uma QAF *sum*. Com isso, tal tarefa será analisada antes que as percebidas diretamente pelo próprio agente.

O grupo denominado "Inibição" congrega as QAFs que consideram que apenas uma das subtarefas deva ser alocada. Assim, uma vez que tarefas interrelacionadas dessa maneira sejam identificadas, os agentes as ordenam aleatoriamente e decidem quais escolher nessa ordem. Quando uma das tarefas for escolhida, as demais passam a ser ignoradas. Os agentes utilizam nesse caso o mesmo mecanismo de comunicação simples com os vizinhos utilizado no grupo anteriormente explicado. Com isso, os agentes podem indicar aos outros que não desperdicem esforços em determinada tarefa pois outra, que está interrelacionada com esta por uma QAF *exactly one*, já foi escalonada.

As QAFs do grupo denominado "Indiferentes" não alteram o modo como os agentes fazem suas escolhas. Tarefas interrelacionadas por QAFs desse grupo são ordenadas aleatoriamente para que os agentes tomem sua decisão. Nenhum mecanismo de cooperação, como os mencionados acima, são utilizados nesse caso.

Além dos têS grupos definidos aqui, algumas QAFs caracterizam-se por exigir que as tarefas relacionadas por elas sejam executadas seqüencialmente ou simultaneamente. Estas QAFs, cujos nomes tem prefixo *seq* ou *sync*, também influenciam o processo de tomada de decisão que está em discussão. A abordagem proposta lida com essa questão de duas formas diferentes:

- As tarefas interrelacionadas pelas QAFs *seq min*, *seq sum*, *seq max* e *seq last* devem ser escalonadas na ordem que aparecem na estrutura de tarefas. Os agentes devem considerá-las nessa ordem no processo de tomada de decisão, ou seja, a segunda tarefa só é analisada depois que a primeira estiver escalonada, e assim por diante.

- As tarefas interrelacionadas pela QAF *sync sum* devem ter o mesmo tempo de início no escalonamento. Esta QAF pertence ao grupo Priorização, o que faz com que os agentes tentem escalonar todas as tarefas simultaneamente ou gerem mensagens para recrutar seus vizinhos a se engajar no escalonamento das não escalonadas. Para garantir o sincronismo do escalonamento, as mensagens nesse caso incluem o tempo de início em que a primeira tarefa foi escalonada. Os agentes que recebem tais mensagens, além de priorizar a escolha dessas tarefas, também procuram escaloná-la no mesmo tempo de início.

Não respeitar as QAFs que possuem as características destacadas acima implica em produzir um escalonamento da supertarefa inválida ou muito ineficiente, onde pouca ou nenhuma qualidade é obtida. Como o processo de tomada de decisão é probabilístico e os agentes são limitados pelos seus recursos disponíveis, pode acontecer da primeira subtarefa de muitas não ser escalonada. Isso impede o escalonamento das demais e prejudica a qualidade do escalonamento. Além disso, os agentes podem não ter disponibilidade para escalonar uma tarefa em determinado intervalo de tempo por já ter outra ocupando

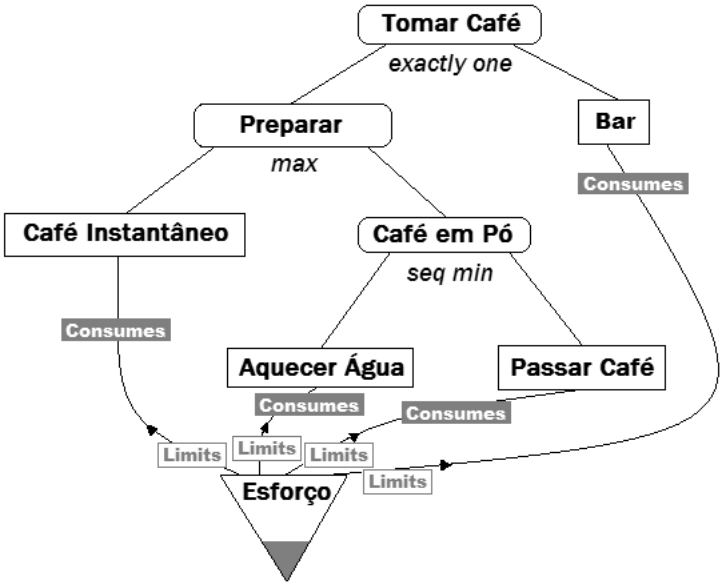


Figura 4.2: Exemplo de uma TS (*task structure*) subjetiva em TÆMS para o objetivo de "Tomar Café".

esse intervalo. Mais uma vez, o escalonamento gerado não produziria a qualidade desejada. Para lidar com essa questão, os agentes podem repetir o processo de escalonamento, tantas vezes quantas forem necessárias, para produzir um escalonamento que satisfaça as restrições de seqüenciamento e sincronismo.

As ações descritas acima devem ser tomadas pelos agentes de acordo com cada um dos grupos e em relação a todas as QAFs da TS, partindo da raiz (TG) até suas folhas (métodos).

Seja a estrutura de tarefas TS em TÆMS (figura 4.2) integralmente percebida por um determinado agente (sua visão subjetiva). Este agente tem que escolher quais métodos (retângulos brancos) dessa TS serão escolhidos em cada momento para compor um escalonamento, respeitando o interrelacionamento entre tais métodos e suas supertarefas (retângulos de cantos arredondados).

Partindo da raiz, o agente tem que determinar qual das subtarefas da tarefa "Tomar Café" (TG) este irá considerar. Como a QAF desta tarefa pertence ao grupo Inibição, o agente ordena aleatoriamente os métodos filhos da tarefa "Preparar" (todos) e o método "Bar".

Assume-se, para fins didáticos, que o primeiro método na ordem aleatória comentada acima seja "Bar" e que sua tendência seja de 0.2. Se a roleta sortear um número maior que este, o método não será alocado e o próximo na ordem será analisado. Caso isso aconteça, o próximo método é necessariamente um dos filhos da tarefa "Preparar". Com isso, a próxima QAF a ser tratada será a desta tarefa, a qual pertence ao grupo Indiferentes. Nesse caso, o agente apenas ordena aleatoriamente os métodos filhos de "Preparar" e analisa cada um deles. O método "Passar Café" fica de fora pois seu antecessor na seqüência ainda não foi escalonado (QAF *seq min* de sua tarefa pai).

Neste exemplo está sendo levado em conta que o agente demanda esforços para obter os ingredientes, preparar o café e (ou) se deslocar entre diferentes locais. Considerando então que o primeiro método nesta última ordenação aleatória seja o "Aquecer Água", o agente então verifica se a quantidade do recurso "Esforço" que este possue são suficientes

dada a demanda do método. Caso esse seja o caso, o agente então escolhe este método e diminui do seu total de recursos disponíveis a quantidade empregado nesse escalonamento.

Como a supertarefa do método "Aquecer Água" possui uma QAF do grupo Priorização, todas as tarefas a esta interrelacionadas devem ser priorizadas. Assim, o agente imediatamente analisa o método "Passar Café". Se o agente não possuir recursos para escolher esse método, este envia uma mensagem para um de seus vizinhos solicitando seu engajamento nessa tarefa, haja visto que esse já decidiu escalonar a outra da interrelação.

O próximo método da ordenação dos filhos de "Preparar" que ainda não foi analisado é o "Café Instantâneo". O agente, possuindo uma quantidade do recurso "Esforço" disponível suficiente para esse caso, computa a tendência para esse método e decide ou não escaloná-lo. Uma vez que o método "Café Instantâneo" seja escolhido, mais nenhuma outra ação resulta dessa decisão visto que a QAF que interrelaciona este método com a tarefa "Café em Pó" pertence ao grupo Indiferentes. Assim, o processo de tomada de decisão está concluído.

4.3.2 Interdependência entre Tarefas (NLEs)

Os NLEs são classificados como restritivos e não restritivos por definição, como discutido na seção 2.3. Os NLEs restritivos habilitam ou impedem que uma tarefa dependente de outra seja escalonada caso esta última tenha sido escalonada primeiro. Os NLEs não restritivos apenas impõem um bônus ou um ônus para o escalonamento de uma tarefa dependente de outra quando esta última é escalonada primeiro. A abordagem proposta aqui lida com os NLEs de forma diferente de acordo com estas características.

Os NLEs restritivos impedem ou possibilitam que determinadas tarefas sejam analisadas pelo processo de tomada de decisão. Os agentes devem obedecer as restrições impostas por estes NLEs, só considerando para a tomada de decisão aquelas tarefas que estiverem habilitadas para o escalonamento. Já os NLEs não restritivos, diferentemente do que foi apresentado até agora neste capítulo, têm impacto sobre a decisão dos agentes influenciando no estímulo que os métodos interdependentes produzem.

Os agentes alteram o estímulo original das tarefas para lidar com os NLEs não restritivos. Este estímulo passa a incorporar também os estímulos de suas dependentes. Com isso, a tendência dos agentes para escolher uma tarefa com esse tipo de NLE é influenciado também pelo estímulo de todas as tarefas da qual a tarefa analisada é dependente.

Sejam $\mathcal{H}_j$ e $\mathcal{F}_j$ os conjuntos que contêm as tarefas que se relacionam com a tarefa j através dos NLEs não restritivos *hinder* e *facilitate*, respectivamente. O conjunto $\mathcal{F}_j$ contém as tarefas que são facilitadas pela tarefa j e o conjunto $\mathcal{H}_j$ contém as tarefas que são dificultadas pela tarefa j.

O novo estímulo s'_{ij} do agente i relacionado à tarefa j é calculado em função de seu estímulo s_{ij} e a média dos estímulos das tarefas que pertencem a cada um destes conjuntos, como mostra a equação 4.2. Com isso, o novo estímulo da tarefa diminui e (ou) aumenta proporcionalmente ao estímulo das tarefas da qual esta tem um relacionamento de interdependência. Essa variação é limitada pelos valores que o estímulo pode assumir, ou seja, $0 \leq s'_{ij} \leq 1$.

$$s'_{ij} = s_{ij} - \frac{\sum_{k \in \mathcal{H}_j} s_k}{|\mathcal{H}_j|} + \frac{\sum_{k \in \mathcal{F}_j} s_k}{|\mathcal{F}_j|} \tag{4.2}$$

onde:

s_{ij} estímulo do método j para o agente i, limitado a $0 \leq s'_{ij} \leq 1$

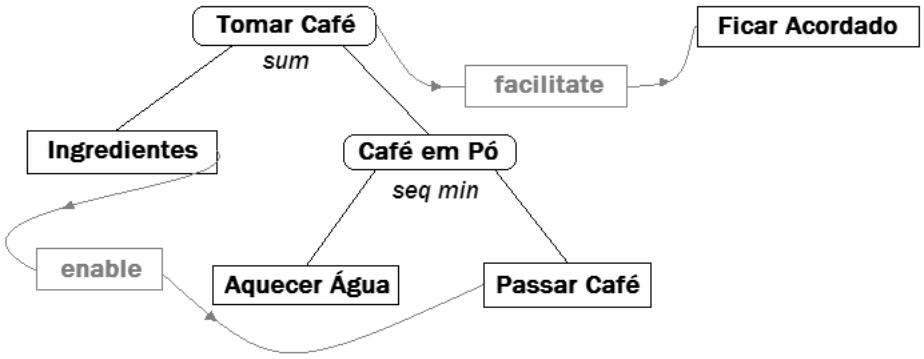


Figura 4.3: Exemplo de parte de uma TS (*task structure*) subjetiva em TÆMS com tarefas interdependentes.

As tarefas interdependentes, de forma restritiva ou não restritiva, são tratadas como aquelas interrelacionadas pelas QAFs do grupo Priorização. Quando uma determinada tarefa é escolhida, aquelas que são dependentes dessa, e que por sua vez não foram escalonadas, são comunicadas aos outros agentes por meio do mecanismo de mensagens explicado na seção anterior (4.3.1).

Sendo a estrutura de tarefas parcial TS em TÆMS (figura 4.3) percebida integralmente por um determinado agente (parte de sua visão subjetiva), este agente tem que escolher quais métodos da TS serão escalonados por ele, respeitando o interdependência entre métodos e tarefas dadas pelos NLEs (retângulos cinzas).

Entre os métodos que podem ser analisados pelo agente estão “Ingredientes”, “Aquecer Água” e “Ficar Acordado”. O método “Passar Café” está impedido pelo NLE restritivo *enable*, que cria uma dependência entre este método e o método “Ingredientes”. Supondo que todos os métodos tenham estímulo igual a 0.3, os métodos “Ingredientes”, “Aquecer Água” e “Passar Café” têm seus estímulos incrementados pelo valor do estímulo do método “Ficar Acordado”. Isso acontece porque este último é dependente de uma tarefa da qual tais métodos são filhos, netos, etc. Assim, os estímulos em questão passam a ser 0.6 e seus respectivos métodos têm uma tendência significativamente maior de ser escalonados antes do método que estes facilitam. O método “Ficar Acordado” pode ser escalonado antes dos métodos dos quais ele depende, mas sua qualidade necessariamente será menor por definição.

4.4 Polimorfismo

A especialização dos indivíduos de acordo com sua forma, como mencionado no capítulo 3, é denominada pelos biólogos de *polimorfismo*. O polimorfismo pode ser modelado através do limiar interno de resposta, mostrado na seção 4.2. Para diferenciar os agentes e obter a especialização morfológica basta iniciar os limiares internos de cada agente de acordo com as tarefas as quais sua “forma” é mais adaptada.

Assim, o limiar interno do agente será iniciado para refletir sua competência. Esta inicialização só poderá ocorrer para as tarefas previamente conhecidas pelos agentes. Caso esse não seja o caso, os agentes são iniciados com um limiar (igual a 0), que produz a maior tendência possível para tais tarefas, com o objetivo que o mecanismo de especialização, que será visto na seção 4.5, funcione adequadamente.

A equação 4.3 calcula o valor inicial do limiar interno θ_{ij} como sendo determinado

pelo inverso da competência k_{ij} do agente i para executar a tarefa j. Como dito anteriormente na seção 4.2, quanto menor é o limiar, mais competente é o agente na realização da tarefa.

$$\theta_{ij} = 1 - k_{ij} \tag{4.3}$$

onde:

k_{ij} competência do agente i executar a tarefa j, $0 \leq k_i(j) \leq 1$

4.5 Especialização

A abordagem proposta aqui caracteriza-se essencialmente pela tomada de decisão probabilística por parte dos agentes a respeito de quais tarefas escalonar. O polimorfismo, discutido na seção 4.4, permite que os agentes tenham sua tendência para escolher uma determinada tarefa fortemente atrelada a sua competência para escaloná-la. Porém, outros dois fatores são bastante relevantes no processo de tomada de decisão: o estímulo da tarefa, que é o outro fator na determinação da tendência, como discutido na seção 4.2; e a ordem em que as tarefas são analisadas, conforme discutido na seção 4.3.

Se o limiar interno do agente diminuir significativamente para algumas tarefas e aumentar significativamente para outras, a influência dos fatores comentados acima diminui drasticamente. Os agentes que tem seu limiar interno próximo a 0 para determinadas tarefas têm tendência próxima à máxima para alocar tais tarefas. Da mesma forma, os agentes que têm seu limiar interno próximo a 1 praticamente não têm tendência para escolher as tarefas relativas a esse limiar quase máximo. A figura 4.1 ilustra esse comportamento da tendência. Assim, os agentes podem modificar seus limiares internos para especializarem-se nas tarefas nas quais são mais bem sucedidos e evitar que outros fatores os levem a escalonar tarefas que implicam em um pior desempenho do sistema.

Além disso, uma das características dos cenários que esta abordagem se propõe a lidar é sua observabilidade parcial. Em algumas situações o polimorfismo, discutido na seção 4.4, não pode ser usado pela falta de informação do agente sobre as tarefas que percebeu. Um agente pode perceber uma tarefa na qual este não sabe sua competência para alocá-la. Em outras palavras, o limiar interno do agente não pode ser configurado para refletir suas competências. Com isso, o agente não consegue determinar uma tendência, que seja adequada a sua competência, para escolher tal tarefa. Nessa situação os agentes precisam adaptar seus limiares internos de resposta dinamicamente para garantir que os agentes mais capazes no escalonamento de determinadas tarefas estejam de fato escalonando-as.

Para lidar com essas questões, propõe-se condicionar a atualização dos limiares internos do agente ao seu sucesso em alocar uma tarefa, seguindo o modelo discutido na seção 3.2 (equações 3.3 e 3.4). Assim, quanto mais eficiente é o agente no escalonamento de uma tarefa, de acordo com sua competência, mais especializado nesta tarefa o agente será.

A qualidade q_{ij} quando o agente i aloca a tarefa j deve ser calculada em função de sua competência. Cada agente i tem uma competência k_{ij}, onde $0 \leq k_{ij} \leq 1$, para alocar a tarefa j. Assim, a qualidade q_{ij} do escalonamento de um método é dada pela equação 4.4.

$$q_{ij} = k_{ij} \tag{4.4}$$

onde:

k_{ij} competência do agente i em realizar a tarefa j

Assume-se que $\mathcal{M}_n$ é o conjunto de tarefas escalonadas e que $start_n(j)$ é a função que retorna o tempo em que a tarefa j começa, ambos com relação ao escalonamento n. O conjunto de escalonamentos armazenados pode ser representado por $\mathcal{S}$, onde $\mathcal{S}_n = \{start_n(1), start_n(2), ..., start_n(j), ..., start_n(|\mathcal{M}_n|)\}$. A qualidade q_{ij} de cada tarefa j, de acordo com o agente i que a escalonou, é usada para determinar a qualidade total acumulada na estrutura de tarefas para cada escalonamento $\mathcal{S}_n \in \mathcal{S}$.

A qualidade total da TS, $QAF_{TG}(\mathcal{S}_n)$, para o escalonamento $\mathcal{S}_n$, é computada segundo as QAFs de cada tarefa pai (supertarefa) da tarefa j na TS. Como explicado na seção 2.3, este procedimento de calcular a qualidade da supertarefa em função de suas subtarefas ou métodos segue recursivamente até o TG (das folhas até a raiz), onde então se determina a qualidade da TS como um todo. A definição da função $QAF_{TG}(\mathcal{S}_n)$ é parte da definição da plataforma TÆMS.

Sendo $\mathcal{M}$ o conjunto de métodos escalonados, o custo acumulado $CA_{TG}(\mathcal{S}_n)$ do escalonamento $\mathcal{S}_n$ é o somatório do custo c_j, definido na TS, de cada método $j \in \mathcal{M}$. Este cálculo se dá como definido na equação 4.5.

$$CA_{TG}(\mathcal{S}_n) = \sum_{j \in \mathcal{M}} c_j \tag{4.5}$$

onde:

c_j custo do método j, definido na TS

O desempenho $D(\mathcal{S}_n)$ dos agentes, no escalonamento $\mathcal{S}_n$, de acordo com o custo e a qualidade acumulados, é dado pela equação 4.6. Para o cálculo deste desempenho assumi-se que, em problemas em que o custo é desconsiderado ($CA_{TG}(\mathcal{S}_n) = 0$), o desempenho será determinado apenas pela qualidade acumulada total $QAF_{TG}(\mathcal{S}_n)$.

$$D(\mathcal{S}_n) = \begin{cases} \frac{QAF_{TG}(\mathcal{S}_n)}{CA_{TG}(\mathcal{S}_n)} & se\ \ CA_{TG}(\mathcal{S}_n) \neq 0 \\ QAF_{TG}(\mathcal{S}_n) & caso\ contrário \end{cases} \tag{4.6}$$

onde:

$QAF_{TG}(\mathcal{S}_n)$ qualidade acumulada total segundo as QAFs para o escalonamento $\mathcal{S}_n$

$CA_{TG}(\mathcal{S}_n)$ custo total do escalonamento $\mathcal{S}_n$

Cada intervalo de tempo necessário para se obter um escalonamento completo é denominado "rodada". O desempenho $D(\mathcal{S}_n)$ dos agentes será utilizado para, a cada μ rodadas, atualizar o limiar interno dos agentes com o objetivo de especializá-los em relação ao escalonamento de maior qualidade.

O escalonamento de maior qualidade em $\mathcal{S}$, cujo índice é n^+, é determinado pela equação 4.7. O escalonamento $\mathcal{S}_{n^+}$ é aquele onde o agente obteve o melhor desempenho dentre os armazenados em $\mathcal{S}$ nas últimas μ rodadas.

$$n^+ = argmax_{t-\mu}^{t} D(\mathcal{S}_n) \tag{4.7}$$

onde:

$D(\mathcal{S}_n)$ desempenho dos agentes no escalonamento $\mathcal{S}_n$

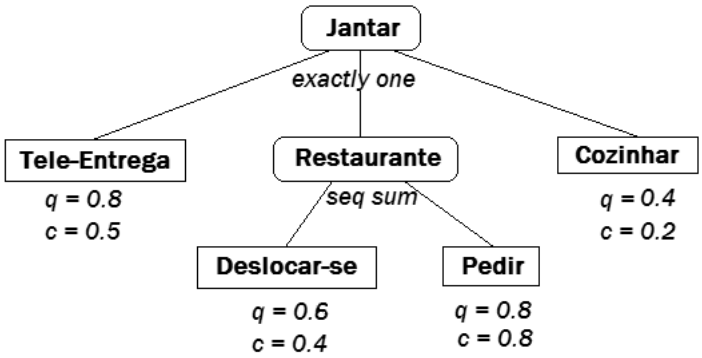


Figura 4.4: Exemplo de uma TS (*task structure*) subjetiva em TÆMS para o objetivo de "Jantar".

Tabela 4.1: Competência k e limiar interno θ de um agente hipotético para cada método na TS da figura 4.4.

	Métodos			
	"Tele-Entrega"	"Deslocar-se"	"Pedir"	"Cozinhar"
k	0.8	0.6	0.8	0.4
θ	0.2	0.4	0.2	0.6

Cada agente i deve ter seu limiar interno atualizado em relação a tarefa j que executou no escalonamento $\mathcal{S}_{n^+}$. Os coeficientes ξ e ρ das equações, 4.8 e 4.9, são utilizados para aumentar e diminuir, respectivamente, os limiares internos dos agentes em relação as tarefas onde os agentes obtiveram o melhor desempenho.

O limiar de resposta θ_{ij} de um agente i em relação à tarefa j é atualizado segundo a equação 4.8, quando o agente i executa j em $\mathcal{S}_{n^+}$, e pela equação 4.9, caso contrário. O limiar de resposta deve variar limitado a $0 \leq \theta_{ij} \leq 1$.

$$\theta_{ij}(t) = \theta_{ij}(t - \mu) - \xi \tag{4.8}$$

onde:

ξ coeficiente de aprendizado

$$\theta_{ij}(t) = \theta_{ij}(t - \mu) + \rho \tag{4.9}$$

onde:

ρ coeficiente de esquecimento

Seja a estrutura de tarefas TS em TÆMS (figura 4.4) integralmente percebida por um determinado agente (sua visão subjetiva). A qualidade q_j e o custo c_j dos métodos encontram-se identificados na TS. Considera-se, por simplificação, que o estímulo s_j para todos os métodos é o mesmo e tem valor 0.5. As competências do agente para os métodos dessa TS podem ser vistas na tabela 4.1. Segundo o mecanismo de especialização, em discussão nesta seção, o agente irá realizar sucessivos escalonamentos completos da TS com o objetivo de especializar-se nos métodos que maximizem seu desempenho.

Partindo da raiz, o agente tem que determinar qual das subtarefas da tarefa "Jantar" (TG) este irá considerar. Como a QAF desta tarefa (*exactly one*) pertence ao grupo Inibição (discutido em detalhes na seção 4.3), o agente ordena aleatoriamente os métodos

filhos da tarefa “Restaurante” (todos) juntamente com os métodos “Cozinhar” e “Tele-Entrega”.

Supondo, para fins didáticos, que o primeiro método da ordenação seja “Tele-Entrega”, a tendência do agente para escolher este método é calculada pela equação 4.1, sendo então $\frac{0.5^2}{0.5^2+0.2^2} = 0.86$. Se a roleta sortear um número menor que este, o método será escolhido e os demais serão desconsiderados. Caso isso aconteça, a qualidade obtida pelo agente ao escalonar este método será de 0.8 (Equação 4.4). A qualidade acumulada da TS, dada a QAF da TG, será igual a qualidade do método, $QAF_{\text{“}Jantar''} = 0.8$. O custo acumulado da TS será igual ao custo do método alocado (um apenas), $CA_{\text{“}Jantar''} = 0.5$. Assim, o desempenho do agente para este escalonamento, denominado $\mathcal{S}_1$, é $D(\mathcal{S}_1) = 1.6$ (equação 4.6).

Concluído o primeiro escalonamento (S_1), o processo inicia novamente. Supõem-se agora que o primeiro método da ordenação seja “Deslocar-se”. Para simplificar o exemplo, considera-se que, com uma tendência dada por $\frac{0.5^2}{0.5^2+0.4^2} = 0.61$, o agente irá alocar esse método. A qualidade obtida para o método nesse caso será de 0.1 (Equação 4.4). Dado que o grupo da QAF da supertarefa desse método (*seq sum*) é o Priorização, o agente imediatamente computa sua tendência para o método “Pedir”, tendo esta o valor $\frac{0.5^2}{0.5^2+0.2^2} = 0.86$. O agente então escolhe este método, com qualidade 1 (Equação 4.4). Desta vez a qualidade acumulada da TS, dada a QAF da tarefa “Restaurante” e de sua supertarefa “Jantar” (TG), será igual a soma das qualidades dos dois métodos, $QAF_{\text{“}Jantar''} = 1.4$. O custo acumulado da TS será igual ao somatório do custo do métodos alocados (equação 4.5) , $CA_{TG} = 1.2$. Finalmente, o desempenho do agente para este novo escalonamento, denominado $\mathcal{S}_2$, é $D(\mathcal{S}_2) = 1.17$ (equação 4.6).

Considerando o intervalo entre a atualização dos limiares $\mu = 2$, o agente agora deve atualizar seus limiares internos segundo os métodos escalonados no melhor escalonamento entre $\mathcal{S}_1$ e $\mathcal{S}_2$ (equação 4.7). Nesse caso, considerando o intervalo de duas rodadas ($\mu = 2$), $\mathcal{S}_1$ é o escalonamento de maior qualidade. Supondo que ξ e ρ assumam ambas o valor 0.05, utilizando-se as equações 4.9 e 4.8 o limiar do agente para o método escalonado “Tele-Entrega” será decrementado de 0.05, passando a ser $\theta_{\text{“}Tele-Entrega''} - \xi = 0.2 - 0.05 = 0.75$, enquanto que os limiares para os demais métodos não escalonados serão incrementados de 0.05. Por exemplo, o limiar interno para o método “Cozinhar” será incrementado como segue $\theta_{\text{“}Cozinhar''} + \rho = 0.6 + 0.05 = 0.65$.

Esse mecanismo faz com que, na próxima vez que esse agente hipotético analisar a TS em questão, sua tendência para escalonar os métodos que lhe renderam o melhor resultado seja maior. Desta forma, o agente consegue aumentar suas chances de obter novamente um melhor escalonamento para a TS.

4.6 Adaptação do Estímulo

Na abordagem proposta neste trabalho os agentes escalonam tarefas tentando minimizar o intervalo de tempo necessário para isso, respeitando todas as restrições e características dos cenários.

Como dito anteriormente, as tarefas produzem um estímulo para o agente. Este estímulo pode ser fixo, de acordo com a importância da tarefa em um determinado cenário, ou variável, para aumentar a tendência do agente escolhê-la por algum motivo específico.

Propõe-se aumentar periodicamente o estímulo das tarefas que compõe o escalonamento com tempo final menor de forma crescente conforme o tempo de início das tarefas avança no intervalo de tempo do escalonamento.

Utiliza-se o viés de atualização $\Gamma(j)$ para determinar o quanto aumentar no estímulo das tarefas dado um determinado escalonamento. Este viés, calculado pela equação 4.10, é menor conforme a unidade de tempo do início da tarefa distancia-se do início do escalonamento. Aumentando o estímulo de acordo com este viés de atualização faz-se com que os agentes tenham um maior estímulo para escalonar as tarefas na mesma ordem em que foram escalonadas previamente.

$$\Gamma(j) = 1 - \frac{start_{n^*}(j)}{|\mathcal{S}_{n^*}|} \tag{4.10}$$

onde:

$start_{n^*}(j)$ unidade de tempo em que o método j tem início no melhor escalonamento $n \in \mathcal{S}$

$|\mathcal{S}_{n^*}|$ duração total do melhor escalonamento $\mathcal{S}_{n^*}$

Considera-se que $\mathcal{M}_n$ é o conjunto de tarefas escalonadas e que $start_n(j)$ é a função que retorna o tempo em que a tarefa j começa, ambos com relação ao escalonamento n. O conjunto de μ escalonamentos armazenados pode ser representado por $\mathcal{S}$, onde $\mathcal{S}_n = \{start_n(1), start_n(2), ..., start_n(j), ..., start_n(|\mathcal{M}_n|)\}$.

A duração total $|\mathcal{S}_n|$ do escalonamento $\mathcal{S}_n$ é dada pelo tempo em que a última tarefa foi terminada, como mostra a equação 4.11.

$$|\mathcal{S}_n| = max_{j=1}^{|\mathcal{M}_n|}[start_n(j) + d_j] \tag{4.11}$$

onde:

$start_n(j)$ tempo onde o método j teve sua alocação iniciada no escalonamento n

$|\mathcal{M}_n|$ número de tarefas escalonados no escalonamento n

d_j duração da tarefa j

O desempenho de um escalonamento completo é medido de acordo com o número de tarefas escalonadas e sua duração total.

Inicialmente, os escalonamentos em $\mathcal{S}$ que são compostos pelo maior número de tarefas nas últimas μ rodadas tem seu índice n pertencente ao conjunto $\mathcal{N}^*$, determinado pela equação 4.12.

$$\mathcal{N}^* = \{m^* \mid m^* = argmax_t^{t-\mu}|\mathcal{M}_n|\} \tag{4.12}$$

onde:

$|\mathcal{M}_n|$ número de tarefas escalonadas no escalonamento n

O melhor escalonamento em $\mathcal{S}$, cujo índice é n^*, é então determinado pela equação 4.13. O escalonamento $\mathcal{S}_{n^*}$ é aquele que tem o menor tempo de duração dentre o conjunto de escalonamentos que têm o maior número de tarefas (aqueles que têm seu índice em $\mathcal{N}^*$).

$$n^* = argmin_{n \in \mathcal{N}^*}|\mathcal{S}_n| \tag{4.13}$$

onde:

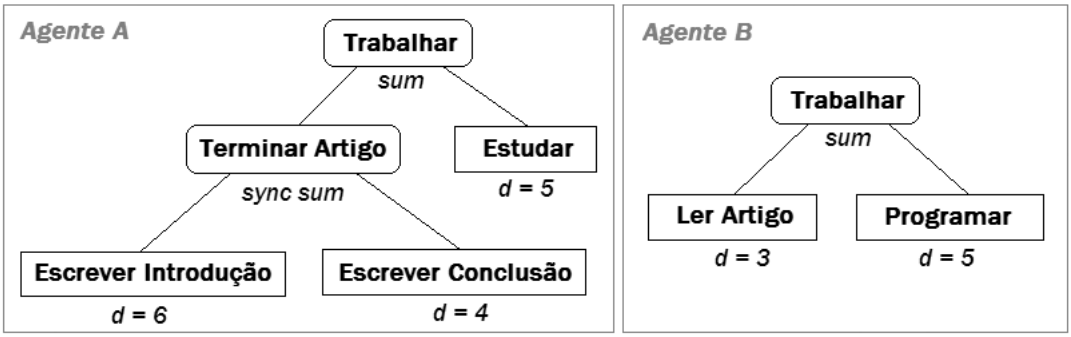


Figura 4.5: Exemplo de duas estruturas de tarefas TSs subjetivas em TÆMS, pertencente aos agentes hipotéticos A e B, para o objetivo "Trabalhar".

$|\mathcal{S}_n|$ duração total do escalonamento $\mathcal{S}_n$

O estímulo local $s_{ij}(t)$ relacionado à tarefa j e ao agente i no tempo t é dado pela equação 4.14. Neste caso, o estímulo decresce com o passar do tempo, segundo a constante δ e apenas cresce segundo o viés de atualização $\Gamma(j)$, computado de acordo com o melhor escalonamento obtido pelos agentes, dado por $\mathcal{S}_{n^*}$.

A atualização do estímulo segundo a equação 4.14 se dá com freqüência μ. Desta forma, durante μ escalonamentos os agentes armazenam as soluções obtidas e, apenas na última delas, utilizam a melhor solução como base para o cálculo do viés de atualização $\Gamma(j)$. O estímulo varia limitado a $0 \leq s_{ij} \leq 1$.

$$s_{ij}(t) = s_{ij}(t - \mu) * \delta + \Gamma(j) * \alpha \tag{4.14}$$

onde:

$s_{ij}(t)$ estímulo local da tarefa j na rodada t

$s_{ij}(t - \mu)$ estímulo local da tarefa j na rodada $t - \mu$

δ constante para diminuir o valor do estímulo com o passar do tempo

$\Gamma(j)$ viés de atualização do estímulo de j

α taxa de desconto para diminuir o impacto do viés de atualização sobre o estímulo

Sejam as estruturas de tarefas TSs em TÆMS (figura 4.5) integralmente percebida por dois agentes hipotéticos A e B (suas visões subjetivas), considera-se, por simplificação: que o estímulo s_{ij} é o mesmo para todos os métodos e ambos os agentes, tendo valor 0.5; que os agentes têm recursos para alocar apenas um método a cada unidade de tempo; e que os agentes têm competência máxima para todos os métodos, ou seja, o limiar interno θ_{ij} dos agentes para todos os métodos é zero. Os tempos de duração dos métodos encontram-se identificados nas TSs.

O agente A pode escolher inicialmente um entre os métodos "Estudar", "Escrever Introdução" e "Escrever conclusão", que são analisados pelo agente em ordem aleatória (segundo o processo de tomada de decisão da abordagem proposta, previamente discutido e exemplificado). Caso o agente em questão escolha primeiramente o método "Estudar", este método será escalonado no primeiro intervalo possível de sua linha de tempo. Nesse

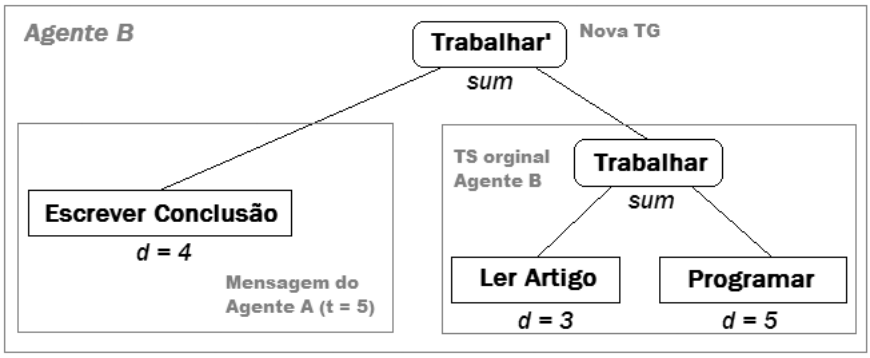


Figura 4.6: Modificação da estrutura de tarefas TS em TÆMS, pertencente ao agente B (figura 4.5), após este receber uma mensagem do agente A.

caso, o método terá início na unidade de tempo 0 e ocupará as 5 primeiras unidades de tempo ($d = 5$) do escalonamento.

Na seqüencia do processo de tomada de decisão o agente A escolhe novamente outro método. Supondo que o método escolhido seja o "Escrever Introdução", o agente A necessariamente enviará uma mensagem para o agente B solicitando sua cooperação para o escalonamento do método "Escrever conclusão". Isso ocorre pois os métodos "Escrever Introdução" e "Escrever conclusão" precisam ser escalonados na mesma unidade de tempo inicial (são interrelacionados pela QAF *sync sum*) e os agentes não tem recursos para alocar mais de um método simultaneamente. O método escolhido pelo agente A irá iniciar na unidade de tempo 5 de seu escalonamento e irá ocupar 6 unidades de tempo ($d = 6$). A mensagem mencionada acima é enviada ao agente B indicando que o método "Escrever conclusão" precisa ser escalonado nesta mesma unidade de tempo.

Assim, o método "Escrever conclusão" é incluído pelo agente B, que estende sua estrutura de tarefas TS como mostra a figura 4.6. Considerando que o agente B já havia escolhido o método "Ler Artigo" de sua estrutura de tarefas e que este foi acomodado no início de seu escalonamento ($d = 3$), o método "Escrever conclusão" pode ser escalonado na mesma unidade de tempo que o agente A utilizou. Caso o agente B escolha escalonar este método, este terá início na unidade de tempo 5 do seu escalonamento. Na seqüencia, caso o agente B escolha escalonar o método "Programar", este iniciará na unidade de tempo 9 pois não há espaço antes na linha de tempo do escalonamento.

Após o processo descrito acima, um escalonamento completo é obtido pelos agentes A e B para suas respectivas estruturas de tarefas (figuras 4.5 e 4.6). Este escalonamento, denominado de escalonamento 1, pode ser visto na figura 4.7. Além disso, a figura 4.7 também mostra um outro escalonamento possível, denominado escalonamento 2, que poderia ser obtido a partir de escolhas diferentes que os agentes realizassem em cada passo do processo de tomada de decisão discutido a pouco. A duração do escalonamento 1 ($|\mathcal{S}_1|$) é 11 para o agente A e 14 para o agente B, enquanto que a duração do escalonamento 2 ($|\mathcal{S}_2|$) é igual ao escalonamento 1 para o agente A e 12 para o agente B.

Segundo o mecanismo de adaptação do estímulo, em discussão nesta seção, o agente irá realizar sucessivos escalonamentos completos da TS para modificar os estímulos das tarefas com o objetivo de minimizar o tempo total de escalonamento.

Assume-se, para simplificar o exemplo, apenas o comportamento do agente B e que são realizados dois escalonamentos entre as adaptações dos estímulos ($\mu = 2$ na equação 4.14). Este agente agora deve atualizar os estímulos para as tarefas por ele percebida de acordo com o melhor escalonamento entre $\mathcal{S}_1$ e $\mathcal{S}_2$ (calculado pela equação 4.11). Neste

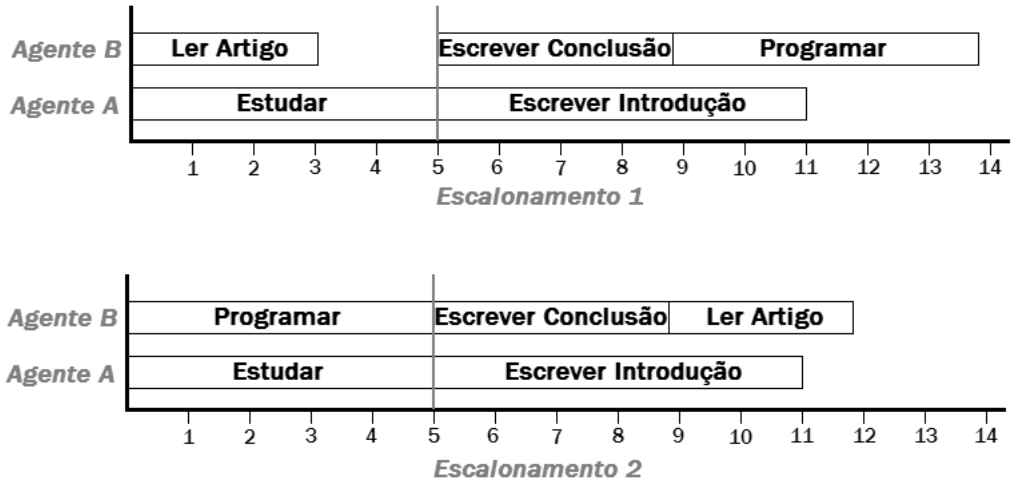


Figura 4.7: Exemplo de dois escalonamentos possíveis para as TSs (*task structures*) dos agentes A (figura 4.5) e B (figura 4.6).

Tabela 4.2: Viés de atualização Γ para cada método alocado pelo agente B.

	Métodos		
	"Programar"	"Escrever conclusão"	"Ler Artigo"
Γ	1	0.58	0.25

caso, o escalonamento que obteve maior qualidade foi o $\mathcal{S}_2$. O viés de atualização para cada um dos métodos alocados em $\mathcal{S}_2$, computados pela equação 4.10, pode ser visto na tabela 4.2.

Para a atualização do estímulo calculado pela equação 4.14, considera-se que a constante para diminuir o valor do estímulo com o passar do tempo δ assuma o valor 0.8 e que a taxa de desconto para diminuir o impacto do viés de atualização sobre o estímulo α assuma o valor 0.1. Com isso, o estímulo s_j de todos os métodos da TS do agente B irá diminuir dos originais 0.5 para 0.4. Além disso, estes irão aumentar de acordo com o viés de atualização $\Gamma(j)$, mostrado na tabela 4.2, multiplicado pela taxa de desconto 0.1. Assim, os estímulos assumirão os seguintes valores: $s_{\text{“}Programar\text{”}} = 0.5*0.8+1*0.1 = 0.5$; $s_{\text{“}Escreverconclusao\text{”}} = 0.5*0.8+0.58*0.1 = 0.46$ e $s_{\text{“}LerArtigo\text{”}} = 0.5*0.8+0.25*0.1 = 0.42$. Esse mecanismo fez com que o estímulo das tarefas se modificassem para induzir os agentes a escolhe-las na ordem em que produziram o escalonamento de menor duração.

4.7 Aplicação

Aplicar todos os mecanismos propostos neste capítulo simultaneamente dificultaria em demasia o estudo do seu comportamento, a realização de experimentos de validação e a análise dos resultados obtidos. Assim, como comentado no capítulo 1, dois problemas da área de pesquisa operacional são utilizados neste trabalho para aplicar a abordagem proposta. O *Generalized Assignment Problem* (GAP) (MARTELLO; TOTH, 1990) e o *Resource-Constrained Project Scheduling Problem* (RCPSP) (BRUCKER et al., 1999) representam partes complementares do problema tratado neste trabalho e modelam vários aspectos de aplicações reais.

Dada a natureza da abordagem proposta, tais problemas são resolvidos por um sistema

multiagente onde os agentes decidem a respeito do escalonamento de tarefas utilizando o mecanismo apresentado na seção 4.3. Esta decisão é tomada probabilisticamente e baseia-se na tendência que cada agente possui para escolher cada uma das tarefa percebidas, apresentada na seção 4.2. Os agentes utilizam o mecanismo, discutido na seção 4.3.1, para lidar com possíveis interrelações entre as tarefas. Este mecanismo demanda que exista um canal de comunicação entre os agentes que compõe o sistema.

Os mecanismos relacionados com o polimorfismo e a especialização, apresentados nas seções 4.4 e 4.5, respectivamente, são explorados pelo GAP. Neste problema as tarefas são alocadas, ou seja, agentes escolhem tarefas imediatamente, não havendo planejamento. Os agentes têm competências diferentes para alocar tais tarefas e busca-se que os agentes mais competentes para alocar determinadas tarefas o façam de fato. Além disso, no GAP as tarefas não têm relação de interdependência entre si. O mecanismo de tratamento de interdependência entre tarefas, discutido na seção 4.3.2, e de adaptação do estímulo para a minimização do tempo total dos escalonamentos, discutido na seção 4.6, não são empregados neste problema.

No RCPSP, por sua vez, os agentes não têm diferentes competências para executar as tarefas, não utilizando os mecanismos de polimorfismo e especialização explorados pelo GAP. No RCPSP, os agentes devem determinar quais tarefas alocar, considerando que têm competência máxima, e organizar a alocação destas tarefas em relação ao tempo, ou seja, as tarefas são escalonadas. Para isso, os agentes dispõem de tempo e informação suficientes para planejar a alocação das tarefas e sua disposição no tempo deve ser feita de maneira a minimizar o tempo total necessário para o escalonamento. O mecanismo de modificação do estímulo, apresentado na seção 4.6, é empregado neste problema com o objetivo de obter essa minimização. No RCPSP as tarefas são interdependentes e, com isso, aplica-se o mecanismo de tratamento de interdependência entre tarefas, discutido na seção 4.3.2.

A abordagem proposta será aplicada primeiramente na versão entendida do GAP (E-GAP) e, na seqüência, na versão distribuída do RCPSP (DRCPSP). Cabe ressaltar que os mecanismos aplicados no E-GAP terão seu desempenho comparado com os resultados obtidos por um algoritmo equivalente, o LA-DCOP (SCERRI et al., 2005) discutido no capítulo 2. Os mecanismos aplicados no DRCPSP terão seu desempenho comparado com resultados padrão, utilizados para comparações desse tipo (*benchmarks*), obtidos por algoritmos centralizados. Não são conhecidos algoritmos distribuídos que possam ser utilizados para comparação nesse caso.

O GAP e o RCPSP, bem como suas extensões e adaptações, serão discutidos e formalizados nos capítulos 5 e 6, respectivamente. Estes capítulos discutem como a abordagem proposta é aplicada nesses problemas e apresentam os resultados obtidos pelos experimentos realizados. A seção 7 apresentará uma discussão sobre a aplicação simultânea de todos os mecanismos da abordagem proposta.

5 ALOCAÇÃO DE TAREFAS NO GAP ESTENDIDO

Este capítulo apresenta os resultados da aplicação da abordagem proposta neste trabalho em uma versão estendida do *Generalized Assignment Problem* (GAP) (MARTELLO; TOTH, 1990) proposta originalmente por SCERRI et al. (2005), denominado *Extended*-GAP (E-GAP).

Os mecanismos relacionados com o polimorfismo e a especialização, apresentados nas seções 4.4 e 4.5, respectivamente, são explorados especificamente neste problema. Os mecanismos de tomada de decisão, apresentado na seção 4.3, de tendência na escolha de tarefas, apresentada na seção 4.2, e de tratamento de interrelações entre as tarefas, discutido na seção 4.3.1, também são experimentados.

O presente capítulo está organizado da seguinte forma: na seção 5.1 o problema em questão é definido; a seção 5.2 apresenta o algoritmo que implementa a abordagem proposta para a solução do E-GAP; na seção 5.3 são discutidos os experimentos realizados e os resultados obtidos; e, finalmente, na seção 5.4 é apresentada a conclusão desta aplicação experimental;

5.1 Definição do Problema

O *Generalized Assignment Problem* (GAP) (MARTELLO; TOTH, 1990) é um problema geral que trata da alocação de tarefas a agentes, respeitando sua capacidade (recursos que estes têm disponíveis), buscando maximizar uma recompensa total associada às competências destes agentes para tal alocação.

O GAP foi estendido por SCERRI et al. (2005) para lidar com problemas dinâmicos e com a interrelação entre tarefas. Esta extensão, denominada *Extended* GAP (E-GAP), captura adequadamente as características dos problemas complexos em questão neste trabalho. O E-GAP foi usado originalmente para modelar um problema de gerência de equipes de busca e salvamento em catástrofes tratado como um problema de coordenação em SMA.

O GAP pode ser definido como segue. Seja $\mathcal{J}$ o conjunto de tarefas e $\mathcal{I}$ o conjunto de agentes. Cada agente $i \in \mathcal{I}$ tem capacidade r_i associada a uma quantidade limitada de recursos. Um único tipo de recurso é considerado. Quando uma tarefa $j \in \mathcal{J}$ é alocada por um agente i, esta consome u_{ij} unidades do recurso deste agente. Cada agente tem uma competência para alocar cada tarefa j dada por k_{ij} ($0 \leq k_{ij} \leq 1$).

A matriz de alocação A, onde a_{ij} é o valor da i-ésima linha e a j-ésima coluna, é dada pela equação 5.1.

$$a_{ij} = \begin{cases} 1 & se \text{ j } está\ alocado\ por \text{ i} \\ 0 & caso\ contrário \end{cases} \tag{5.1}$$

O objetivo do GAP é encontrar a matriz A que maximiza a recompensa do sistema dada pela equação 5.2, respeitando as limitações de recursos e a limitação de ter apenas um agente alocado a cada tarefa.

$$A = argmax_{A'} \sum_{i \in \mathcal{I}} \sum_{j \in \mathcal{J}} k_{ij} * a'_{ij} \tag{5.2}$$

tal que

$$\forall i \in \mathcal{I}, \sum_{j \in \mathcal{J}} u_{ij} * a_{ij} \leq r_i$$

e

$$\forall j \in \mathcal{J}, \sum_{i \in \mathcal{I}} a_{ij} \leq 1$$

O E-GAP estende o GAP de duas maneiras diferentes, as quais são:

Interrelacionamentos entre tarefas. As tarefas no E-GAP podem ser interrelacionadas por restrições do tipo AND. Todas as tarefas interrelacionadas por esta restrição devem ser alocadas simultaneamente para serem consideradas no cálculo da recompensa. Esta é uma extensão importante porque o GAP, por definição, restringe a alocação de cada tarefa a um agente apenas. Quando uma tarefa requer mais recursos que um único agente pode oferecer, esta tarefa deve ser decomposta em subtarefas interrelacionadas pela restrição AND. De acordo com SCERRI et al. (2005), define-se $\bowtie = \{\alpha_1, ..., \alpha_k\}$, onde $\alpha_k = \{j_{k_1}, ..., j_{k_q}\}$ denota o k-ésimo conjunto de tarefas interrelacionadas pela restrição AND. Assim, a recompensa local w_{ij} para a alocação da tarefa j ao agente i é dada pela equação 5.3.

$$w_{ij} = \begin{cases} k_{ij} * a_{ij} & if\ \forall \alpha_k \in \bowtie,\ j\ \notin \alpha_k \\ k_{ij} * a_{ij} & if\ \exists \alpha_k \in \bowtie\ with\ j\ \in \alpha_k \wedge \\ & \forall j_{k_q} \in \alpha_k,\ a_{x j_{k_q}} \neq 0\ with\ x \in \mathcal{I} \\ 0 & caso\ contrário \end{cases} \tag{5.3}$$

Recompensa determinada dinamicamente. A recompensa total W é calculada no E-GAP como o somatório das recompensas parciais w^t_{ij} obtidas em cada intervalo de tempo t. Nesse caso, o que está sendo considerado é a recompensa resultante de uma seqüência de alocações, diferentemente do GAP que considera uma única alocação. Com isso, todos os conjuntos e parâmetros da equação 5.2 são indexadas pelo tempo. Além disso, um custo v_j pode ser usado para punir os agentes quando a tarefa j não é alocada no tempo t. O objetivo do E-GAP é maximizar a recompensa total W dada pela equação 5.4.

$$W = \sum_t \sum_{i^t \in \mathcal{I}^t} \sum_{j^t \in \mathcal{J}^t} w^t_{ij} * a^t_{ij} - \sum_t \sum_{j^t \in \mathcal{J}^t} (1 - \sum_{i^t \in \mathcal{I}^t} a^t_{ij}) * v^t_j \tag{5.4}$$

tal que

$$\forall t \forall i^t \in \mathcal{I}^t, \sum_{j^t \in \mathcal{J}^t} u^t_{ij} * a^t_{ij} \leq r^t_i$$

e

$$\forall t \forall j^t \in \mathcal{J}^t, \sum_{i^t \in \mathcal{I}^t} a^t_{ij} \leq 1$$

Tabela 5.1: Competência k_{ij} dos agentes para alocar cada tarefa e sua quantidade de recursos r_i, considerando uma instância hipotética do GAP.

	Agentes	
	X	Y
k_A	1	2
k_B	5	7
k_C	9	3
r	10	5

Tabela 5.2: Todas as alocações possíveis para uma instância hipotética do GAP.

	Agentes			
	X		Y	
Alocações	$X_{\mathcal{J}}$	$c_{X_{\mathcal{J}}}$	$Y_{\mathcal{J}}$	$c_{Y_{\mathcal{J}}}$
(1)	$\{B, C\}$	9	$\{A\}$	3
(2)	$\{A, C\}$	10	$\{B\}$	2
(3)	$\{C\}$	7	$\{A, B\}$	5
(4)	$\emptyset$	0	$\{A\}$	3
(5)	$\emptyset$	0	$\{B\}$	2
(6)	$\emptyset$	0	$\{A, B\}$	5
(7)	$\{A, B\}$	5	$\emptyset$	0
(8)	$\{B, C\}$	9	$\emptyset$	0
(9)	$\{A, C\}$	10	$\emptyset$	0
(10)	$\{A\}$	3	$\emptyset$	0
(11)	$\{B\}$	2	$\emptyset$	0
(12)	$\{C\}$	7	$\emptyset$	0
(13)	$\emptyset$	0	$\emptyset$	0

Os agentes devem determinar quais tarefas alocar de maneira que suas competências associadas a estas tarefas sejam as maiores possíveis. A quantidade de tarefas que os agentes podem alocar simultaneamente é restrita pela quantidade de recursos que estes têm disponível em relação a quantidade demandada pela tarefas a serem alocadas. A duração d_j destas tarefas é irrelevante para a tomada de decisão pois os agentes precisam definir quais alocar a cada unidade de tempo. As competências associadas aos agentes e as tarefas por eles alocadas determinam a recompensa do sistema como um todo. As recompensas parciais são obtidas em cada unidade de tempo e a total é o somatório das recompensas parciais durante um intervalo de tempo.

Seja uma instância simples do E-GAP, com 2 agentes (X e Y), tendo suas especificações definidas na tabela 5.1, e 3 tarefas (A, B e C) que demandam a quantidade de recursos c_{ij} igual a 3, 2 e 7, respectivamente, para todos os agentes $i \in \mathcal{I}$. Cada agente i tem uma competência k_{ij} para alocar cada uma das tarefas dada pelas primeiras linhas da tabela 5.1. A última linha desta tabela indica a quantidade de recursos r_i que cada agente i pode despender na alocação das referidas tarefas. Além disso, as tarefas B e C são interrelacionadas pela restrição AND.

Seja $i_{\mathcal{J}}$ o conjunto de tarefas $\mathcal{J}$ alocadas pelo agente i e $u_{i_{\mathcal{J}}}$ o somatório dos custos u_{ij} (recursos demandados) das tarefas do conjunto $\mathcal{J}$, a tabela 5.2 mostra todas as alocações possíveis para as três tarefas em questão considerando a limitação de recursos dos agentes, $u_{i_{\mathcal{J}}} \leq r_i$. Por exemplo, o agente X não pode alocar o conjunto de tarefas $\{A, B, C\}$

porque a soma dos custos dessas tarefas é igual a 12 e este agente tem apenas 10 unidades de recurso disponíveis.

Para as alocações válidas da tabela 5.2 os agentes têm diferentes competências, sendo que para a alocação (1) o somatório destas competências é igual a 16 ($k_{XC} + k_{XB} + k_{YA}$), para a alocação (2) igual a 17 ($k_{XA} + k_{XC} + k_{YB}$), para a (3) igual a 18 ($k_{XC} + k_{YA} + k_{YB}$) e assim por diante. As alocações a partir da (4), inclusive, têm o somatório das competências menor que os apontados aqui, e a maior recompensa é obtida pela alocação (3). As alocações da tarefa B que não incluem a tarefa C, dada a interrelação AND entre elas, fazem com que a alocação de B não conte no somatório das competências. O mesmo vale para as alocações de C que não incluem B. Por exemplo, o somatório das competências dos agentes na alocação (6) resulta em 2 (k_{YA}).

A recompensa parcial no E-GAP é o somatório das competências dos agentes para as tarefas que estes estão alocando no tempo t. Considerando 3 unidades de tempo e supondo que os agentes realizem as alocações (2), (2) e (6) consecutivamente, estes obterão uma recompensa total de 36.

Um E-GAP qualquer pode ser modelado em TÆMS. Para criar uma TS (*task structure*) da visão objetiva (problema como um todo) do E-GAP, basta seguir os seguintes passos:

1. Cria-se uma tarefa raiz TG (*task group*) para TS do E-GAP utilizando a QAF *sum* para interrelacionar suas subtarefas;

2. Para cada tarefa j do E-GAP criar uma subtarefa da TG utilizando a QAF *exactly one* para interrelacionar suas subtarefas;

3. Para cada grupo de tarefas interrelacionadas pela restrição AND cria-se uma subtarefa da TG utilizando a QAF *sync sum* para interrelacionar suas subtarefas. Todas as tarefas interrelacionadas serão filhas desta nova tarefa e não da TG;

4. Para cada agente i cria-se um método filho da tarefa j na TS e indica-se que este método pertence ao respectivo agente;

5. Configura-se o atributo qualidade q de cada um dos métodos criados de acordo com a competência do respectivo agente i, de maneira que se tenha $q = k_{ij}$;

6. Cria-se um recurso para cada agente i com o estado configurado de acordo com a quantidade r_i de recursos deste agente, identificando-o explicitamente como sendo seu;

7. Cria-se um relacionamento *consumes* e um relacionamento *limits* entre cada recurso e os métodos relacionados com o agente. Ambos os relacionamentos devem conter a quantidade de recursos c_{ij} que a tarefa j relacionada ao método consome.

8. Configura-se os demais atributos dos métodos atribuindo: o valor 0 para o custo, uma vez que no E-GAP este não é considerado da maneira como o TÆMS o define; e o valor -1 para a duração, pelo motivo explicado a seguir. O TÆMS, por definição, não permite que mais de uma tarefa seja escalonada ao mesmo tempo por um agente. No problema em questão a duração das tarefas não é relevante e é importante que os agentes aloquem mais de uma tarefa. Como não se quer minimizar o tempo do escalonamento, pode-se considerar que as tarefas escalonadas sucessivamente serão na verdade alocadas em paralelo. A documentação técnica do TÆMS recomenda a utilização de -1 quando este for o caso.

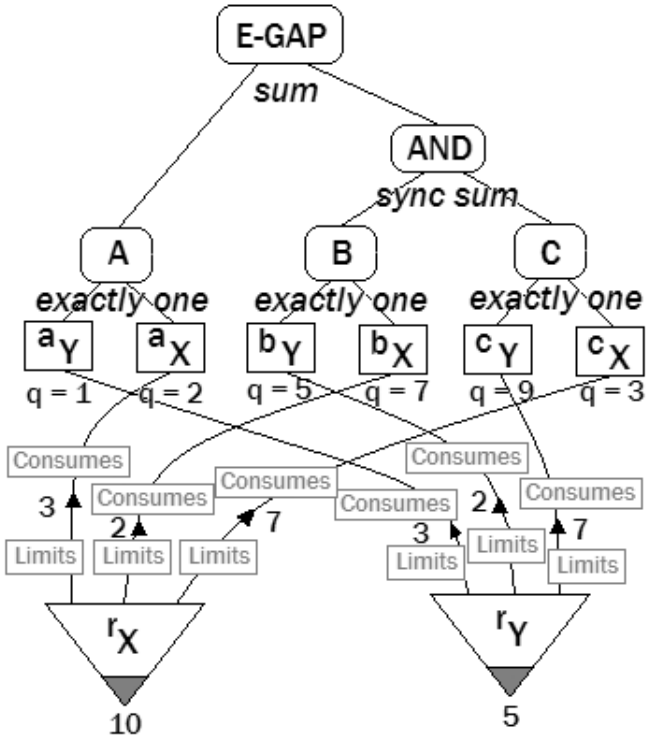


Figura 5.1: TS (*task structure*) objetiva em TÆMS para uma instância simples do E-GAP.

A figura 5.1 mostra a instância do E-GAP discutida nesta seção modelada como uma TS objetiva em TÆMS através da metodologia descrita acima. O nome dos métodos e dos recursos estão sendo usados para identificar com qual dos agentes estes estão relacionados. Na prática utiliza-se o campo *agent* dos métodos e dos recursos em TÆMS para essa identificação.

5.2 Swarm-GAP

O algoritmo Swarm-GAP, apresentado no algoritmo 1, implementa os mecanismos da abordagem proposta necessários para a solução do E-GAP. Os agentes executando o Swarm-GAP se comunicam utilizando mensagens simples, reagindo a dois eventos: a percepção de tarefas e o recebimento de mensagens. Quando um agente percebe algumas tarefas este cria um objeto *taskAlloc* composto por estas tarefas (linhas 4 a 7). Da mesma forma, quando um agente recebe uma mensagem de outro agente (linha 9) este também cria um objeto *taskAlloc*, porém desta vez contendo as tarefas que compõem a mensagem recebida. Uma vez que isso é feito, o agente passa a determinar quais destas tarefas alocar (linhas 11 a 23).

Utilizando o mecanismo discutido na seção 4.3, os agentes analisam primeiramente as tarefas que estes sabem possuir interrelacionamentos AND com outras tarefas. Como definido pelo mecanismo de tomada de decisão, estas tarefas são priorizadas. Para isso, o algoritmo principal aplica a função *TryAllocate()*, descrita no algoritmo 2, em cada tarefa. Através desta função, que implementa o mecanismo discutido na seção 4.2, o agente decide qual tarefa alocar de acordo com sua tendência, restrito pela questão de que sua quantidade de recursos disponível é suficiente para alocá-la (linha 3). A medida que o agente decide alocar uma tarefa, este a atualiza para o estado *allocated* no objeto

taskAlloc (linha 4). A quantidade de recursos que o agente possui é decrementada pela quantidade requerida pela tarefa alocada (linha 5). O algoritmos retorna *true* se a tarefa for alocada ou *false* caso contrário.

Depois disso, os agentes usam a mesma função *TryAllocate()*, algoritmo 2, para tomar suas decisões sobre as tarefas que estes não sabem ainda possuir ou não o interrelacionamentos AND (linha 18). Quando o agente decide alocar uma tarefa, somente nesse caso, este verifica se a tarefa escolhida possui interrelação AND com outras tarefas (linha 19). Se este for o caso, tais tarefas são analisadas na seqüencia (linhas 21 a 23) pelo função *TryAllocate()*, algoritmo 2. Cada uma das tarefas interrelacionadas que não é alocada nesse momento, é incluída em um grupo de tarefas para serem compartilhadas com outros agentes (linha 23).

Os agentes, através do mecanismo apresentado na seção 4.5, podem se adaptar utilizando a função detalhada no algoritmo 3. Para isso, os agentes mantém armazenada a melhor alocação realizada. Esta melhor alocação é atualizada se a recompensa local obtida (w_{ij} na definição do E-GAP) pela nova alocação é maior que a recompensa obtida pela melhor alocação armazenada (linhas 1 e 2). Se o agente está na última de μ rodadas, este diminui seus limiares relacionados as tarefas alocadas na melhor alocação até então obtida (linhas 4 a 6), e aumenta seus limiares para as demais tarefas (linhas 7 a 9). A melhor alocação armazenada é reiniciada neste momento (linha 10).

No último estágio do algoritmo principal, se os agentes ainda possuírem tarefas não alocadas, uma mensagem é então enviada para um dos agentes vizinhos aleatoriamente selecionado (linhas 27 a 33). Este agente não pode ter recebido nenhuma mensagem na rodada atual, para isso este é marcado como visitado (linha 30). O tamanhos destas mensagens é proporcional ao seu número de tarefas.

O Swarm-GAP não lida com a resolução distribuída de conflitos na percepção das tarefas. Como mencionado na seção 1, assume-se que a tarefa pode ser identificada como alocada ou simplesmente que as tarefas não são percebidas simultaneamente pelos agentes. O algoritmo LA-DCOP, utilizado neste capítulo como parâmetro de comparação com o Swarm-GAP, também assume estas premissas.

5.3 Experimentos e Resultados

Os experimentos com o Swarm-GAP foram realizados em um simulador abstrato independente de domínio, escrito em Java, e no simulador da Robocup Rescue (KITANO, 2000; SKINNER; BARLEY, 2006). As simulações realizadas no simulador abstrato foram configuradas exatamente com os mesmos parâmetros usado por SCERRI et al. (2005) para experimentar o LA-DCOP e, de acordo com os autores, diferentes valores para estes parâmetros não afetaram os resultados obtidos por eles. A motivação principal em adotar os mesmos parâmetros que SCERRI et al. (2005) é para que se possa realizar uma comparação adequada entre o LA-DCOP e o Swarm-GAP.

O simulador abstrato permite a experimentação com um grande número de agentes e tarefas. A menos que se explicite o contrário, nos experimentos aqui realizados foram utilizadas 2000 tarefas. Estas tarefas são organizadas em classes e a competência dos agentes é associada a estas classes, e não às tarefas diretamente. Foram adotadas 5 classes diferentes aleatoriamente atribuídas, sendo alocadas por um número de agentes variando entre 500 e 4000 (este último, o dobro do número de tarefas). As tarefas têm custo (recursos que consomem) distribuído uniformemente entre os valores $\{0.25, 0.50, 0.75\}$.

Cada agente tem 60% de probabilidade de ter competência diferente de zero para cada

Algorithm 1 Swarm-GAP($agentId$)

```
1: loop
2:    ev ← waitEvent()
3:    if ev = task perception then
4:       J ← set of new unallocated tasks
5:       taskAlloc ← newTaskAlloc()
6:       for all t ∈ J do
7:          taskAlloc.addTask(t)
8:    else
9:       taskAlloc ← newTaskAlloc(receiveMessage())
10:   {analisa tarefas interelacionadas}
11:   τ_n ← taskAlloc.ANDTasks.getTasks()
12:   for all t ∈ τ_n do
13:      if TryAllocate(taskAlloc, t) then
14:         taskAlloc.ANDTasks.excludeTask(t)
15:   {analisa tarefas não interelacionadas}
16:   τ_a ← taskAlloc.getNoANDTasks()
17:   for all t ∈ τ_a do
18:      if TryAllocate(taskAlloc, t) then
19:         if taskAlloc.isMemberOfAnyANDSet(t) then
20:            τ_g ← taskAlloc.getANDSetTasks(t)
21:            for all t' ∈ τ_g / t' ≠ t do
22:               if !TryAllocate(taskAlloc, t') then
23:                  taskAlloc.ANDTasks.includeTask(t')
24:   {realiza especialização}
25:   Adapt(taskAlloc)
26:   {envia mensagem}
27:   τ ← taskAlloc.getAvaliableTasks()
28:   if |τ| > 0 then
29:      message ← newMessage(taskAlloc)
30:      message.visitedAgent(agentId)
31:      ι ← message.avaiableAgents()
32:      i ← rand(ι)
33:      sendMessage(i)
```

Algorithm 2 TryAllocate($taskAlloc$, t)

```
1: if taskAlloc.avaliable(t) then
2:    r ← avaiableResources()
3:    if roulette() < T_{θ_t(s)} and r ≥ c_t then
4:       taskAlloc.allocateTask(t)
5:       r ← r − c_t
6:       return true
7: return false
```

Algorithm 3 Adapt($taskAlloc$)

1: **if** $taskAlloc$.localReward()> $bestAlloc$.localReward() **then**
2: $\quad bestAlloc \leftarrow taskAlloc$
3: **if** last of μ cycles **then**
4: $\quad \tau_a \leftarrow bestAlloc$.getAllocatedTasks()
5: $\quad$ **for all** $t \in \tau_a$ **do**
6: $\quad\quad \theta_t \leftarrow \theta_t - \xi$
7: $\quad \tau_u \leftarrow bestAlloc$.getAvaliableTasks()
8: $\quad$ **for all** $t \in \tau_u$ **do**
9: $\quad\quad \theta_t \leftarrow \theta_t + \rho$
10: $\quad bestAlloc \leftarrow NULL$

classe anteriormente. Neste caso, as competências são uniformemente distribuídas com valores no intervalo $[0, 1]$. 60% das tarefas são interrelacionadas por restrições AND em conjuntos de 5. Quando uma nova tarefa surge em cada rodada da simulação, esta é aleatoriamente percebida pelos agentes. O número total de tarefas é mantido constante, o que significa que na verdade apenas as características das tarefas mudam (10% das tarefas têm uma probabilidade de 10% de ter sua classe e seu custo (os recursos por ela demandados) alterados.

Em cada rodada da simulação os agentes são limitados a enviar no máximo 20 mensagens cada um. A recompensa total é computada depois de 1000 rodadas da simulação. Uma rodada é definida como sendo o processo onde todos os agentes, de forma síncrona, recebem todas as mensagens direcionadas a eles e enviam todas as suas mensagens. No final de cada rodada uma alocação completa é realizada.

O swarm-GAP é comparado com outros dois algoritmos: o LA-DCOP, mencionado anteriormente neste trabalho; e com um algoritmo guloso (*greedy*). Na estratégia gulosa, adotada como parâmetro de comparação para os outros dois algoritmos mencionados, as tarefas são alocadas aos agentes disponíveis mais competentes de uma maneira centralizada. Todos os dados discutidos neste capítulo são a média de 20 execuções do algoritmo. Em muitos casos o desvio padrão não aparece nos gráficos que ilustram os experimentos pois seu valor é cerca de 1% do valor da média.

Além do simulador abstrato, como comentado no início desta seção, o simulador da Robocup Rescue também foi utilizado. O objetivo da *RoboCup Rescue Simulation League* é oferecer um ambiente de testes, para times de agentes de busca e salvamento atuando em desastres urbanos de larga escala, no formato de uma competição anual. Na *RoboCup Rescue* cada agente tem sua percepção limitada a uma determinada área ao seu redor. Os agentes mantém contato por rádio, mas têm um número limitado de mensagens para trocar. Com isso, como nenhum agente tem informação de todo o estado do ambiente, seu domínio é parcialmente observável por cada agente e pelo conjunto deles (NAIR; TAMBE; MARSELLA, 2003). Isso significa que, mesmo se for considerado o conhecimento de todos os agentes, ainda assim não se tem garantias de informação perfeita.

A eficiência dos agentes nesse ambiente só pode ser obtida através de um adequado processo de coordenação. Abordagens fortemente baseadas em comunicação não são apropriadas nesse caso dada a severa restrição imposta para a comunicação entre os agentes. De acordo com NAIR et al. (2002), os desafios impostos pelo simulador em questão podem ser decompostos em problemas de alocação de tarefas. O subproblema de alocação considerado neste trabalho trata da alocação eficiente de agentes, representando brigadas

de incêndio, às tarefas de extinguir focos de incêndio.

O Swarm-GAP também é comparado com o LA-DCOP neste caso, e ambos os algoritmos são comparados com um algoritmo centralizado que utiliza uma heurística gulosa baseada na distância mais curta, similar ao usado por muitos times da liga.

Os experimentos conduzidos neste capítulo são discutidos como segue: primeiramente, na seção 5.3.1 é empiricamente determinado o estímulo s que maximiza a recompensa total do sistema de acordo com as diferentes características dos cenários; depois disso, na seção 5.3.2, o mecanismo de especialização do limiar interno proposto neste trabalho é explorado, onde procura-se mostrar como este funciona na prática, seu desempenho e limitações; na seqüência, as seções 5.3.3 e 5.3.4, mostram o desempenho do Swarm-GAP enfrentando situações difíceis: quando as tarefas têm interrelação AND entre si e quando são impostas restrições mais severas na comunicação entre os agentes; finalmente, na seção 5.3.5, o Swarm-GAP é comparado ao LA-DCOP nos dois simuladores comentados há pouco.

5.3.1 Estímulo Ótimo

Alguns problemas, como o RCPSP discutido no capítulo 6, impõem prioridade ou interdependência na alocação de tarefas, onde uma tarefa deve ser precedida ou preceder outras tarefas. O E-GAP, diferentemente, não considera esta possibilidade. Por esta razão, o estímulo s_{ij} usado pelo mecanismo discutido na seção 4.2 é o mesmo para todos os agentes e para todas as tarefas do cenário. O limiar interno θ_{ij} de cada agente é configurado utilizando o mecanismo de polimorfismo, discutido na seção 4.4, de acordo com a competência de cada agente para as classes de tarefas. As questões que se pretende responder aqui são: qual é o estímulo ótimo que maximiza a recompensa total do E-GAP (dada pela equação 5.4) na abordagem proposta? Este valor para o estímulo é influenciado pelas características dos cenários? É possível capturar esta influência na abordagem proposta?

Como é mostrado aqui, este estímulo (chamado a partir de agora de s apenas) influencia a recompensa total obtida pelo sistema de acordo com a proporção entre os agentes e as tarefas, bem como de acordo com os diferentes níveis de distribuição na percepção dos agentes. Os agentes podem perceber as tarefas de três diferentes maneiras:

- Distribuída, com os agentes percebendo um pequeno subconjunto de todas as tarefas disponíveis, em torno de 5%;
- Centralizada, com todos os agentes percebendo todas as tarefas;
- Parcialmente-distribuída, com todos os agentes percebendo um subconjunto de tarefas, com aproximadamente 20% de todas as tarefas disponíveis, de uma maneira distribuída (em uma eventual maneira parcialmente-centralizada um subconjunto de agentes perceberia todas as tarefas).

O E-GAP, por definição, permite que apenas um agente aloque cada tarefa. Contudo, os subconjuntos de tarefas podem se sobrepor, com agentes percebendo as mesmas tarefas. Assume-se aqui que a percepção simultânea de tarefas é possível e que o primeiro agente que decidir alocar uma determinada tarefa o fará, marcando esta tarefa como desabilitada para os demais. O Swarm-GAP, como dito anteriormente, não lida com a resolução distribuída de conflitos na percepção dos agentes, por isso esta simplificação precisa ser adotada.

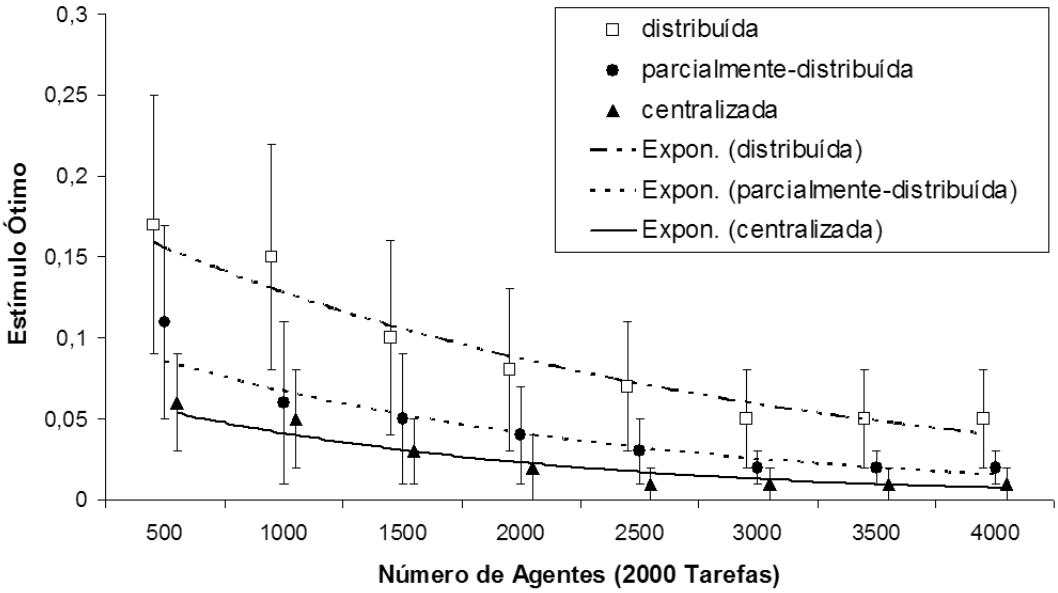


Figura 5.2: Comparando o estímulo ótimo de acordo com diferentes quantidades de agentes para as percepções distribuída, centralizada e parcialmente-distribuída.

Para determinar o estímulo que conduz as melhores alocações dada a influência da configuração dos cenários (denominado de agora em diante de "estímulo ótimo"), são experimentados valores para o estímulo s no intervalo $[0.01, 0.2]$. Valores para o estímulo abaixo de 0.01 levam a uma tendência muito baixa para alocar tarefas (menor que 1%), considerando os valores que o limiar interno pode assumir. Da mesma forma, valores para o estímulo maiores que 0.2 representam tendências que, dependendo do limiar interno, podem chegar a valores muito altos (maiores que 80%). Estes limites para a tendência motivaram o intervalo escolhido para a experimentação do estímulo. São utilizados diferentes quantidades de agentes (de 500 a 4000) para experimentar as diferentes proporções entre agentes e tarefas, cujo número é mantido constante e igual a 2000. O estímulo ótimo, de acordo este número dos agentes, para cada uma das maneiras de percepção caracterizadas anteriormente, é mostrado na figura 5.2 (pontos identificados segundo cada maneira de percepção - "distribuída", "parcialmente-distribuída" e "centralizada").

Como se pode ver, a medida que o número de agentes aumenta, o estímulo que maximiza a recompensa total diminui. Por exemplo, na maneira distribuída de percepção, a maior recompensa para 500 agentes é obtida com $s = 0.17$, para 2000 agentes com $s = 0.08$ e para 4000 agentes com $s = 0.05$. O mesmo ocorre para todas as maneiras diferentes de percepção. Comparando a curva de cada maneira de percepção pode-se ver que as diferenças entre a percepção também causa impacto sobre o estímulo ótimo. Na percepção centralizada, por exemplo, o estímulo ótimo para 500 agentes é $s = 0.05$, para 2000 é $s = 0.02$, e para 4000 é $s = 0.01$. Estes valores são bem diferentes dos valores ótimos para as percepções distribuída e parcialmente-distribuída.

No gráfico da figura 5.2 estão plotadas curvas de tendência exponenciais para os estímulos ótimos determinadas por regressão não linear usando o método dos mínimos quadrados, denominadas "Expon. ({distribída, parcialmente-distribída, centralizada})". É possível ver que estas curvas se encontram dentro do intervalo do desvio padrão dos valores para todas as diferentes maneiras de percepção. Outros tipos de linhas de tendência, por exemplo determinadas por regressão linear, não apresentam esse comportamento.

Esta análise levou a conclusão de que o estímulo ótimo varia exponencialmente de acordo com o número de agentes para todas as maneiras de percepção.

A equação 5.5 foi determinada através do método dos mínimos quadrados e captura a relação entre o estímulo ótimo de acordo com os fatores em questão. Esta equação é utilizada doravante para calcular o estímulo ótimo dada a proporção x. A constante λ captura as diferentes maneiras na percepção dos agentes, sendo $\lambda = 0.23$ para a percepção distribuída (5% das tarefas percebidas simultaneamente por agente), $\lambda = 0.19$ para a percepção parcialmente-distribuída (20% das tarefas por agente), e $\lambda = 0.07$ para a percepção centralizada (100% das tarefas por agente). Cabe ressaltar que a equação 5.5 foi obtida com base nos experimentos com a configuração apresentada no início da seção 5.3. Outras configurações podem implicar em uma variação do parâmetro λ. É conveniente, neste caso, determinar empiricamente o estímulo ótimo e verificar a adequação dos parâmetros dessa equação.

$$s = \lambda e^{-x} \tag{5.5}$$

tal que

$$x = \frac{|\mathcal{I}|}{|\mathcal{J}|}$$

onde $|\mathcal{J}|$ é número de tarefas e $|\mathcal{I}|$ é número de agentes

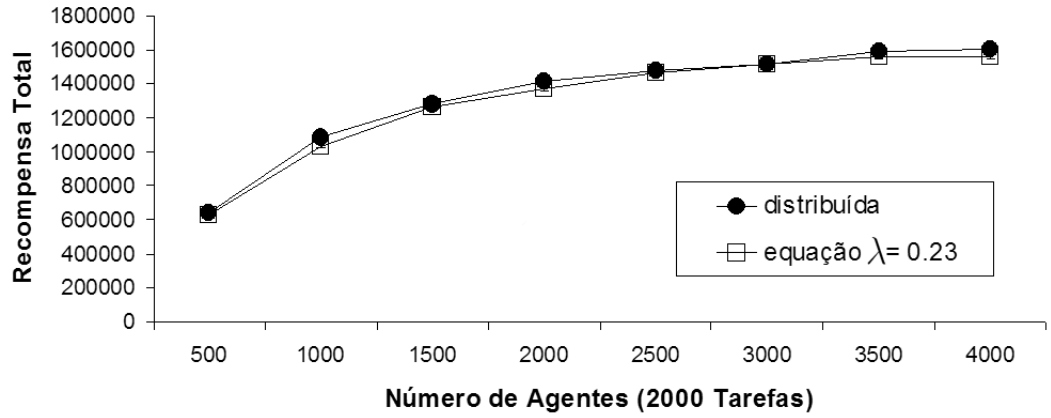
A figura 5.3 mostra as curvas para a recompensa total obtida pelo estímulo calculado pela equação **??** e pelo estímulo ótimo empiricamente determinado, para cada maneira de percepção. Como se pode ver nos gráficos 5.3(a), 5.3(b) e 5.3(c), a recompensa total obtida por ambos estímulos, para todas as maneiras de percepção, são equivalentes.

5.3.2 Especialização

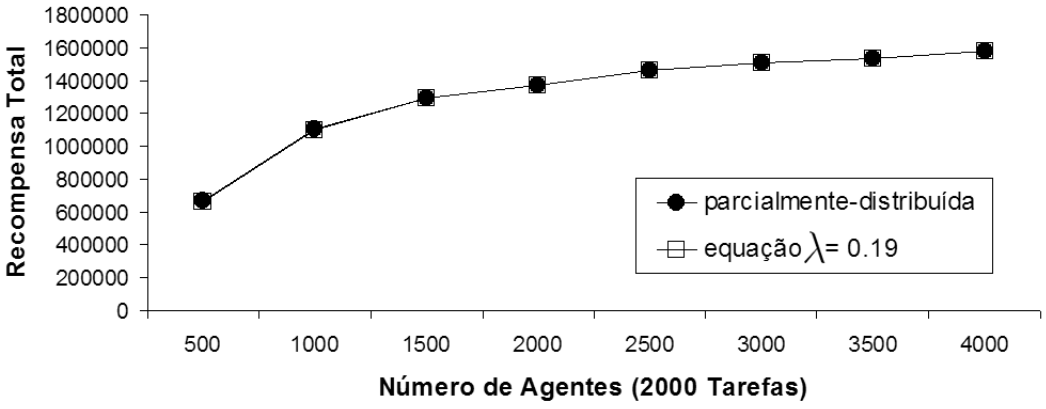
O Swarm-GAP permite que os agentes adaptem seus limiares internos para alocar tarefas de classes desconhecidas, mantendo o desempenho do sistema. Classes desconhecidas são aquelas para as quais os agentes não conhecem suas competências. O mecanismo de especialização, apresentado na seção 4.5, permite que os agentes aumentem ou diminuam tais limiares de acordo com a recompensa local que obtém. Esta recompensa local é calculada pelos agentes como o somatório de suas competências para as tarefas alocadas em determinado momento.

A cada μ rodadas, os coeficientes ξ e ρ são utilizados, respectivamente, para aumentar e diminuir os limiares internos das tarefas alocadas na melhor alocação local, ou seja, aquela com a maior recompensa local obtida neste período. É importante que os agentes sejam capazes de determinar essa recompensa local ou recebam informação equivalente a respeito do seu desempenho nas alocações. Assume-se que todos os agentes são capazes de se engajar na alocação de qualquer tarefa desconhecida. Quando um agente não é capaz de alocar uma tarefa a recompensa obtida é nula (zero), o que para o sistema representa um resultado equivalente a não ter nenhum agente alocando esta tarefa.

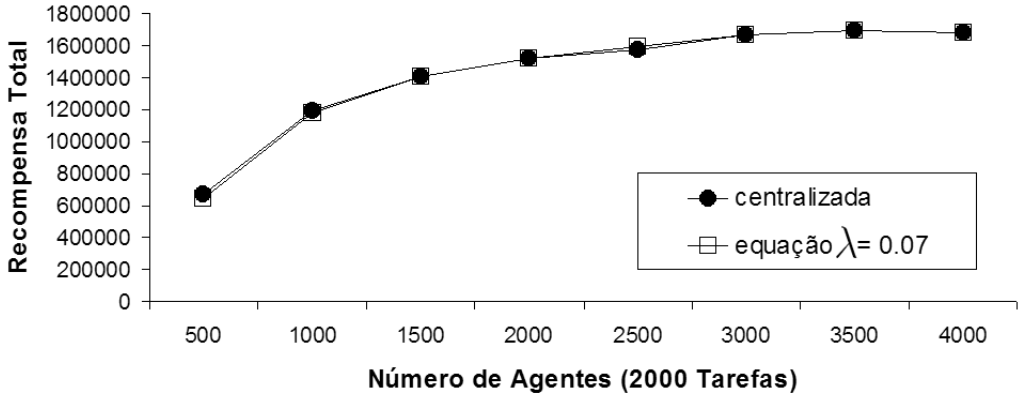
Nesta seção pretende-se explorar a aplicabilidade do mecanismo de especialização. Inicialmente é discutido como a especialização ocorre na prática para permitir que agentes adaptem-se às tarefas de classes desconhecidas. Em seguida, é mostrado como os agentes que conhecem sua competência, através do mecanismo de polimorfismo (seção 4.4), podem usar a adaptação para aumentar seu desempenho. Finalmente, o desempenho do Swarm-GAP utilizando a adaptação é analisado e as limitações desse mecanismo são discutidas.



(a)



(b)



(c)

Figura 5.3: Comparando a recompensa total obtida pelo estímulo ótimo determinado empiricamente e o estímulo ótimo calculado pela equação 5.5 de acordo com a percepção distribuída (a), parcialmente-distribuída (b) e centralizada (c), para diferentes quantidades de agentes

5.3.2.1 Partida a Frio

Nos experimentos conduzidos nesta seção os agentes não conhecem sua competência para alocar as tarefas do cenário. Esta situação é extrema e improvável em aplicações reais. Contudo, este é um cenário interessante para analisar e validar o mecanismo de especialização. A idéia é verificar como agentes que têm tendência máxima para alocar todas as tarefas se comportam e, a partir disso, modificar seus limiares através de um mecanismo de especialização por recompensa (um rudimentar aprendizado por reforço). Com isso busca-se obter resultados equivalentes aqueles que seriam obtidos se o comportamento dos agentes fosse orientado por suas competências.

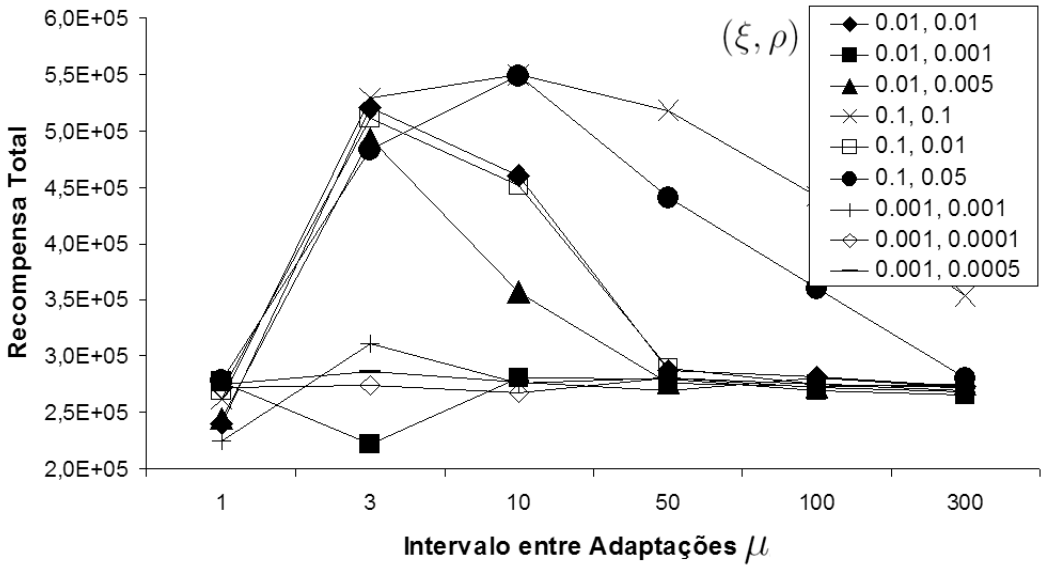
Os limiares internos dos agentes são configurados inicialmente com valor 0 ($\theta_{ij} = 0$). O que significa que os agentes têm tendência máxima para alocar todas as tarefas do cenário. Esta configuração é denominada "partida a frio". Cada agente alocará todas as tarefas que perceber, restrito apenas pela quantidade de recursos disponíveis que possuir.

Para medir o impacto dos diferentes valores que os parâmetros da especialização podem assumir (ξ, ρ e μ) na eficiência deste processo, foram experimentadas diferentes combinações de diferentes pares (ξ, ρ) de acordo com diferentes intervalos de adaptação μ. A figura 5.4(a) mostra a recompensa total obtida por 500 agentes alocando 2000 tarefas, onde as melhores combinações dos parâmetros mencionados são $(0.1, 0.1)$ e $(0.1, 0.05)$ para as constantes de especialização e $\mu = 10$ para o intervalo de adaptação. Estendendo este experimento para um número maior de agentes (1000, 2000, 3000 and 4000), como mostra a figura 5.4(b), pode-se ver que o par $(0.1, 0.05)$ tem melhor desempenho em cada uma das demais quantidades de agentes e na média entre estes casos.

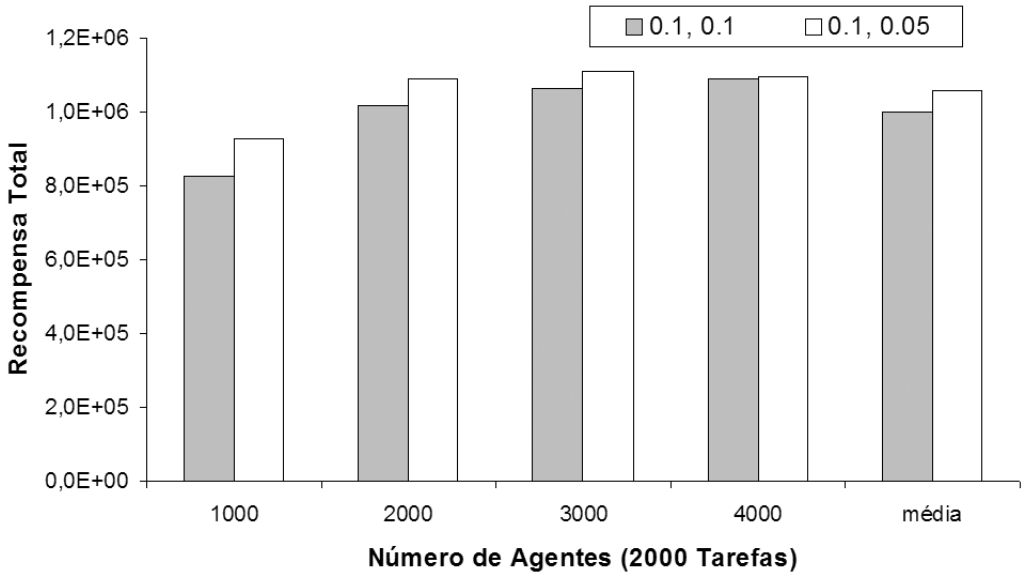
O processo de adaptação é mostrado em detalhes na figura 5.5. No pior caso (500 agentes), a recompensa parcial obtida em cada rodada da simulação aumenta a medida que os agentes alocam as tarefas nas primeiras 200 rodadas da simulação. Depois disso, a recompensa parcial converge para um comportamento estável. Os limiares internos atingem seus limites (valores no intervalo $[0, 1[$) e cada agente se especializa na classe de tarefas para a qual possui maior competência (que será denominada apenas de "melhor classe" daqui por diante). Os limiares internos não assumem o valor 1, o que é essencial em cenários dinâmicos onde tarefas novas podem surgir a qualquer momento, fazendo com que os agentes precisem se readaptar à nova situação. O número de tarefas alocadas durante o processo de adaptação e o total de recursos empregados não muda significativamente, como mostra a figura 5.6.

A figura 5.7 mostra o número de agentes especializados em cada classe de tarefas de acordo com suas competências. Os agentes estão organizados de acordo com as classes em que estes estão especializados e a competência que estes possuem relacionada a cada uma destas classes. No final de cada μ rodadas da simulação uma determinada quantidade de agentes se especializa na sua melhor classe, outra quantidade na sua segunda melhor classe, e assim por diante. A figura em questão mostra os três primeiros intervalos de μ rodadas (rodadas 10, 20 e 30) e a ultima rodada da simulação (1000). Como se pode observar, 500 agentes se especializam de maneira bastante eficiente. No final da simulação mais de 95% dos agentes se especializaram na sua melhor classe. Como esperado, com mais de 2000 agentes a adaptação perde desempenho: 75% de 2000 agentes e 50% de 4000 agentes se especializam na sua melhor classe.

Contudo, em todos os casos se pode verificar uma queda significativa no número de agentes especializados nas classes em que estes não têm nenhuma competência, o que também aumenta o desempenho da especialização. Como dito no início deste capítulo, cada agente tem 60% de probabilidade de ter competência maior que zero para cada



(a)



(b)

Figura 5.4: Comparando os diferentes valores para os parâmetros da especialização (ξ, ρ e μ). A figura (a) mostra a recompensa total obtida por 500 agentes alocando 2000 tarefas. A figura (b) mostra a recompensa total obtida para $(\xi, \rho) = \{(0.1, 0.1), (0.1, 0.05)\}$ e $\mu = 10$ de acordo com um número maior de agentes e a média entre estes.

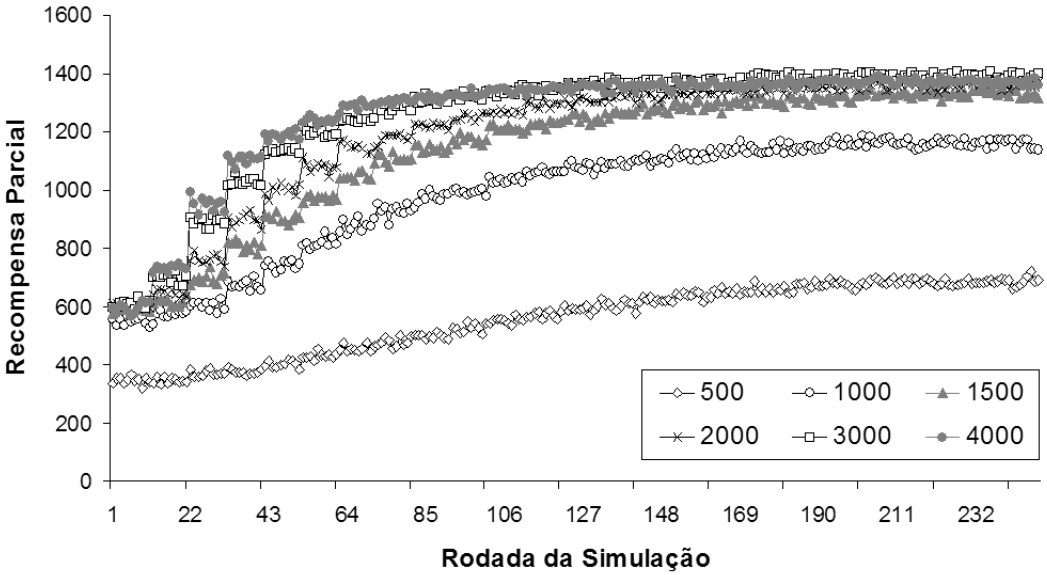


Figura 5.5: Analisando o desempenho para a partida a frio de acordo com a recompensa parcial obtida em cada rodada da simulação de acordo com diferentes quantidades de agentes.

classe. Em outras palavras, com alguma probabilidade, os agentes podem não ser competentes para alocar tarefas de uma ou mais classes. Dado que os agentes não conhecem sua competência no início do processo de adaptação, estes podem temporariamente se especializar em classes para as quais não têm nenhuma competência.

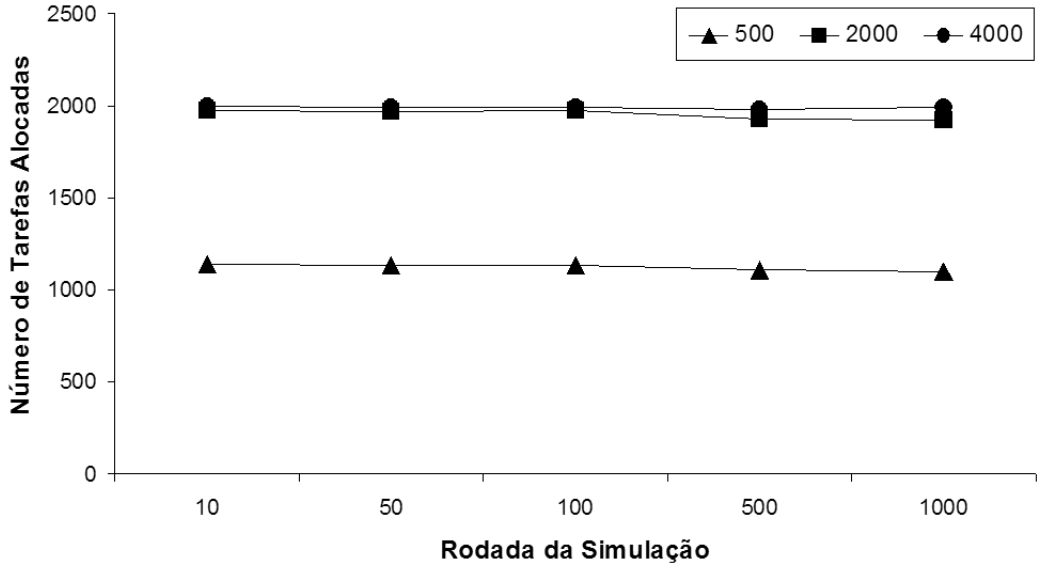
O mecanismo de especialização tem que ser flexível o suficiente para lidar com tarefas de classes desconhecidas que eventualmente passem a fazer parte do sistema a qualquer momento. Esta hipótese foi experimentada através da introdução de um conjunto de tarefas de uma classe desconhecida pelo sistema numa rodada durante a simulação. A figura 5.8 mostra as recompensas parciais em cada rodada da simulação para diferentes quantidades de agentes. Os pontos e as barras de erro foram omitidos nesta figura para manter sua clareza. O desvio padrão máximo da recompensa parcial ocorrido nas curvas mostradas é em torno de 14.

As novas tarefas são introduzidas na rodada 500 da simulação. O processo de adaptação dos limiares internos inicialmente estabiliza por volta da rodada 200 para todos os casos. Como se pode ver, quando as novas tarefas aparecem no cenário, o processo de adaptação reinicia e são necessárias mais 200 rodadas para que tudo se estabilize novamente. Uma conclusão importante é que, mesmo quando os limiares internos estão especializados, é possível que estes voltem a se adaptar para lidar com situações novas e imprevisíveis.

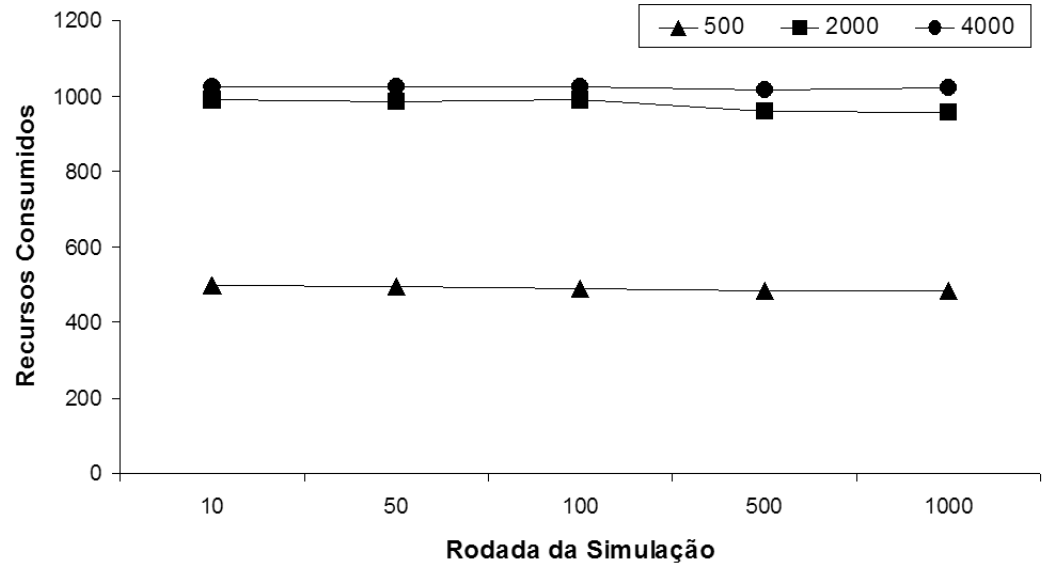
5.3.2.2 Partida com o Polimorfismo

Nesta seção foi conduzido um experimento para explorar a especialização de agentes que possuem seus limiares internos refletindo suas competências, através do mecanismo de polimorfismo (seção 4.4), que iniciam um processo de adaptação idêntico ao da partida a frio, discutido na seção anterior.

A figura 5.9 mostra a recompensa parcial para cada rodada da simulação com a espe-

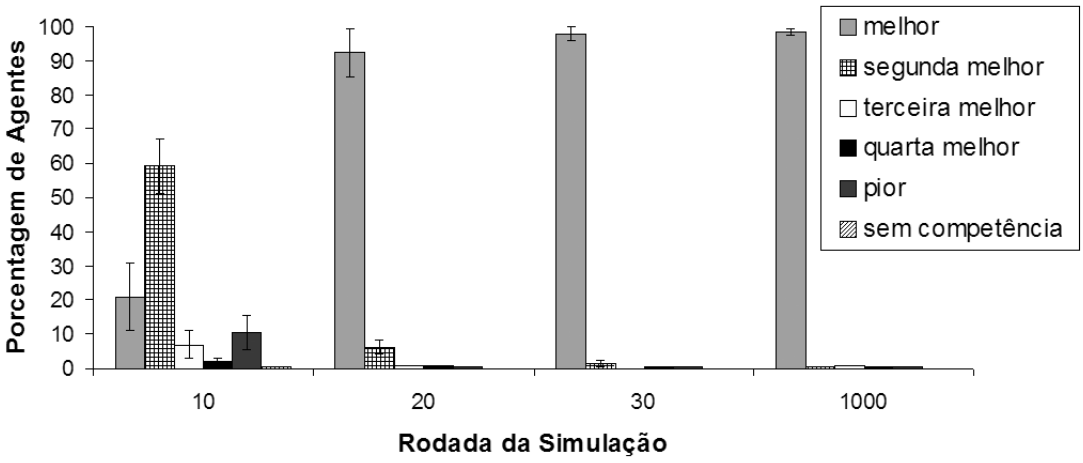


(a)

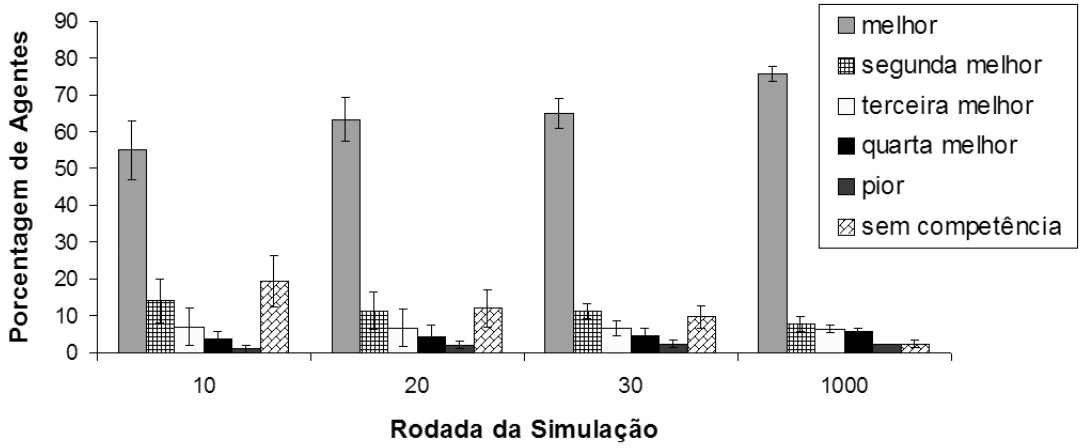


(b)

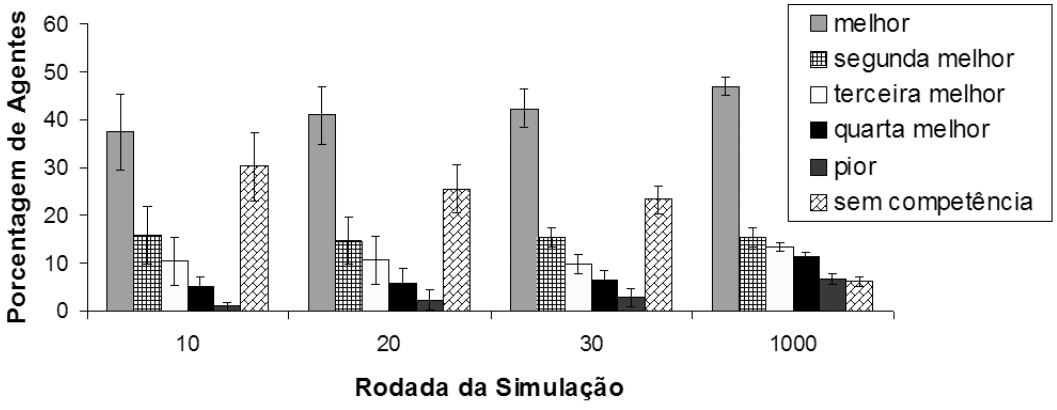
Figura 5.6: Analisando o desempenho para a partida a frio de acordo com as tarefas alocadas (a) e os recursos consumidos (b) em rodadas arbitrárias da simulação (10, 50, 100, 500, 1000), para diferentes quantidades de agentes.



(a)



(b)



(c)

Figura 5.7: Analisando a especialização dos limiares nas classes de tarefas. As figuras (a), (b) e (c) mostram o número de agentes especializados em cada classe de acordo com suas competências para alocar tarefas destas classes, para 500, 2000 e 4000 agentes, respectivamente.

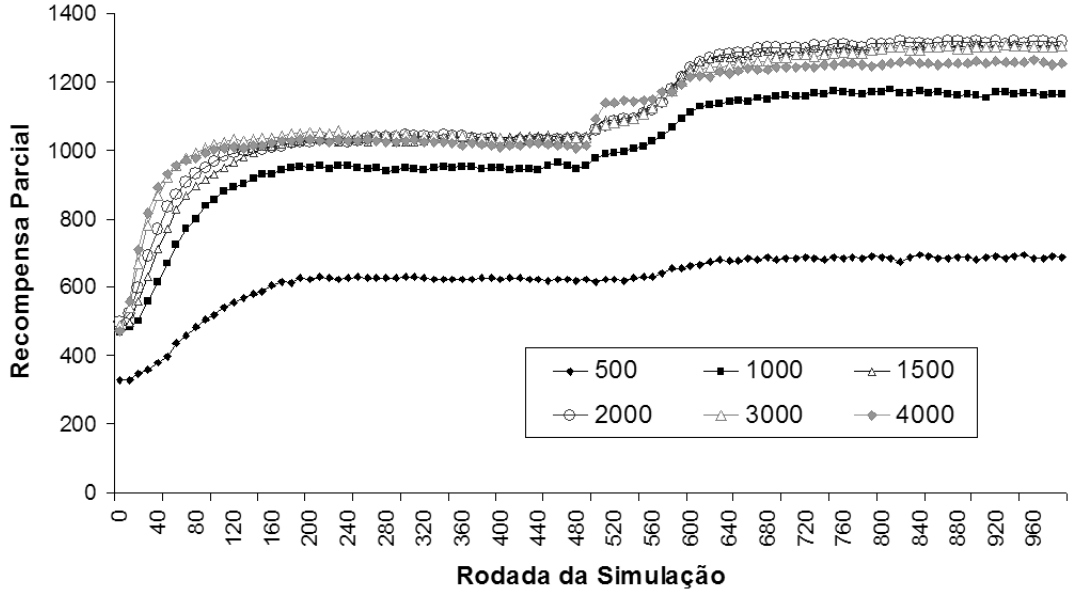


Figura 5.8: Comparando a recompensa parcial em cada rodada da simulação de acordo com diferentes quantidades de agentes.

cialização de agentes polimórficos, variando a quantidade de agentes. É possível verificar uma melhora na recompensa parcial nas primeiras rodadas da simulação, que passa a estabilizar em determinado momento para 500 e 1000 agentes. Porém, acima de 1500 agentes esta melhora cessa em um determinado patamar e começa a cair, estabilizando em uma recompensa parcial igual ou pior em relação às rodadas iniciais. A especialização perde desempenho quando o número de agentes ultrapassa o número de tarefas, sempre convergindo em recompensas parciais piores do que quando os agentes utilizam o polimorfismo sem esta adaptação.

Assim, quando os agentes conhecem suas competências (sendo polimórficos), a especialização tem real benefício para o desempenho do sistema como um todo, quando a quantidade de agentes por tarefa é 0.5 (o dobro de tarefas), somente no início do processo de especialização. Com isso, é necessário pausar tal processo quando a recompensa parcial atinge seu valor máximo. A adaptação pode voltar a acontecer a qualquer momento a medida que novas tarefas aparecem. Contudo, se o sistema é extremamente dinâmico, com várias mudanças ocorrendo o tempo todo, esta pausa não é útil e esta limitação é intratável.

5.3.2.3 Desempenho

O desempenho geral do mecanismo de especialização mostra sua relevância, como pode-se ver na figura 5.10. Quando os limiares internos não são alterados (curva *Threshold 0*), a recompensa total é drasticamente menor que a obtida usando o polimorfismo (sem a especialização). Contudo, quando os limiares são adaptados com a partida a frio (curva *Especialização - Partida a Frio*), a recompensa total aumenta para valores próximos aos obtido pelo polimorfismo sem a especialização. Além disso, a curva *Especialização (pausa) - Polimorfismo* mostra a melhora de desempenho que se pode obter ao especializar agentes polimórficos com a pausa estratégica comentada no final da seção 5.3.2.2. O processo de adaptação, neste último caso, obtém uma recompensa total

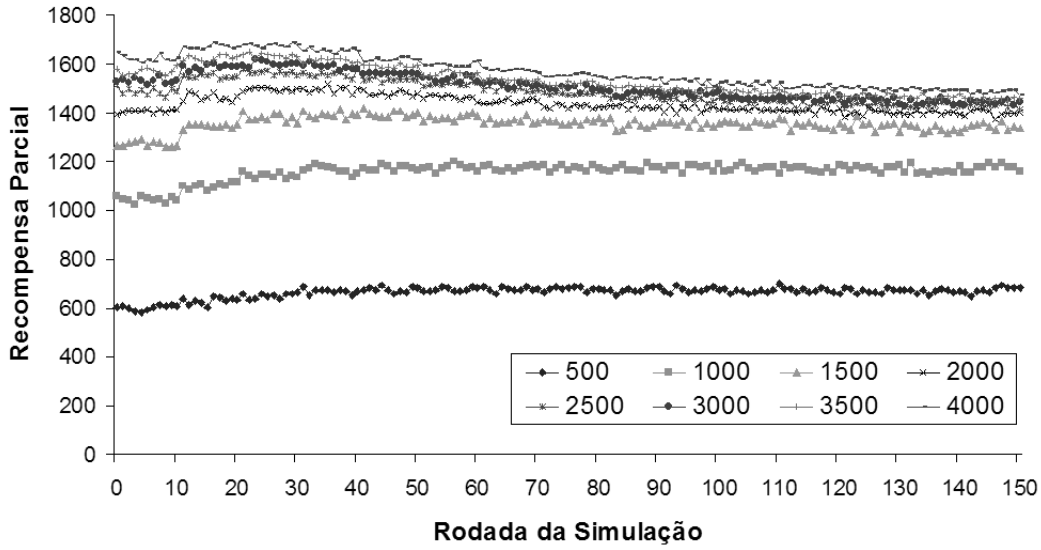


Figura 5.9: Comparando a recompensa parcial obtida pela especialização de agentes polimórficos a cada rodada, para diferentes quantidades de agentes.

superior ao polimorfismo sem especialização.

5.3.2.4 Limitações

A figura 5.10 mostra que há diferenças significativas na recompensa total obtida para mais de 2000 agentes, onde o polimorfismo é melhor que a especialização partindo a frio e partindo com o polimorfismo (sem a pausa estratégica). Isto significa que a especialização tem limitações quando o número de agentes ultrapassa o número de tarefas. Conforme a proporção entre agentes e tarefas aumenta, passam a sobrar um número pequeno de tarefas para serem alocadas pelo agentes.

Pelo mesmo motivo, quando se aumenta o número de classes o desempenho da especialização cai drasticamente. A figura 5.11 mostra a recompensa total obtida pelo mecanismo de polimorfismo e pelo mecanismo de especialização (partindo a frio) para quantidades de classes diferentes. Quando as 2000 tarefas são divididas em 100 classes o desempenho da especialização cai drasticamente e, considerando 500 classes, a especialização é absolutamente ineficiente.

De maneira geral, agentes que se especializam através de mecanismos como o que está sendo proposto neste trabalho, precisam experimentar a alocação de um razoável número de diferentes tarefas por um determinado tempo para que a adaptação surta efeito. Ao se reduzir a possibilidade de experimentação que os agentes necessitam para especializarem-se, há uma inevitável perda na qualidade da alocação obtida. Com isso, a quantidade alta de classes, um número maior de agentes que de tarefas, e o dinamismo do cenário, com tarefas aparecendo ou mudando de características, interferem de forma crítica no processo de adaptação orientado por recompensas. Este comportamento é esperado e foi documentado diversas vezes antes na literatura, onde deve-se destacar um recente trabalho a respeito de simulação social (ZOETHOUT; JAGER; MOLLEMAN, 2008).

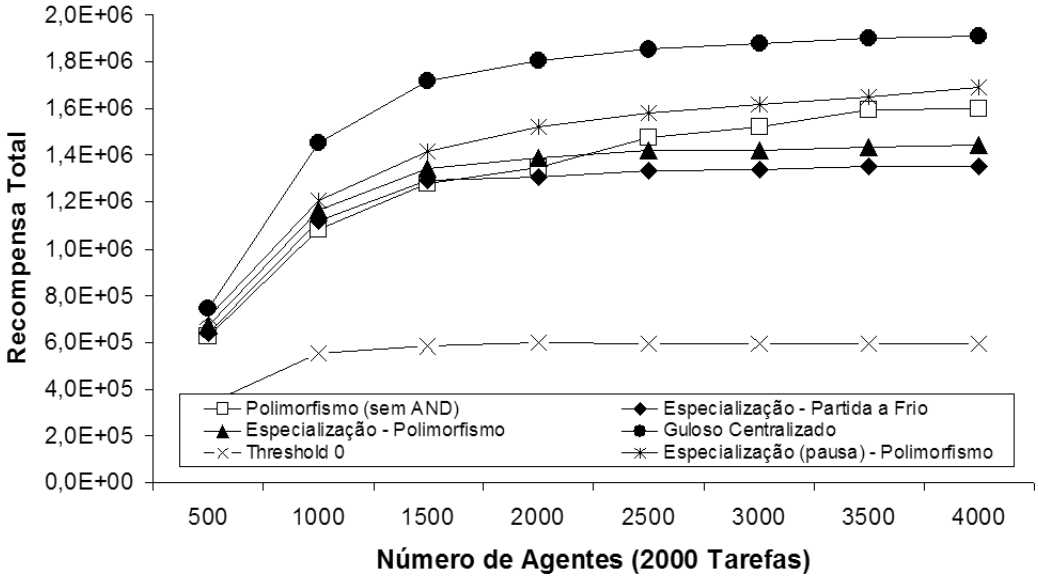


Figura 5.10: Analisando o desempenho da adaptação através da comparação entre uma abordagem gulosa centralizada, limiares internos fixos em zero, o polimorfismo e a adaptação partindo do polimorfismo, partindo do polimorfismo com a pausa estratégica e partindo a frio.

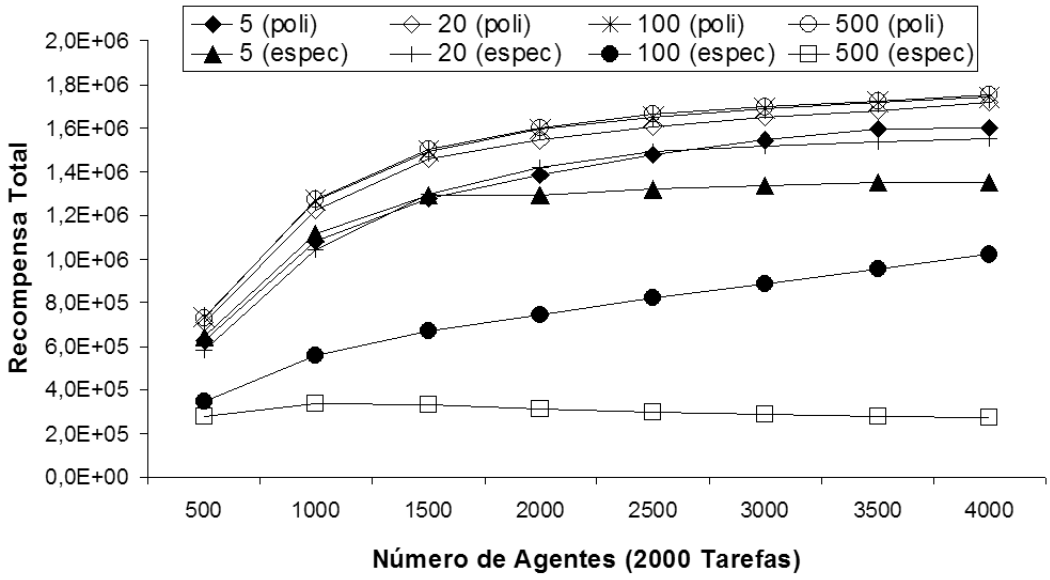


Figura 5.11: Comparando a recompensa total obtida pelo polimorfismo e pela especialização (partindo a frio) para diferentes quantidades de classes.

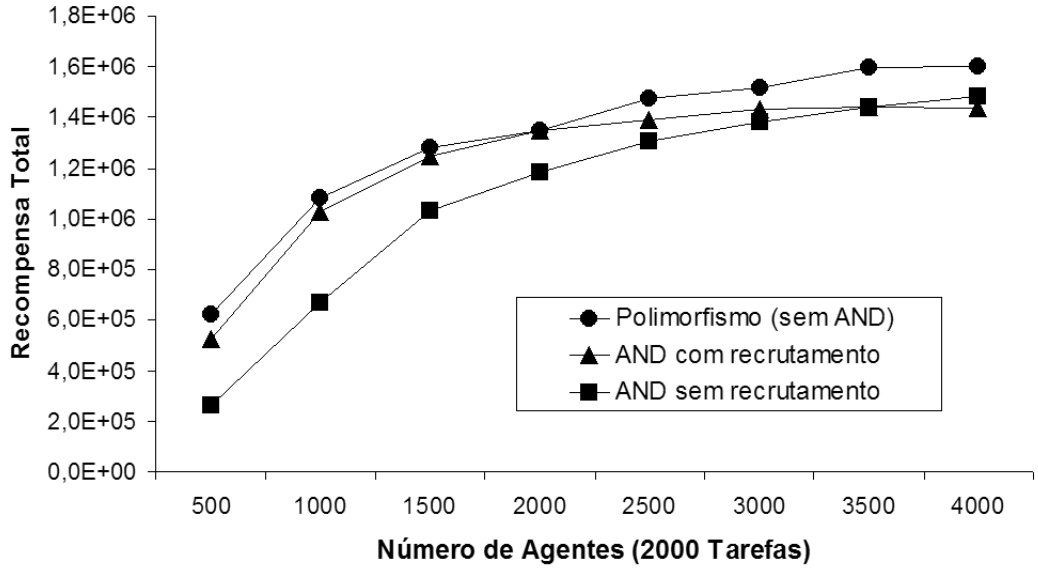


Figura 5.12: Comparando a recompensa total na presença de interrelações AND com e sem a utilização de recrutamento, de acordo com diferentes quantidades de agentes.

5.3.3 Interrelacionamento entre Tarefas

A interrelação entre tarefas do tipo AND constringe os agentes a alocar todas as tarefas de um mesmo grupo para que seja obtida a recompensa pela alocação individual destas tarefas. A figura 5.12 (curva *AND sem recrutamento*) mostra a queda no desempenho do sistema na presença deste tipo de interrelação.

A abordagem proposta neste trabalho imita o processo de recrutamento dos insetos sociais. Como explicado na seção 4.3.1, um agente é capaz de enviar uma mensagem aos vizinhos para que estes priorizem a alocação das tarefas que se interrelacionam com outras que ele tenha alocado. A curva *AND com recrutamento*, da figura 5.12, mostra o aumento na recompensa total quando o mecanismo de recrutamento é utilizado, atingindo valores próximos a quando não existem as interrelações AND.

Quando a quantidade de agentes ultrapassa o número de tarefas, o processo de recrutamento gradualmente perde eficiência. Somente quando o número de agentes ultrapassa o dobro do número de tarefas esse processo realmente falha, obtendo resultados equivalentes a quando este não é utilizado. Quando um agente recruta um vizinho, este adiciona tarefas para serem analisadas antes das tarefas percebidas, aumentando as chances de que estas sejam alocadas. Conforme o número de agentes aumenta, passa a existir uma alta probabilidade de um agente decidir alocar mais tarefas influenciado por seus vizinhos e menos tarefas para as quais sua competência é a melhor, podendo haver agentes mais competentes que ele para tais tarefas.

5.3.4 Restrição na Comunicação

Agentes executando o Swarm-GAP trocam mensagens para lidar com os interrelacionamentos AND e para aumentar sua percepção. No experimento realizado nesta seção, são introduzidas falhas no canal de comunicação entre os agentes. A figura 5.13 mostra a recompensa total obtida pelo Swarm-GAP quando o canal de comunicação entre os agentes é restrito a 25% da sua capacidade normal. Com isso, os agentes passam a trocar

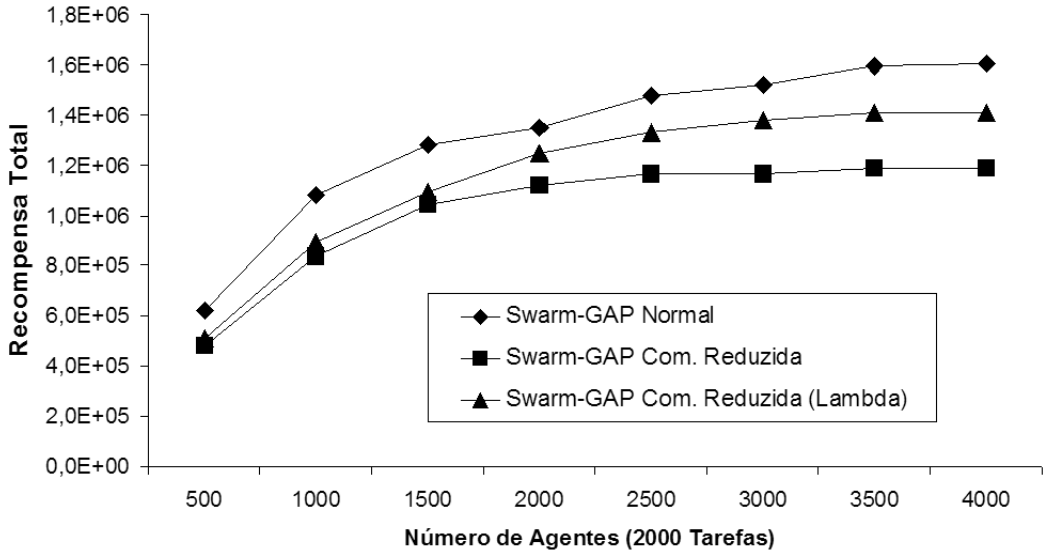


Figura 5.13: Comparando a recompensa total sob restrições de comunicação, para diferentes quantidades de agentes.

5 mensagens por rodada da simulação, ao invés das 20 usuais. A figura 5.13 mostra a redução na recompensa total quando esta restrição é (curva *Com. Reduzida*) e quando não é (curva *Normal*) imposta. De certa forma, restringir o canal de comunicação entre os agentes equivale a reduzir sua percepção. Assim, esta perda de qualidade é esperada.

Contudo, é possível minimizar este efeito aumentado a constante λ da equação 5.5. Como mencionado anteriormente, conforme varia a maneira que os agentes percebem as tarefas, é possível ajustar a equação 5.5 para aumentar a recompensa total. Com isso, a constante λ foi modificada para $\lambda = 0.4$, o que é o dobro de seu valor normal. A curva *Com. Reduzida (Lambda)*, da figura 5.13, mostra a eficiência desta modificação. O Swarm-GAP consegue lidar com falhas nos canais de comunicação pois, como se pode ver, a abordagem proposta apresenta uma queda na recompensa total obtida de maneira geral, mas mantendo um desempenho razoável.

5.3.5 Swarm-GAP x LA-DCOP

Os agentes que executam o Swarm-GAP nos experimentos descritos em toda esta seção têm uma percepção disjunta das tarefas, o que significa que cada uma é percebida por um agente apenas. Assume-se esta simplificação aqui para que seja possível uma comparação adequada com o LA-DCOP, que também a considera.

5.3.5.1 Complexidade de Tempo

Como comentado na seção 2.4.2, o LA-DCOP utiliza um algoritmo para a solução de um Problema da Mochila Binário (PMB). Os agentes executando o LA-DCOP utilizam o *threshold* para selecionar quais tarefas, entre as pecebidas, estes pretendem alocar em função de suas competências (vide seção 2.4.2). Dada a restrição de recursos que cada agente possui, é preciso então definir quais tarefas, dentre essas que foram previamente selecionadas, serão de fato alocadas. O LA-DCOP busca a otimização do emprego dos recursos naquelas tarefas para as quais o agente é mais competente. Este problema pode

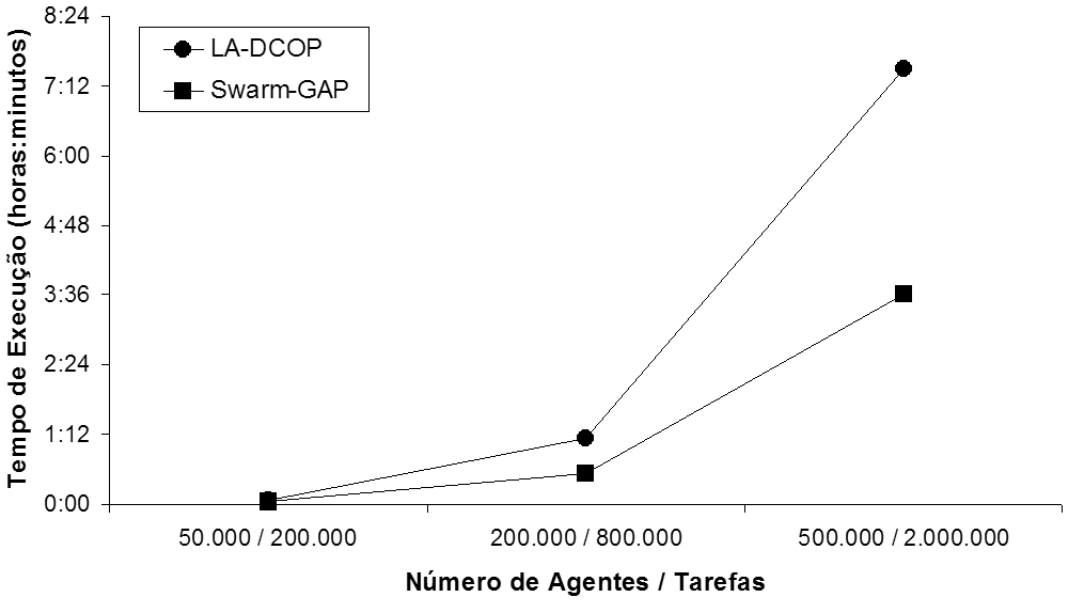


Figura 5.14: Comparando o tempo de execução do Swarm-GAP e do LA-DCOP sendo executados em um PC comum.

ser visto como um PMB. Um algoritmo baseado em programação dinâmica para este tipo de problema tem complexidade de tempo na ordem O(n^2), onde n é o número de tarefas. Como se pode perceber, a medida que mais tarefas são selecionadas, por corresponderem a competências maiores que o *threshold*, maior será o n.

O Swarm-GAP tem vantagens em relação à complexidade de tempo se comparado com o LA-DCOP pois seu processo de tomada de decisão é bem mais simples (vide seção 4.3). Os agentes apenas percorrem aleatoriamente a lista de tarefas percebidas e analisam cada uma delas utilizando a equação da tendência (equação 4.1). Este processo tem complexidade de tempo na ordem de O(n), onde n é o número de tarefas percebidas.

Para comparar o desempenho dos algoritmos Swarm-GAP e LA-DCOP de maneira prática optou-se por simular o LA-DCOP na pior situação mencionada anteriormente, ou seja, utilizando o *threshold* igual a zero. A figura 5.14 mostra o tempo de execução (em horas e minutos) consumidos pelos algoritmos em questão sendo executados em um PC comum e utilizando o simulador abstrato descrito anteriormente. São mostrados os resultados para 50.000 agentes com 200.000 tarefas, 200.000 agentes com 800.000 tarefas, e 500.000 agentes com $2.000.000$ tarefas. Como esperado, o tempo consumido pelo LA-DCOP é significativamente maior que o consumido pelo Swarm-GAP.

Apesar de ter complexidade de tempo superior ao LA-DCOP, o Swarm-GAP tem um bom desempenho. Como será visto a seguir, o Swarm-GAP obtém recompensas totais que são comparáveis as obtidas pelo LA-DCOP.

5.3.5.2 Simulador Abstrato

No primeiro experimento desta seção os resultados obtidos pelo Swarm-GAP são comparados com uma abordagem gulosa centralizada e com o LA-DCOP. A figura 5.15 mostra a recompensa total, para diferentes quantidades de agentes, obtida pelo Swarm-

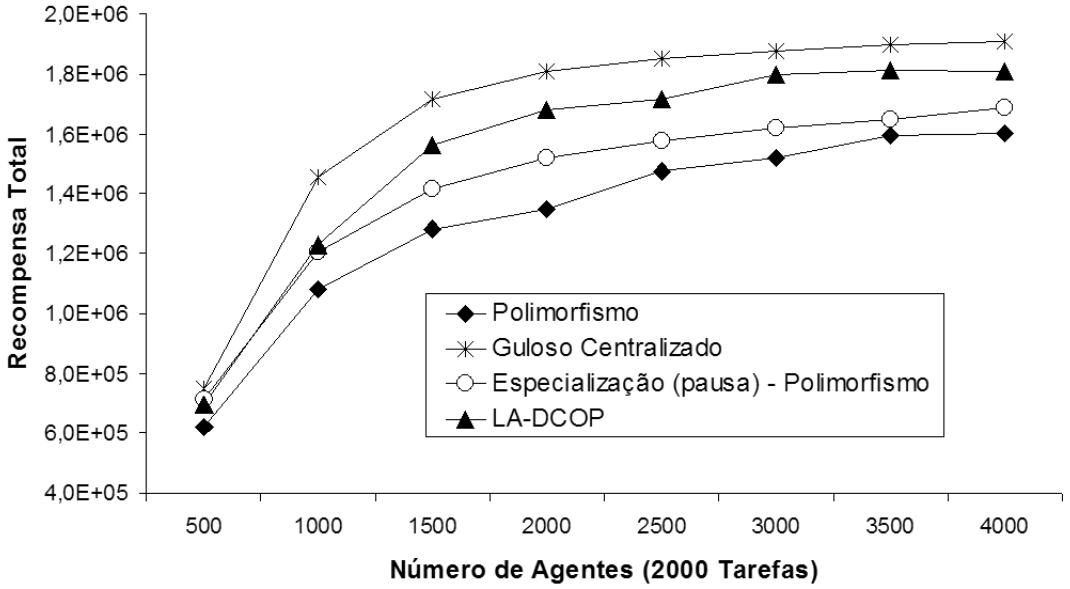


Figura 5.15: Comparando a recompensa total obtida pelo Swarm-GAP, LA-DCOP e um algoritmo guloso centralizado, para diferentes quantidades de agentes.

GAP e pelo LA-DCOP utilizando seu *threshold* [1] ótimo.

Como esperado, a abordagem gulosa centralizada supera o Swarm-GAP e o LA-DCOP. O Swarm-GAP, com os agentes utilizando o mecanismo de polimorfismo sem a especialização (curva *Polimorfismo*), tem um bom desempenho, obtendo recompensas totais 20% (em média) menores que as obtidas pela abordagem gulosa e cerca de 15% menores que as obtidas pelo LA-DCOP. Quando utilizando a especialização de agentes partindo do polimorfismo (curva *Especialização (pausa) - Polimorfismo*), o Swarm-GAP obtém resultados equivalentes ao LA-DCOP para 500 e 1000 agentes. Para mais de 1500 agentes o Swarm-GAP obtém resultados em média menores que 8% em relação ao LA-DCOP e 15% em relação a abordagem gulosa. Em geral, os algoritmos alocam o mesmo número de tarefas, como mostra a figura 5.16.

Em outro experimento realizado é medido o número médio de mensagens trocadas durante cada rodada da simulação pelo Swarm-GAP e pelo LA-DCOP, para diferentes quantidades de agentes. Como se pode ver na figura 5.17, para menos de 2500 agentes ambos algoritmos trocam praticamente o mesmo número de mensagens. Quando a quantidade de agentes aumenta, o número de mensagens trocadas pelos agentes executando o Swarm-GAP é mais alta que as trocadas pelos agentes executando o LA-DCOP.

Ambos algoritmos apresentam uma queda no número de mensagens trocadas quando o número de agentes é maior que o número de tarefas. Este comportamento é esperado pois, com uma grande quantidade de agentes, a alocação é obtida mais rapidamente do que quando o número de agentes é menor e, portanto, são trocadas menos mensagens. O mecanismo de tomada de decisão probabilístico do Swarm-GAP é responsável pela queda menos acentuada que se percebe na curva do Swarm-GAP. Mesmo com uma proporção maior de agentes que tarefas, agentes com alta competência para alocar determinadas tarefas podem decidir por não fazê-lo. Estas tarefas não alocadas geram as mensagens

[1] Os agentes executando o LA-DCOP decidem sobre quais tarefas alocar utilizando um *threshold* (limiar) global. Este *threshold* representa a competência do sistema como um todo para alocar cada tarefa, podendo ser fixo ou dinamicamente calculado.

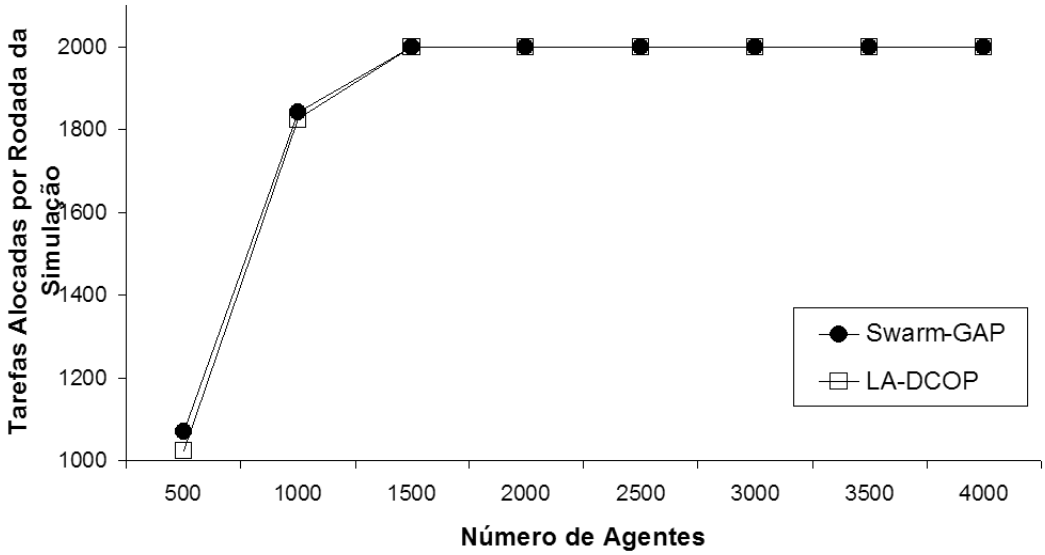


Figura 5.16: Comparando o número de tarefas alocadas pelo Swarm-GAP e LA-DCOP, para diferentes quantidades de agentes.

que diminuem a queda na referida curva.

O último experimento desta seção é uma repetição do experimento que impôs restrições à comunicação entre agentes no Swarm-GAP, realizado na seção 5.3.4. A figura 5.18 mostra a recompensa total obtida pelo LA-DCOP e pelo Swarm-GAP com e sem restrições na comunicação, de acordo com diferentes quantidades de agentes. Como mostrado anteriormente, com um canal de comunicação adequado, o LA-DCOP tem um desempenho melhor do que o Swarm-GAP. Contudo, quando a comunicação é restrita a 25% do normal (apenas 5 mensagens ao invés de 20), o LA-DCOP não troca tantas mensagens quanto precisaria. O desempenho do LA-DCOP passa a ser pior do que o desempenho do Swarm-GAP com a modificação da constante λ, discutida na seção 5.3.4, e praticamente igual ao do Swarm-GAP sem esta variação. O desempenho robusto do Swarm-GAP em condições de comunicação restritas é uma vantagem significativa sobre o LA-DCOP quando aplicando ambos em problemas reais.

5.3.5.3 Robocup Rescue Simulator

Como dito no início deste capítulo, o simulador da RoboCup Rescue oferece um ambiente de teste bastante realístico para problemas de coordenação em SMA. Os experimentos desta seção avaliam o Swarm-GAP em comparação ao LA-DCOP neste simulador. Ambos algoritmos são comparados com outro baseado em uma estratégia gulosa centralizada bastante similar às usadas pelos times da liga. Os resultados mostram que o Swarm-GAP e o LA-DCOP obtém resultados equivalentes, tendo o Swarm-GAP atingido resultados melhores que a abordagem gulosa em alguns casos.

Os cenários escolhidos para utilização no simulador foram dois mapas parciais da cidade japonesa de Kobe[2] (normalmente usados nas simulações da RoboCup Rescue), com 30 agentes representando brigadas de incêndio no primeiro deles ($Kobe$) e o dobro

[2]A cidade de Kobe foi atingida por um terremoto denominado Great Hanshi-Awaji em 1995 que afetou mais de 1 milhão de pessoas e teve seu dano total estimado em um bilhão de dólares americanos.

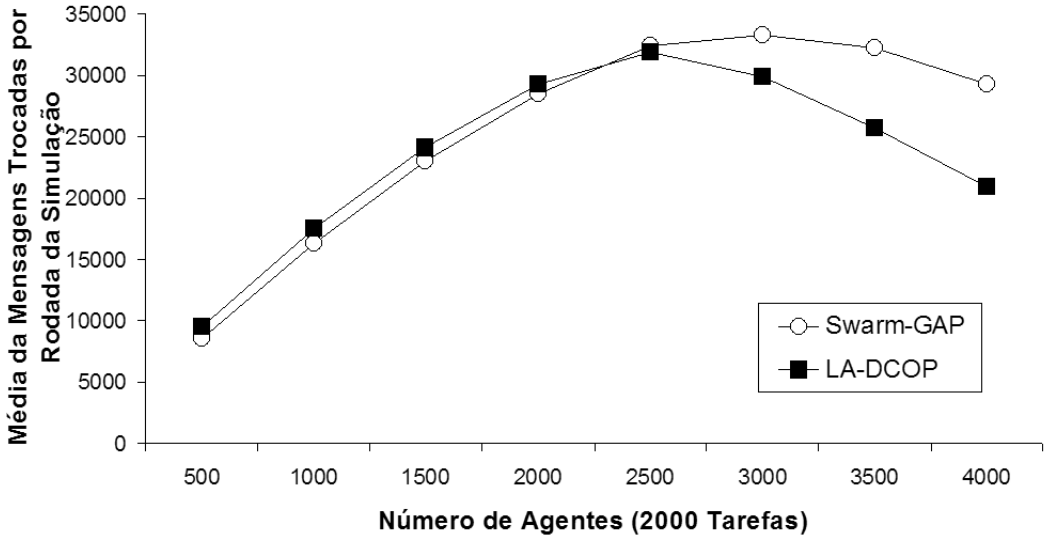


Figura 5.17: Comparando o número de mensagens trocadas em cada rodada da simulação pelo Swarm-GAP e LA-DCOP, para diferentes quantidades de agentes.

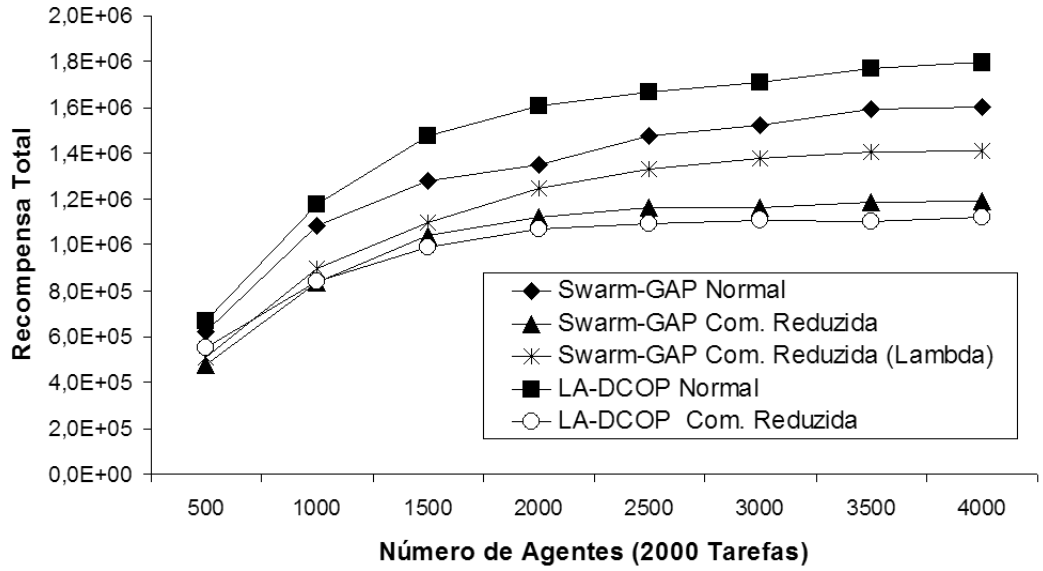


Figura 5.18: Comparando a recompensa total obtida pelo Swarm-GAP e LA-DCOP, para diferentes quantidades de agentes, com e sem restrições no canal de comunicação.

no segundo e maior ($Kobe\_4$).

A figura 5.19 mostra estes mapas com a distribuição inicial dos agentes (pontos pretos). Para cada mapa foram testados dois cenários, diferenciados pela quantidade de focos de incêndio: 30 focos ($Kobe_1$), 42 focos ($Kobe_2$), 43 focos ($Kobe\_4_1$) e 59 focos ($Kobe\_4_2$). Estes focos foram distribuídos uniformemente nos mapas em questão.

Para medir o desempenho dos algoritmos a métrica utilizada foi a área total construída que foi preservada depois de um determinado intervalo de tempo. Nos experimentos realizados, quando os agentes não percebem nenhuma tarefa ou não escolhem nenhuma, estes exploram o ambiente movendo-se aleatoriamente.

Para calcular a competência dos agentes no simulador da RoboCup Rescue foi utilizada a distância euclidiana entre o agente (brigada de incêndio) e cada foco de incêndio. A equação 5.6 mostra como essa medida é normalizada pelos agentes.

$$k_{ij} = \frac{\max\{distancia(j,i), \forall j \in \mathcal{J}\} - distancia(j,i)}{\max\{distancia(j,i), \forall j \in \mathcal{J}\}} \tag{5.6}$$

onde

k_{ij} competência do agente i em relação a tarefa j

$distancia(j,i)$ distância euclidiana no mapa entre o agente i e a tarefa j

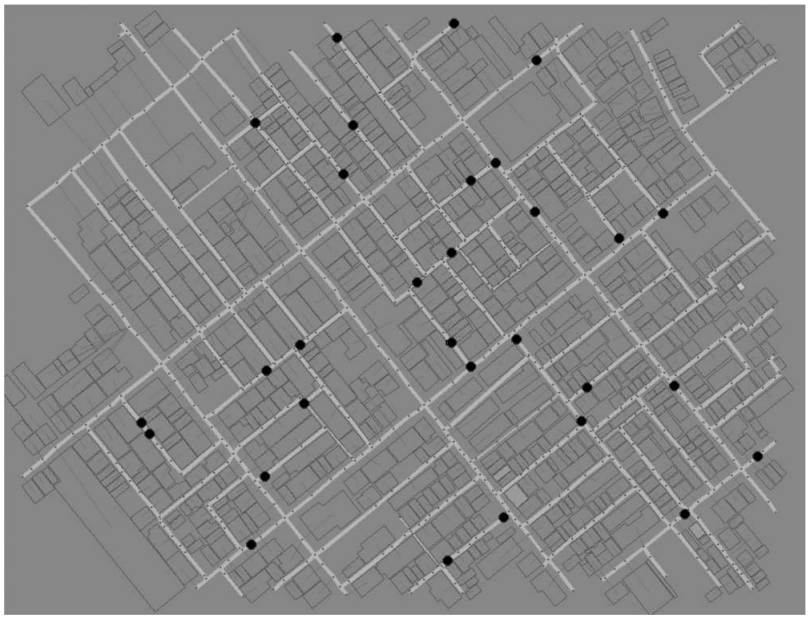
$\mathcal{J}$ conjunto de tarefas percebidas

A figura 5.20 mostra a média (20 execuções) dos resultados obtidos pelos algoritmos Swarm-GAP, LA-DCOP e o centralizado de estratégia gulosa, para cada cenário dos mapas $Kobe$ e $Kobe\_4$. O Swarm-GAP, no primeiro cenário de cada mapa ($Kobe_1$ e $Kobe\_4_1$), tem melhor desempenho que os outros algoritmos por uma pequena diferença. Em situações simples, quando existem poucos focos de incêndio, todos os algoritmos tem desempenho equivalente.

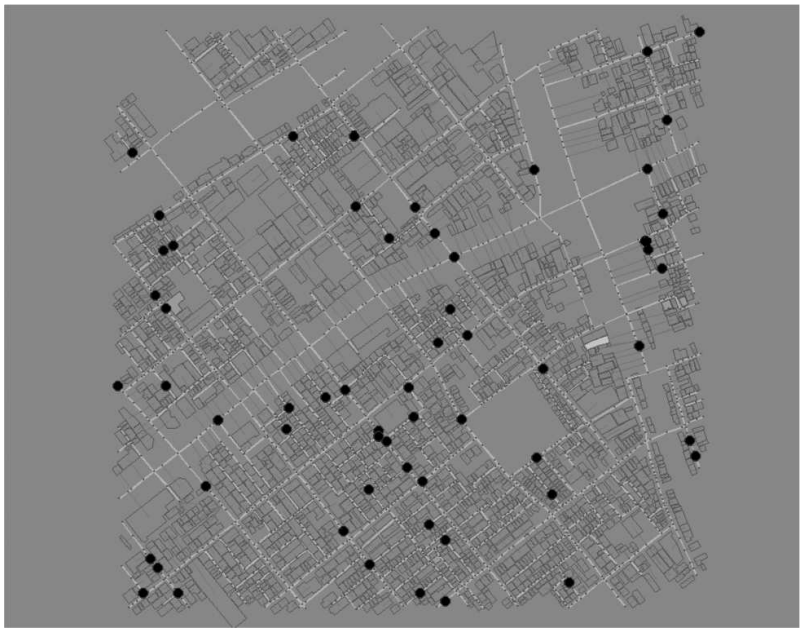
No segundo cenário de cada mapa ($Kobe_2$ e $Kobe\_4_2$) o Swarm-GAP se saiu melhor em relação ao LA-DCOP, mas não em relação ao algoritmo guloso. Quando existem mais focos de incêndios e estes se tornam extensos, passa a existir uma demanda de um número maior de agentes concentrada em cada grupo de focos. Neste caso, a tomada de decisão probabilística não permite que os agentes tomem ações mais concentradas, fazendo com que estes percam tempo na simulação deslocando-se de um foco de incêndio para outro. Já no LA-DCOP, os agentes simplesmente rejeitam tarefas para as quais o *threshold* é maior que sua competência. Uma vez que o LA-DCOP foi executado com *threshold* fixo, sempre existe a possibilidade que os agentes não aloquem tarefas durante uma rodada, ou fiquem alocando tarefas distantes por um longo tempo.

Os agentes executando o Swarm-GAP também podem não alocar nenhuma tarefa ou aceitar tarefas distantes, mas quando o número de tarefas é alto (como no segundo cenário do mapa $Kobe\_4$) isto ocorre com baixa probabilidade. De fato, na média geral, o número de tarefas não alocadas pelo LA-DCOP é aproximadamente 80% mais alta que no caso do Swarm-GAP. Com isso, se o LA-DCOP for executado com *threshold* dinâmico, diminuindo seu valor quando a tarefa não está alocada, é possível que o LA-DCOP torne-se melhor que o Swarm-GAP.

Os resultados mostrados aqui indicam que o Swarm-GAP, por possuir um mecanismo de tomada de decisão mais flexível, permite que os agentes se dividam melhor na alocação das tarefas, alcançando melhores resultados que o LA-DCOP e obtendo resultados equivalentes aos de um algoritmo guloso centralizado em algumas situações.



(a)



(b)

Figura 5.19: Os dois mapas, Kobe (a) e Kobe_4 (b), usados nos experimentos com o simulador da RoboCup Rescue, onde os pontos pretos representam a posição inicial das brigadas de incêndio.

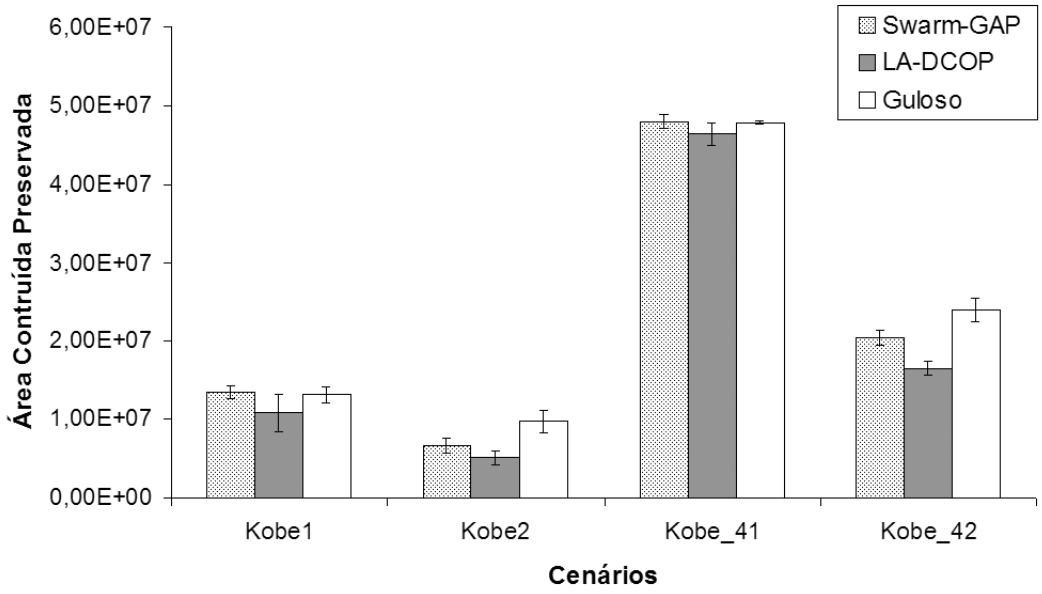


Figura 5.20: Comparando o melhor resultado obtido pelos algoritmos Swarm-GAP, LA-DCOP e o de estratégia gulosa centralizado, para cada cenário dos mapas $Kobe$ e $Kobe\_4$.

5.4 Conclusão

O algoritmo apresentado neste capítulo implementa vários mecanismos da abordagem proposta, os quais são: o mecanismo de tomada de decisão (seção 4.3), com ênfase no processo que lida com o interrelacionamento entre as tarefas (subseção 4.3.1), que é utilizado para que os agentes decidam sobre quais tarefas alocar, respeitando as interrelações AND entre elas; o mecanismo da tendência (seção 4.2) e o do polimorfismo (seção 4.4) que são utilizados em conjunto para que os agentes decidam quais tarefas alocar para maximizar o desempenho do sistema como um todo; e o mecanismo de especialização (seção 4.5) que é utilizado para que os agentes possam melhorar seu desempenho de maneira geral e, principalmente, na presença de tarefas para as quais os agentes não conhecem suas competências.

O Swarm-GAP foi experimentado para que se pudesse determinar o estímulo ótimo que maximiza o desempenho do algoritmo. Uma equação foi estabelecida, pelo método dos mínimos quadrados, para capturar a influência no estímulo ótimo da proporção entre agentes e tarefas e do quão distribuída é a percepção dos agentes. Os agentes foram capazes de alocar as tarefas quando tomaram suas decisões baseados na sua tendência de escolha, utilizando este estímulo ótimo e seus limiares internos configurados de acordo com suas competências.

Os agentes executando o Swarm-GAP também foram capazes de adaptar seus limiares internos para se especializar em tarefas para as quais têm maior competência, mesmo sem conhecê-las, através de um processo de adaptação mediante recompensa. A especialização permitiu melhorar o desempenho do sistema de maneira geral. As interrelações AND entre as tarefas foram tratadas adequadamente pela troca de mensagens simples e a redução nos canais de comunicação entre os agentes, apesar de implicar em uma redução de desempenho, foi contornada pelo algoritmo.

O LA-DCOP, um algoritmo recentemente proposto que apresentou resultados proeminentes, foi utilizado para que se pudesse comparar e avaliar o desempenho do Swarm-

GAP. Nos experimentos realizados em um simulador abstrato o LA-DCOP mostrou-se melhor que o Swarm-GAP em situações normais, considerando o desempenho com relação as competências empregadas nas alocações e no número de mensagens trocadas. Contudo, sob restrições de comunicação, o Swarm-GAP se equivaleu ou foi melhor que o LA-DCOP. Nos experimentos realizados no simulador da RoboCup Rescue o Swarm-GAP e o LA-DCOP obtiveram desempenhos semelhantes. Apesar disso, como discutido no início da seção 5.3.5.2, a complexidade computacional do LA-DCOP é significativamente maior que do Swarm-GAP.

Concebido com base na abordagem proposta, o algoritmo Swarm-GAP é simples e efetivo na solução do E-GAP de maneira aproximada. Todos os mecanismos foram experimentados e validados. Os resultados empíricos destes experimentos mostraram que a abordagem proposta, baseada em modelos teóricos da divisão do trabalho pelos insetos sociais, pode ser utilizada com sucesso para a alocação distribuída de tarefas, fazendo com que os agentes tomem suas decisões de forma coordenada, com baixa comunicação, para maximizar o desempenho do sistema. Cabe ressaltar que o desempenho, neste caso, está atrelado a maximização da competência dos agentes na alocação das tarefas.

6 ESCALONAMENTO DE TAREFAS NO RCPSP DISTRIBUÍDO

Este capítulo apresenta os resultados da aplicação da abordagem proposta neste trabalho em uma versão Distribuída do *Resource-Constrained Project Scheduling Problem* (RCPSP) (BRUCKER et al., 1999), denominado *Distributed* RCPSP (DRCPSP).

O mecanismo de modificação do estímulo, apresentado na seção 4.6, é experimentado especificamente neste problema. Além disso, dado que no DRCPSP as tarefas são interdependentes, aplica-se exclusivamente neste caso o mecanismo de tratamento de interdependência entre tarefas, discutido na seção 4.3.2. Os mecanismos de tomada de decisão, apresentado na seção 4.3, de tendência na escolha de tarefas, apresentada na seção 4.2, e de tratamento de interrelações entre as tarefas, discutido na seção 4.3.1, também são experimentados.

No DRCPSP, diferentemente do E-GAP discutido na capítulo anterior, os agentes não têm diferentes competências para escalonar as tarefas, não utilizando os mecanismos de polimorfismo e especialização, apresentados nas seções 4.4 e 4.5, respectivamente.

O presente capítulo está organizado da seguinte forma: na seção 6.1 o problema em questão é definido; a seção 6.2 apresenta o algoritmo que implementa a abordagem proposta para a solução do DRCPSP; na seção 6.3 são discutidos os experimentos realizados e os resultados obtidos; e, finalmente, na seção 6.4 é apresentada a conclusão desta aplicação experimental.

6.1 Definição do Problema

O *Resource-Constrained Project Scheduling Problem* (RCPSP) (BRUCKER et al., 1999) é um problema de otimização onde deve-se escalonar as tarefas de um projeto (originalmente denominadas de atividades) de modo que o tempo total do escalonamento seja minimizado. As tarefas são interdependentes, existindo restrições de precedência entre estas que devem ser respeitadas. Além disso, as tarefas demandam recursos para que possam ser escalonadas, os quais são restritos pela sua quantidade total disponível.

O RCPSP é uma generalização de vários problemas de escalonamento. Os cenários complexos utilizados por MAHESWARAN et al. (2004) para validar os algoritmos de alocação de tarefas, como descrito na seção 2.4.2, são problemas generalizáveis como RCPSPs. Esta característica contribuiu para a escolha deste problema para compor um dos cenários onde a abordagem proposta foi experimentada. Vários problemas reais da área de sistemas multiagente, como o agendamento múltiplo de reuniões e o controle de redes de sensores distribuídos, foram modelados na literatura como instâncias de problemas generalizáveis como RCPSPs.

Considerando que $\mathcal{J}$ representa o conjunto de tarefas e que o conjunto de recursos disponíveis para ser empregado nestas tarefas é denotado por $\mathcal{R}$, cada tarefa $j \in \mathcal{J}$ tem uma duração d_j e requer uma determinada quantidade r_j de cada recurso $r \in \mathcal{R}$, por unidade de tempo. A quantidade total disponível associada a cada recurso $r \in \mathcal{R}$ é dada por $Q_r > 0$. O conjunto de predecessoras diretas de j é dado por $P(j)$.

Um escalonamento é representado por $\mathcal{S}_n$, onde $\mathcal{S}_n = \{start_n(1), start_n(2), ..., start_n(j), ..., start_n(|\mathcal{J}|)\}$. Sendo que $start_n(j)$ é a unidade de tempo em que a tarefa j inicia no escalonamento $\mathcal{S}_n$. Assim, a unidade de tempo em que a tarefa é concluída é dada por $end(j) = start(j) + d_j$. A unidade de tempo em que um escalonamento inicia é a mesma unidade de tempo onde começa a tarefa que inicia mais cedo $min\{start_n(j) \ \forall \ j \in \mathcal{J}\}$, enquanto que a unidade de tempo em que o escalonamento termina é a mesma unidade de tempo onde termina a última tarefa $max\{start_n(j) + d_j \ \forall \ j \in \mathcal{J}\}$. O tempo total do escalonamento é a diferença entre as unidades de tempo em que este começa e termina. O RCPSP é um problema de otimização cujo objetivo é minimizar este tempo total.

Para ser considerado válido, um escalonamento $\mathcal{S}_n$ deve satisfazer as restrições mencionadas no início dessa seção, as quais são:

- A tarefa j não pode iniciar antes que todas as suas predecessoras terminem, $start_n(j) \geq start_n(h) + d_h \quad \forall h \in P(j)$;
- As restrições dos recursos têm de ser satisfeitas, ou seja, para cada unidade de tempo t, o somatório da quantidade de cada recurso r requisitados por todas as tarefas escalonadas não pode exceder a quantidade total destes recursos, $\sum_{j \in \mathcal{J}, start_n(j) \leq t < start_n(j)+d_j} r_j <= Q_r$.

O DRCPSP é uma modificação do RCPSP que o transforma em um problema distribuído. No DRCPSP cada unidade de cada recurso r pertence a um, e somente um, agente. Em outras palavras, cada agente possui apenas uma unidade de um recurso $r \in \mathcal{R}$, podendo escalonar apenas uma tarefa em cada unidade de tempo. Com isso, um agente i, que possui uma unidade do recurso $r \in \mathcal{R}$, têm competência k_{ij} para escalonar uma tarefa j de acordo com a quantidade do recurso r demandado por esta tarefa, dado pela equação 6.1.

$$k_{ij} = \begin{cases} 1 & se\ r_j \neq 0 \\ 0 & caso\ contrário \end{cases} \tag{6.1}$$

Em sua versão distribuída, as tarefas do RCPSP são divididas em tantas subtarefas quanto for a quantidade de recursos demandados por elas. Assim, cada tarefa j é divida em um conjunto $\mathcal{B}_j$ de subtarefas, na quantidade equivalente ao somatório da quantidade r_j de cada recurso $r \in \mathcal{R}$ que esta tarefa demanda. Portanto, o tamanho do conjunto de subtarefas $|\mathcal{B}_j|$ é igual a quantidade total de recursos que a tarefa demanda para ser escalonada.

A distribuição do RCPSP, da maneira como se está propondo, impõe outras duas restrições para que um escalonamento $\mathcal{S}_n$ seja considerado válido:

- Considerando $\mathcal{I}_{j_n}$ como o conjunto de agentes que alocaram as subtarefas de j no escalonamento $\mathcal{S}_n$, todas as subtarefas $j_m \in \mathcal{B}_j$ devem ser escalonadas, $|\mathcal{I}_{j_n}| = |\mathcal{B}_j|$;

Tabela 6.1: Duração d_j e demanda de agentes $\mathcal{R}_j$ para cada tarefa, considerando uma instância hipotética do DRCPSP.

	Tarefas				
	A	**B**	**C**	**D**	**E**
d	4	9	2	2	2
$\mathcal{R}$	{1,0}	{1,0}	{0,2}	{2,0}	{2,1}
$P()$	-	-	B	-	A

- Sendo $start_n(j_m)$ a unidade de tempo em que uma subtarefa $j_m \in \mathcal{B}_j$ foi escalonada no escalonamento $\mathcal{S}_n$, todas as demais subtarefas de j, suas tarefas irmãs, devem ser escalonadas na mesma unidade de tempo t por todos os agentes $i \in \mathcal{I}_{j_n}$, $\{start_n(j_1) = start_n(j_2), start_n(j_2) = start_n(j_3), ..., start_n(j_{|\mathcal{B}_j|-1}) = start_n(j_{|\mathcal{B}_j|})\}$.

Assim, os agentes devem buscar, de forma distribuída, minimizar o tempo necessário para compor um escalonamento para as tarefas percebidas por eles, respeitando todas as restrições elencadas nessa seção que tornam tal escalonamento válido. Isso faz com que a coordenação entre os agentes seja uma questão bastante complexa. Os agentes precisam coordenar-se para, em número pelo menos igual ao demandado, alocar tarefas simultaneamente. Além disso, para que esses escalonamentos sejam possíveis, os agentes têm que escalonar as tarefas em uma ordem que respeite as relações de precedência entre as tarefas.

Seja uma instância simples do DRCPSP com 4 agentes $\mathcal{I} = (V, X, Y, Z)$ e 5 tarefas $\mathcal{J} = (A, B, C, D, E)$, com suas especificações definidas na tabela 6.1. Cada tarefa tem sua duração específica como mostra a primeira linha da tabela. Os conjuntos da segunda linha da tabela indicam a quantidade de cada um dos recursos disponíveis $\mathcal{R} = \{R1, R2\}$ demandados por cada tarefa. Na última linha da tabela se pode ver as relações de precedência entre as tarefas.

Segundo a subdivisão das tarefas de acordo com os recursos demandados por elas, anteriormente descrita, tem-se: as tarefas A e B permanecem inalteradas, demandando uma unidade do recurso $R1$ cada uma; a tarefa C é dividida em duas subtarefas ($C1$ e $C2$) que demandam uma unidade do recurso $R2$ cada uma; a tarefa D é dividida em duas subtarefas ($D1$ e $D2$) que demandam uma unidade do recurso $R1$ cada uma; e a tarefa E é dividida em três subtarefas ($E1$, $E2$ e $E3$), as duas primeiras demandando uma unidade do recurso $R1$ e a $E3$ demandando uma unidade do recurso $R2$. Os agentes V e X possuem uma unidade do recurso $R1$ e os agentes Y e Z possuem uma unidade do recurso $R2$.

A figura 6.1 mostra um escalonamento possível para a instância em discussão. Como se pode ver, a interdependência entre as tarefas foi respeitada, com as subtarefas de C e E escalonadas depois de suas predecessoras. O escalonamento simultâneo das subtarefas de C, D e E também foi realizado adequadamente pelos agentes.

Um DRCPSP qualquer pode ser facilmente modelado em TÆMS (discutido na seção 2.3). Para criar uma estrutura de tarefas TS com a visão objetiva (problema como um todo) de um DRCPSP basta realizar os seguintes passos:

1. Cria-se uma tarefa raiz TG (*task group*) para a TS do DRCPSP utilizando a QAF *sum* para interrelacionar suas subtarefas;

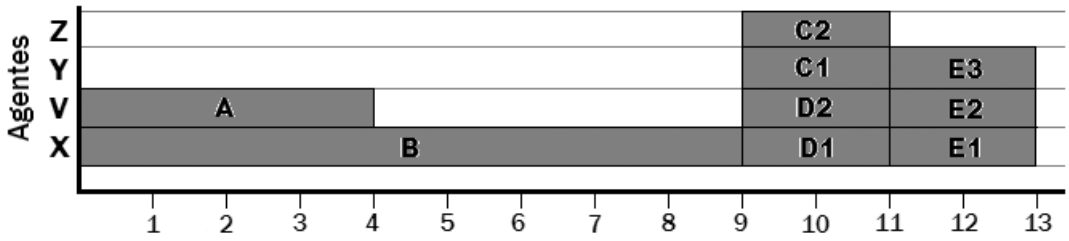


Figura 6.1: Exemplo de um escalonamento possível para uma instância hipotética simples do DCRPSP.

2. Para cada tarefa j do DRCPSP que possui subtarefas $\mathcal{B}_j \neq \emptyset$, cria-se uma subtarefa da TG utilizando a QAF *sync sum* para interrelacionar suas subtarefas;

3. Para cada tarefa j do DRCPSP que não possui subtarefas $\mathcal{B}_j = \emptyset$, cria-se um método filho da TG;

4. Para cada subtarefa $j_m \in \mathcal{B}_j$ cria-se um método filho da tarefa j na TS;

5. Cria-se uma NLE *enable* para cada interdependência entre tarefas, partindo de um método qualquer da tarefa origem para a tarefa que é sua dependente;

6. Cria-se um recurso em TÆMS, renovável a cada unidade de tempo, para cada recurso $r \in \mathcal{R}$, com o estado configurado para 1 unidade. Além disso, identifica-se os agentes que possuem cada recurso;

7. Cria-se um relacionamento *consumes* e um relacionamento *limits* entre cada recurso e os métodos relacionados às tarefas que demandam estes recursos. Ambos os relacionamentos devem conter 1 como a quantidade de recursos que cada método consome;

8. Configura-se os atributos dos métodos de acordo com os atributos das tarefas da seguinte maneira: a duração d_j da tarefa j como a duração de cada um dos métodos dessa tarefa em TÆMS; 0 para a qualidade e custo dos métodos, uma vez que no DRCPSP estes não são relevantes.

A figura 6.2 mostra a instância do DRCPSP discutida nesta seção modelada como uma estrutura de tarefas TS objetiva em TÆMS através da metodologia descrita acima.

6.2 Swarm-RCPSP

O algoritmo Swarm-RCPSP, apresentado no algoritmo 4, implementa os mecanismos da abordagem proposta necessários para a solução do DRCPSP. Os agentes executando o Swarm-RCPSP se comunicam utilizando mensagens simples, reagindo a dois eventos: a percepção de tarefas e o recebimento de mensagens. Quando um agente percebe algumas tarefas este cria um objeto *taskSched* composto por estas tarefas (linhas 4 a 7). Da mesma forma, quando um agente recebe uma mensagem de outro agente (linha 9) este também cria um objeto *taskSched*, porém desta vez contendo as tarefas que compõem a mensagem recebida. Uma vez que isso é feito, o agente passa a tentar obter um escalonamento válido para tais tarefas (linhas 12 a 41).

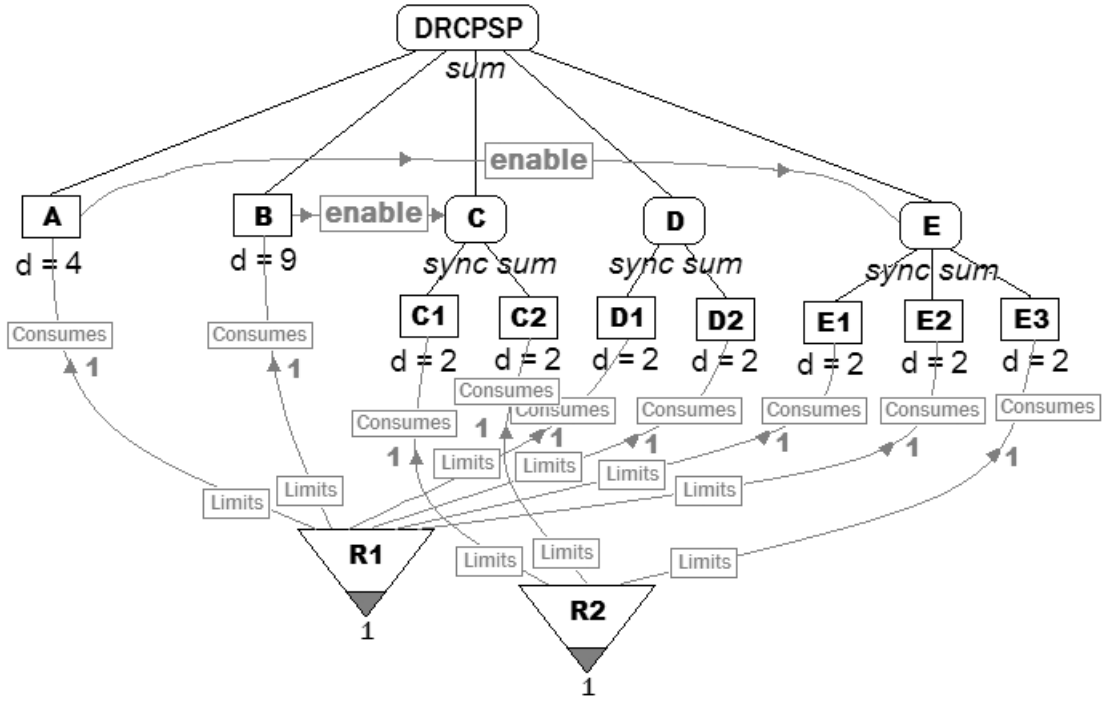


Figura 6.2: Estrutura de tarefas TS objetiva em TÆMS para uma instância simples do DRCPSP.

Utilizando o mecanismo discutido na seção 4.3, a cada iteração do laço mais externo (linhas 13 a 41), os agentes analisam todas as tarefas que não tem predecessoras ou que suas predecessoras foram alocadas com sucesso numa iteração anterior. Assim, a cada nova execução deste laço, o escalonamento vai sendo gradualmente obtido respeitando as interrelações de precedência entre as tarefas. Considerando que uma série de restrições tem que ser respeitadas para que o escalonamento seja considerado válido (mencionadas na seção 6.1), é possível que o agente não obtenha um escalonamento para as tarefas em análise nas iterações do laço em questão. Para verificar isso, o agente utiliza a função *successfullyScheduled()* (linha 34). Uma vez que não foi possível obter um escalonamento bem sucedido, o algoritmo é interrompido e é indicado que esta tentativa foi falha. Caso contrário, o melhor escalonamento obtido passa a ser o escalonamento definitivo para todas as tarefas analisadas até então (linha 35). Em seguida, o agente comunica-se com seus vizinhos para informar-lhes sobre seu novo escalonamento, utilizando a função *communicateNeighbours()*.

A função *successfullyScheduled()*, algoritmo 5, determina se um conjunto de tarefas tiveram todas as suas tarefas irmãs escalonadas no mesmo intervalo de tempo. Tarefas irmãs são aquelas que pertencem a uma mesma supertarefa e, conforme discutido na seção 6.1, precisam ser todas escalonadas simultaneamente para que o escalonamento seja considerado válido. A função *communicateNeighbours()*, algoritmo 6, permite que o agente troque informações sobre o escalonamento que o sistema está tentando realizar. Esta função permite que o agente envie (linha 2) e receba (linha 3) de seus vizinhos as informações referentes a um determinado escalonamento. O agente mantém o objeto *taskSched* com as informações referentes aos escalonamentos que ele acredita que seus vizinhos estão realizando no momento.

No laço das linhas 15 a 32, do algoritmo 4, o agente tenta por um determinado número de vezes (η) obter o melhor escalonamento válido possível para as tarefas correntes. Cabe

lembrar que só são analisadas as tarefas que não têm predecessoras ou cujas predecessoras foram escalonadas com sucesso numa iteração anterior (linha 18). A função *trySchedule()* define se e quando cada uma destas tarefas será escalonada (linha 19). O agente então comunica o resultado desse processo aos seus vizinhos (linha 21). As tarefas que foram escalonadas são novamente tratadas no laço mais interno do algoritmo (linhas 23 a 27), que será discutido a seguir. De qualquer maneira, depois desse laço mais interno, se o agente obter um escalonamento válido, este, se for o caso, substituirá o melhor armazenado (linha 31). Antes da próxima tentativa de obter um escalonamento melhor, o agente atualiza os estímulos para cada tarefa escalonada nessa iteração utilizando o mecanismo discutido na seção 4.6 (linha 31) e implementado na função *adaptStimulus()*, algoritmo 8.

No laço mais interno do algoritmo 4 (linhas 23 a 27), mencionado no parágrafo anterior, o agente passa a analisar repetidamente as tarefas escolhidas para serem alocadas nos passos anteriores (linhas 18 a 21). O agente refaz o processo de escalonamento destas tarefas até que este escalonamento seja um escalonamento válido ou que um determinado número de iterações tenha sido realizada (χ). Isso se dá porque dificilmente o primeiro escalonamento obtido (linhas 18 a 21) respeita todas as restrições da definição do DRCPSP (discutidas na seção 6.1) e que invalidam o escalonamento obtido pelo sistema como um todo. Assim, os agentes interagem sucessivas vezes, escalonando e re-escalonando as tarefas para buscar um escalonamento válido. Se este processo for mal sucedido, ou seja, se o número de tentativas não for suficiente, o algoritmo segue sua execução como discutido no parágrafo anterior.

Os agentes utilizam a função *trySchedule()*, algoritmo 7, para decidir se uma determinada tarefa será escalonada e em que unidade de tempo do escalonamento isso ocorrerá. Caso a tarefa ainda não tenha sido escalonada de forma válida (segundo a função *successfullyScheduled()*), o agente toma sua decisão a respeito de escaloná-la ou não utilizando o mecanismo discutido na seção 4.2 (linha 2). Se a tarefa tiver sido escalonada de forma válida e o agente não estiver escalonando a tarefa em um mesmo intervalo que os demais agentes escalonaram suas tarefas irmãs, este retira a tarefa do seu escalonamento (linhas 12 e 13). O mesmo ocorre quando o agente decide escalonar a tarefa que já estava no seu escalonamento (linhas 3 e 4). Como a tarefa será re-escalonada, esta é primeiramente retirada do escalonamento. Em seguida, o agente determina o intervalo de tempo em que a tarefa será escalonada tentando primeiramente (linhas 5 e 6) aquele que ele acredita que seus vizinhos utilizaram (armazenado no objeto $taskSched$). Caso o agente não tenha essa informação ou este intervalo não esteja disponível no seu escalonamento, este escolhe o intervalo mais cedo possível, respeitando as restrições de interdependência (NLEs *enable*) com suas antecessoras (linha 8). Com o intervalo de tempo determinado, o agente insere a tarefa em seu escalonamento (linha 9).

6.3 Experimentos e Resultados

Os experimentos com o Swarm-RCPSP foram realizados em um simulador abstrato, independente de domínio, escrito em Java. Estas simulações foram configuradas de acordo com três instâncias diferentes de RCPSPs. Tais instâncias pertencem ao conjunto de dados "j120" da biblioteca *Project Scheduling Problem Library* (PSPLIB)[1] (KOLISCH; SPRECHER, 1997). Estas instâncias são originalmente para problemas centralizados, mas, como discutido na seção 6.1, podem ser mapeadas como problemas distribuí-

[1]A PSPLIB é uma das mais conhecidas e bem conceituadas biblioteca de conjuntos de teste para o RCPSP e encontra-se disponível na URL http://129.187.106.231/psplib/

Algorithm 4 Swarm-RCPSP($agentId$)

1: **loop**
2: $ev \leftarrow$ waitEvent()
3: **if** $ev =$ *task perception* **then**
4: $\mathcal{J} \leftarrow$ *set of new unscheduled tasks*
5: $taskSched \leftarrow$ newTaskSched()
6: **for all** $t \in \mathcal{J}$ **do**
7: $taskSched$.addTask(t)
8: **else**
9: $taskSched \leftarrow$ newTaskSched(receiveMessage())
10: $\tau \leftarrow taskSched$.getTasks()
11: *{tenta obter um escalonamento completo válido}*
12: $\mathcal{S} \leftarrow \emptyset$
13: **repeat**
14: *{tenta obter o melhor escalonamento parcial válido}*
15: **repeat**
16: $\mathcal{S}' \leftarrow \mathcal{S}$
17: $\mathcal{M} \leftarrow \emptyset$
18: **for all** $t \in \tau \ \cap \mathcal{S} \ / \ \forall h \in P(t) \ \exists h \in \mathcal{S} \ \vee \ P(t) = \emptyset$ **do**
19: **if** trySchedule(t, $\mathcal{S}'$, $\mathcal{M}$, $taskSched$) **then**
20: $\mathcal{M} \leftarrow \mathcal{M} \ \cup t$
21: communicateNeighbours($\mathcal{S}'$, $taskSched$)
22: *{tenta obter um escalonamento parcial válido para as tarefas escolhidas nos passos anteriores}*
23: **repeat**
24: **for all** $t \in \mathcal{M}$ **do**
25: trySchedule(t, $\mathcal{S}'$, $\mathcal{M}$, $taskSched$)
26: communicateNeighbours($\mathcal{S}'$, $taskSched$);
27: **until** successfullyScheduled($\mathcal{M}$, $taskSched$) or η
28: *{realiza a adaptação dos estímulos}*
29: **if** successfullyScheduled($\mathcal{M}$, $taskSched$) and $|\mathcal{S}^*| < |\mathcal{S}'|$ **then**
30: $\mathcal{S}^* \leftarrow \mathcal{S}'$
31: adaptStimulus($\mathcal{M}$, $taskSched$)
32: **until** χ
33: *{abandona a execução se não obteve um escalonamento parcial válido}*
34: **if** successfullyScheduled($\mathcal{M}$, $taskSched$) **then**
35: $\mathcal{S} \leftarrow \mathcal{S}^*$
36: $\mathcal{S}^* \leftarrow \emptyset$
37: communicateNeighbours($\mathcal{S}$, $taskSched$)
38: **else**
39: **return** fail attempt
40: *{tenta realizar o escalonamento das tarefas não escolhidas, caso existam}*
41: **until** successfullyScheduled(τ, $taskSched$)

Algorithm 5 successfullyScheduled($\mathcal{M}, taskSched$)

for all $t \in \mathcal{M}$ **do**
  $\mathcal{J} \leftarrow taskSched$.getSisterTasks($t$)
  for all $j \in \mathcal{J}$ **do**
    if $taskSched$.isTaskScheduled(j) **then**
      if !$taskSched$.start(j) = $taskSched$.start(t) **then**
        return false
    else
      return false
return true

Algorithm 6 communicateNeighbours($\mathcal{S}', taskSched$)

for all $i \in neighbourhood$ **do**
  sendMessage(i, $\mathcal{S}'$)
  $taskSched$.updateTaskInfo(receiveMessage(i))

Algorithm 7 trySchedule(t, $\mathcal{S}'$, $\mathcal{M}'$, $taskSched$)

if !successfullyScheduled($\{t\}, taskSched$) and roulette() $< T_{\theta(s_t)}$ **then**
  if $taskSched$.isTaskAllocatedByMe(t) **then**
    $\mathcal{S}' \leftarrow \mathcal{S}' \cap \{$start($t$)$\}$
  if $\mathcal{S}' \cup taskSched$.start($t$) **then**
    start(t) $\leftarrow taskSched$.start(t)
  else
    start(t) $\leftarrow$ first possible time unit where $start(t) > start(h) \ \forall h \in P(t)$
  $\mathcal{S}' \leftarrow \mathcal{S}' \cup \{$start($t$)$\}$
  return true
else
  if $taskSched$.isTaskAllocatedByMe(t) **then**
    $\mathcal{S}' \leftarrow \mathcal{S}' \cap \{$start($t$)$\}$
return false

Algorithm 8 adaptStimulus($\mathcal{M}$, $taskSched$)

if μ schedules have been done **then**
  for all $t \in \mathcal{M}$ **do**
    $s_t = s_t * \delta + \Gamma(t) * \alpha$

Tabela 6.2: Quantidade de tarefas com diferentes números de sucessoras na configuração das simulações do DRCPSP.

Instância	**3 sucessoras**	**2 sucessoras**	**1 sucessora**
"j1201_1"	23	15	82
"j12030_1"	32	32	56
"j12060_1"	54	24	42

Tabela 6.3: Quantidade de agentes disponível em cada classe na configuração das simulações do DRCPSP.

Instância	**R1**	**R2**	**R3**	**R4**
"j1201_1"	14	12	13	9
"j12030_1"	31	28	31	38
"j12060_1"	40	50	55	46

dos.

As três instâncias utilizadas para os experimentos deste capítulo são as de maior escala disponíveis, denominadas "j1201_1", "j12030_1" e "j12060_1". Em cada uma delas são utilizadas 120 tarefas. As tarefas são fortemente interdependentes em todas as instâncias. Cada tarefa tem um número de sucessoras no intervalo discreto $[1, 3]$ e em proporções diferentes de acordo com as características de cada instância, como mostra a tabela 6.2.

As tarefas têm duração também em um intervalo discreto, neste caso $[1, 10]$, e são utilizados 4 recursos diferentes denominados $R1$, $R2$, $R3$ e $R4$. Cada tarefa pode demandar uma quantidade de cada recurso no intervalo discreto $[0, 10]$. A tabela 6.3 mostra a quantidade de cada recurso disponível para cada uma das instâncias.

As instâncias em questão também diferem-se pelo número de recursos diferentes que demandam. Na instância "j1201_1" todas as tarefas demandam uma quantidade diferente de zero para apenas um recurso. Já na instância "j12030_1" o número de recursos diferentes requeridos varia no intervalo discreto $[1, 3]$ e na instância "j12060_1" uma tarefa pode demandar que todos os recursos estejam envolvidos em seu escalonamento.

Considerando o processo de divisão das tarefas do RCPSP em subtarefas no DRCPSP, tem-se um total de 646, 1334 e 2554 tarefas para as instâncias "j1201_1", "j12030_1" e "j12060_1", respectivamente. Cada uma destas tarefas demanda uma unidade de um determinado recurso de acordo com as características das supertarefas em cada instância especificamente.

Uma vez que cada agente possui um recurso, o número de agentes necessários para tratar estas instâncias como um DRCPSP é igual ao somatório da quantidade disponível de cada de recurso, mostrados na tabela 6.3. Este somatório é igual a 48, 128 e 191 para as instâncias "j1201_1", "j12030_1" e "j12060_1", respectivamente, sendo esta última a maior disponível no conjunto de dados.

Como se pode ver, a complexidade dos problemas cresce substancialmente de uma instância para outra, considerando que aumentam simultaneamente o número de interdependências entre as tarefas, a quantidade total de agentes e a quantidade de classes de agentes envolvidas no escalonamento de cada tarefa. Uma tentativa de escalonamento só é considerada bem sucedida se as tarefas escalonadas têm respeitadas todas as restrições que caracterizam o DRCPSP discutidas em detalhes na seção 6.1.

Um algoritmo equivalente ao Swarm-RCPSP foi implementado baseado em uma estratégia gulosa (*greedy*) distribuída. Este algoritmo é uma variação simples do Swarm-

RCPSP, não tendo implementado o mecanismo de variação do estímulo e nem tão pouco a tomada de decisão probabilística orientada pela tendência. Os agentes, através deste algoritmo, escolhem as tarefas sempre que possível. Estas tarefas, assim como no Swarm-RCPSP, são escalonadas na ordem em que são escolhidas de acordo com a disponibilidade de tempo no escalonamento do agente. Optou-se por implementar esta estratégia para que se pudesse analisar e comparar as questões de desempenho associadas à busca da solução de forma distribuída. Com isso, pode-se ver claramente o impacto da utilização dos mecanismos que compõem a abordagem na qualidade da solução obtida para o DRCPSP.

Além disso, a PSPLIB possui as melhores soluções conhecidas obtidas de forma aproximada para as instâncias utilizadas nesse capítulo, que servirão como parâmetro de comparação para avaliar o desempenho geral do Swarm-RCPSP e do algoritmo guloso. As instâncias "j1201_1" e "j12030_1" tem sua melhor solução obtida através do trabalho de BRUCKER; KNUST (2000), enquanto que a instância "j12060_1" tem sua melhor solução obtida por LUO et al. (2006). Cabe ressaltar que estas soluções foram determinadas de forma centralizada, utilizando diversas técnicas que, na maioria das vezes, não têm compromisso com sua aplicabilidade real (por exemplo, não consideram se o tempo necessário para obter a solução é viável ou não). Estas técnicas são empregadas com o único objetivo de buscar uma solução melhor do que a melhor conhecida.

Para garantir que os resultados obtidos com os algoritmos distribuídos possam ser comparados adequadamente à melhor solução conhecida obtida de forma centralizada, nos experimentos conduzidos aqui garante-se que todas as tarefas serão percebidas pelo menos por um agente. O processo de recrutamento dos vizinhos, para lidar com as interrelações entre tarefas, garante que uma tarefa percebida por um agente, que necessite de mais agentes escalonando suas irmãs, será compartilhada. Desta forma, os demais agentes terão sua percepção estendida para conter estas tarefas. Isso é importante para que o sistema possa obter um escalonamento completo. Pelo mesmo motivo, as mensagens são enviadas para todos os agentes simultaneamente (*broadcasting*).

Os experimentos conduzidos neste capítulo são discutidos como segue: primeiramente, na seção 6.3.1 é empiricamente determinado o limiar interno θ inicial que minimiza o tempo total dos escalonamentos de acordo com o estímulo ótimo determinado na seção 5.3.1; depois disso, na seção 6.3.2, o mecanismo de adaptação do estímulo proposto neste trabalho é explorado; na seqüência, a seção 6.3.3, mostra o desempenho do Swarm-RCPSP quando são impostas restrições na comunicação entre os agentes; finalmente, na seção 6.3.4, o Swarm-RCPSP é comparado com a abordagem gulosa distribuída e com os casos de teste comentados anteriormente.

6.3.1 Limiar Interno Ótimo

Problemas como o E-GAP, discutido no capítulo 5, consideram que cada agente pode ter competências específicas para cada tarefa. Isso se dá, por exemplo, de acordo com sua habilidade em executá-la ou a distância que este se encontra dela. No caso do RCPSP, os agentes têm ou não competência para realizar cada tarefa, ou seja, sua competência é nula ou máxima. Por isso, o mecanismo de polimorfismo, discutido na seção 4.4, não se aplica na solução do DRCPSP. Desta forma, é preciso determinar que combinação dos parâmetros na equação da tendência (4.1) obtém o escalonamento de menor tempo total.

Uma abordagem óbvia é a utilização da equação 6.1, que determina a competência k_{ij} do agente de acordo com a definição do DRCPSP. Assim, utilizando os valores 0 ou 1 para a competência dos agentes, poderia-se determinar empiricamente o estímulo ótimo que leva a uma melhor solução para o DRCPSP, da mesma forma como o que feito para

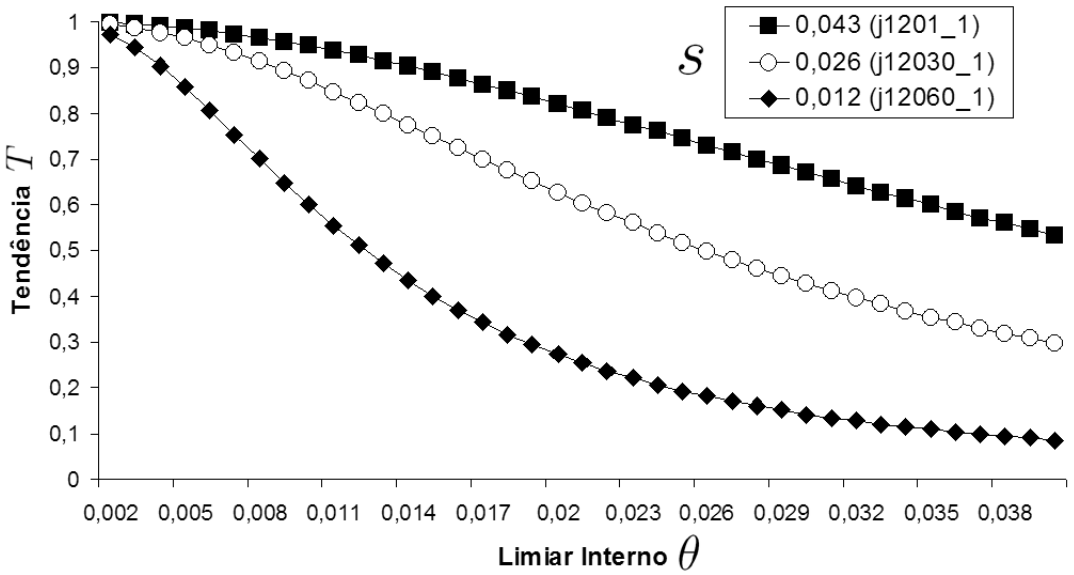


Figura 6.3: Comparando a tendência de acordo com o estímulo ótimo em cada instância de experimentação do DRCPSP, para diferentes valores do limiar interno.

o E-GAP na seção 5.3.1.

Contudo, para permitir a análise mais precisa da integração de todos os mecanismos da abordagem proposta, optou-se por adotar aqui o estímulo que foi determinado como ótimo para o E-GAP na seção 5.3.1. Com isso, é possível verificar o impacto no processo de tomada de decisão de ambos os aspectos combinados: a busca pela melhor alocação de acordo com as competências dos agentes; e a busca pelo escalonamento de menor tempo total. Cabe lembrar que nos experimentos realizados nesta seção o valor do estímulo é fixo. Este valor servirá como ponto de partida para o mecanismo de adaptação do estímulo, explorado na próxima seção (6.3.2), sendo incrementado e decrementado de acordo com este mecanismo (seção 4.6).

Experimentos preliminares mostraram que, no Swarm-RCPSP, a tendência que os agentes têm para escolher cada tarefa em um determinado instante tem de ser maior que 0.5, para que seja possível obter um escalonamento válido em um tempo hábil, e menor que 0.9, para que os agentes não escolham simplesmente as tarefas na ordem em que as perceberam. A figura 6.3 mostra a tendência dos agentes de acordo com o estímulo ótimo (equação 5.5) em cada instância de experimentação do DRCPSP, para diferentes valores do limiar interno. Pode-se verificar que os valores para o limiar interno que produzem tendências na faixa viável entre 0.5 e 0.9 variam para cada instância. Para a instância "j1201_1" este valor varia no intervalo $[0.02, 0.04]$, enquanto que para a instância "j12030_1" varia no intervalo $[0.008, 0.024]$, e para a instância "j12060_1" varia no intervalo $[0.004, 0.012]$.

A figura 6.4 mostra a variação do tempo total do escalonamento, de acordo com os valores dos limiares internos nos intervalos discutidos no parágrafo anterior, para as instâncias de experimentação "j1201_1", "j12030_1" e "j12060_1". Como se pode ver, a variação nos valores para o limiar interno não tem grande impacto sobre o tempo total dos escalonamentos realizados. Em todas as instâncias a diferença entre os tempos totais não é estatisticamente significativa (teste ANOVA), ao nível de 5% ($P < 0.05$), dada a variância destes valores (cabe lembrar que o tempo total dos escalonamentos mostrados

nos gráficos dos experimentos é a média de 20 execuções). Pode-se ver uma diferença crescente entre a variância das diferentes instâncias a medida que estas são mais complexas. Isso se dá porque, como existem mais restrições a serem respeitadas conforme as instâncias ficam mais complexas, é mais difícil obter um escalonamento completo válido e estes escalonamentos, quando obtidos, ou são realizados rapidamente com um tempo total baixo, ou precisam de mais interações do algoritmo e têm um tempo total alto. Em suma, conforme o problema fica mais complexo, os escalonamentos tendem a obter tempos totais mais extremos, ou muito baixos ou muito altos, tendo poucos tempos totais medianos.

De maneira geral, para todas as instâncias experimentadas, o valor para tendência que obteve o melhor resultado é o de aproximadamente 0.7. Com isso, é possível determinar o limiar interno ótimo que produz o escalonamento de menor tempo total através da equação 6.2. Esta equação foi obtida simplesmente substituindo os termos e isolando o limiar na equação da tendência (4.1). Cabe destacar que a equação 6.2 foi baseada em um valor para a tendência que resultou nos escalonamentos de menor duração para os casos experimentados. Outros casos, com diferentes configurações, podem levar a outros valores para a tendência. É indicado determinar empiricamente o limiar interno ótimo em cada caso, que seja significativamente diferente dos considerados aqui, para verificar a adequação desta equação. O Swarm-RCPSP adota esta equação para determinar o limiar interno inicial para os agentes. Este limiar é igual para todos os agentes e para todas as tarefas, e é denominado apenas de θ.

$$\theta = \sqrt{\frac{s^2}{0.7} - s^2} \tag{6.2}$$

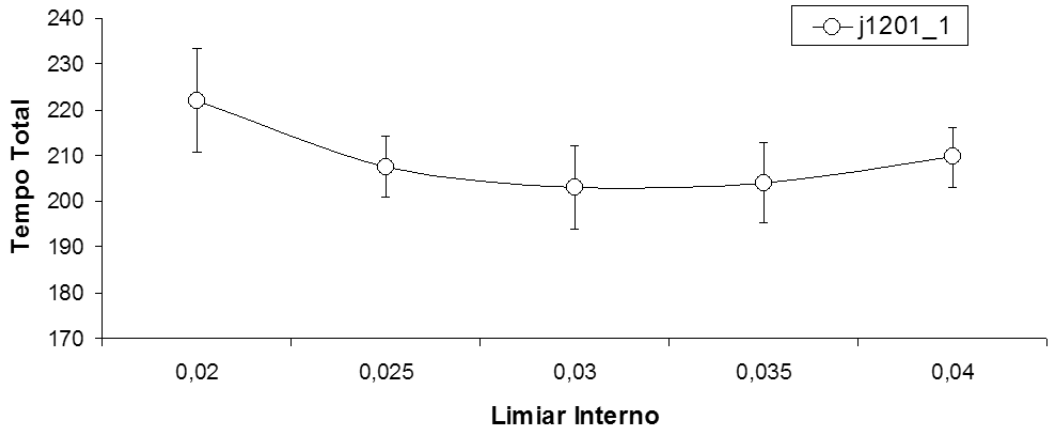
onde:

s_{ij} estímulo associado com a tarefa j para o agente i, calculado pela equação 5.5.

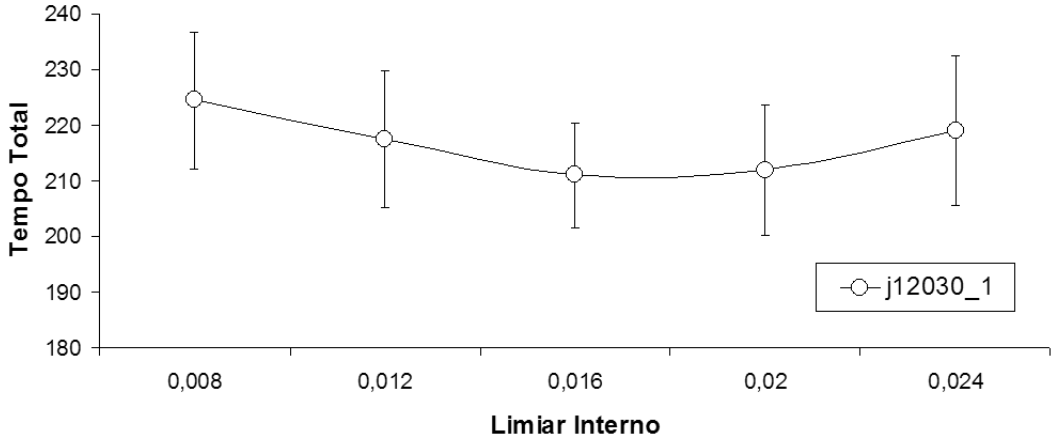
Os experimentos realizados com a instância de experimentação "j1201_1" destacam-se pelo fato de que, somente nesse caso, o Swarm-RCPSP abandona sua execução prematuramente sem obter um escalonamento completo válido. Como discutido no início da seção 6.3, esta instância é a que possui o menor número de restrições, tanto de interrelação quanto de interdependência, e o menor número de agentes para realizar os escalonamentos. Quando estas restrições são poucas em proporção ao tamanho do problema e o número de agentes é pequeno, o Swarm-RCPSP tem mais dificuldades em obter um escalonamento completo válido. Com isso, o número de possibilidades para explorar são maiores e, como o processo de tomada de decisão deste algoritmo é probabilístico e não existem mecanismos complexos de coordenação explícita implementados, fica mais difícil obter escalonamentos válidos. Tais escalonamentos ocorrem em uma quantidade drasticamente menor que a quantidade total de combinações possíveis.

A figura 6.5 mostra a porcentagem de tentativas falhas durante o processo que obteve os 20 escalonamentos pretendidos. Esta porcentagem aumenta significativamente quando o limiar interno chega a 0.04, o que produz uma tendência de 0.5. Nesse caso, aproximadamente 70% das tentativas de se obter um escalonamento são falhas.

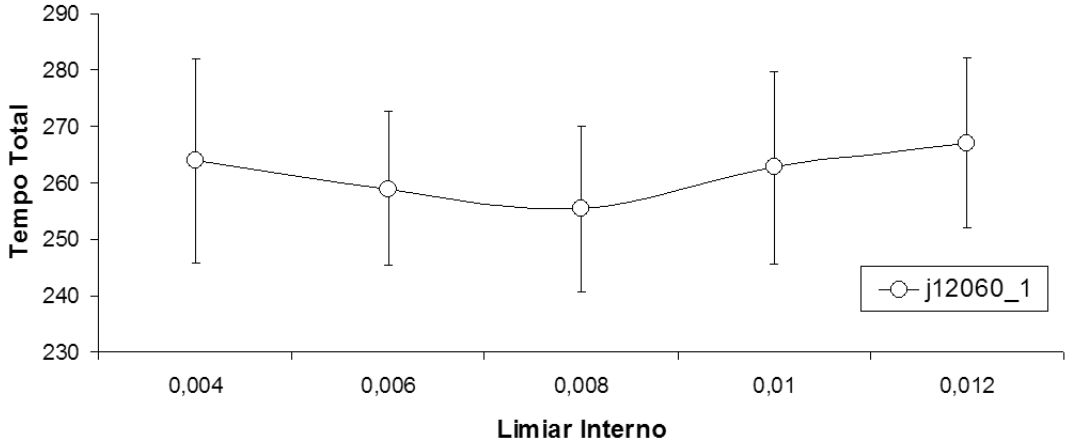
O parâmetro η do Swarm-RCPSP, discutido na seção 6.2, é utilizado para impor um limite às sucessivas tentativas de se obter um escalonamento completo válido e, com isso, garantir que o algoritmo seja executado em um tempo razoável. As tentativas falhas discutidas há pouco ocorrem porque este limite foi atingido. A figura 6.6 mostra a porcentagem de tentativas falhas de acordo com diferentes valores do parâmetro de insistência (η), para



(a)



(b)



(c)

Figura 6.4: Analisando a variação do tempo total do escalonamento de acordo com diferentes valores para o limiar interno, para cada instância de experimentação do DRCPSP.

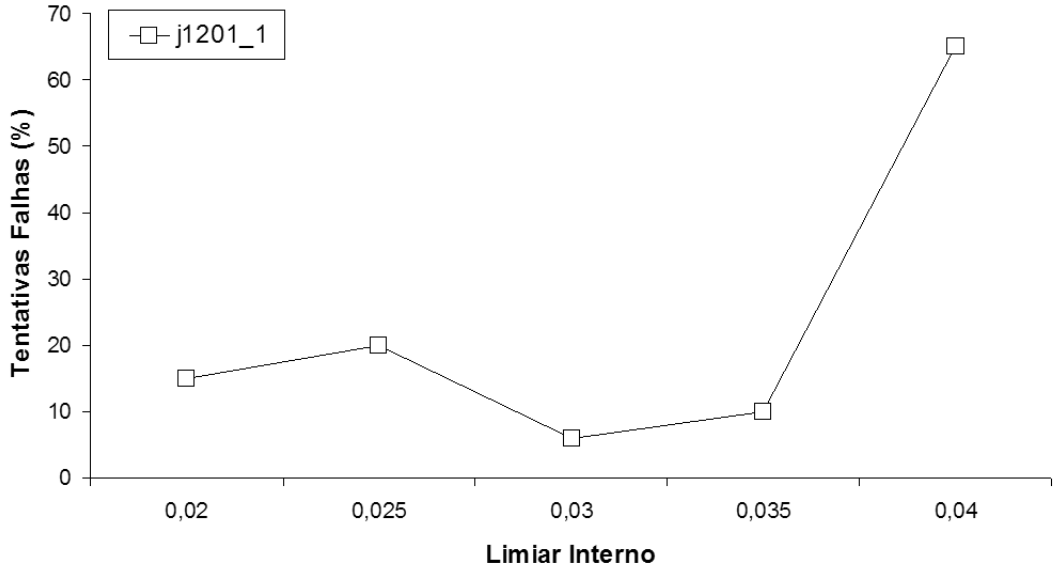


Figura 6.5: Analisando a porcentagem de tentativas falhas de acordo com diferentes valores para o limiar interno, para a instância de experimentação "j1201_1".

a instância de experimentação "j1201_1". Estas porcentagens são a média entre as tentativas falhas nos experimentos com os diferentes valores para o limiar interno. Pode-se ver que o número de falhas é menor quando a insistência η passa de 300.

No entanto, valores altos para a insistência têm impacto no número de interações que algoritmo utiliza para produzir um escalonamento válido e, conseqüentemente, no número de mensagens trocadas entre os agentes. Nos experimentos que envolvem a instância "j1201_1" é utilizado o valor 300 para a insistência. Para as demais instâncias, a insistência pode assumir valores menores que 50 e, ainda assim, o algoritmo não tem sua execução interrompida prematuramente. Nestes casos o valor 50 é utilizado. Com isso, nota-se que a insistência (η) é um parâmetro do Swarm-RCPSP que deve ser adequado às características específicas de cada instância do DRCPSP que pretende-se resolver.

A grandeza dos valores para os limiares internos que foram discutidos nesta seção é bem menor que os valores dos limiares internos discutidos na aplicação do mecanismo de polimorfismo ao E-GAP (capítulo 5). Esta diferença tem implicações na integração dos mecanismos de polimorfismo e adaptação do estímulo. Estas implicações e suas conseqüências serão detalhadas na seção 7, que discute a combinação dos mecanismos empregados separadamente nos algoritmos Swarm-GAP e Swarm-RCPSP.

6.3.2 Adaptação do Estímulo

O Swarm-RCPSP permite que os agentes adaptem os estímulos das tarefas que estão escalonando para obter escalonamentos com menor duração total. O mecanismo de adaptação do estímulo, apresentado na seção 4.6, faz com que os agentes modifiquem o estímulo de cada tarefa de acordo com a posição destas na linha de tempo do escalonamento. O estímulo de cada tarefa diminui com o passar do tempo e aumenta proporcionalmente em relação a unidade de tempo onde está seu início no escalonamento. Além de obter escalonamentos melhores, a adaptação dos estímulos permite que escalonamentos completos válidos sejam obtidos mais rapidamente e evita que o algoritmo seja interrompido

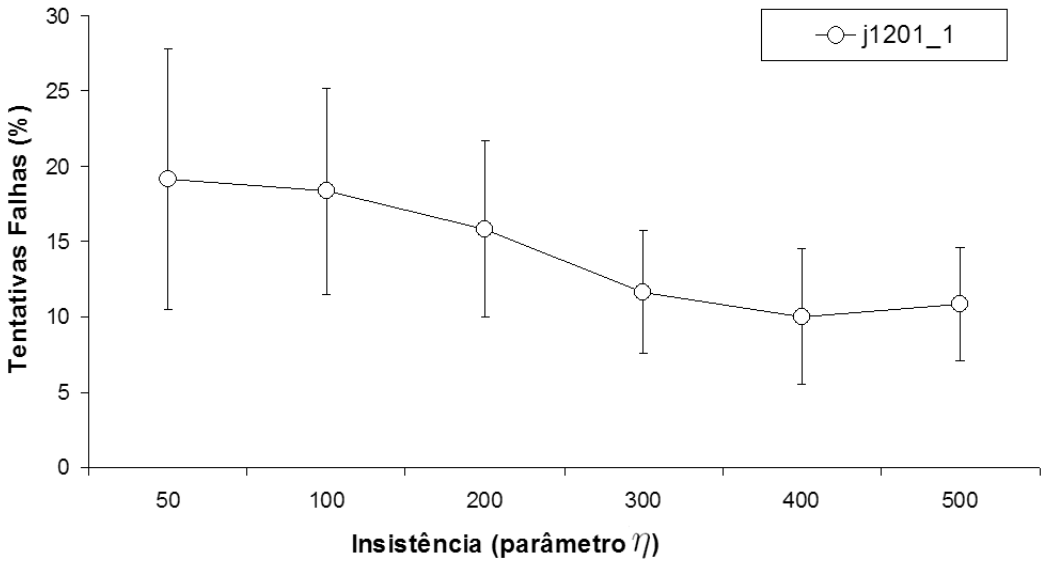


Figura 6.6: Analisando a porcentagem de tentativas falhas de acordo com diferentes valores para o parâmetro de insistência (η), para a instância de experimentação "j1201_1".

prematuramente sem sucesso.

A cada μ rodadas, os estímulos de todas as tarefas diminuem de acordo com a constante δ e aumentam proporcionalmente de acordo com o viés de atualização $\Gamma(j)$ e sua taxa de desconto α, conforme discutido na seção 4.6. Tarefas escalonadas com sucesso mais cedo no escalonamento terão um maior estímulo, fazendo com que os agentes tenham uma maior tendência de escolheê-las na mesma ordem em que foram escalonadas previamente em um escalonamento de melhor qualidade.

Os agentes tentam realizar sucessivos escalonamentos válidos de acordo com o parâmetro χ. Nos experimentos realizados nesta seção os agentes adaptam seus estímulos baseados em 200 escalonamentos válidos, $\chi = 200$. Os parâmetros mencionados anteriormente (μ, δ e α) foram experimentados com diferentes valores proporcionais e este número de escalonamentos. Diferentes valores para o número de repetições χ, desde que mantidas as proporções com os demais parâmetros, não alteram os resultados discutidos aqui.

Para medir o impacto na eficiência da adaptação do estímulo dos diferentes valores que seus parâmetros (δ, α e μ) podem assumir, foram experimentadas diferentes combinações de diferentes pares (δ, α) de acordo com diferentes intervalos de adaptação μ. A figura 6.7 mostra o tempo total dos escalonamentos obtidos na experimentação da instância "j1201_1" para os parâmetros da adaptação do estímulo (δ, α) e de acordo com o intervalo de adaptação μ. Esta instância foi adotada aqui pois, como dito anteriormente, este foi o único caso em que o algoritmo abandonou prematuramente sua execução sem obter um escalonamento completo válido. Como se pode ver, as melhores combinações destes parâmetros são $(0.003, 0.01)$ com intervalo de adaptação 5 e $(0.001, 0.01)$ com intervalo de adaptação 10. Apesar de não haver praticamente diferença no tempo total dos escalonamentos obtidos por estas duas configurações de parâmetros, a primeira obteve o melhor tempo total para o escalonamento desta instância (164) e será adotada pelo Swarm-RCPSP para aplicar o mecanismo de adaptação do estímulo, inclusive para as demais instâncias de experimentação.

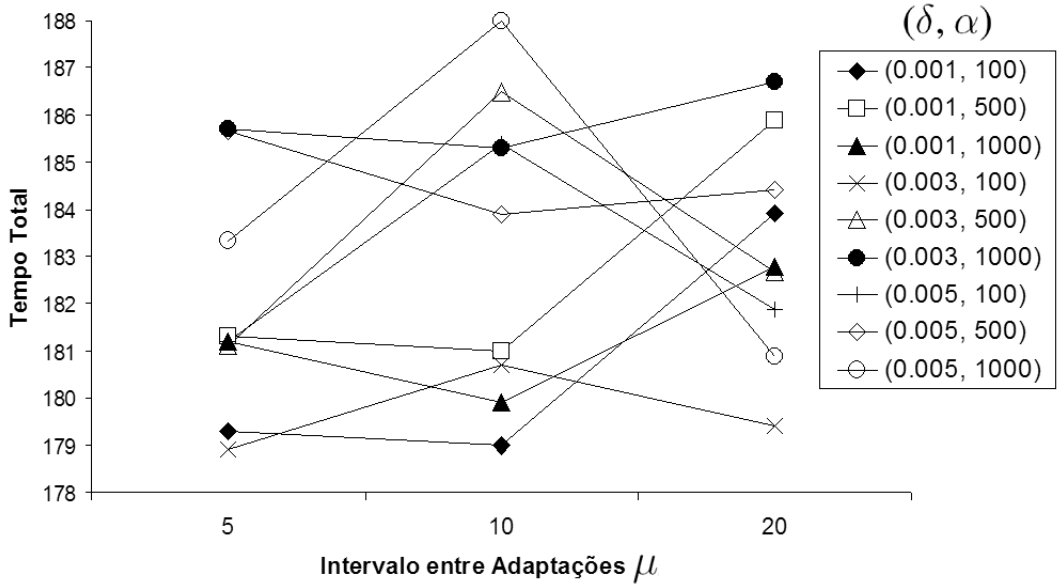


Figura 6.7: Comparando os diferentes valores para os parâmetros da adaptação do estímulo (δ, α) de acordo com o intervalo de adaptação μ, para a instância de experimentação "j1201_1".

Além de trazer uma melhora na qualidade do escalonamento obtido pelo Swarm-RCPSP, como é melhor discutido na proxima seção, a adaptação do estímulo das tarefas faz com que o algoritmo não mais seja interrompido sem que um escalonamento completo válido seja obtido.

6.3.3 Restrição na Comunicação

Agentes executando o Swarm-RCPSP trocam mensagens para lidar com os interrelacionamentos e as interdependências entre as tarefas. Sem mecanismos de coordenação explícita, os agentes precisam se comunicar para conseguir lidar com tarefas que têm que ser escalonadas simultaneamente e para ter conhecimento de quais tarefas podem ser escolhidas a partir do escalonamento prévio de suas antecessoras. Estas questões dificultam significativamente a obtenção de escalonamentos completos válidos no DRCPSP.

No experimento realizado nesta seção são introduzidas falhas no canal de comunicação entre os agentes. Tais falhas fazem com que, com uma probabilidade de 20%, as mensagens enviadas pelos agentes não sejam recebidas pelo seu destinatário. Com essa perda, nenhum escalonamento completo pode ser estabelecido pelo Swarm-RCPSP. Em geral, para as instâncias do DRCPSP experimentadas, foram escalonadas de forma válida 18% das tarefas. Com probabilidades de perda menores que 20% foi possível obter ao menos um escalonamento completo válido. A figura 6.8 mostra a porcentagem média de tarefas escalonadas de forma válida para cada uma das instâncias de experimentação para o DRCPSP.

Essa queda significativa de desempenho é esperada. Apesar do algoritmo continuar obtendo escalonamentos parciais válidos, como os agentes perdem muitas mensagens que convocariam outros agentes a se engajar no escalonamento de tarefas interrelacionadas (irmãs), tais tarefas são escalonadas com sucesso em menor número. Isso se reflete nas tarefas sucessoras destas, que não podem ser escalonadas sem que suas predecessoras

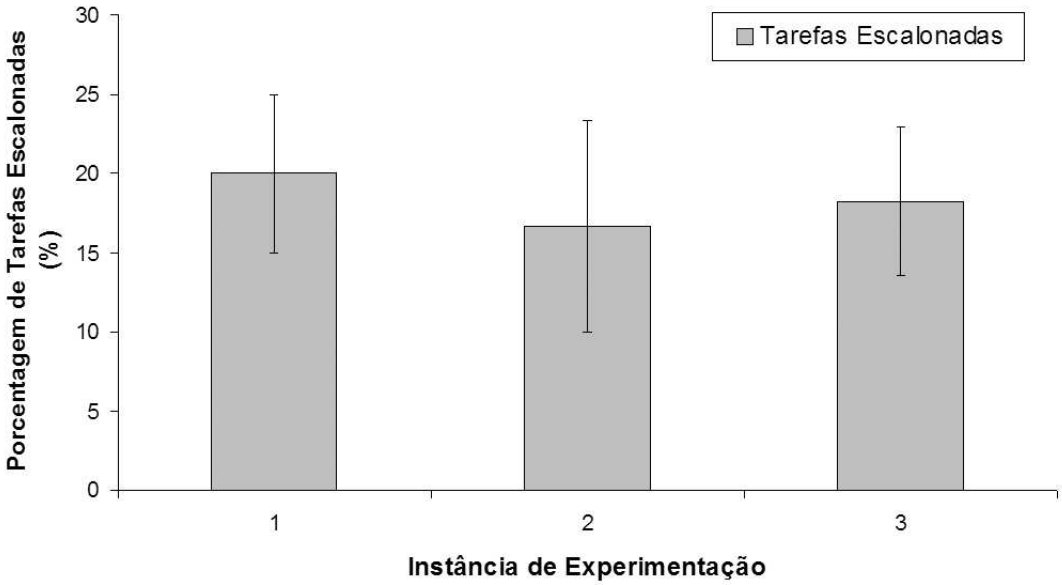


Figura 6.8: Comparando o número de tarefas escalonadas de forma válida para cada uma das instâncias de experimentação para o DRCPSP.

estejam escalonadas de forma válida.

A intrincada relação entre as tarefas no DRCPSP torna fundamental que os agentes consigam se comunicar para que o escalonamento completo seja obtido. Contudo, mesmo perdendo muitas mensagens, o Swarm-RCPSP conseguiu obter escalonamentos parciais válidos.

6.3.4 Desempenho

Nos experimentos realizados nesta seção os resultados obtidos pelo Swarm-RCPSP são comparados com os obtidos por uma abordagem gulosa distribuída e com o melhor resultado obtido pelo estado-da-arte na solução das instâncias do RCPSP utilizadas neste trabalho (BRUCKER; KNUST, 2000; LUO et al., 2006). A figura 6.9 mostra a comparação entre o tempo total para o escalonamento obtido pelo Swarm-RCPSP, com e sem a adaptação do estímulo, pelo estado-da-arte em algoritmos aproximativos centralizados e por um algoritmo guloso, para as instâncias de experimentação do DRCPSP.

O algoritmo de estratégia gulosa apresentou o pior resultado, obtendo escalonamentos com tempo total de duas a seis vezes maior que o melhor tempo total conhecido para tais escalonamentos, dependendo da complexidade da instância de experimentação. O Swarm-RCPSP obteve escalonamentos cujo tempo total foi, no máximo, o dobro do melhor tempo total conhecido. O algoritmo mostrou-se significativamente mais eficiente que a abordagem gulosa, principalmente para as instâncias mais complexas. Com relação a melhor solução centralizada conhecida, era esperado que o Swarm-RCPSP não obtivesse resultados equivalentes. Além de não tratar o problema de maneira distribuída, como dito anteriormente, os algoritmos utilizados para obter estas soluções utilizam diversas técnicas que não têm comprometimento com sua aplicabilidade real. Mesmo assim, pode-se observar que a diferença entre os resultados obtidos com o Swarm-RCPSP e a melhor solução conhecida não têm uma variação acentuada conforme as instâncias ficam mais complexas.

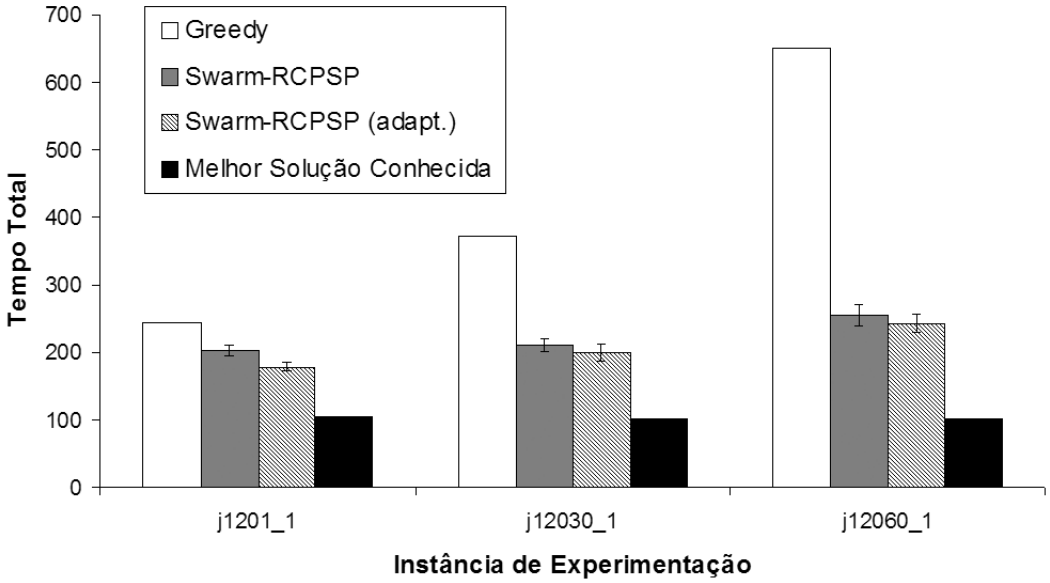


Figura 6.9: Comparando o tempo total obtido pelo Swarm-RCPSP, com e sem a adaptação do estímulo, pelo estado-da-arte em algoritmos aproximativos centralizados e por um algoritmo guloso, para as instâncias de experimentação do DRCPSP.

Como mostra a figura 6.9, a adaptação do estímulo foi efetiva para a instância "j1201_1", onde esta obteve uma melhora de mais de 10% no tempo total do escalonamento em relação a sua não utilização. Além disso, como comentado anteriormente, deve-se destacar que a adaptação do estímulo permitiu que o algoritmo sempre obtivesse uma solução válida (escalonamento completo) para a instância "j1201_1", o que não ocorre sem ela. Nas demais instâncias, por serem bem mais complexas e envolverem um número maior de agentes, não se obteve uma diferença significativa, chegando a 5% em média para ambas.

A figura 6.10 mostra o número de mensagens trocadas pelos agentes executando o Swarm-RCPSP, com e sem a adaptação do estímulo, e o algoritmo guloso, para as instâncias de experimentação do DRCPSP. Os agentes executando o algoritmo guloso, dado seu mecanismo, trocam poucas mensagens em relação ao Swarm-RCPSP. Este último teve uma diminuição significativa na troca de mensagens para a instância "j1201_1" quando a adaptação do estímulo foi utilizada, chegando a 35% em média. Nas outras instâncias não houve mudança significativa.

6.4 Conclusão

O algoritmo apresentado neste capítulo implementa vários mecanismos da abordagem proposta, os quais são: o mecanismo de tomada de decisão (seção 4.3), com ênfase no processo que lida com a interdependência entre as tarefas (subseção 4.3.2); o mecanismo da tendência (seção 4.2); e o mecanismo de adaptação do estímulo (seção 4.6), que é utilizado para que os agentes possam buscar um escalonamento completo e válido para as tarefas no menor intervalo de tempo possível.

O Swarm-RCPSP foi experimentado para que se pudesse determinar o limiar interno ótimo que minimiza o tempo total do escalonamento. Isso foi feito pois foi adotado para este algoritmo o mesmo estímulo ótimo calculado para o Swarm-GAP (seção 5.3.1).

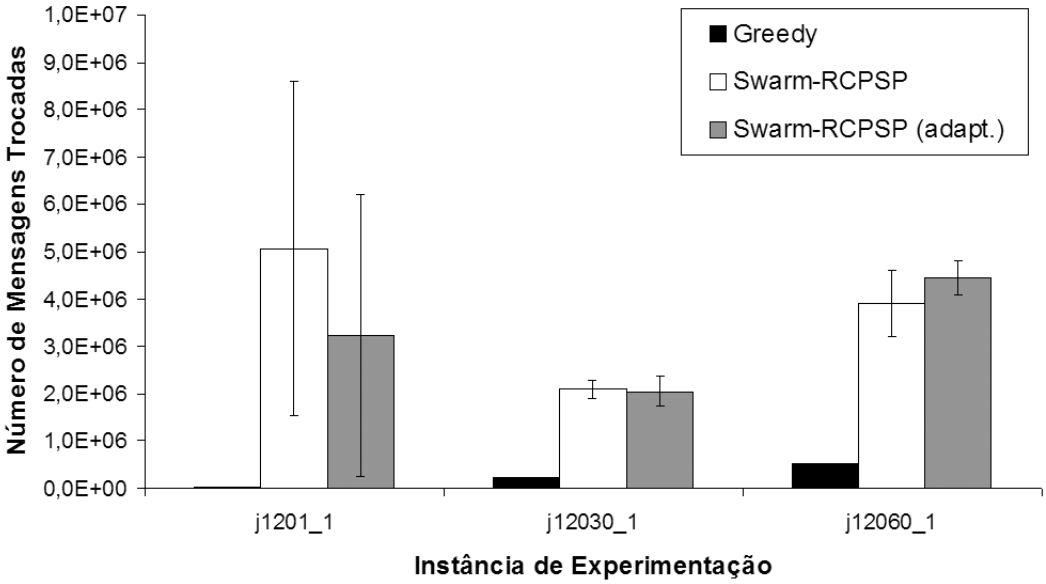


Figura 6.10: Comparando o número de mensagens trocadas pelo agentes executando o Swarm-RCPSP, com e sem a adaptação do estímulo, e um algoritmo guloso, para as instâncias de experimentação do DRCPSP.

Os agentes obtiveram escalonamentos completos válidos quando tomaram suas decisões baseados na sua tendência utilizando o referido estímulo ótimo e este limiar interno ótimo determinado para o problema em questão.

Os agentes executando o Swarm-RCPSP foram capazes de adaptar os estímulos das tarefas, partindo do ótimo, aumentando e diminuindo estes estímulos de acordo com seu tempo de início nos melhores escalonamentos obtidos. Esta adaptação permitiu em alguns casos obter um menor tempo total para o escalonamento e garantiu que o algoritmo sempre chegasse a uma solução. As interrelações e as interdependências entre as tarefas foram tratadas adequadamente pela troca de mensagens simples. A introdução de falhas nos canais de comunicação entre os agentes, apesar de implicar em uma redução de desempenho, foi suportada pelo algoritmo.

O Swarm-RCPSP foi comparado com um algoritmo guloso (*greedy*) e mostrou-se significativamente melhor que este último, sendo que a principal diferença entre os dois algoritmos é a adoção dos mecanismos da abordagem proposta pelo Swarm-RCPSP. Como esperado, o Swarm-RCPSP não obteve resultados melhores que a melhor solução conhecida. Contudo, a comparação entre os resultados obtidos pelo Swarm-RCPSP e pelo estado-da-arte em algoritmos aproximativos centralizados mostrou que a diferença entre estes cresceu moderadamente em relação ao aumento da complexidade das instâncias.

Concebido com base na abordagem proposta, o algoritmo Swarm-RCPSP é simples e efetivo na solução do DRCPSP de maneira aproximada. Todos os mecanismos da abordagem proposta empregados no Swarm-RCPSP, mencionados no início desta seção, foram experimentados e validados. Os resultados empíricos destes experimentos mostraram que a abordagem proposta, baseada em modelos teóricos da divisão do trabalho pelos insetos sociais, pode ser utilizada com sucesso para o escalonamento distribuído de tarefas, fazendo com que os agentes tomem suas decisões de forma coordenada, com baixa comunicação, para maximizar o desempenho do sistema. Cabe ressaltar que o desempenho, neste caso, está atrelado a minimização do tempo total para o escalonamento das tarefas.

7 COMBINANDO OS CENÁRIOS (E-GAP + DRCPSP)

Uma vez que os experimentos conduzidos com a abordagem proposta foram realizados de maneira que seus mecanismos foram testados e validados isoladamente nos capítulos 5 e 6, este capítulo discute em detalhes como estes mecanismos podem ser integrados e analisa o impacto desta integração sobre os resultados obtidos nos referidos capítulos.

O presente capítulo está organizado da seguinte forma: na seção 7.1 é apresentada uma discussão inicial sobre a utilização da abordagem proposta com todos os seus mecanismos aplicados simultaneamente; A análise de como a combinação dos mecanismos da abordagem proposta afeta os resultados obtidos neste trabalho é analisado na seção 7.2; e, finalmente, na seção 7.3 a aplicação combinada de todos os mecanismos da abordagem proposta é demonstrada.

7.1 Discussão

Este trabalho propõe uma abordagem para coordenar um grande número de agentes em um sistema multiagente para atuar em cenários complexos, utilizando como base os modelos teóricos da divisão do trabalho em colônias de insetos sociais. Esta abordagem baseia-se nestes modelos teóricos para conduzir os agentes a decidirem individualmente quais tarefas percebidas estes irão executar.

Na abordagem proposta, o objetivo do sistema multiagente é descrito utilizando a plataforma TÆMS. Através de uma estrutura construída nessa plataforma decompõe-se este objetivo em tarefas. Estas tarefas, por sua vez, são organizadas em uma estrutura hierárquica onde são descritas as suas características e definidas as suas interrelações e interdependências. Os agentes se coordenam para atuar nas tarefas através de sua estrutura organizacional. A organização de um SMA, utilizando a abordagem proposta, parte da estrutura de tarefas em TÆMS e se completa de forma emergente em função das escolhas dos agentes sobre quando e quais tarefas executar.

De maneira geral, a medida de qualidade desta organização está relacionada à qualidade com que as tarefas são executadas e o tempo total demandado por esta execução. A execução das tarefas pelos agentes, na abordagem proposta, está associada à sua alocação ou ao seu escalonamento. Considera-se a alocação das tarefas se o agente tiver que deliberar sobre que tarefa realizar imediatamente, sem tempo suficiente para planejar. Por outro lado, considera-se o escalonamento das tarefas se o agente puder deliberar previamente (planejar) quais tarefas executar, e em que ordem, em um intervalo de tempo futuro.

Com isso objetiva-se que o agente mais competente na realização de uma tarefa tenha alta probabilidade para escolher esta tarefa, podendo atribuir qualidade e (ou) reduzir o custo dessa execução, e que as tarefas sejam realizadas no menor espaço de tempo possível. Os mecanismos da abordagem proposta são capazes de lidar com estes objetivos,

como foi detalhado no capítulo 4. Contudo, nos experimentos realizados, tais objetivos foram testados de maneira independente: a maximização da competência dos agentes alocados as tarefas (capítulo 5), através do E-GAP; e a minimização do tempo total do escalonamento de tarefas (capítulo 6), através do DRCPSP.

A organização de um SMA, quando definida na plataforma TÆMS, considera que os agentes devem escalonar as tarefas de maneira a maximizar a qualidade e a minimizar o tempo total destes escalonamentos, simultaneamente. Dado que a abordagem que está sendo proposta neste trabalho aplica-se a estrutura organizacional definida utilizando a plataforma TÆMS, é preciso considerar como esta se comporta quando os mecanismos, que tratam separadamente seus objetivos, forem combinados.

O processo de tomada de decisão dos agentes é o principal responsável pelo sucesso da abordagem em cada um dos objetivos mencionados, individualmente. A tendência dos agentes para escolher as tarefas conduz a uma processo de tomada de decisão probabilístico que, amparado pela comunicação entre os agentes, produzem os resultados discutidos nos capítulos 5 e 6.

Tais resultados, que serão detalhados e melhor discutidos na seção seguinte (7.2), apontam que o processo de tomada de decisão para maximizar a competência dos agentes alocados às tarefas e minimizar o tempo total do escalonamento não pode ser único. Apesar da equação da tendência (4.1) utilizar o estímulo e o limiar interno no cálculo da tendência de escolha do agente, os valores para esta tendência que obtém a otimização de cada um dos critérios difere significativamente.

Para que seja possível atingir todos os objetivos da abordagem proposta, é preciso realizar um processo de tomada de decisão combinado, mas não unificado. A análise da tendência relativa às tarefas deve ser feita em dois passos. Isso significa que os agentes devem primeiro, utilizando os mecanismos aplicados no Swarm-GAP, decidir quais tarefas alocar. Depois disso, os agentes passam a utilizar os mecanismos do Swarm-RCPSP para decidir quando estas tarefas serão escalonadas.

Assim, é possível combinar todos os mecanismos da abordagem proposta, experimentados em separado a partir da implementação do Swarm-GAP e do Swarm-RCPSP, mantendo o desempenho de cada um individualmente. A seção 7.3 apresenta detalhes de como esse processo de dois passos se dá.

7.2 Análise dos Resultados

O cerne da abordagem proposta está na tendência de alocação das tarefas utilizando o modelo teórico de divisão do trabalho dos insetos sociais. Com isso, a busca pela melhor solução que a abordagem proposta realiza está atrelada a equação da tendência (equação 4.1) e aos seus parâmetros. Nos resultados dos experimentos, discutidos nas seções 5.3.1 e 6.3.1, pode-se notar que os valores para a tendência que levam o Swarm-GAP a atingir seu objetivo é diferente destes valores para o Swarm-RCPSP.

No Swarm-GAP, os limiares internos dos agentes variam de acordo com suas competências, estando uniformemente distribuídos no intervalo $[0.01, 0.1]$. Os agentes podem utilizar o mecanismo de especialização e modificar estes limiares. Foi determinado empiricamente um estímulo ótimo que maximiza a qualidade das alocações e fazem com que o algoritmo em questão seja bem sucedido. Este estímulo ótimo e os valores adotados para o limiar interno fazem com que a tendência dos agentes em alocar as tarefas seja muito pequena, sendo sempre menor que 0.2. Esta tendência, inserida no processo de decisão implementado no Swarm-GAP, restringe os agentes a alocar as tarefas para as quais estes

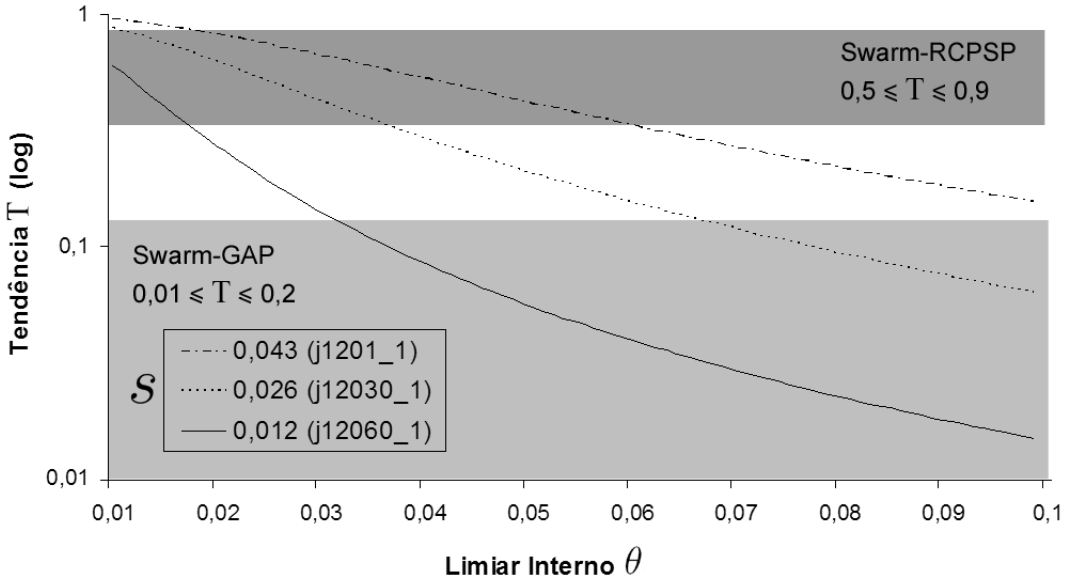


Figura 7.1: Analisando a variação da tendência utilizando o estímulo ótimo para as instâncias de experimentação do DRCPSP, de acordo com diferentes valores para o limiar interno.

têm maior competência, que é o objetivo neste caso.

Já no Swarm-RCPSP, os agentes não tem competências específicas associadas às tarefas. Optou-se então por, utilizando o estímulo ótimo para o E-GAP, determinar empiricamente um valor ótimo para o limiar interno. Este valor ótimo é modificado pelo mecanismo de adaptação do estímulo, mas sua grandeza não é significativamente alterada (vide os parâmetros adotados para este mecanismo na seção 6.3.2). A tendência produzida pelos valores para o estímulo e o limiar interno ótimo é sempre maior que 0.5. Esta é uma tendência de alocação grande e está em uma faixa de valores diferente da utilizada pelo Swarm-GAP (abaixo de 0.2, mencionada no parágrafo anterior).

A figura 7.1 mostra a variação da tendência, considerando o estímulo ótimo adotado para cada umas das instâncias dos experimentos com o DRCPSP, com o limiar interno variando no intervalo $[0.01, 0.1]$. Estes estímulos ótimos foram utilizados nos experimentos com o Swarm-RCPSP, como discutido na seção 6.3.1, tendo sido determinados da mesma forma que o estímulo ótimo para o E-GAP. Pode-se ver na figura em questão as faixas de valor para a tendência que conduzem o Swarm-GAP (área em cinza claro) e o Swarm-RCPSP (área em cinza escuro) a obter seu melhor desempenho. Fica claro, observando as curvas para cada valor de estímulo, que as faixas de valores que os limiares internos devem assumir, para que a tendência adequada seja obtida, são diferentes em cada caso. Por exemplo, para a curva do estímulo de valor 0.012, os limiares internos dos agentes devem assumir valores entre $[0.01, 0.016]$ para que a tendência possa variar na faixa de valores utilizada pelo Swarm-RCPSP, e, em contrapartida, devem assumir valores entre $[0.025, 0.1]$ para que a tendência possa variar na faixa de valores utilizada pelo Swarm-GAP.

Desta forma, se os limiares internos dos agentes forem configurados, através do polimorfismo (equação 4.3), para que o Swarm-RCPSP possa lidar com diferentes competências e busque a maximização destas competências na atribuição das tarefas aos agentes, não se obterá mais a minimização do tempo total do escalonamento. A diferença entre os

Tabela 7.1: Competência k_ij de cada agente para o escalonamento de cada tarefa, considerando uma estrutura de tarefas hipotética em TÆMS

	Tarefas								
	A	**B**	**C1**	**C2**	**D1**	**D2**	**E1**	**E2**	**E3**
V	0.8	0.8	0	0	0.2	0.6	0.8	0.8	0
X	0.4	0.4	0	0	0.5	0.2	0.3	0.3	0
Y	0	0	0.5	0.7	0	0	0	0	0.4
Z	0	0	0.8	0.2	0	0	0	0	0.8

valores para a tendência, que obtém os melhores resultados nos experimentos realizados, mostram que a busca simultânea pela maximização da competência dos agentes alocados às tarefas e a minimização do tempo total do escalonamento não pode ser obtida apenas combinando os dois algoritmos diretamente.

Na próxima seção será mostrado como contornar essa situação utilizando um processo de tomada de decisão em dois passos. Para ilustrar este processo um terceiro algoritmo, que combina adequadamente as características do Swarm-GAP e do Swarm-RCPSP, será apresentado e discutido.

7.3 Aplicação em Problemas Definidos em TÆMS

Para discutir a aplicação da abordagem proposta em uma estrutura de tarefas completa em TÆMS, o exemplo de DRCPSP apresentado na seção 6.1 será aqui estendido e novamente discutido. A estrutura de tarefas que representa este exemplo se encontra na figura 6.2. Pretende-se com isso ilustrar o processo de tomada de decisão em dois passos, em discussão neste capítulo, e demonstrar como a abordagem proposta é capaz de obter a minimização do tempo total de escalonamento e a maximização das competências empregadas nas alocações, simultaneamente e sem perda de eficiência.

Para que o DRCPSP contemple todos os aspectos relevantes para este trabalho relacionados com uma estrutura de tarefas em TÆMS, é preciso que as tarefas tenham uma qualidade diferente de zero associada ao seu escalonamento. Para isso, o exemplo utilizado para mostrar o DRCPSP modelado com uma estrutura de tarefas em TÆMS pode ser estendido de maneira que a qualidade do escalonamento das tarefas varie de acordo com a competência dos agentes que escaloná-las.

Com isso, considera-se que os agentes têm diferentes competências para o escalonamento das tarefas. Supõem-se, então, que os agentes percebem a estrutura de tarefas mostrada na figura 6.2 e têm suas competências individuais dadas pela tabela 7.1. Esta modificação faz com que escalonamentos de mesma duração possam ter qualidades diferentes dependendo do agente que estiver escalonando cada tarefa.

O escalonamento mostrado na figura 6.1 continua sendo um escalonamento possível para o problema em questão mesmo em sua versão entendida. Neste caso, a qualidade do escalonamento pode ser determinada da mesma forma que a recompensa W é calculada no GAP (equação 5.4), ou seja, como o somatório das competências dos agentes que estão escalonando as tarefas em determinado escalonamento. A recompensa obtida para o escalonamento em questão é $W = 4.5$. A figura 7.2 mostra um outro escalonamento possível para o mesmo problema. Nesse caso, algumas tarefas foram escalonadas por agentes com mais competência em relação a estas tarefas. Desta forma, a qualidade do escalonamento sobe para $W = 5.7$.

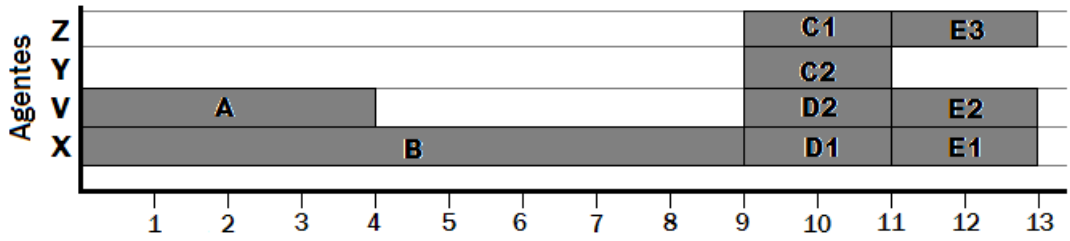


Figura 7.2: Exemplo de um escalonamento possível para uma instância hipotética simples do DCRPSP, cuja recompensa obtida é $W = 5.7$.

Para conseguir esse aumento da qualidade, os agentes precisam primeiro escolher as tarefas de acordo com sua tendência ($T_{\theta_j}(s*)$)). Esta tendência, neste momento, é calculada a partir do estímulo ótimo (denominado aqui de $s*$, calculado pela equação 5.5) e de seu limiar interno (θ_j), que é configurado de acordo com sua competência para a alocação de cada tarefa j. Em seguida, tais tarefas passam por um outro processo de tomada de decisão. Desta vez, os agentes escolhem as tarefas de acordo com sua tendência ($T_{\theta*}(s*)$)), utilizando o mesmo estímulo ótimo $s*$, porém não mais terão seus limiares configurados de acordo com suas competências, mas sim utilizarão o limiar interno ótimo (denominado aqui de $\theta*$, calculado pela equação 6.2). As tarefas escolhidas neste segundo passo serão escalonadas na linha de tempo do agente de acordo com a disponibilidade de um intervalo de tempo adequado o mais próximo possível ao tempo de início do escalonamento.

O algoritmo Swarm-RCPSP pode ser modificado para que se possa ver como este processo pode ser implementado na prática. O algoritmo 9 traz uma versão do Swarm-RCPSP que incorpora esta modificação utilizando o algoritmo 2, que é parte do algoritmo Swarm-GAP. As linhas de 18 a 24 contém o primeiro passo do processo de tomada de decisão. Neste passo, todas as tarefas disponíveis para o escalonamento são escolhidas utilizando o mecanismo empregado no Swarm-GAP e passam a compor um conjunto de tarefas que serão analisadas pelo segundo passo do processo, que inicia na linha 22 do algoritmo em questão. A partir do conjunto construído na primeiro passo, as tarefas serão escalonadas como no Swarm-RCPSP.

A adoção dos dois passos na tomada de decisão pode levar a escalonamentos mais eficientes em qualidade, mas que acabam tendo um tempo maior de duração total. A figura 7.3 mostra um escalonamento possível para o problema em discussão nesta seção que obtém recompensa $W = 6.1$, mas cujo tempo total de duração é 17 ao invés de 13 como nos outros escalonamentos discutidos aqui. A abordagem proposta tende a não priorizar a recompensa ou a duração dos escalonamentos. Ambas as situações podem ocorrer uma vez que a aboragem proposta utiliza um processo de tomada de decisão probabilístico. O DTC, como discutido na seção 2.3, é o responsável pelos escalonamentos na plataforma TÆMS. Este algoritmo, diferentemente da abordagem proposta, permite que seja explicitamente definida a prioridade entre os critérios de otimização para as estruturas de tarefas.

Os mecanismos de especialização dos limiares internos (seção 4.5) e de adaptação do estímulo (seção 4.6), utilizados pelos algoritmos para obter soluções ainda mais eficientes, podem ser empregados da mesma forma, sem perda de qualidade.

Algorithm 9 Swarm-TAEMS($agentId$)

1: **loop**
2: $ev \leftarrow$ waitEvent()
3: **if** $ev =$ *task perception* **then**
4: $\mathcal{J} \leftarrow$ *set of new unscheduled tasks*
5: $taskSched \leftarrow$ newTaskSched()
6: **for all** $t \in \mathcal{J}$ **do**
7: $taskSched$.addTask(t)
8: **else**
9: $taskSched \leftarrow$ newTaskSched(receiveMessage())
10: $\tau \leftarrow taskSched$.getTasks()
11: *{tenta obter um escalonamento completo válido}*
12: $\mathcal{S} \leftarrow \emptyset$
13: **repeat**
14: *{tenta obter o melhor escalonamento parcial válido escolhendo primeiro as tarefas para as quais o agente é mais competente}*
15: **repeat**
16: $\mathcal{S}' \leftarrow \mathcal{S}$
17: $\mathcal{M} \leftarrow \emptyset$
18: **for all** $t \in \tau \cap \mathcal{S} \ / \ \forall h \in P(t) \ \exists h \in \mathcal{S} \ \vee \ P(t) = \emptyset$ **do**
19: $taskAlloc$.addTask(t)
20: **if** tryAllocate($taskAlloc$,t) **then**
21: $\mathcal{M} \leftarrow \mathcal{M} \cup t$
22: **for all** $t \in \mathcal{M}$ **do**
23: **if** !trySchedule(t, $\mathcal{S}'$, $\mathcal{M}$, $taskSched$) **then**
24: $\mathcal{M} \leftarrow \mathcal{M} \cap t$
25: communicateNeighbours($\mathcal{S}'$, $taskSched$)
26: *{tenta obter um escalonamento parcial válido para as tarefas escolhidas nos passos anteriores}*
27: **repeat**
28: **for all** $t \in \mathcal{M}$ **do**
29: trySchedule(t, $\mathcal{S}'$, $\mathcal{M}$, $taskSched$)
30: communicateNeighbours($\mathcal{S}'$, $taskSched$);
31: **until** successfullyScheduled($\mathcal{M}$, $taskSched$) or η
32: *{realiza a adaptação dos estímulos}*
33: **if** successfullyScheduled($\mathcal{M}$, $taskSched$) and $|\mathcal{S}^*| < |\mathcal{S}'|$ **then**
34: $\mathcal{S}^* \leftarrow \mathcal{S}'$
35: adaptStimulus($\mathcal{M}$, $taskSched$)
36: **until** χ
37: *{abandona a execução se não obteve um escalonamento parcial válido}*
38: **if** successfullyScheduled($\mathcal{M}$, $taskSched$) **then**
39: $\mathcal{S} \leftarrow \mathcal{S}^*$
40: $\mathcal{S}^* \leftarrow \emptyset$
41: communicateNeighbours($\mathcal{S}$, $taskSched$)
42: **else**
43: **return** fail attempt
44: *{tenta realizar o escalonamento das tarefas não escolhidas, caso existam}*
45: **until** successfullyScheduled(τ, $taskSched$)
46:

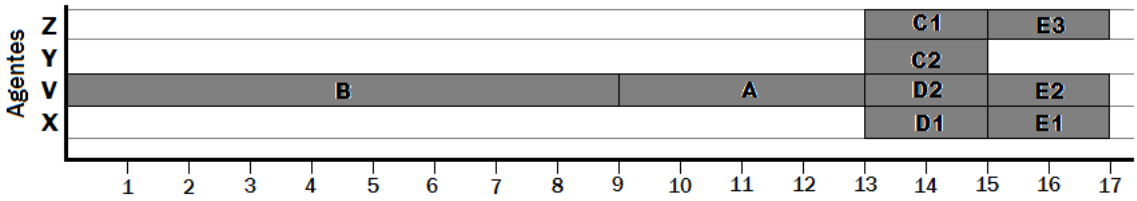


Figura 7.3: Exemplo de um escalonamento possível para uma instância hipotética simples do DCRPSP, cuja recompensa obtida é $W = 6.1$.

7.4 Conclusão

Os experimentos apresentados nos capítulos 5 e 6 mostraram que a abordagem proposta é eficaz e robusta para o tratamento do E-GAP e do DRCPSP representados como estruturas de tarefas em TÆMS. O presente capítulo discutiu a aplicação da abordagem proposta em problemas, também representados em TÆMS, mas que consideram a busca pelo escalonamento com menor duração possível, objetivo do DRCPSP, em conjunto com a busca pela maior qualidade na atribuição dos agentes às tarefas, objetivo do E-GAP. Problemas com estas características são o escopo de aplicação da abordagem proposta.

A análise detalhada do comportamento dos mecanismos empregados em cada um dos algoritmos mostrou que a simples combinação do processo de tomada de decisão não é possível. Um processo de tomada de decisão em dois passos foi apresentado e sua aplicação foi ilustrada em uma versão extendida do Swarm-RCPSP. Mostrou-se que os mecanismos que buscam escalonamentos com a maior recompensa, empregados no Swarm-GAP, e de menor duração, empregados no Swarm-RCPSP, podem ser utilizados em passos isolados e com parâmetros diferentes, não produzindo qualquer interferência mútua. Desta forma, os resultados obtidos separadamente nos experimentos realizados neste trabalho (capítulos 5 e 6) são válidos quando se considera a aplicação da abordagem proposta integralmente.

8 CONCLUSÃO

Este trabalho propõe uma abordagem para a emergência da organização dos agentes em relação à sua atuação nas tarefas que compõem os objetivos de um sistema multiagente, visando obter a coordenação destes agentes na alocação e (ou) escalonamento destas tarefas em cenários dinâmicos, parcialmente observáveis e de larga escala (cenários complexos). Esta abordagem adota uma paradigma baseado em colônias de insetos sociais, onde existem plenas evidências de sucesso ecológico a despeito da não existência de coordenação explícita. Estes insetos se adaptam às mudanças no ambiente e às necessidades da colônia usando os mecanismos de divisão de trabalho explicados aqui.

A abordagem proposta faz com que os agentes decidam probabilisticamente sobre quais tarefas estes irão se engajar. Esta probabilidade é influenciada pelo estímulo da tarefa, por sua demanda em relação às outras tarefas (interrelações e interdependências) e por um limiar interno do agente. O estímulo é atualizado de acordo com a eficiência dos escalonamentos obtidos, fazendo com que os agentes busquem escalonar as tarefas no menor tempo total. O limiar interno pode ser determinado pela competência do agente em realizar uma tarefa e é atualizado de acordo com seu desempenho no escalonamento de cada uma delas. Com isso, os agentes mais competentes para cada tarefa tendem a escolhê-las.

Extensões de problemas de pesquisa operacional denominados *Generalized Assignment Problem* (GAP) (MARTELLO; TOTH, 1990) e *Resource-Constrained Project Scheduling Problem* (RCPSP) (BRUCKER et al., 1999), as quais são denominadas E-GAP e DRCPSP, serviram como cenários para a experimentação e validação da abordagem proposta. Tais problemas se complementam para cobrir o escopo de cenários em que a abordagem proposta pode atuar, permitindo que seu desempenho seja analisado detalhadamente e comparado com outras abordagens. Dois algoritmos distribuídos aplicados a estes problemas foram propostos, implementados e experimentados (Swarm-GAP e Swarm-RCPSP). Mostrou-se que o paradigma dos insetos sociais pode ser utilizado para coordenar sistemas multiagente quando atuando em cenários complexos.

A seguir, na seção 8.1, são revistas e detalhadas as contribuições deste trabalho e, na seção 8.2, são apresentados as questões em aberto que podem ser cobertas em trabalhos futuros.

8.1 Contribuições

As principais contribuições deste trabalho, tanto para o avanço do estado-da-arte no estudo e desenvolvimento de SMA, quanto na modelagem e aplicação de técnicas de inteligência de enxames em problemas computacionais, são revistas e detalhadas a seguir:

- A abordagem proposta, apresentada no capítulo 4 para coordenação distribuída de agentes em cenários complexos é nova, eficaz e robusta. Assume-se, neste trabalho, que a tomada de decisão deve ser realizada de maneira distribuída. Existem diversos cenários reais em que não se pode centralizar este processo dada a privacidade das informações individuais, dada a necessidade de se ter um determinado nível de tolerância a falhas, etc. Todos os mecanismos desta abordagem, apresentados no capítulo 4, foram experimentados e validados, como mostram os capítulos 5 e 6. Os resultados empíricos destes experimentos mostraram que a abordagem proposta, baseada em modelos teóricos da divisão do trabalho pelos insetos sociais, pode ser utilizada com sucesso para a alocação e (ou) escalonamento distribuído de tarefas, fazendo com que os agentes tomem suas decisões de forma coordenada, com baixa comunicação, para obter o melhor desempenho do sistema possível. Esta abordagem é integrada ao TÆMS, como se vê no capítulo 7, e compõem uma técnica para a emergência da organização dos agentes na alocação da estrutura de tarefas em um SMA, podendo ser vista como um método aproximado para a coordenação de agentes atuando em problemas descritos em TÆMS.

- O algoritmo Swarm-GAP, apresentado na seção 5.2, concebido a partir da abordagem proposta, resolve de forma aproximada o E-GAP, definido na seção 5.1. O Swarm-GAP foi experimentado para que se pudesse determinar o estímulo ótimo que maximiza o desempenho do algoritmo. Os agentes obtiveram bons resultados quando tomaram suas decisões baseados na sua tendência de alocar cada tarefa, utilizando este estímulo ótimo e seus limiares internos configurados de acordo com suas competências, como mostra a seção 5.3.1. Além disso, os agentes executando o Swarm-GAP também foram capazes de adaptar seus limiares internos para se especializar em tarefas par as quais têm maior competência, mesmo sem conhecê-las, através de um processo de adaptação mediante recompensa. A especialização permitiu melhorar o desempenho do sistema de maneira geral, como pode ser visto na seção 5.3.2. As interrelações AND entre as tarefas foram tratadas adequadamente pela troca de mensagens simples, como mostra a seção 5.3.3, e a redução nos canais de comunicação entre os agentes, apesar de implicar em uma redução de desempenho, foi contornada pelo algoritmo, como discute a seção 5.3.4. Em experimentos realizados em um simulador abstrato, mostrados na seção 5.3.5.2, o LA-DCOP mostrou-se melhor que o Swarm-GAP em situações normais, considerando o desempenho com relação as competências empregadas nas alocações e no número de mensagens trocadas. Sob restrições de comunicação, o Swarm-GAP se equivaleu ou foi melhor que o LA-DCOP. Nos experimentos realizados no simulador da RoboCup Rescue, mostrados na seção 5.3.5.3, o Swarm-GAP e o LA-DCOP obtiveram desempenhos semelhantes. A complexidade computacional do LA-DCOP é significativamente maior que a do Swarm-GAP, como foi discutido na seção 2.4.2.

- O algoritmo Swarm-RCPSP, apresentado na seção 6.2, também concebido a partir da abordagem proposta, resolve de forma aproximada o DRCPSP, definido na seção 6.1. O Swarm-RCPSP foi experimentado para que se pudesse determinar o limiar interno ótimo que minimiza o tempo total do escalonamento. Isto foi feito pois foi adotado o mesmo estímulo ótimo calculado para o Swarm-GAP. Os agentes obtiveram bons resultados quando tomaram suas decisões baseados na sua tendência de alocar cada tarefa, utilizando o referido estímulo ótimo e este limiar interno ótimo determinado para o problema em questão, como mostra a seção 6.3.1. Além disso,

os agentes executando o Swarm-RCPSP foram capazes de adaptar os estímulos das tarefas, permitindo em alguns casos obter um menor tempo total para o escalonamento e garantindo que o algoritmo sempre chegue a uma solução, como pode ser visto na seção 6.3.2. As interrelações e as interdependências entre as tarefas foram tratadas adequadamente pela troca de mensagens simples. A redução nos canais de comunicação entre os agentes, apesar de implicar em uma redução de desempenho, foi razoavelmente suportada pelo algoritmo, como mostra a seção 6.3.3. A comparação do Swarm-RCPSP com o algoritmo guloso e com a melhor solução conhecida para cada instância experimentada, como discuti a seção 6.3.4, mostrou que o primeiro é capaz de obter bons resultados apesar do processo de tomada de decisão distribuído. O algoritmo guloso obteve resultados significativamente piores que os obtidos pelo Swarm-RCPSP, sendo que a principal diferença entre as duas abordagens é a tomada de decisão dos agentes orientada pela tendência e a variação do estímulo adotados pelo Swarm-RCPSP.

- Extensões e modificações nos modelos teóricos de organização dos insetos sociais para resolver problemas de forma distribuída foram propostas, avaliadas e validadas. O processo de tomada de decisão dos insetos sociais, baseado nos modelos de tendência e especialização dos limiares internos, foram aplicados pela primeira vez em problemas de alocação e escalonamento distribuídos e de larga escala, obtendo a eficácia esperada se comparado a abordagens proeminentes. A utilização do polimorfismo para capturar a competência dos agentes foi proposta e validada. O processo de recrutamento dos insetos sociais foi utilizado de para a resolução da interdependência entre tarefas e permitiu que os agentes se coordenassem, conseguindo realizar escalonamentos complexos com sucesso. Foi proposto um modelo inédito de variação do estímulo, fortemente inspirado no comportamento dos insetos sociais, que permitiu que escalonamentos mais eficientes pudessem ser obtidos.

8.2 Trabalhos Futuros

Este trabalho deixa questões em aberto que apontam direções para trabalhos futuros, as quais são:

- O processo de tomada de decisão em dois passos, discutido no capítulo 7, foi a solução empregada para que a busca pelos critérios de otimização relacionados a competência (explorados no E-GAP) e do tempo do escalonamento (explorados no DRCPSP) fossem combinados. No entanto, não foi aprofundada a discussão quanto a interferência entre estes dois critérios. Pode-se enchergar isto como um problema de otimização multiobjetivo. Problemas deste tipo consistem em estabelecer o ponto Pareto eficiente entre seus múltiplos objetivos. A idéia é combinar os diversos critérios (dois no caso deste trabalho) de maneira que a otimização de cada um deles não produza piora na qualidade de algum outro. Estudar como os mecanismos empregados no E-GAP e no DRCPSP podem ser combinados utilizando métodos de otimização multiobjetivo é um relevante trabalho em aberto.

- Os cenários estudados neste trabalho, apesar de complexos e desafiadores, não exploraram estruturas de tarefas (TS) em TÆMS de concepção mais sofisticada. Tanto o E-GAP, quanto o DRCPSP, não utilizaram TSs com diversos níveis hierárquicos em sua árvore. A altura máxima para a árvore de tarefas da TS explorada por estes

problemas foi 3. Além disso, não foram experimentados alguns interrelacionamentos (QAFs) possíveis de serem especificadas em TÆMS. Apesar da abordagem proposta teoricamente lidar com todas essas questões, é relevante estender sua validação, encontrando outros problemas que, modelados em TÆMS, produzam TSs mais sofisticadas. Para isso também é importante considerar outras abordagens resolvam tais problemas ou que hajam soluções e dados para teste que possam ser utilizados como parâmetro de comparação.

- O TÆMS permite que os atributos (qualidade, custo e duração) de seus métodos tenham probabilidades associadas a seus valores. Um método, por definição, pode ter uma probabilidade de 10% de ter duração X e de 90% de ter duração Y, dependendo do que acontecer no cenário. Desta forma o TÆMS lida com a incerteza acerca dos resultados que a execução de um método pode produzir. A abordagem proposta neste trabalho não trata esta questão. Conhecer as abordagens propostas para TÆMS que buscam lidar com isso e estudar mecanismos que possam ser integrados a abordagem proposta, contribuindo para que esta lide com tais incertezas, é um trabalho em aberto.

- Quando se trata problemas de alocação e (ou) escalonamento dinâmicos, considera-se que os agentes estão tomando suas decisões enquanto resolvem o problema. Neste trabalho este dinamismo foi tratado considerando que os agentes alocavam ou escalonavam as tarefas de forma independente de sua execução. Nas simulações com o DRCPSP, por exemplo, não foi considerado explicitamente que, enquanto os agentes escolhem como escalonar as tarefas, estes estão executando as primeiras tarefas que foram escalonadas. Da mesma forma no E-GAP, quando o agente escolhe alocar a tarefa, não se considera que ele irá executá-la em seguida. Contudo, é relevante estudar o comportamento da abordagem quando se considera que o agente despende algum esforço computacional para executar uma tarefa e não poderia tomar novas decisões nesse mesmo instante. No caso do E-GAP, hipoteticamente é como se o agente estivesse executando a tarefa quando decide alocá-la, e não devem haver maiores implicações. Já no caso do DRCPSP, como o agente utiliza um processo incremental para a construção do escalonamento, é possível que o Swarm-RCPSP apresente vantagens significativas se comparado com outras abordagens. O Swarm-RCPSP faz com que os agentes deliberem sobre um conjunto de tarefas e não mais altere seu escalonamento, permitindo que estas tarefas possam ser executadas. Outras abordagens, que precisem fazer o escalonamento completo de uma só vez, necessariamente teriam desvantagens nesse caso.

- Um aspecto interessante que interfere na aplicação real de abordagens para a alocação e (ou) escalonamento de tarefas é a associação entre o tempo de execução de uma tarefa com a competência do agente que está alocando-a. Cada tarefa teria uma determinada duração dependendo do agente que passar a alocá-la, interferindo diretamente no tempo total do escalonamento. Essa questão não foi abordada neste trabalho. Estudar o comportamento da abordagem proposta diante de escalonamentos nesse tipo de situação, comparando-a com outras abordagens que lidem adequadamente com essa questão, é um trabalho relevante a ser realizado.

REFERÊNCIAS

BONABEAU, E.; THERAULAZ, G.; DORIGO, M. **Swarm Intelligence**: from natural to artificial systems. New York, USA: Oxford University Press, 1999. 307p.

BROOKS, R. A. Intelligence without Representation. **Artificial Intelligence**, [S.l.], v.47, p.139–159, 1991.

BRUCKER, P. et al. Resource-constrained project scheduling: notation, classification, models, and methods. **European Journal of Operational Research**, [S.l.], v.112, n.1, p.3–41, January 1999.

BRUCKER, P.; KNUST, S. A linear programming and constraint propagation-based lower bound for the RCPSP. **European Journal of Operational Research**, [S.l.], v.127, n.2, p.355–362, December 2000.

CAMPOS, M. et al. Dynamic Scheduling and Division of Labor in Social Insects. **Adaptive Behavior**, [S.l.], v.8, n.2, p.83–95, 2000.

CICIRELLO, V.; SMITH, S. Wasp-like Agents for Distributed Factory Coordination. **Journal of Autonomous Agents and Multi-Agent Systems**, Amsterdam, v.8, n.3, p.237–266, May 2004.

DECKER, K. S.; LESSER, V. R. Quantitative Modeling of Complex Computational Task Environments. In: INTERNATIONAL WORKSHOP ON DISTRIBUTED ARTIFICIAL INTELLIGENCE, 12., 1993, Hidden Valley, Pennsylvania. **Proceedings...** [S.l.: s.n.], 1993. p.67–82.

DECKER, K. S.; LESSER, V. R. Designing a Family of Coordination Algorithms. In: INTERNATIONAL CONFERENCE ON MULTI-AGENT SYSTEMS, 1., 1995, San Francisco, CA, USA. **Proceedings...** Menlo Park: California: AAAI Press, 1995.

DORIGO, M. **Optimization, Learning and Natural Algorithms**. 1992. Tese (Doutorado em Ciência da Computação) — Politecnico di Milano, Milão.

DORIGO, M.; DI CARO, G.; GAMBARDELLA, L. Ant Colony Optimization: A new meta-heuristic. In: CONGRESS ON EVOLUTIONARY COMPUTATION, 1999, Washington D.C., USA. **Proceedings...** [S.l.]: IEEE Press, 1999. v.2, p.1470–1477.

FERREIRA JÚNIOR, P. R.; BAZZAN, A. L. C. Distributed Meeting Schedule through Cooperative Mediation: analysing OptAPO's performance in a complex scenario. In: INTERNATIONAL WORKSHOP ON DISTRIBUTED CONSTRAINT REASONING, DCR, 6., 2005. **Proceedings...** [S.l.: s.n.], 2005. p.101–113.

GORDON, D. The Organization of Work in Social Insect Colonies. **Nature**, [S.l.], v.380, p.121–124, 1996.

GORDON, D. **Formigas em ação**: como se organiza uma sociedade de insetos. [S.l.]: J. Zahar Ed., 2002. 144p.

HORLING, B.; BENYO, B.; LESSER, V. Using Self-Diagnosis to Adapt Organizational Structures. In: INTERNATIONAL CONFERENCE ON AUTONOMOUS AGENTS, 5., 2001, Montreal. **Proceedings...** New York: ACM Press, 2001. p.529–536.

HORLING, B.; LESSER, V. A survey of multi-agent organizational paradigms. **Knowledge Engineering Review**, [S.l.], v.19, n.4, p.281–316, 2004.

HORLING, B. et al. **The TAEMS White Paper**. [S.l.]: University of Massachusetts at Amherst, USA, 1999.

ISHIDA, T.; YOKOO, M.; GASSER, L. An Organization Approach to Adaptative Production Systems. In: NATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE, 1990. **Proceedings...** [S.l.: s.n.], 1990. p.52–58.

KIRN, S.; GASSER, L. Organizational Approaches to Coordination in Multi-Agent Systems. **Informationstechnik und Technische Informatik (IT+TI)**, [S.l.], v.1, n.4, p.23–29, 1998.

KITANO, H. RoboCup Rescue: a grand challenge for multi-agent systems. In: INTERNATIONAL CONFERENCE ON MULTIAGENT SYSTEMS, 4., 2000, Boston, USA. **Proceedings...** Los Alamitos: IEEE Computer Society, 2000. p.5–12.

KOLISCH, R.; SPRECHER, A. PSPLIB – A project scheduling problem library. **European Journal of Operational Research**, [S.l.], v.96, n.1, p.205–216, January 1997.

KUBE, R.; BONABEAU, E. Cooperative Transport by Ants and Robots. **Robotics and Autonomous Systems**, [S.l.], v.30, n.1/2, p.85–101, 2000.

LEMAITRE-LEÓN, C.; EXCELENTE-TOLEDO, C. Multiagent Organization Approach. In: IBEROAMERICAN WORKSHOP ON DISTRIBUTED ARTIFICIAL INTELLIGENCE AND MULTI-AGENT SYSTEMS, 2., 1998. **Proceedings...** [S.l.: s.n.], 1998.

LESSER, V. Reflections on the Nature of Multi-Agent Coordination and Its Implications for an Agent Architecture. **Autonomous Agents and Multi-Agent Systems**, [S.l.], v.1, p.89–111, January 1998.

LESSER, V. R. Evolution of the GPGP/TÆMS domain-independent coordination framework. In: INTERNATIONAL JOINT CONFERENCE ON AUTONOMOUS AGENTS AND MULTIAGENT SYSTEMS, AAMAS, 1., 2002, Bologna, Italy. **Proceedings...** New York: ACM Press, 2002. p.1–2.

LUO, X. et al. An Improved PSO Algorithm for Resource-Constrained Project Scheduling Problem. **Intelligent Control and Automation**, [S.l.], v.1, n.1, p.3514–3518, June 2006.

MAHESWARAN, R. T. et al. Taking DCOP to the Real World: efficient complete solutions for distributed multi-event scheduling. In: INTERNATIONAL JOINT CONFERENCE ON AUTONOMOUS AGENTS AND MULTIAGENT SYSTEMS, AAMAS,

3., 2004, New York, USA. **Proceedings...** Washington: DC: IEEE Computer Society, 2004. v.1, p.310–317.

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

MARTELLO, S.; TOTH, P. **Knapsack problems**: algorithms and computer implementations. [S.l.]: John Wiley & Sons, Inc., 1990. 308p.

MERKLE, D.; MIDDENDORF, M.; SCHMECK, H. Ant Colony Optimization for Resource-Constrained Project Scheduling. **IEEE Transactions on Evolutionary Computation**, [S.l.], v.6, n.4, p.333–146, 2002.

MODI, P. J. et al. An Asynchronous Complete Method for Distributed Constraint Optimization. In: INTERNATIONAL JOINT CONFERENCE ON AUTONOMOUS AGENTS AND MULTIAGENT SYSTEMS, 2., 2003, Melbourne, Australia. **Proceedings...** New York: ACM Press, 2003. p.161–168.

MUSLINER, D. J. et al. Coordinated Plan Management Using Multiagent MDPs. In: AAAI SPRING SYMPOSIUM ON DISTRIBUTED PLAN AND SCHEDULE MANAGEMENT, 2006. **Proceedings...** [S.l.: s.n.], 2006.

NAIR, R. et al. Task Allocation in the Rescue Simulation Domain: a short note. In: BIRK, A.; CORADESCHI, S. (Ed.). **RoboCup 2001**: robot soccer world cup V. Berlin: Springer-Verlag, 2002. p.751–754. (Lecture Notes in Computer Science, v.2377).

NAIR, R.; TAMBE, M.; MARSELLA, S. Team Formation for Reformation in Multiagent Domains like RoboCupRescue. In: KAMINKA, G.; LIMA, P.; ROJA, R. (Ed.). **RoboCup 2002**: robot soccer world cup VI. Berlin: Springer-Verlag, 2003. p.150–161. (Lecture Notes in Computer Science, v.2752).

NWANA, H. S.; LEE, L. C.; JENNINGS, N. R. Coordination in Software Agent Systems. **The British Telecom Technical Journal**, [S.l.], v.14, n.4, p.79–88, 1996.

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

PUTERMAN, M. L. **Markov Decision Processes**: discrete stochastic dynamic programming. New York, NY, USA: Wiley–Interscience, 2005. (Wiley Series in Probability and Statistics).

ROBISON, G. E. Regulation of Division of Labor in Insect Societies. **Annual Review of Entomology**, [S.l.], v.37, p.637–665, 1992.

RUSSEL, S.; NORVIG, P. **Artificial Intelligence**: a modern approach. [S.l.]: Prentice-Hall, 2003. 1057p.

SANTOS, F. **Estudo Comparativo de Métodos de Otimização de Restrições Distribuída**. [S.l.]: Instituto de Informática, UFRGS, 2007.

SCERRI, P. et al. Allocating tasks in extreme teams. In: INTERNATIONAL JOINT CONFERENCE ON AUTONOMOUS AGENTS AND MULTIAGENT SYSTEMS, AAMAS, 4., 2005, Utrecht, The Netherlands. **Proceedings...** New York: ACM Press, 2005. p.727–734.

SCHILLO, M. et al. Self-Organization in Multiagent Systems. In: INTERNATIONAL WORKSHOP ON MODELLING ARTIFICIAL SOCIETIES AND HYBRID ORGANIZATIONS, MASHO, 3., 2002. **Proceedings...** [S.l.: s.n.], 2002.

SHEHORY, O. et al. Agent cloning: an approach to agent mobility and resource allocation. **IEEE Communications Magazine**, [S.l.], v.36, n.7, p.58–67, 1998.

SKINNER, C.; BARLEY, M. Robocup Rescue Simulation Competition: status report. In: BREDENFELD, A.; JACOFF, A.; NODA, I.; TAKAHASHI, Y. (Ed.). **RoboCup 2005**: robot soccer world cup IX. Berlin: Springer-Verlag, 2006. p.632–639. (Lecture Notes in Computer Science, v.4020).

SO, Y.; DURFEE, E. An organizational self-design model for organizational change. In: WORKSHOP ON AI AND THEORIES OF GROUPS AND ORGANIZATIONS: CONCEPTUAL AND EMPIRICAL RESEARCH, 1993. **Proceedings...** [S.l.: s.n.], 1993. p.8–15.

SULTANIK, E.; MODI, P. J.; REGLI, W. C. On Modeling Multiagent Task Scheduling as a Distributed Constraint Optimization Problem. In: INTERNATIONAL JOINT CONFERENCE ON ARTIFICIAL INTELLIGENCE, IJCAI, 20., 2007. **Proceedings...** [S.l.: s.n.], 2007. p.1531–1536.

TARRANT, F.; BRIDGE, D. When Ants Attack: comparing ant algorithms for constraint problems. In: IRISH CONFERENCE ON ARTIFICIAL INTELLIGENCE AND COGNITIVE SCIENCE, 15., 2004. **Proceedings...** [S.l.: s.n.], 2004. p.167–176.

THERAULAZ, G.; BONABEAU, E.; DENEUBOURG, J. Response Threshold Reinforcement and Division of Labour in Insect Societies. **Royal Society of London Series B - Biological Sciences**, [S.l.], v.265, p.327–332, 2 1998.

WAGNER, T. A.; RAJA, A.; LESSER, V. R. Modeling Uncertainty and its Implications to Sophisticated Control in TEAMS Agents. **Autonomous Agents and Multi-Agent Systems**, [S.l.], v.13, n.3, p.235–292, November 2006.

WAGNER, T.; LESSER, V. R. Design-to-Criteria Scheduling: real-time agent control. In: SPRING SYMPOSIUM ON REAL-TIME AUTONOMOUS SYSTEMS, AAAI, 2000, Stanford, CA. **Proceedings...** [S.l.: s.n.], 2000. p.89–96.

WILSON, E. O. **Sociobiology**: the new synthesis. [S.l.]: Harvard Univ. Press, 2000.

WOOLDRIDGE, M. J. **An Introduction to MultiAgent Systems**. Chichester: John Wiley & Sons, 2002. 340p.

WOOLDRIDGE, M. J.; JENNINGS, N. R. Agent Theories, Architectures, and Languages: a survey. In: WORKSHOP ON AGENT THEORIES, ARCHITECTURES, AND LANGUAGES, 1994. **Proceedings...** Berlin: Springer-Verlag, 1994. p.1–39. (Lecture Notes in Artificial Intelligence, v.890).

ZHANG, W.; WITTENBURG, L. Distributed Breakout Revisited. In: NATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE, 18., 2002, Edmonton, Alberta, Canada. **Proceedings...** Menlo Park: CA: American Association for Artificial Intelligence, 2002. p.352–357.

ZOETHOUT, K.; JAGER, W.; MOLLEMAN, E. Task dynamics in self-organising task groups: expertise, motivational, and performance differences of specialists and generalists. **Autonomous Agents and Multi-Agent Systems**, Hingham, MA, USA, v.16, n.1, p.75–94, 2008.