Auto-parametrização de Meta-heurísticas para Escalonamento Dinâmico

Ivo Pereira
Departamento de Engenharia Informática
Instituto Superior de Engenharia
Porto, Portugal
iasp@isep.ipp.pt

Ana Madureira
Departamento de Engenharia Informática
Instituto Superior de Engenharia
Porto, Portugal
amd@isep.ipp.pt

***Resumo*— Este artigo aborda o problema da parametrização de Técnicas de Optimização Inspiradas na Biologia (BIT - *Biological Inspired Optimization Techniques*), também conhecidas como Meta-heurísticas, considerando a importância que estas técnicas têm na resolução de situações de mundo real, sujeitas a perturbações externas. É proposto um módulo de aprendizagem com o objectivo de permitir que um Sistema Multi-Agente (SMA) para Escalonamento seleccione automaticamente uma Meta-heurística e escolha a parametrização a usar no processo de optimização. Para o módulo de aprendizagem foi usado o Raciocínio baseado em Casos (RBC), permitindo ao sistema aprender a partir da experiência acumulada na resolução de problemas similares. Através da análise dos resultados obtidos é possível concluir acerca das vantagens da sua utilização.**

Palavras-chave- Auto-parametrização, Raciocínio baseado em Casos, Meta-heurísticas, Sistema Multi-Agente, Escalonamento

I. INTRODUÇÃO

O problema do Escalonamento é caracterizado por uma grande quantidade de incerteza associada ao dinamismo do sistema, revelando-se um aspecto importante de automação em sistemas de produção. Existe uma grande necessidade de desenvolver abordagens de escalonamento que possam ser aplicadas a vários problemas de escalonamento com impacto significativo no desempenho das organizações empresariais. Emerge assim um desafio para a construção de sistemas de suporte ao escalonamento para ambientes de produção onde a adaptação dinâmica e a optimização se tornam cada vez mais importantes. Neste cenário, a adaptação dinâmica surge como a capacidade de um agente monitorizar o seu estado e desempenho, e proactivamente se auto-parameterizar para responder aos estímulos no ambiente [13].

A auto-regulação de sistemas é um processo complexo que depende de vários factores que podem mudar durante o tempo de vida operacional do sistema. O processo para a coordenação da auto-regulação deve também ser adaptável a perturbações que possam ocorrer durante o tempo de execução.

Existe um interesse crescente na investigação de técnicas para a automação do desenvolvimento de técnicas BIT. Estes esforços tentam substituir ou reduzir o papel do perito humano no processo de desenvolvimento e especificação de algoritmos capazes de resolver, eficiente e eficazmente, problemas de optimização. Várias contribuições têm sido propostas, as quais incorporam técnicas de Aprendizagem Automática (AA), para a parametrização dos parâmetros que controlam o comportamento ou os componentes do algoritmo que constituem aplicações híbridas de BIT. Algumas destas contribuições podem ser encontradas em [4], [6], e [8].

A parametrização de BIT pode ser considerada a dois níveis: *offline* (quando os parâmetros são definidos antes do começo do processo de optimização, podendo ser considerados como configuração da técnica BIT) e *online* (quando os parâmetros podem ser alterados em tempo de execução).

As BIT têm-se mostrado eficazes na resolução de problemas de optimização complexos nos domínios da indústria, economia, e científicos [18]. Alguns exemplos importantes são os Algoritmos Genéticos, o *Simulated Annealing*, a Pesquisa Tabu, a *Scatter Search*, os Algoritmos Meméticos, a Colónia de Formigas e o *Particle Swarm Optimization*. Quando se considera e se percebe as soluções baseadas na natureza, é possível usar o conhecimento adquirido na resolução de problemas complexos em domínios diferentes.

Uma vez que o processo de parametrização de BIT se revela uma tarefa difícil, que requer conhecimento de perito acerca do domínio de aplicação e quais as técnicas e parâmetros a usar, é importante dar aos sistemas a capacidade de efectuar essa parametrização automaticamente. Para permitir esta auto-parametrização, é possível usar conhecimento acerca da experiência passada, com o RBC a revelar-se uma abordagem promissora. Através do uso de RBC, os sistemas podem recordar casos anteriormente resolvidos e automaticamente decidir qual a BIT e respectivos parâmetros devem usar para a resolução do novo problema.

As restantes secções estão organizadas da seguinte forma: a Secção II faz uma descrição do RBC e apresenta alguns exemplos de aplicação à resolução de problemas de Escalonamento em sistemas de fabrico. A Secção III apresenta um SMA para a resolução de problemas de Escalonamento com a incorporação de um módulo de aprendizagem baseado em RBC. Na secção IV é apresentado o estudo computacional realizado, com a comparação dos resultados obtidos de várias instâncias de problemas de Escalonamento académicos. Finalmente, a Secção V sumariza algumas conclusões.

II. RBC APLICADO AO ESCALONAMENTO

O Raciocínio baseado em Casos é uma metodologia de AA com o objectivo de resolver novos problemas através da

utilização da informação acerca da resolução de problemas anteriores similares [9]. Burke et al. [4] referiram que o RBC é uma abordagem apropriada para sistemas de escalonamento com conhecimento de perito e enfatizam um potencial de investigação em escalonamento dinâmico.

No RBC, os casos anteriormente resolvidos e as suas soluções são memorizados como **casos**, guardados na **base de casos** (repositório), para serem reutilizados futuramente [3]. Em vez de definir um conjunto de regras ou linhas gerais, um sistema de RBC resolve um novo problema através da reutilização de casos similares previamente resolvidos [15].

O RBC consiste num ciclo (Figura 1), normalmente descrito como os '4 Rs' [1][3]:

1. **Recuperação (*Retrieve*)** do(s) caso(s) mais similar(es);
2. **Reutilização (*Reuse*)** da informação e conhecimento recuperado;
3. **Revis**ão (***Revise*)** da solução proposta;
4. **Retenção (*Retain*)** da solução revista para ser útil na resolução de um problema futuro.

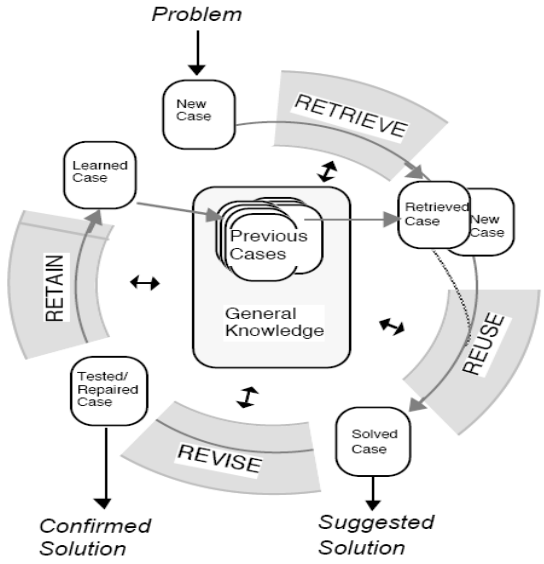

Figura 1. Ciclo RBC (extraído de [1])

Os sistemas RBC aplicados ao Escalonamento podem ser divididos em três categorias [15]: reutilização de operadores, reutilização de algoritmos, e reutilização de soluções.

Os sistemas RBC de escalonamento pertencentes à primeira categoria reutilizam os operadores para a resolução do novo problema [4]. Um caso descreve um contexto no qual um problema de escalonamento útil é usado para adaptar/reparar o plano de escalonamento para melhorar a sua qualidade, em termos de satisfação de restrições [3]. Burke et al. [4] propuseram uma hiper-heurística RBC para a resolução de problemas de calendarização. Beddoe et al. [3] desenvolveram um sistema de resolução de problemas de escalonamento de enfermeiros.

A reutilização de algoritmos assume que é provável que uma abordagem eficaz para a resolução de um problema específico também o será na resolução de um problema similar. Nestes sistemas, um caso consiste na representação do problema e num algoritmo eficiente conhecido para a sua resolução. Schmidt [17] propôs uma estrutura RBC para seleccionar o método mais apropriado para a resolução de problemas de escalonamento em máquinas de produção. Schirmer [16] implementou um sistema RBC para seleccionar algoritmos de escalonamento na resolução de problemas de escalonamento de projectos, e mostrou experiencialmente que alguns algoritmos funcionam melhor do que outros, em algumas instâncias de problemas.

Na reutilização de soluções são usadas todas ou parte das soluções para a construção da solução do novo problema. Um caso contém a descrição do problema e a sua solução, ou parte da solução. Este método foi usado para a resolução de problemas de escalonamento da produção [11] e calendarização de cursos [4]. Foi também usado para a construção das soluções iniciais de BIT, tais como o *Simulated Annealing* [5] e os Algoritmos Genéticos [14].

III. Sistema Multi-Agente para Escalonamento baseado em RBC

Nesta secção é descrito o SMA baseado em RBC para a resolução de problemas de escalonamento dinâmico. Este sistema contém um módulo de aprendizagem para a auto-parametrização de Meta-heurísticas com o objectivo de melhorar o desempenho do sistema.

A. Definição do Problema

O problema focado neste artigo tem algumas extensões e diferenças importantes relativamente ao problema clássico de Escalonamento Job-Shop (JSSP) e é designado de *Extended Job-Shop Scheduling Problem* (EJSSP) [12]. Uma tarefa é definida como uma ordem de produção para um item final, que pode ser simples ou complexo [12]. Os principais elementos do problema EJSSP podem ser modelados como: um conjunto de tarefas multi-operação $J_1,\ldots,J_n$ são escalonadas num conjunto de máquinas $M_1,\ldots,M_n$. d_j representa a data de conclusão da tarefa J_j. t_j é o tempo de processamento inicial da tarefa J_j. r_j representa a data de lançamento da tarefa J_j. Existem também operações na mesma tarefa, em diferentes partes e componentes, processadas simultaneamente em máquinas diferentes, seguidas por operações de componentes (tarefas multi-nível).

Além disso, o EJSSP deve cumprir as seguintes restrições:

a) A existência de datas de lançamento r_j e datas de conclusão d_j diferentes;
b) A possibilidade da definição de prioridades das tarefas, reflectindo a importância de satisfazer as suas datas de conclusão, sendo similares aos pesos atribuídos às tarefas na teoria do escalonamento;
c) As novas tarefas podem chegar em intervalos imprevisíveis. As tarefas podem ser canceladas e podem ocorrer alterações nos atributos das tarefas:

tempos de processamento, datas de lançamento, datas de conclusão e prioridades;

d) Existência de restrições de precedência entre operações de tarefas diferentes;
e) Cada operação O_{ijkl} deve ser processada numa máquina do conjunto M_i, onde p_{ijkl} representa o tempo de processamento da operação O_{ijkl} na máquina M_i.
f) Uma máquina pode processar mais do que uma operação da mesma tarefa (recirculação).
g) Pode haver a existência de máquinas alternativas, idênticas ou não.

B. Arquitectura do Sistema

A arquitectura do sistema (Figura 2) é baseada em seis tipos de agentes [13]. 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.

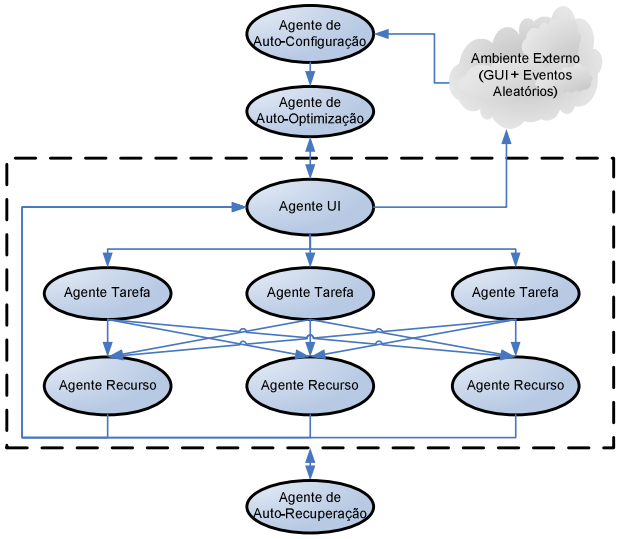

Figura 2. Arquitectura do sistema [13]

Os **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.

C. Módulo de Aprendizagem Proposto

As Meta-heurísticas, tais como Algoritmos Genéticos, *Simulated Annealing*, Pesquisa Tabu, entre outras, são bastante úteis para a obtenção de boas soluções, ou mesmo óptimas, em tempos de computação aceitáveis. Mas, para ser possível obter soluções óptimas ou quase-óptimas, é necessário a correcta afinação dos parâmetros das Meta-heurísticas. Esta revela-se ser uma tarefa árdua, pois é necessário algum conhecimento de perito acerca da Meta-heurística a usar e do problema a tratar. Por vezes torna-se necessário usar o método de tentativa-erro e as dificuldades aumentam quando existe a possibilidade de usar mais do que uma Meta-heurística, porque requer uma escolha a priori da técnica e só depois a afinação dos parâmetros.

Para resolver este problema, é proposto neste artigo um módulo de aprendizagem, descrito nesta secção, com o objectivo de afinar os vários parâmetros da Meta-heurística considerada. É impossível prever todos os problemas a tratar, por isso o sistema deve ser capaz de aprender acerca da sua experiência durante o tempo de vida. Para efectuar este mecanismo de aprendizagem, é proposto o uso de RBC, que se revelou apropriado, uma vez que considera que problemas similares podem usar soluções similares.

É importante que o sistema seja capaz de decidir qual Meta-heurística e quais os parâmetros respectivos que deve usar, porque nem todas as Meta-heurísticas são adequadas para todos os tipos de problema. A abordagem RBC usada pelo sistema consiste em recuperar os casos mais similares ao problema, independentemente da Meta-heurística a usar. Os casos que contém a Meta-heurística e os respectivos parâmetros a usar são recuperados. O caso recuperado é então reutilizado como solução para o novo caso.

Tal como anteriormente descrito, um sistema RBC é constituído por quatro fases, cada uma delas descrita de seguida, com alguns detalhes acerca da implementação.

Antes da descrição de cada fase, é importante mencionar alguns detalhes acerca da base de casos, constituída por seis tabelas, uma para cada Meta-heurística e uma para guardar os atributos dos casos no RBC. As tabelas das Meta-heurísticas representam a Pesquisa Tabu, Algoritmos Genéticos, *Simulated Annealing*, Optimização por Colónia de Formigas, e *Particle Swarm Optimization*. Cada uma destas tabelas contém os parâmetros dos casos da Meta-heurística respectiva.

1) Fase de Recuperação: O objectivo da fase de Recuperação é pesquisar a base de casos, encontrar os casos

mais similares e recuperá-los para análise, de forma a seleccionar um e reutilizá-lo na próxima fase.

Em primeiro lugar, é efectuada uma pré-selecção de casos com o objectivo de ignorar os casos pouco similares com o novo caso. Após esta pré-selecção, os casos são analisados um por um, para ser possível seleccionar os casos que têm similaridade suficiente com o novo caso, através do cálculo da medida de similaridade.

A medida de similaridade é um valor entre zero (0) e um (1), correspondendo respectivamente a casos não similares e casos iguais. Os atributos a considerar na medida de similaridade são o número de tarefas (*NJobs*), número de máquinas (*NMachines*), tipo de problema (*ProblemType*), operações multi-nível (*MultiLevel*), e o valor de C_{max} óptimo (*CmaxOptimal*). Estes atributos são pesados distintamente tal como é mostrado na equação (1).

$$\begin{aligned}\text{Sim} = 0.5 * \text{Sim}_{\text{Njobs}} + 0.25 * \text{Sim}_{\text{Nmachines}} + 0.15 * \\ \text{Sim}_{\text{ProbType}} + 0.05 * \text{Sim}_{\text{MultiLevel}} + 0.05 * \\ \text{Sim}_{\text{CmaxOpt}}\end{aligned} \quad (1)$$

É dada uma maior importância ao número de tarefas e ao número de máquinas, uma vez que estes são os valores que melhor definem a dimensão do problema, característica essencial para a parametrização de Meta-heurísticas. Os parâmetros para a medida de similaridade foram obtidos através dum método de tentativa-erro.

As similaridades de *NJobs* e *NMachines* são calculadas de forma similar, correspondendo à divisão do menor valor pelo maior valor (2)(3), estando no intervalo [0;1].

$$\text{Sim}_{\text{Njobs}} = \frac{\min(\text{Njobs}_1, \text{Njobs}_2)}{\max(\text{Njobs}_1, \text{Njobs}_2)} \quad (2)$$

$$\text{Sim}_{\text{Nmachines}} = \frac{\min(\text{Nmachines}_1, \text{Nmachines}_2)}{\max(\text{Nmachines}_1, \text{Nmachines}_2)} \quad (3)$$

A similaridade de *ProblemType* e *MultiLevel* pode ser um (1) ou zero (0) se os atributos são os mesmos ou não, respectivamente (4)(5).

$$\text{Sim}_{\text{ProbType}} = \begin{cases} 0, & \text{ProbType}_1 \neq \text{ProbType}_2 \\ 1, & \text{ProbType}_1 = \text{ProbType}_2 \end{cases} \quad (4)$$

$$\text{Sim}_{\text{MultiLevel}} = \begin{cases} 0, & \text{MultiLevel}_1 \neq \text{MultiLevel}_2 \\ 1, & \text{MultiLevel}_1 = \text{MultiLevel}_2 \end{cases} \quad (5)$$

A similaridade de *CmaxOptimal* (6) é calculada similarmente ao número de tarefas/máquinas, se os valores de ambos os casos são positivos. Se algum dos valores for negativo, isso significa que o valor é desconhecido, sendo neste caso a similaridade igual a zero.

No final, é recuperada a lista de casos mais similares.

$$\text{Sim}_{\text{CmaxOpt}} = \begin{cases} \dfrac{\min(\text{CmaxOpt}_1, \text{CmaxOpt}_2)}{\max(\text{CmaxOpt}_1, \text{CmaxOpt}_2)}, \\ \quad \text{CmaxOpt}_1 \geq 0 \text{ if } \text{CmaxOpt}_2 \geq 0 \\ 0, \quad \text{CmaxOpt}_1 < 0 \text{ or } \textit{CmaxOpt}_2 < 0 \end{cases} \quad (6)$$

2) Fase de Reutilização: Nesta fase é seleccionado um caso da lista de casos mais similares resultantes da fase anterior. Este caso sugere a sua solução como uma possível solução para resolver o novo caso. Com isto, a Meta-heurística e respectivos parâmetros são devolvidos para serem usados na resolução do novo caso.

No início é verificado se a lista de casos recuperados está vazia ou não. Se estiver vazia, isso significa que não existem casos com similaridade suficiente com o novo caso, sendo neste caso usados os parâmetros pré-definidos pelo utilizador/perito, na interface gráfica, funcionando como uma parametrização inicial. Estes parâmetros pré-definidos são um ponto de partida para a resolução futura de casos similares ao novo caso.

Se a lista não estiver vazia, significa que os melhores casos foram seleccionados, numa perspectiva de eficácia/eficiência, ou, se não existirem casos suficientemente bons, é seleccionado o caso mais similar. Quando existem casos muito similares entre eles (7), é necessário calcular o rácio entre *CmaxOpt* e *Cmax* (8), de forma a detectar os casos mais eficazes/eficientes. Quando não existem casos com alta similaridade entre eles, é então seleccionado o caso mais similar de entre todos.

$$\frac{\min(\text{Sim}_{\text{Case1}}, \text{Sim}_{\text{Case2}})}{\max(\text{Sim}_{\text{Case1}}, \text{Sim}_{\text{Case2}})} > 0.95 \quad (7)$$

$$\text{RatioCase}_i = \frac{\text{CmaxOpt}_{\text{Casei}}}{\text{Cmax}_{\text{Casei}}} \quad (8)$$

Após a selecção dos melhores casos, considerando os aspectos de eficácia/eficiência, é seleccionado aleatoriamente um caso, garantido assim a selecção de um caso bom, em vez de seleccionar o melhor caso entre todos. Isto revela-se importante para evitar a estagnação do sistema e para evitar a escolha do mesmo caso muitas vezes, porque, se isso acontecer, o sistema não consegue evoluir. Se for seleccionado o melhor caso entre todos, aconteceria que esse caso seria sempre escolhido, a não ser que os novos casos obtivessem melhores resultados, o que não acontece sempre devido à aleatoriedade subjacente às Meta-heurísticas. Se não houver casos com um rácio suficiente de eficácia/eficiência, é seleccionado o caso mais similar entre todos.

3) Fase de Revisão: Nesta fase, a solução sugerida pela fase de Reutilização é adaptada, uma vez que o uso directo das soluções se revelou inapropriado pois conduzia a uma estagnação do sistema, não permitindo a evolução para resultados melhores. Assim, foi implementado um algoritmo que aplica alguma diversidade e perturbação aos parâmetros sugeridos, usando uma abordagem de crédito global para atribuir algum valor desse crédito aos vários parâmetros.

Se o caso reutilizado tiver uma similaridade superior a 95%, então o crédito global toma o valor mínimo de 15 (9). Se não, o crédito global é inversamente proporcional à similaridade do caso reutilizado. Isto significa que quanto menos similar um caso for, maior é a perturbação a ser incluída nos parâmetros da Meta-heurística.

$$\begin{aligned}\text{CreditoMinimo} &= 10 + (1 - 0.95) * 100 \\ &= 15\end{aligned} \quad (9)$$

Após a inicialização do crédito global, este é distribuído aleatoriamente por cada parâmetro da Meta-heurística. Após

esta distribuição, a Meta-heurística revista é devolvida com os parâmetros actualizados. Esta actualização dos parâmetros é feita como ilustrado nas equações (10) e (11), respectivamente para valores inteiros e de vírgula flutuante.

$$Parameter_i = Parameter_i + round\left(\left(Parameter_i * \frac{Credit_{Parameter_i}}{100}\right), 0\right) \quad (10)$$

$$Parameter_i = Parameter_i + round\left(\left(Parameter_i * \frac{Credit_{Parameter_i}}{100}\right), 2\right) \quad (11)$$

Com este procedimento, o sistema é capaz de introduzir alguma perturbação nas soluções sugeridas, e desta forma pode escapar a estagnação e evoluir para melhores resultados.

4) Fase de Retenção: A retenção de casos é feita em dois passos. Primeiro, os valores de C_{max} e de tempo de execução são recolhidos do SMA. Depois disto, é criado um novo registo na tabela RBC com os dados do caso, sendo também criado um novo registo na tabela da Meta-heurística usada para soluccionar o problema. Esta fase conclui o ciclo do RBC. Na próxima execução, o caso resolvido já estará disponível para ser usado na resolução do novo caso.

IV. Estudo Computacional

Nesta secção é apresentado o estudo computacional realizado para avaliar o módulo de aprendizagem proposto. Foram usadas várias instâncias de problemas de escalonamento académicos de Fisher e Thompson [7], Lawrence [10] e de Adams, Balas e Zawack [2], num total de 15 instâncias. O objectivo é minimizar o tempo de conclusão do plano (C_{max}).

O estudo computacional está dividido em duas fases. Primeiro, o módulo de aprendizagem proposto é avaliado para se perceber se é capaz de evoluir durante o seu tempo de vida. Na segunda fase, os resultados obtidos após a incorporação do módulo de aprendizagem são comparados com os resultados obtidos antes da integração. Pretende-se perceber as melhorias resultantes da incorporação do módulo de aprendizagem proposto.

A. Avaliação do Módulo de Aprendizagem Proposto

Para avaliar a evolução do módulo de aprendizagem proposto, foram testadas as 15 instâncias durante 100 execuções. Os resultados foram obtidos após 25, 50, 75, e 100 execuções para avaliar a evolução do MAP. Para ilustrar esta avaliação e para se perceber como funciona são apresentados os resultados para duas instâncias nas figuras 3 e 4.

A Figura 3 apresenta os melhores valores, valores médios e valor óptimo para a instância La01 e a Figura 4 apresenta os mesmos valores para a instância ABZ6. Em ambas as instâncias é possível analisar e verificar a evolução dos melhores resultados e dos resultados médios.

Para a instância La01, após 100 execuções, o valor médio teve uma melhoria de 10 unidades de tempo e o melhor valor foi melhorado em 7 unidades de tempo, quando comparado com os resultados obtidos após 25 execuções.

Analisando a instância ABZ6, após 50 execuções o valor médio foi melhorado em 7 unidades de tempo e o melhor valor teve uma melhoria de 3 unidades de tempo, quando comparado com os resultados obtidos após 25 execuções. Estes valores são mantidos nas 25 execuções seguintes e após as 100 execuções o valor médio foi melhorado em 13 unidades de tempo e o melhor valor melhorado em 8, quando comparado com os resultados obtidos após 25 execuções.

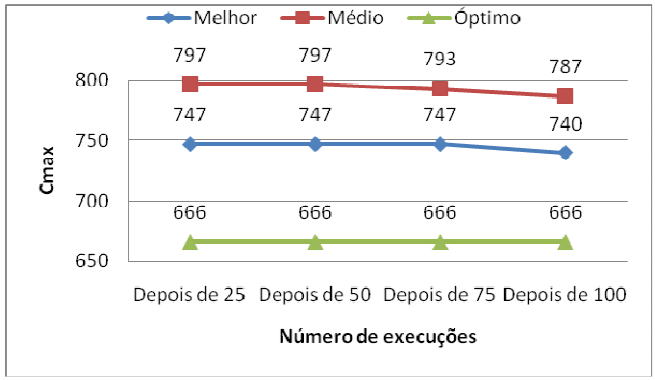

Figura 3. Melhores valores, Médios e Óptimos para a instância La01, depois de 25, 50, 75, e 100 execuções

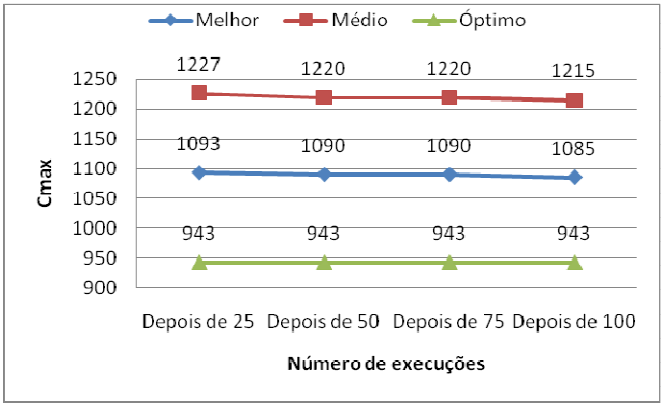

Figura 4. Melhores valores, Médios e Óptimos para a instância ABZ6, depois de 25, 50, 75, e 100 execuções

Com isto, é possível concluir que o módulo de aprendizagem proposto é capaz de evoluir, sendo capaz de melhorar os resultados ao longo do tempo de vida.

B. Comparação de Resultados

Na segunda fase do estudo computacional, os melhores resultados obtidos pelo módulo de aprendizagem proposto são comparados com os melhores resultados previamente obtidos, antes da sua incorporação, das várias Meta-heurísticas em utilização. Estes resultados são apresentados na Tabela I.

As Meta-heurísticas em análise são a Pesquisa Tabu (PT), Algoritmos Genéticos (AG), *Simulated Annealing* (SA), Optimização por Colónia de Formigas (OCF), e *Particle Swarm Optimization* (PSO). Para cada uma destas Meta-heurísticas, os melhores valores obtidos pelo SMA para a resolução de problemas Job-Shop académicos são apresentados na Tabela I.

Com a incorporação do módulo de aprendizagem proposto, o sistema usou apenas uma destas Meta-heurísticas, mas com uma parametrização melhorada, definida de acordo com os

casos anteriormente resolvidos, usando a metodologia RBC (como explicado anteriormente).

Da análise dos resultados obtidos, é possível concluir que os resultados são melhorados em quase todas as instâncias de problema. Para FT10, La02, La03, La09, La15, La34, ABZ6, e ABZ8, os resultados obtidos a partir do módulo de aprendizagem proposto são melhores do que os resultados anteriores. Para as instâncias FT06, La01, La10, e La14 os resultados obtidos são iguais aos resultados previamente obtidos, com a particularidade de terem sido atingidos os valores óptimos nas instâncias La10 e La14. Apenas para as instâncias La24, La30, e ABZ5 os resultados não foram melhorados totalmente, uma vez que a PT com a antiga parametrização obteve melhores resultados.

TABELA I. RESULTADOS DE PT, AG, SA, OCF, PSO E MÓDULO DE APRENDIZAGEM PROPOSTO (MAP)

Instância de problema	**C_{max} Opt.**	**PT**	**AG**	**SA**	**OCF**	**PSO**	**MAP**
FT06 6x6	*55*	**59**	65	60	**59**	**59**	**59**
FT10 10x10	*930*	1319	1399	1301	1376	1306	**1276**
La01 10x5	*666*	**740**	782	763	764	771	**740**
La02 10x5	*655*	839	839	839	855	839	**808**
La03 10x5	*597*	749	804	793	935	847	**747**
La09 15x5	*951*	965	994	973	1004	1024	**960**
La10 15x5	*958*	***958***	978	***958***	***958***	***958***	***958***
La14 20x5	*1292*	***1292***	***1292***	***1292***	***1292***	***1292***	***1292***
La15 20x5	*1207*	1457	1440	1463	1531	1463	**1378**
La24 15x10	*935*	**1242**	1376	1311	1281	1260	1308
La30 20x10	*1355*	**1654**	1781	1704	1779	1729	1680
La34 30x10	*1721*	2205	2229	2175	2241	2141	**2105**
ABZ5 10x10	*1234*	**1411**	1552	1523	1468	1474	1431
ABZ6 10x10	*943*	1093	1173	1093	1203	1164	**1085**
ABZ8 20x15	*645*	936	1045	946	974	928	**925**

Sumarizando, os resultados obtidos foram melhorados em 80% das instâncias analisadas. Isto mostra ser promissor e aponta para um grande interesse em dotar os sistemas com a capacidade de auto-regulação aos eventos externos. Além disso, os sistemas dinâmicos são capazes de reagir e de se adaptarem a novos cenários através da auto-parametrização das BIT, o que pode melhorar significativamente o seu desempenho.

V. CONCLUSÃO

Neste artigo foi proposto um módulo de aprendizagem com o objectivo de seleccionar e parametrizar automaticamente Meta-heurísticas tais como Pesquisa Tabu, Algoritmos Genéticos, *Simulated Annealing*, Optimização por Colónia de Formigas, e *Particle Swarm Optimization*, de modo a resolver problemas de escalonamento. De forma a usar uma abordagem baseada na experiência passada, foi aplicado RBC. Foi descrito o módulo proposto, assim como o SMA onde foi integrado.

Para ser possível avaliar o módulo de aprendizagem, foi realizado um estudo computacional, que permitiu concluir que o módulo proposto é capaz de evoluir e melhorar o desempenho do sistema na resolução de problemas de escalonamento, aprendendo com a experiência e parametrizando automaticamente Meta-heurísticas, para resolver cada diferente caso que surja. O módulo proposto pode ser também aplicado a escalonamento dinâmico, uma vez que um evento dinâmico representa um novo caso. Concluindo, o módulo de aprendizagem proposto revelou-se uma melhoria significativa para os sistemas baseados em BIT uma vez que se tornam capazes de reagir a perturbações externas e de se auto-adaptar a novos cenários, através da auto-parametrização.

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.

REFERÊNCIAS

[1] A. Aamodt and E. Plaza, “Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches”, Artificial Intelligence Communications, 7, 39-52, 1994.

[2] J. Adams, E. Balas, and D. Zawack, “The shifting bottleneck procedure for job shop scheduling”, Management Science 34, 391-401, 1988.

[3] G. Beddoe, S. Petrovic, and J. Li, “A hybrid metaheuristic case-based reasoning system for nurse rostering”, Journal of Scheduling 12, 2009.

[4] E. K. Burke, B. L. MacCarthy, S. Petrovic, and R. Qu, “Knowledge Discovery in a Hyper-Heuristic for Course Timetabling Using Case-Based Reasoning”, PATAT, 2002.

[5] P. Cunningham and B. Smyth, “Case-Based Reasoning in Scheduling: Reusing Solution Components”, The International Journal of Production Research, 35, 2947-2961, 1997.

[6] H.L. Fang, P. Ross, and D. Corne, “A Promising Hybrid GA/Heuristic Approach for Open-Shop Scheduling Problems”, in The 11th European Conference on Artificial Intelligence (ECAI’94), 1994.

[7] H. Fisher and G.L. Thompson, “Probabilistic learning combinations of local job-shop scheduling rules”, Industrial Scheduling, Prentice Hall, Englewood Cliffs, New Jersey, 225-251, 1963.

[8] S. Grolimund and J.G. Ganascia, “Driving Tabu Search with case-based reasoning”, European Journal of Operational Research, 103(2), 1997.

[9] J. Kolodner, “Case-Based Reasoning”, Morgan Kaufmann PubInc, 1993.

[10] S. Lawrence, “Resource constrained project scheduling: an experimental investigation of heuristic scheduling techniques (Supplement)”, Graduate School of Industrial Administration, Pennsylvania, 1984.

[11] B. L. MacCarthy and P. Jou, “Case-based reasoning in scheduling”, in Proceedings of the Symposium on Advanced Manufacturing Processes, Systems and Techniques (AMPST96), 1996, MEP Publications Ltd.

[12] A. Madureira, “Meta-heuristics application to scheduling in dynamic environments of discrete manufacturing”, Ph.D. dissertation, University of Minho, Braga, Portugal, 2003, (in portuguese).

[13] A. Madureira and I. Pereira, "Self-Optimization for Dynamic Scheduling in Manufacturing Systems", Technological Developments in Networking, Education and Automation, pp. 421-426, Springer, 2010.

[14] S. Oman and P. Cunningham, “Using case retrieval to seed genetic algorithms”, International Journal of Computational Intelligence and Applications, 1, 1, 71-82, 2001.

[15] S. Petrovic, Y. Yang, and M. Dror, “Case-based selection of initialisation heuristics for metaheuristic examination timetabling”, Expert Syst. Appl. 33, 3 October 2007, 772-785.

[16] A. Schirmer, “Case-based reasoning and improved adaptive search for project scheduling”, Naval Research Logistics 47, 201-222, 2000.

[17] G. Schmidt, “Case-based reasoning for production scheduling”, International Journal of Production Economics, 56-57, 537-546, 1998.

[18] F. Xhafa and A. Abraham, “Metaheuristics for Scheduling in Industrial and Manufacturing Applications”, Springer, 2008, vol. 128.