Mecanismo de Negociação para Sistema de Escalonamento Dinâmico

Ana Madureira
GECAD – Grupo de Investigação em Engenharia do Conhecimento e Apoio à Decisão
Instituto de Engenharia – IPP
Porto, Portugal
amd@isep.ipp.pt

Nelson Sousa
GECAD – Grupo de Investigação em Engenharia do Conhecimento e Apoio à Decisão
Instituto de Engenharia – IPP
Porto, Portugal
nffs@isep.ipp.pt

Ivo Pereira
GECAD – Grupo de Investigação em Engenharia do Conhecimento e Apoio à Decisão
Instituto de Engenharia – IPP
Porto, Portugal
iasp@isep.ipp.pt

***Resumo* – Este artigo propõe um Mecanismo de Negociação para Escalonamento Dinâmico com recurso a Swarm Intelligence (SI). No Mecanismo de Negociação, os agentes devem competir para obter um plano de escalamento global. SI é o termo geral para várias técnicas computacionais que retiram ideias e inspiração nos comportamentos sociais de insectos e outros animais. Este artigo propõe uma abordagem híbrida de diferentes conceitos da Inteligência Artificial (IA), como SI, Negociação em Sistemas Multi-Agente (SMA) e Técnicas de Aprendizagem Automática (AA). Este trabalho concentra a sua atenção na negociação, processo através do qual múltiplos agentes auto-interessados podem chegar a acordo através da troca competitiva de recursos.**

Keywords: Negociação em Sistemas Multi-Agente; Swarm Intelligence; Self-Organization;

I. INTRODUÇÃO

A competição tem sido estudada em diversos campos, incluindo a psicologia, a sociologia e a antropologia. Os psicólogos sociais, por exemplo, estudam a natureza da competição, investigam o estímulo natural da competição e as suas circunstâncias, assim como a dinâmica de grupo, detectando como a competição emerge e quais os seus efeitos [1].

Os sistemas de Informação envolvendo a interacção de agentes autónomos apresentam um novo desafio nas ciências de computação e na engenharia de software. O problema refere-se à especificação de diversas formas de interacção entre os agentes. A interacção pode ser definida para permitir que os agentes coordenem as suas actividades e comportamentos, cooperando para atingir objectivos comuns, ou concorrer para melhor atingir os seus objectivos individuais.

Considerando os sistemas reais de fabrico compostos por múltiplos agentes autónomos, a negociação é uma importante forma de interacção que permite a grupos de agentes atingirem um acordo mútuo através de algum benefício, objectivo ou plano de escalonamento. Particularmente, porque os agentes são autónomos e não podem ser assumidos como benevolentes, os agentes devem influenciar os outros para convencê-los a agir de determinada forma. A negociação é fundamental para o controlo de tais dependências inter-agentes. O processo de negociação pode ser realizado de diferentes formas, mas não se torna claro quão sofisticado os agentes ou os protocolos de interacção devem ser para uma negociação bem sucedida em diferentes ambientes do mundo real.

A literatura refere diferentes definições para coordenação, podendo geralmente ser definida como *"o processo para garantir acções de actores independentes (agentes) num ambiente coerente de alguma forma"*[2]. O desafio surge com o sentido de identificar/desenvolver mecanismos que permitam aos agentes coordenar as suas acções de forma autónoma, sem supervisão humana, um requisito encontrado numa variedade de aplicações do mundo real, como o proposto neste trabalho.

O trabalho descrito neste artigo refere-se à resolução de problemas de escalonamento de sistemas de fabrico em ambientes dinâmicos que tirar partido das vantagens da inteligência colectiva [3] e da computação autónoma [4].

As secções apresentadas neste artigo estão organizadas da seguinte forma: na secção II, são descritos alguns trabalhos relacionados com a negociação para o escalonamento através de SMA. A secção III resume as características dos métodos de SI considerados neste trabalho, como a Optimização por Colónia de Formigas (*Ant Colony Optimization*) e a Optimização por Nuvem de Partículas (*Particle Swarm Optimization*). Na secção IV, é apresentada a arquitectura competitiva para Escalonamento Dinâmico com auto-regulação, e na secção V é descrito o Mecanismo de Negociação proposto, que integra as ideias de Inteligência Colectiva e Negociação em SMA. O estudo computacional é apresentado na secção VI. Finalmente, são apresentas algumas conclusões e perspectivadas algumas ideias para futuros trabalhos.

II. NEGOCIAÇÃO EM SISTEMAS MULTI-AGENTE

A negociação pode ser definida como uma forma de interacção em que um grupo de agentes com interesses conflituosos e um desejo de cooperar tentam chegar a um acordo mutuamente aceitável para a alocação de recursos escassos. Várias definições de negociação são propostas na literatura como "*processo pelo qual um grupo de agentes comunica para tentar chegar a acordos sobre algum assunto de interesse comum*"[3].

Em geral, a literatura, refere mecanismos de negociação baseados em:

- Contract Net Protocol (CNP) [6].
- Leilões: Uma forma de lidar com temas de negociação em ambiente dinâmico de afectação de recursos [7].
- Teoria de Jogos: Permite um melhor ambiente matemático para analisar, emular e capturar comportamentos humanos em contextos económicos [8].
- Argumentação: Propõe-se resolver restrições dos métodos anteriores, como a resposta de uma proposta só poder ser uma contra-proposta; e fazer com que as propriedades, dos produtos de uma negociação em curso, possam ser alteradas [9].

Refere-se ao protocolo e à estratégia como os dois componentes principais de um Mecanismo de Negociação. O protocolo define as regras do encontro entre os participantes da negociação, em geral, inclui um conjunto de normas que condicionam as propostas que os participantes da negociação são capazes de executar. A estratégia define as acções possíveis ou sequência de acções que os agentes podem fazer durante a negociação [9].

A negociação procura encontrar alguma afectação de recursos que seja aceite por todos os participantes. Uma vez que geralmente há uma grande quantidade de afectações possíveis, a negociação pode ser vista como uma pesquisa distribuída através de um espaço de possíveis acordos [10].

O problema da negociação pode ser definido como: dado um conjunto de agentes, um conjunto de recursos, uma afectação de recursos já conhecida e um conjunto de outras possíveis, a negociação procura descobrir uma afectação que seja melhor segundo algum critério, se esta existir. Para atingir esses objectivos, os agentes precisam de algum mecanismo que defina as regras do processo de procura, o que os agentes estão autorizados a dizer e quando, como as atribuições são calculadas, se os cálculos são feitos utilizando um algoritmo centralizado ou de forma distribuída, entre outros.

Um exemplo típico de um Mecanismo de Negociação é um leilão. Este mecanismo inclui um conjunto de jogadores com um vendedor e um número de potenciais compradores. O vendedor tem um recurso único (por exemplo, uma casa), e cada potencial comprador é dotado de dinheiro e uma avaliação específica dos recursos do vendedor. Os potenciais compradores fazem licitações sucessivas, propondo comprar os recursos por um preço específico. O mecanismo valida que nenhuma licitação oferece uma quantia em dinheiro inferior a uma licitação anterior. Trata-se de um protocolo do tipo *Ascending*. A maior licitação ganha.

Diferentes mecanismos podem ter propriedades diferentes. Um conjunto de características desejáveis para os mecanismos de negociação pode ser definidas, das quais se destacam [10]:

- Simplicidade - Um mecanismo que requer menos processamento computacional e sobrecarga de comunicação é preferível.
- Eficiência - Um mecanismo é eficiente se produzir um bom resultado. O que se entende por bom, no entanto, pode variar de um domínio para outro. Um critério utilizado é a *Pareto Optimality*, onde nenhum agente poderia ser melhor num domínio diferente, sem qualquer outro agente a ser pior. Outros critérios poderiam ser a *Global Optimality*, onde os benefícios dos agentes são maximizados ou minimizados.
- Distribuição - É preferível ter um Mecanismo de Negociação que não envolve um centro decisor. A centralização pode levar a gargalos de comunicação ou diminuição de confiabilidade devido a um único ponto de falha.
- Simetria - Implica que o mecanismo não seja tendencioso a favor ou contra algum agente com base em critérios inadequados. O que constitui um critério inadequado depende do domínio em questão.
- Estabilidade - Um mecanismo é estável se nenhum agente tem incentivo para o desviar de alguns acordos estratégicos ou um conjunto de estratégias. Num leilão, por exemplo, é possível exigir que nenhum agente minta, fazendo uma oferta falsa, ou que nenhum grupo de agentes possa formar alianças estratégicas para criar uma desvantagem para outros agentes.
- Flexibilidade - O mecanismo deverá conduzir a um acordo, mesmo que os agentes não tenham informação privada completa e correcta em relação às suas próprias decisões e preferências.

Num SMA, os agentes de uma negociação trabalham como seres egoístas que pretendem melhorar o seu objectivo individual, comunicando entre si para melhorar o seu próprio objectivo através de modificações na solução de outro agente. A negociação diminui as falhas de conhecimento e melhora as capacidades dos agentes [11].

Vários mecanismos de negociação têm sido propostos na literatura. No planeamento e programação de sistemas de produção, a negociação é usada para melhorar a qualidade das soluções finais. Zattar et al. [12] propõe uma adaptação on-line do plano de processamento com alternativas, através da aplicação de um protocolo de negociação adaptado de dois métodos de Usher [13]. É proposto em Singh et al. [14] um modelo baseado em dois agentes cooperativos que estão dispostos a utilizar objectivos do sistema, e são regidos pelo gestor dos processos e gestor de recursos. Em Kim e Cho [15] agentes de negociação são utilizados para "*alocar inúmeras ordens de muitos participantes para a formação da cadeia de abastecimento*".

O protocolo CNP facilita a interacção entre os agentes, implementando um mecanismo autónomo de negociação competitiva por meio de contrato. O CNP aparece pela primeira vez em 1980 numa rede de sensores acústicos simulados distribuídos [6]. O CNP tem algumas abordagens novas para melhorar algumas das falhas do protocolo original como *TRAnsportation COoperation NET,* TRACONET [16], *Cooperative Information Agent* CIA [17], *Trading-Partner Agreement* TPA [18], *Threshold* e *Degree of Availability* DoA [19].

A *Foundation for Intelligent Physical Agents* (FIPA), definiu uma norma de especificação de protocolo para maximizar a interoperabilidade entre sistemas de agentes

heterogéneos [20]. FIPA é uma associação que visa a partilha de conhecimentos dos seus membros no sentido de desenvolver formas de agentes de negociação em diferentes plataformas de comunicação e inter-operacionais entre si. O protocolo *Contract Net with Confirmation Protocol* (CNCP) foi inicialmente visto como uma extensão do CNP original amplamente aplicado em SMA.

Neste trabalho foi seguida uma adaptação do modelo FIPA-CNP utilizando a mesma classificação e esquema de interacção dos agentes.

III. SWARM INTELLIGENCE

A *Swarm Intelligence* é uma abordagem para a resolução de problemas de optimização com inspiração na inteligência colectiva dos enxames, dos comportamentos sociais de insectos e de outros animais [3].

A. *Optimização por Colónia de Formigas*

A Optimização por Colónia de Formigas (ACO) estuda os sistemas artificiais inspirando-se no comportamento real duma colónia de formigas. Simulam o comportamento de um conjunto de agentes (formigas) que cooperam para resolver um problema de optimização por meio de comunicações muito simples. A inteligência da colónia demonstrada pelas formigas baseia-se não só na cooperação entre os indivíduos mas também na coordenação de cada formiga individualmente.

A designação ACO é uma designação genérica que inclui os algoritmos baseados no comportamento de formigas. Desde o início da década de 90, quando o primeiro algoritmo ACO – *Ant System* - foi proposto em [21], diferentes algoritmos e aplicações bem sucedidas têm sido apresentadas [22].

O primeiro sistema baseado na Optimização por Colónia de Formigas foi introduzido por Marco Dorigo [21], sendo designado por Sistema de Formigas (*Ant System*) e o seu desempenho testado extensivamente nos problemas de caixeiro-viajante. Trata-se de uma heurística baseada em probabilidade, criada para a resolução de problemas computacionais que envolvem procura de caminhos em grafos. Este algoritmo foi inspirado na observação do comportamento das formigas ao saírem do formigueiro para encontrar comida.

Um ACO é um algoritmo iterativo. Em cada iteração são consideradas *m* formigas. Cada formiga constrói uma solução deslocando-se de vértice em vértice ao longo do grafo com a restrição de não se deslocar para um vértice já visitado. Em cada passo do processo de construção da solução, uma formiga selecciona o próximo vértice a ser visitado de acordo com um mecanismo estocástico influenciado pela feromona: se o nó *j* ainda não foi visitado, este poderá sê-lo com uma probabilidade que é proporcional à concentração de feromona associada ao caminho/aresta (*i, j*) que interliga os nós *i* e *j* [22].

B. *Optimização por Nuvem de Partículas*

A Optimização por Nuvem de Partículas (PSO) proposto em [23] é um algoritmo evolucionário inspirado no movimento colectivo dos bandos de pássaros, dos cardumes de peixes e de enxames de abelhas. De iteração em iteração, as potenciais soluções, designadas por "partículas", movem-se pelo espaço de pesquisa de acordo com uma regra que rege o seu movimento e que depende de três factores: a inércia (as partículas tendem a mover-se na direcção do movimento anterior), a memória (as partículas tendem a mover-se na direcção da melhor solução encontrada na sua trajectória) e a cooperação (as partículas tendem a mover-se na direcção do melhor solução global encontrada até ao momento).

A Optimização por Nuvem de Partículas (PSO) é uma técnica evolucionária baseada em populações, desenvolvida por Russell Eberhart e James Kennedy [23] que pretende simular um sistema social simplificado.

Estes algoritmos surgem como uma abstracção do comportamento biológico natural onde a procura por uma melhor posição é a pesquisa por uma solução óptima, sendo o conjunto de posições da partícula o espaço de pesquisa ou espaço de soluções possíveis. O comportamento de cada partícula baseia-se na sua experiência anterior e na das outras partículas com que se relaciona.

O algoritmo começa pela inicialização da posição actual e velocidade de todas as partículas [23]. De seguida, enquanto o critério de paragem não é atingido (normalmente número de iterações), em cada iteração é calculado o valor de aptidão (*fitness*) para cada partícula, sendo actualizado o melhor valor local de cada uma (*pBest*). Depois de todas as partículas estarem tratadas, é actualizado o melhor valor global (*gBest*) e são calculadas as novas velocidades e posições de todas as partículas, como se descreve de seguida. No final, a melhor solução é devolvida de acordo com *gBest*.

IV. COMPETITIVE DYNAMIC SCHEDULING ARCHITECTURE

A arquitectura *Competitive Dynamic Scheduling* (*CompeteSched*) consiste num sistema em que as comunidades de agentes modelam um sistema de produção sujeito a perturbações. Os agentes devem ser capazes de aprender e gerir o seu comportamento interno e suas relações com outros agentes autónomos, por meio de negociação, em conformidade com as políticas comerciais definidas pelo gestor de negócio.

A. *Descrição do Problema*

O problema foca a sua atenção no que se designa de *Extended Job-Shop Scheduling Problem* (EJSSP) [24], apresenta algumas extensões e diferenças em relação ao clássico *Job-Shop Scheduling Problem* (JSSP). Neste trabalho define-se uma tarefa como uma ordem de produção de um produto final, que pode se Simples ou Complexo [24]. Os principais elementos de um problema de EJSSP podem ser modelado como: um conjunto de tarefas multi-operações $J_1,\ldots,J_n$ devem ser escalonadas num conjunto de maquinas $M_1,\ldots,M_n$. d_j é a data de entrega da tarefa J_j. t_j é o tempo inicial de processo para a tarefa J_j. a existência de operações na mesma tarefa, em diferentes partes e componentes, processados simultaneamente em diferentes máquinas, seguidas de operações de montagem de componentes (tarefas de multi-nível).

Além disso o problema EJSSP deverá satisfazer as seguintes condições de restrição:

a) Existência de diferentes datas de lançamento de tarefas r_j e datas de entrega d_j.
b) A possibilidade de definição de prioridades das tarefas, reflectindo a importância de satisfazer as suas datas de entrega, sendo similar ao peso atribuído as tarefas na teoria de escalonamento.
c) Restrições de precedência sobre as operações das diferentes tarefas.
d) Novas tarefas podem chegar em intervalos imprevisíveis. As tarefas podem ser canceladas. Alterações nos atributos das tarefas alterados: tempos de processamento, datas de entrega e de lançamento, prioridades.
e) Cada operação O_{ijkl} deve ser processada numa única máquina do conjunto M_i, onde p_{ijkl} é o tempo de processamento da operação O_{ijkl} na máquina M_i.
f) Uma máquina pode processar mais que uma operação da mesma tarefa (recirculação).
g) A existência de máquinas alternativas, idênticas ou não.

B. *Modelo*

O sistema *CompeteSched* é um SMA onde os Agentes Recurso devem ser capazes de descobrir uma solução óptima ou quase-óptima local, através de algoritmos SI (ACO e PSO) negociando com os restantes agentes. O sistema deve ser capaz de lidar com dinamismo (chegada de novas tarefas, cancelamento de tarefas, mudança de atributos, etc.), mudando ou ajustando os parâmetros do algoritmo SI de acordo com a situação actual.

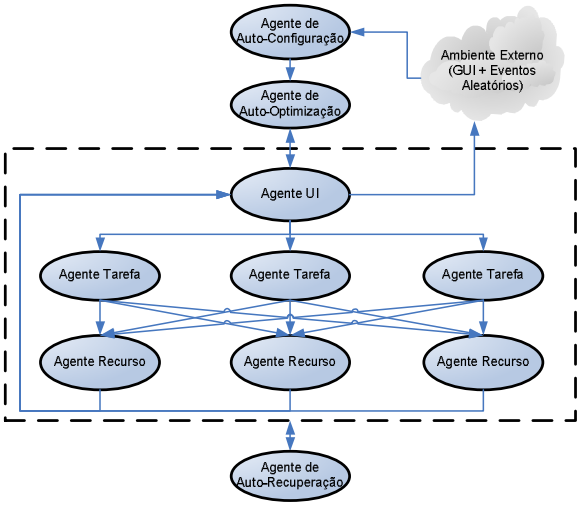

Figura 1. Arquitectura do sistema CompeteSched

A abordagem de escalonamento seguida é ligeiramente diferente da encontrada na literatura, onde cada Agente Recurso é responsável por optimizar o plano de escalonamento para um recurso/máquina através de ACO e PSO. Este considera um tipo específico de interacção social relativa à resolução de problemas concorrenciais, onde um grupo de agentes trabalha em conjunto para alcançar uma boa solução para o problema.

O problema original de escalonamento descrito na subsecção anterior é decomposto numa série de *Single Machine Scheduling Problems* (SMSP). Os Agente Recurso (que tem um método SI associado) obtêm uma solução local e depois negoceiam com o objectivo de superar as restrições locais e conseguir um plano global [20].

A arquitectura do sistema (Figura 1) é baseada em seis tipos de agentes. De forma a permitir uma comunicação contínua com o utilizador, foi implementado um agente controlador da interface gráfica (Agente UI). Além disso, este agente gera dinamicamente os Agentes Tarefa necessários de acordo com o número de tarefas do problema de escalonamento e atribui cada tarefa ao respectivo Agente Tarefa. No final aplica um mecanismo de reparação se a solução final não for exequível. Agentes Tarefa processam a informação necessária acerca da tarefa, i.e., são responsáveis pela geração dos tempos de processamento mais cedo e mais tarde e pela separação das diferentes operações das tarefas pelos Agentes Recurso respectivos.

Os Agentes Recurso são responsáveis por escalonar as operações que requerem processamento na máquina supervisionada pelo agente. Estes agentes executam Meta-heurísticas e procedimentos de pesquisa local de forma a encontrar os planos de escalonamento melhores possíveis e comunicar as soluções ao Agente UI para uma verificação de exequibilidade.

O Agente de Auto-Configuração é responsável por monitorizar o sistema para detectar alterações ocorridas no plano de escalonamento, tendo em vista a adaptação dinâmica. Com este agente, o sistema está preparado para lidar com dinamismo através da adaptação das soluções às perturbações externas.

O Agente de Auto-Optimização é responsável pela aplicação do módulo de aprendizagem proposto, para automaticamente afinar os parâmetros das Meta-heurísticas, de acordo com o problema a tratar. Este módulo proposto usa aprendizagem e experiência, uma vez que aplica RBC, e é descrito na subsecção seguinte.

O Agente de Auto-Recuperação é responsável pela monitorização dos outros agentes para fornecer capacidades de auto-recuperação. Com este agente, o sistema torna-se estável, mesmo se alguma falha ocorrer.

A abordagem utilizada para lidar com este problema é dividida em duas fases. Na primeira, o sistema aguarda que pelas soluções obtidas pelos agentes Recurso e, em seguida, aplica um mecanismo de reparação para adaptar algumas operações até uma solução viável ser obtida. Na segunda fase é estabelecido, um Mecanismo de Negociação entre os agentes relacionados no processo, a fim de interagir uns com os outros para a prossecução dos seus objectivos através de negociação.

V. MECANISMO DE NEGOCIAÇÃO

O objectivo principal deste trabalho é investigar questões relacionadas com a negociação em sistemas de fabrico

dinâmicos. Nesta secção, apresenta-se uma abordagem para dotar o sistema de escalonamento de negociação com inteligência. As principais contribuições estão resumidas nas seguintes subsecções.

A. Algoritmo de Negociação

Com este mecanismo, pretende-se dotar o sistema de inteligência colectiva para analisar o plano de escalonamento gerado pelos Agentes Recurso e melhorá-lo através da redução dos tempos paragem (tempo durante o qual uma máquina não está em funcionamento porque a próxima operação a processar ainda não pode ser executada devido à seu precedente não ter terminado o processamento).

Inicialmente, os Agentes Recurso são egoístas ao gerar a sua solução. O Agente UI analisa as diferentes soluções locais obtidas e aplica um mecanismo de reparação, deslocando as operações para a direita, de acordo com as relações de precedência e os tempos de ocupação nas diferentes máquinas/recursos. Até ao momento, o sistema é totalmente dependente das soluções iniciais dos agentes, tornando-se "cego" e incapaz de evoluir os planos de escalonamento. Com o Mecanismo de Negociação proposto pretende-se dar ao sistema a capacidade de optimizar o plano resultante através de negociação, permitindo-lhe evoluir e produzir melhores planos de escalonamento. O Mecanismo de Negociação proposto é descrito na Tabela I.

TABELA I
ALGORITMO DO MECANISMO DE NEGOCIAÇÃO

Passo 1	Receber o escalonamento/solução de cada Agente de Recursos
	Verifica e ajusta cada operação para cumprir as suas precedências ajustar os tempos de início das soluções
Passo 2	Actualiza os dados de cada Agente Recurso para os tempos de cada operação deste recurso
	Envia dados para os agentes a testar e dá ordem para o início da comunicação entre os agentes fazendo com que o agente com maior tempo de inactividade total inicie o processo
Passo 3	Continua a negociar na próxima operação em atraso no Agente Recurso
Passo 4	Actualiza dados para cada Agente Recurso com os valores da melhor solução encontrada e dá a ordem para terminar a negociação

No sistema *CompeteSched* a negociação pretende eliminar os tempos mortos entre as operações em cada máquina, o que irá melhorar a taxa de utilização de cada máquina, reduzindo os atrasos em geral. A inactividade (tempos mortos) no Agentes Recurso gerada pelas suas operações de precedência, ou seja, uma operação tem de esperar até ao final do processamento da sua anterior no grafo de precedência, a fim do seu processamento poder ser iniciado. O negociador precisa antecipar a execução da precedente ou preencher o tempo de paragem com uma outra operação que pode ser realizada no recurso. Através da troca de operações, a negociação será responsável por verificar os ganhos da antecipação da anterior ou para preencher o tempo paragem com outra operação, escolhendo o cenário mais rentável.

Utilizando uma arquitectura baseada no modelo FIPA [11], um agente pode ser *Initiator* ou *Participant* durante a execução. Um agente *Initiator* é responsável por iniciar o processo de negociação, pedindo a outros agentes alterações nos seus planos para melhorar o seu, verificando também no seu plano melhorias de solução. Um agente *Participant* vende a sua alteração, em troca de algo que lhe será útil. Os agentes podem ter qualquer um dos papéis durante a execução, podem mesmo ter os dois.

A especificação de uma arquitectura em que todos os agentes agem sem restrições, tendo a liberdade de negociar sem ordem tornar-se-ia impossível visto que as alterações nos planos de escalonamento não são independentes, ou seja, a alteração num plano de um agente poderá ou não influenciar um ou mais planos de outros agentes. A criação de restrições no tipo de comunicação entre agentes tornou-se indispensável. Em primeiro lugar a comunicação deve ser do tipo passa-a-palavra, podendo haver apenas um agente *Initiator* em cada instante do decorrer da negociação. Em segundo, um agente só pode solicitar alterações de um nível no grafo de precedências da operação a melhorar.

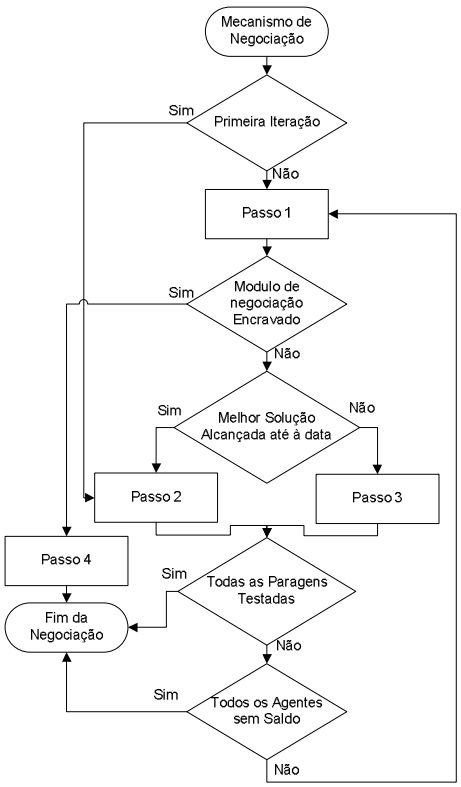

Figura 1. Fluxograma do Mecanismo de Negociação

Para realizar a negociação cada agente tem um saldo correspondente ao somatório das paragens entre operações no seu plano. O saldo pode ser utilizado como moeda de troca para comprar alterações nos planos de outros agentes. Como remuneração pelas alterações ao seu plano o agente recebe um valor correspondente ao tempo que a operação alterada se deslocou, sendo este o saldo ganho pelo agente. Um agente que faça uma alteração no seu próprio plano paga a alteração a si próprio.

Cada agente em negociação tem como objectivo conseguir o plano mais contínuo possível com o maior saldo ganho. O

primeiro agente *Initiator* é o agente com o maior somatório de paragens entre operações. Após a escolha do primeiro agente *Initiator* é iniciada a negociação com algoritmo descrito na Figura 2.

B. Exemplo

Para este exemplo utilizamos o plano de escalonamento resultante do mecanismo de reparação apresentado na Tabela II.

TABELA II
Inicio do Mecanismo de Negociação

T	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
M1	T1.1				T2.1		T3.3								
M2	T3.1		S1			T2.2				T1.2					
M3	S2		T3.2		S3					T2.3				T1.3	
Paragens: S1 → 3 ; S2 → 2 ; S3 → 6 ;															
Paragens Testadas:															

	Saldo			
	Paragens	Gasto	Ganho	Total
M1	0	0	0	0
M2	3	0	0	3
M3	8	0	0	8

Esta solução é guardada temporariamente como a melhor encontrada até ao momento. A negociação começa por analisar os tempos de paragem em cada agente. O agente com maior somatório de tempo de inactividade é o primeiro a ter palavra (M3) e testa a operação com maior tempo de paragem (T2.3). O agente M3 pergunta ao agente M2 se existe a possibilidade de mudar a operação precedente da T2.3, o agente M2 responde que a operação T2.2 não pode ser mudada, porque começa no instante em que a sua precedente termina o processamento.

O agente M3 verifica se pode alterar a operação imediatamente à direita (seguinte) de T2.3. Também não pode ser alterado devido à mesma razão de T2.2. O agente contínua e testa a operação T3.2. O agente M3 pergunta ao agente M2 para alterar a operação precedente de T3.2.O agente M2 responde que não pode fazer nenhuma alteração devido ao processamento de T3.1 iniciar no instante 0. O agente M3 verifica a possibilidade de T3.2 trocar com T2.3, mas não pode ser realizado porque T2.3 começa o processamento imediatamente após o término de T2.2. O agente M3 passa a palavra ao agente com o tempo de paragem seguinte, neste caso o M2.

TABELA III
Mecanismo de negociação iteração 1

T	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
M1	T2.1		T1.1				T3.3								
M2	T3.1		T2.2				T1.2								
M3	S1		T3.2		S2		T2.3				T1.3				
Paragens: S1 → 2 ; S2 → 3 ;															
Paragens Testadas: S2 ; S1 ;															

	Saldo			
	Paragens	Gasto	Ganho	Total
M1	0	0	3	3
M2	0	3	0	0
M3	8	0	0	8

O agente M2 verifica a operação T2.2, perguntando ao agente com a operação precedente para alterar o seu escalonamento (T2.1). M1 responde que essa troca é possível, mas tem um custo de 3 créditos. O agente verifica a possibilidade de troca de T2.2 com T1.2 e verifica que é possível com um custo de 4. Como o agente (M2) não tem fundos suficientes, selecciona a primeira opção, o intercâmbio da operação anterior.

A solução final foi melhorada. É reiniciado o processo de negociação e a nova solução é armazenada.

A Negociação inicia no agente com o maior valor do somatório de tempos de paragens (M3), que testa o maior tempo morto (S2). Pergunta ao agente se pode alterar a operação anterior, T2.3, não é possível devido a iniciar o processamento no final do T2.1. a troca de T2.3 com T1.3 não é possível pelo mesmo motivo. O agente (M3) testa S1. A troca da precedência de T3.2 em M2 máquina não é possível, desde o tempo inicial é 0. A troca de T3.2 com T2.3 não é possível. A negociação termina e o plano (Tabela III) é enviado para o módulo que cria a solução final.

Como conclusão para este exemplo, a negociação melhorou a solução final em 3 unidades de tempo, diminuindo o número e a soma dos tempos de paragem nas máquinas (11 para 5).

TABELA IV
Parametrizações para ACO e PSO

Instancia do Problema		FT06	FT10(PSO), La01, La02, La03, La04, La05, La16 (PSO), La17 (PSO), ABZ5, ABZ6	FT20(PSO), ABZ7, ABZ8, ABZ9, La6, La11, La21(PSO), La29(PSO), La40(PSO)	FT10(ACO), FT20(ACO)	La16(ACO), La17(ACO), La21(ACO), La29(ACO), La40(ACO)	La31, La35, SWV4,SWV5, SWV6, SWV9, YN1, YN2	SWV11, SWV15
ACO	SC	50	100	250	1	1	1	1
	ER	80,00%	80,00%	80,00%	80,00%	80,00%	80,00%	80,00%
	NA	15	25	50	10	15	50	50
	A	1	1	1	1	1	1	1
	B	1	1	1	1	1	1	1
PSO	SC	500	1000	2000	--	--	1	1
	NP	15	25	35	--	--	75	100
	MiV	-4	0,4	-4	--	--	-4	-4
	MaV	4	4	4	--	--	4	4
	MiI	40,00%	40,00%	40,00%	--	--	40,00%	40,00%
	MaI	95,00%	95,00%	95,00%	--	--	95,00%	95,00%
	C1	2	2	2	--	--	2	2
	C2	2	2	2	--	--	2	2
	LL	0	0	0	--	--	0	0
	UL	4	4	4	--	--	4	4

Legenda: SC – Critério de Paragem (*Stopping Criteria*); ER – Taxa de Evaporação (*Evaporation Rate*); NA – Numero de Formigas (*Number of Ants*); A - Alpha; B - Beta; NP – Numero de Particulas (*Number of Particles*); MiV – Velocidade Minima (*Minimum Velocity*):MaV – Velocidade Maxima (*Maximum Velocity*);MiI – Inercia Minima (*Minimum Inertia*);MaI – Inercia Maxima (*Maximum Inertia*);LL – Limite Inferior (*Lower Limit*); UL – Limite Superior (*Upper limit*);

VI. ESTUDO COMPUTACIONAL

Para o estudo computacional, validaram-se problemas de diferentes autores e diferentes dimensões. Após o desenvolvimento do Módulo de Negociação, foram escolhidas 30 instâncias de problemas académicos da OR Library[1]. Para cada instância foram retirados resultados com e sem a incorporação do Mecanismo de Negociação, para ser possível comparar e validar as suas vantagens. As instâncias de problemas académicos analisados são apresentadas na Tabela IV.

Para este estudo computacional utilizamos os seguintes problemas: Fisher e Thompson – FT06, FT10 e FT20 [25], de Lawrence – La1, La2, La03, La04, La05, La06, La11, La16, La17, La2, La29, La31, La35 e La40 [26], de Adams, Balas e Zawack – ABZ5, ABZ6, ABZ7, ABZ8 e ABZ9 [27], de Storer, Wu e Vaccari – SWV4, SWV4, SWV6, SWV9, SWV11 e SWV15 [28] e de Yamada e Nakano YN1 e YN2 [29]. A função objectivo a optimizar é a minimização do *makespan* (C_{max}). A heurística inicial é Earliest Due Date (EDD).

Para a realização dos testes recorreu-se a um PC com as seguintes características: processador Intel® Core™ 2 Quad Q6600 @ 2.40 Ghz; Memoria RAM 4 Gb; disco rígido de 250 Gb 7200 rpm; Sistema operativo Windows 7 64-bit.

Cada instância foi executada 5 vezes e foi armazenada a melhor (*best*), pior (*worst*) e media (*average*) das soluções para a minimização do *makespan* (*Cmax*), maximização da taxa de ocupação das máquinas (U), e a minimização do tempo computacional gasto (T). A comparação entre o sistema com e sem Módulo de Negociação pode ser visualizada na Figura 2, Figura 3 e Figura 4.

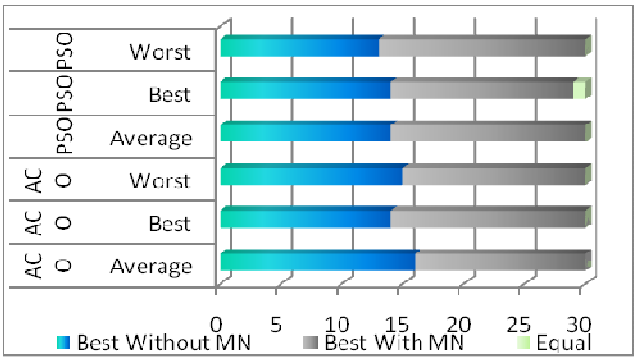

Figura 2. Resultados de minimização do *Makespan* (Cmax)

Da análise dos resultados obtidos para a minimização de *makespan* (Figura 2) é possível concluir que os valores foram melhorados em mais de 50% das instâncias, utilizando ambos os Meta-heurísticas (ACO e PSO), onde foram obtidas duas soluções óptimas.

[1] http://people.brunel.ac.uk/~mastjjb/jeb/info.html

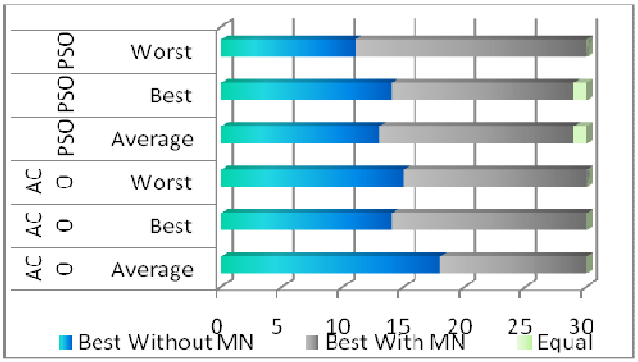

Figura 3. Resultados taxa de ocupação das máquinas (U)

Dos resultados obtidos para a maximização da taxa de ocupação das máquinas (Figura 3) é possível concluir que a incorporação do Mecanismo de Negociação apresenta uma melhoria no desempenho do sistema, com ambas as Meta-heurísticas em estudo (excepto nos resultados médios da ACO). Torna-se muito importante para o sistema, já que um dos objectivos é criar um plano de escalonamento para cada máquina o mais contínuo possível, a fim de evitar tempos de paragem e atrasos na produção.

Da análise dos resultados obtidos para os tempos computacionais (Figura 4) verificamos que estes registaram valores piores, com o Mecanismo de Negociação activo, especialmente quando utilizando o PSO. Com o ACO, obtivemos melhores resultados para o melhor e o pior resultado obtido, os resultados médios obtivemos melhores soluções sem Mecanismo de Negociação. Isto ocorre devido ao aumento do número de leituras, alterações e verificações que a aplicação faz com as soluções do problema.

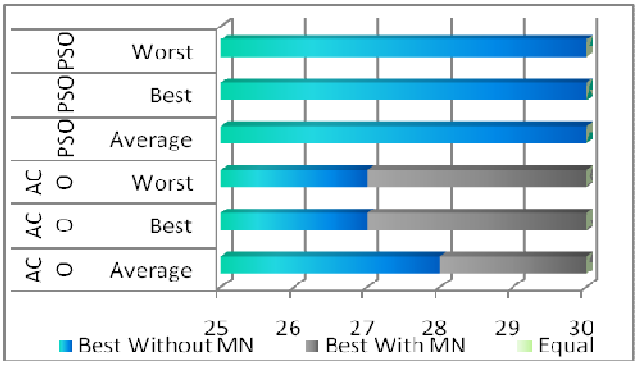

Figura 4. Resultados para Tempos Computacionais (T)

Em geral, o Mecanismo de Negociação melhora a qualidade dos planos de escalonamento, com desvantagens nos tempos computacionais. A introdução do mecanismo de negociação implica na maioria dos casos uma do desempenho do sistema ao nível da eficácia, permitindo melhorar a qualidade das soluções, quando analisada a minimização do *Cmax* e a maximização da tacha de utilização.

VII. CONCLUSÕES E TRABALHOS FUTUROS

Neste artigo foi proposto um Protocolo de Coordenação para uma Arquitectura Multi-Agente de Mercado híbrida baseada num Mecanismo de Negociação entre máquinas com recurso a SI.

Considerando o tipo de problemas que o sistema *CompeteSched* se propõe resolver, o Mecanismo de Negociação proposto permite uma comunicação coordenada entre os agentes, sem comunicações falsas e falhas na implementação das soluções propostas. Os testes computacionais mostram uma melhoria na minimização da *Cmax* e na maximização das taxas de ocupação das máquinas em cerca de metade das instâncias analisadas. Considerando que o Mecanismo de Negociação não degrada a solução, as melhorias registadas permitem afirmar que a utilização deste se torna vantajosa em termos de eficácia apesar da degradação dos tempos computacionais.

Realça-se a necessidade de esforço adicional a ser realizado no sentido da validação do sistema de proposto *CompeteSched* e respectivo Mecanismo de Negociação em ambientes dinâmicos sujeitos a perturbações.

AGRADECIMENTOS

Os autores gostariam de agradecer à FCT, FEDER, POCTI, POSI, POCI, POSC, e COMPETE pelo suporte nos projectos de I&D e ao GECAD.

REFERENCIAS

[1] L. G. Telser, *A Theory of Effective Cooperation e Competition*, Cambridge:Cambridge University Press, 1987.

[2] M. Luck, P. McBurney, O. Shehory, S. Willmoth, *Agent Technology: Computing as Interaction*, A Roadmap for Agent-Based Computing, AgentLink III, 2005.

[3] M. Dorigo, *Swarm Intelligence*, Springer New York, 2007.

[4] J. Kephart e D. Chess, *The Vision of Autonomic Computing*, Computer, 36, 41-50, January 2003.

[5] J.S. Rosenschein e G. Zlotkin, *Rules of Encounter: Designing Conventions for Automated Negotiation Among Computers*, MIT Press, Cambridge, Massachusetts, 1994.

[6] R.G. Smith, *The Contract Net Protocol: High level Communication e Control in a Distributed Problemsolver*, in proceedings of the first International Conference on Distributed Computing Systems, IEEE, New York, pp. 185-192, 1979.

[7] W. Shen, Y. Li, H. Ghenniwa,C. Wang, *Adaptive negotiation for agent-based grid computing*,Proceedings of AAMAS 2002 Workshop on Agentcities: Challenges in Open Agent Environments, Bologna, Italy,pp. 32-36, 2002.

[8] T. Sandholm, *eMediator: A next generation electronic commerce server*, Computational Intelligence, Special issue on Agent Technology for Electronic Commerce, 18(4):656.676, 2002

[9] N.R. Jennings, P. Faratin, A.R. Lomuscio, C. Sieera, e M. Wooldridge, *Automated Negotiation: Prospects, Methods e Challenges*, Int. Journal of Group Decision e Negotiation (GDN2000), Vol.10, No.2, pp.199-215, 2000.

[10] N. R. Jennings, *An agent-based approach for building complex software systems*, Communications of the ACM, 44(4):35.41, 2001.

[11] Z. Alibhai, *What is Contract Net Interaction Protocol?*, IRMS Laboratory, 2003.

[12] I. Zattar, J. Ferreirab, J. Rodrigues e C. Sousa, *A multi-agent system for the integration of process planning e scheduling using operation-based time-extended negotiation protocols*, International Journal of Computer Integrated Manufacturing, 2010.

[13] J. M. Usher, *Negotiation-based routing in job shops via collaborate agents*, Journal of Intelligent Manufacturing, 2003.

[14] A. Singh, D. Juneja e A. K. Sharma, *Introducing Trust Establishment Protocol In Contract Net Protocol*, International Conference on Advances in Computer Engineering, 2010.

[15] H. S. Kim e J. H. Cho, *Supply chain formation using agent negotiation*, Decision Support Systems, 2010.

[16] T. Sandholm, *An Implementation of the Contract Net Protocol Based on Marginal Cost Calculations*, Eleventh National Conference on Artificial Intelligence, pp. 256 – 262, 1993.

[17] E. Verharen, *A Language/Action Perspective on Cooperative Information Agents*, Ph.D. Thesis, Tilburg University, 1997.

[18] M. Sachs, A. Dan e T. Nguyen, *Executable Trading-Partner Agreements in Electronic Commerce*, IBM Research Report, 2000.

[19] C. Xueguang e S. Haigang, *Further Extensions of FIPA Contract Net Protocol: Threshold plus DoA*, ACM Symposium on Applied Computing, 2004.

[20] A. Viana, *Agentes Inteligentes e Sistemas Multi-agente FIPA*, Lisboa: Instituto Superior Tecnico, 2003.

[21] M. Dorigo, Optimization, *Learning e Natural Algorithms*, PhDThesis, Politecnico di Milano, Italy, (in Italian), 1992.

[22] M. Dorigo, M. Birattari, e T. Stützle, *Ant Colony Optimization - Artificial Ants as a Computational Intelligence Technique*, IEEE Computational Intelligence Magazine, 2006.

[23] J. Kennedy e R.C. Eberhart, Particle Swarm Optimization, in Proceedings of the IEEE International Conference Neural Networks, pp 1942-1948, 1995.

[24] A. Madureira, J. Santos, N. Gomes, e C. Ramos, *Proposal of a cooperation mechanism for team-work based multi-agent system in dynamic scheduling through meta-heuristics*, in ISAM07, 2007, pp. 233–238.

[25] H. Fisher e G.L. Thompson, *Probabilistic learning combinations of local job-shop scheduling rules*, J.F. Muth, G.L. Thompson (eds.), Industrial Scheduling, Prentice Hall, Englewood Cliffs, New Jersey, 225-251, 1963.

[26] S. Lawrence, *Resource constrained project scheduling: an experimental investigation of heuristic scheduling techniques (Supplement)*, Graduate School of Industrial Administration, Carnegie-Mellon University, Pittsburgh, Pennsylvania, 1984.

[27] J. Adams, E. Balas, e D. Zawack, *The shifting bottleneck procedure for job shop scheduling*, Management Science 34, 391-401, 1988.

[28] R.H. Storer, S.D. Wu, R. Vaccari,*New search spaces for sequencing instances with application to job shop Management Science 38*, 1495-1509, 1992.

[29] T. Yamada, R. Nakano, *A genetic algorithm applicable to large-scale job-shop instances*, R. Manner, B. Manderick (eds.), Parallel instance solving from nature 2, North-Holland, Amsterdam, 281-290, 1992.