X Congresso Brasileiro de Inteligência Computacional (CBIC'2011), 8 a 11 de Novembro de 2011, Fortaleza, Ceará

ALGORITMO HÍBRIDO DE ACS E AG PARA MINIMIZAR O MAKESPAN DO PROBLEMA DE JOB SHOP COM TEMPO DE PROCESSAMENTO INCERTO

Márcia Braga de Carvalho, Akebo Yamakami
DT/FEEC/UNICAMP
{marciab, akebo}@dt.fee.unicamp.br

Tatiane Regina Bonfim
Inovação, Educação e Soluções Tecnológicas
tatiane.bonfim@gmail.com

Resumo – Neste artigo apresentaremos uma hibridização entre os algoritmos de otimização por colônia de formigas e o algoritmo genético (AG) para resolver o problema de escalonamento *job shop* com tempo de processamento incerto e com o objetivo de minimizar o *makespan fuzzy* do problema. Como o problema *job shop* é considerado NP-difícil, sua resolução através de métodos convencionais torná-se inviável. Desta forma neste artigo propomos uma nova abordagem híbrida que trabalha com os algoritmos ant colony system (ACS) e algoritmo genético com busca local conhecido como algoritmo memético (MA) com a finalidade de obter um conjunto de escalonamentos com alto grau de possibilidade de serem ótimos. Implementamos 4 versões híbridas para resolver o problema e testamos estas versões em 10 problemas conhecidos da literatura.

Palavras-chave – Escalonamento job shop, tempo de processamento incerto, sistema de colônia de formigas, algoritmo genético.

Abstract – In this paper we present a hybridization between the algorithms of ant colony optimization and genetic algorithm (GA) to solve the job shop scheduling problem with uncertain processing time and with the objective of minimizing the makespan of the fuzzy problem. As the job shop problem is considered NP-hard, its resolution through conventional methods make it feasible. Thus this paper proposes a new hybrid approach that works with ant colony system algorithms (ACS) and genetic algorithm with local search known as memetic algorithm (MA) in order to obtain a set of schedules with a high possibility of being great . We implemented four hybrid versions to solve the problem and test these versions in ten known problems from literature.

Keywords – Job shop scheduling, fuzzy processing time, ant colony system, genetic algorithm.

1. INTRODUÇÃO

Em problemas reais é comum encontrarmos incertezas nas informações, como tempos de processamento, disponibilidade de equipamentos, tempos de transporte e custos. Desta forma o uso de modelos com incertezas é adequado pois permite aproximar o problema da realidade.

A incerteza pode ser modelada utilizando a lógica *fuzzy* introduzida por Zadeh em 1965 [11]. Esta lógica tem a finalidade de aproximar a precisão matemática clássica da inexatidão do mundo real.

O problema de escalonamento *fuzzy* foi formulado por Dubois et al. em 1995 [4] como um problema de otimização com satisfação de restrições *fuzzy*, onde as restrições associadas ao problema são representadas por um conjunto *fuzzy*. No problema de escalonamento *job shop fuzzy* (JSSPF) os tempos de processamento das operações são imprecisos, desta forma a noção de "escalonamento ótimo"é considerada também imprecisa, já que um escalonamento é avaliado através de um número *fuzzy*. Neste caso, a noção de escalonamento ótimo é avaliado seguindo um critério de ordenação entre os valores dos escalonamentos. Esse critério é importante e pode ser utilizado em situações onde se tem vários escalonamentos e se quer saber qual é o maior ou menor entre eles.

O JSSPF é um problema de alta complexidade computacional, visto que os tempos de processamentos das tarefas são números *fuzzy* e a comparação entre estes é dificíl, sendo às vezes impossível determinar qual o melhor escalonamento. Devido a isso, muitos trabalhos da literatura utilizam índices de *defuzzificação* para assim resolver um problema clássico (*crisp*). Outros utilizam algum método de ordenação (rankeamento) de números *fuzzy* que associa a cada número *fuzzy* um valor *crisp* e com este valor resolvem um problema clássico associado. Ou ainda, utilizam métodos de ordenação lexicográfica ou comparação baseada em α-cortes com o objetivo de otimizar um escalonamento em termos do *makespan fuzzy*.

Dentre os principais trabalhos da literatura, alguns que consideramos importantes são: Lin (2000), Bonfim (2006), González et al. (2007) e Niu et al. (2008). Em Lin [7] as incertezas nos tempos de processamento são modeladas usando tanto números triangulares *fuzzy* como *λ-cortes*. Os autores minimizam o *makespan* resolvendo o problema do *job shop crisp* que resulta da *defuzzificação* dos tempos de processamento. Bonfim [2] resolve o problema utilizando um algoritmo memético com população

estruturada e utiliza o conceito de possibilidade para avaliar qual é o *makespan* de cada indivíduo. Neste trabalho o *makespan* é calculado através de uma janela de tempo onde cada valor é representado por um número *crisp* associado. No trabalho de González et al. [5] os autores propõem um algoritmo memético para resolver o problema de *job shop fuzzy* e introduz um modelo de escalonamento baseado no valor experado do *makespan* ($E[C_{max}(x, \xi, \nu)]$). Este valor associa a cada variável *fuzzy* um valor experado e assim resolve o problema com o objetivo de minimizar o *makespan* experado. Niu et al. [8] propõem um algoritmo de *Particle Swarm* combinado com operadores genéticos denominado GPSO para resolver o problema de *job shop* com tempo de processamento *fuzzy*. Neste trabalho os tempos de processamento das operações são descritos por números triangulares *fuzzy* e o objetivo do problema é encontrar um escalonamento que minimize o *makespan* e sua incerteza. Os autores utilizam um método de classificação de números *fuzzy* para estimar o valor do *makespan fuzzy* e sua incerteza.

Os trabalhos citados acima possuem a desvantagem de lidarem somente com um problema associado, ou seja, em todos os trabalhos é utilizado algum método que associa um número real ao número *fuzzy* para então encontrar o *makespan* do problema. Neste trabalho é proposto uma abordagem que contorna esta dificuldade mantendo as incertezas em todo processo de resolução do problema.

2. DESCRIÇÃO DO PROBLEMA

O problema de escalonamento *job shop fuzzy* consiste de um conjunto de n tarefas que precisam ser processadas em m máquinas. Neste problema cada tarefa consiste de uma sequência de operações que especificam a ordem das máquinas pelas quais as tarefas deverão ser processadas. Cada tarefa é composta por uma lista ordenada de operações contendo a sequência das máquinas que irá processá-la e os correspondentes tempos de processamento. O tempo de processamento aqui é considerado incerto e representado por números *fuzzy*. Cada operação precisa ser processada por uma dada máquina durante um período de tempo, sem interrupção. O objetivo do problema é encontrar uma ordem de escalonar as tarefas nas máquinas de tal forma que minimize o *makespan fuzzy*.

2.1 TEMPO DE PROCESSAMENTO INCERTO

Em aplicações reais frequentemente nos deparamos com situações onde o tempo que uma tarefa leva para ser processada numa determinada máquina não é conhecido. Desta forma é comum utilizar números *fuzzy* para representar um tempo de processamento incerto. Esta incerteza dentro da teoria *fuzzy* pode ser modelada utilizando números triangulares *fuzzy* (*NTF*) da forma $\widetilde{A} = (a_i, a_m, a_s)$, onde a_m é o valor modal e os espalhamentos inferiores e superiores são a_i e a_s respectivamente. A função de pertinência deste número *fuzzy* $\widetilde{A}$ assume a seguinte forma triangular.

$$\mu_{\widetilde{A}}(x) = \begin{cases} 0, & \text{se } x < a_i \\ \frac{x-a_i}{a_m-a_i}, & \text{se } a_i \leqslant x \leqslant a_m \\ \frac{x-a_s}{a_m-a_s}, & \text{se } a_m < x \leqslant a_s \\ 0, & \text{se } a_s < x \end{cases} \quad (1)$$

A Tabela 1 apresenta uma instância gerada para o problema com 3 tarefas e 3 máquinas. Para a geração desses números utilizou-se o benckmark [10], onde o valor modal a_m corresponde ao valor *crisp* deste benckmark e os espalhamentos a esquerda ($a_m - a_i$) e a direita ($a_s - a_m$) foram gerados segundo uma distribuição uniforme no intervalo [0,1].

Tabela 1: Instância do problema *job shop fuzzy* (3×3).

Tarefas	Ordem das operações (máquina, tempo de processamento)		
1	O_{11}(1, [2,40 3 3,23])	O_{12}(2, [2,45 3 3,35])	O_{13}(3, [2,00 3 3,36])
2	O_{21}(1, [1,18 2 2,79])	O_{23}(3, [2,22 3 3,44])	O_{22}(2, [3,37 4 4,31])
3	O_{32}(2, [2,41 3 3,44])	O_{31}(1, [1,63 2 2,31])	O_{33}(3, [0,96 1 1,61])

Na instância exemplificada acima a ordem de processamento das tarefas nas máquinas é pré-estabelecida. De acordo com a Tabela 1, a tarefa 1 precisa ser processada inicialmente na máquina 1 com um tempo *fuzzy* de [2,40 3 3,23], em seguida pela máquina 2 com tempo *fuzzy* de [2,45 3 3,35] e assim por diante. Um escalonamento é factível quando a ordem das operações é preservada e cada máquina processa apenas uma operação de cada vez.

2.2 RELAÇÃO DE ORDEM

Para determinar o *makespan* do problema do *job shop fuzzy* é utilizado o algoritmo proposto por Hernandes [6] com algumas modificações. Este algoritmo tem como finalidade encontrar todos os caminhos não-dominados entre o nó origem e o nó destino do grafo com parâmetros *fuzzy* e utiliza a relação de ordem proposta por Okada e Soper [9]. No algoritmo proposto utiliza-se números triangulares *fuzzy* e é determinado o seguinte critério de dominância parcial *fuzzy* [9].

Definição 2.2.1 *(Dominância Parcial) Sejam* $\widetilde{A} = (a_i, a_m, a_s)$ *e* $\widetilde{B} = (b_i, b_m, b_s)$ *dois números triangulares fuzzy e* $\varepsilon \in [0, 1]$, *então* $\widetilde{A}$ *domina* $\widetilde{B}$ *com grau* ε, *denotado por* $\widetilde{A} \prec_\varepsilon \widetilde{B}$, *se e somente se* $(a_m) \leqslant (b_m)$, $(a_i)_\varepsilon \leqslant (b_i)_\varepsilon$, $(a_s)_\varepsilon \leqslant (b_s)_\varepsilon$ *e* $\widetilde{A} \neq \widetilde{B}$.

2.3 COMPARAÇÃO DE NÚMEROS TRIANGULARES *FUZZY*

A comparação entre os números triangulares *fuzzy* é um recurso utilizado no desenvolvimento do algoritmo proposto neste artigo, pois é necessário, entre vários números *fuzzy* definir qual é o maior ou menor. O método de comparação utilizado [3] consiste em definir um número real que represente o número triangular *fuzzy*. A seguir apresentaremos os 3 critérios utilizados para ordenar os números triangulares *fuzzy*.

$$Cr_1(\tilde{A}) = \frac{a_i + 2a_m + a_s}{4}, \quad Cr_2(\tilde{A}) = a_m, \quad Cr_3(\tilde{A}) = a_s - a_i \tag{2}$$

- Ordene os NTF_s de acordo com o primeiro critério de ordenação Cr_1; se existirem NTF_s com valor idêntico a Cr_1 então ordene os NTF_s de acordo com Cr_2; se existirem NTF_s com valor idêntico a Cr_1 e Cr_2 então ordene usando Cr_3.

Uma observação importante é que este critério de ordenação é utilizado apenas para ordenar a população de acordo com seu valor de *makespan fuzzy*.

3. MODELAGEM POR GRAFO DISJUNTIVO COM PARÂMETROS *FUZZY*

A característica *fuzzy* de um problema pode ser encontrada em diversos níveis: da estrutura do grafo (nós e arestas) aos parâmetros associados ao grafo (custo, tempo, etc). Problemas com estrutura de grafo *crisp* e parâmetros *fuzzy* é um dos mais estudados da literatura. Estes são problemas em que a estrutura do grafo é bem conhecida e rígida e os parâmetros associados são representados por números *fuzzy*.

Um dos problemas de grafos com parâmetros *fuzzy* mais estudados na literatura é o de caminho mínimo. Um outro problema de otimização que pode ser modelado na forma de grafo com parâmetros *fuzzy* bastante complexo e estudado na literatura é o problema de escalonamento *job shop*.

Inicialmente estes dois problemas não apresentam nenhuma relação em comum, já que o objetivo do problema de caminho mínimo é encontrar um menor caminho entre dois nós do grafo, ou seja, minimizar a função objetivo ($min\{f(x)\}$) e o objetivo do problema de *job shop* é minimizar o *makespan* (caminho máximo) do grafo, ou seja, $min(max\{f(x)\})$. Entretando, podemos estabelecer uma relação entre estes dois problemas, o que é bastante interessante pois o problema de caminho máximo é NP-completo e não existe algoritmo exato para resolvê-lo.

No trabalho de Hernandes (2007) [6] é proposto um algoritmo de caminho mínimo baseado no algoritmo clássico de Bellman-Ford-Moore (FBM) que pode ser aplicado em grafos com parâmetros *fuzzy* negativos. Desta forma, foi necessário fazer algumas modificações no algoritmo para calcular o caminho crítico (*makespan*) para o problema do *job shop fuzzy*.

Na modelagem do problema JSSPF através do grafo disjuntivo com parâmetros *fuzzy*, os tempos de processamento das tarefas nas máquinas são representados por números *fuzzy* (Figura 1).

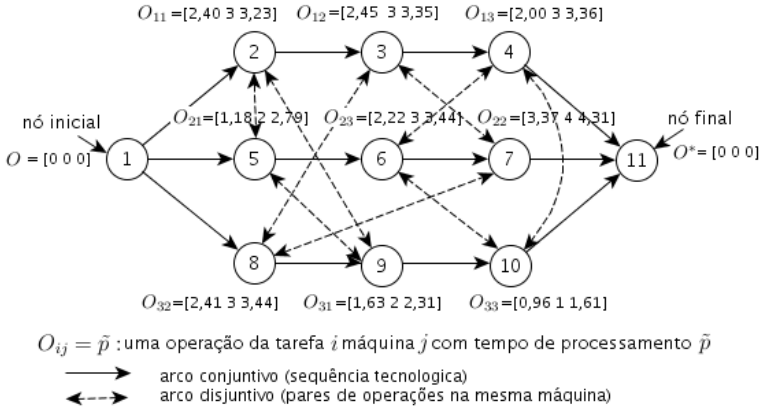

Figura 1: Grafo disjuntivo com parâmetros *fuzzy* do JSSPF ($n = 3$ e $m = 3$).

Inicialmente o grafo disjuntivo *fuzzy* não representa uma solução para o JSSPF. Quando todas as direções dos arcos disjuntivos são definidas, obtém-se um grafo direcionado, se este grafo direcionado for acíclico temos uma solução (escalonamento) factível para o problema.

Para calcular o *makespan* basta calcular o caminho crítico do grafo acíclico *fuzzy*. Este caminho é determinado pela maior distância entre o nó inicial e o nó final do grafo e será calculado pelo algoritmo descrito na Seção 3.1.

3.1 ALGORITMO PARA CALCULAR O *MAKESPAN* DO PROBLEMA *JOB SHOP FUZZY*

Hernandes (2007) propõe um algoritmo para encontrar o caminho mínimo em grafos com parâmetros *fuzzy*. Tal algoritmo é baseado no algoritmo clássico de Bellman-Ford-Moore [1] e utiliza a relação de ordem (descrita na Seção 2.2) para determinar o

conjunto de caminhos não-dominados. Desta forma são necessárias algumas modificações no algoritmo proposto de Hernandes (2007) para calcular o *makespan* do problema *job shop fuzzy*.

Um Pseudocódigo do algoritmo é apresentado no Algoritmo 1. Este é um algoritmo iterativo possuindo como critério de parada o número de iterações ou a não alteração dos custos (tempos de processamento) de todos os caminhos (escalonamentos) encontrados na iteração anterior com relação à iteração atual. Como a relação de Okada e Soper (2000) pode apresentar entre dois nós, mais de um caminho não-dominado, cada caminho recebe um rótulo (etiqueta) para que este seja construído no passo final do algoritmo.

Algoritmo 1: Pseudocódigo do algoritmo calcular o *makespan*.

Informações sobre o algoritmo: V: conjunto dos nós; it: contador de iterações; $(a_s)^i$: limitante superior do tempo de processamento do nó i; $\widetilde{l}_{ji}$: tempo de processamento do arco (j,i); $\widetilde{c}^{it}_{(i,k)}$: tempo de processamento do caminho entre os nós 1 e i, com a etiqueta k, na iteração it; M: um número com valor grande, substitui o ∞ do algoritmo clássico de FBM; Γ_i^{-1}: conjunto dos nós predecessores de i.

Passo 1: Inicialização das variáveis

1. $\widetilde{c}^{0}_{(1,1)} = (0,0,0)$
2. $\widetilde{c}^{0}_{(j,1)} = (M+2,1,1)$, $j = 2,3,\ldots,r$
   talque:
   - r: número de nós; $\quad M = \sum_{i=1}^{na} |(a_s)^i|$; $\quad na$: número de arcos.
3. $it \leftarrow 1$.

Passo 2: Determinação dos caminhos e verificação da dominância

1. $\widetilde{c}^{it}_{(1,1)} = (0,0,0)$
2. $\forall j \in \Gamma_i^{-1}$, $i = 2,3,\ldots,r$, faça:
   - $\widetilde{c}^{it}_{(i,k1)} = \widetilde{c}^{it-1}_{(j,k2)} \oplus \widetilde{l}_{ij}$ (construção dos custos dos caminhos)
3. Verificação da dominância entre as etiquetas do nó i (relação de Okada e Soper):
   - Se $\widetilde{c}^{it}_{(i,m)} \succ \widetilde{c}^{it}_{(i,n)} \Rightarrow$ elimine a ***m-ésima*** etiqueta
   - Se $\widetilde{c}^{it}_{(i,m)} \prec \widetilde{c}^{it}_{(i,n)} \Rightarrow$ elimine a ***n-ésima*** etiqueta

Passo 3: Critério de parada

1. Se ($\widetilde{c}^{it}_{(i,k1)} = \widetilde{c}^{it-1}_{(i,k1)}, \forall i \in V$) ou ($it = r$) faça:
   - $it = r$ e $\widetilde{c}^{it}_{(i,k1)} \neq \widetilde{c}^{it-1}_{(i,k1)} \Rightarrow$ **Passo 5**
2. Senão $it \leftarrow it + 1 \Rightarrow$ volte ao **Passo 2**

Passo 4: Composição dos caminhos

- Encontre todos os caminhos não-dominados entre os nós 1 e i

Passo 5: Saída: Todos os caminhos não-dominados e seus tempos de processamento.

4. ABORDAGEM PROPOSTA

Neste trabalho propomos quatro algoritmos híbridos que trabalham com os algoritmos de sistema de colônia de formigas (ACS) e o algoritmo genético (AG) com busca local para resolver o problema de escalonamento *job shop* com tempo de processamento incerto. As técnicas de buscas locais serão descritas na Seção 4.1. O algoritmo de colônia de formigas é utilizado para criar uma população inicial de bons escalonamentos factíveis e o algoritmo genético com busca local também conhecido como algoritmo memético (MA) é utilizado para evoluir esses escalonamentos.

O algoritmo híbrido é utilizado para encontrar boas soluções para o problema e para avaliar a melhor solução é utilizado o método de comparação descrito na Seção 2.3. Este método têm a finalidade de definir entre vários números *fuzzy* qual é o maior (*makespan*).

4.1 ALGORITMO HÍBRIDO DE SISTEMA DE COLÔNIA DE FORMIGAS E MEMÉTICO PARA O JSSPF

Um conceito importante e crítico na execução de um algoritmo é a escolha da representação de possíveis soluções para o problema. Neste trabalho o problema de *job shop* com parâmentros *fuzzy* (Tabela 1) é representado por um grafo disjuntivo *fuzzy* (Figura 1) e cada solução é representada através de um vetor V de $n.m+2$ colunas, onde n é o número de tarefas e m o número de máquinas. Cada campo do vetor v_k com $1 \leqslant k \leqslant n.m+2$ é preenchido por um número k que indica o número do nó no grafo disjuntivo. Este vetor possui uma sequência que corresponde a ordem que as tarefas são atendidas nas máquinas. Como dito anteriormente na literatura não foi encontrado nenhum trabalho que resolve este problema através desta representação, sendo essa mais uma razão para esta abordagem.

1. Geração da população inicial: O primeiro passo do algoritmo híbrido é a geração da população inicial e esta foi gerada utilizando o algoritmo de colônia de formigas.

2. Cálculo do makespan: Para calcular o *makespan* do JSSPF utilizou-se o algoritmo descrito na Seção 3.1. Este algoritmo encontra todos os caminhos não-dominados entre o nó inicial e o nó final do grafo, sendo necessário a utilização de um método capaz de decidir qual é o maior entre os valores encontrados, pois este representa o *makespan* do JSSPF.

3. Passos seguintes do algoritmo híbrido: Os passos seguintes do algoritmo híbrido onde são executados os operadores genéticos e os métodos de busca local são realizados enquanto o critério de parada não for alcançado. O critério de parada utilizado foi o número de gerações. Os operadores genéticos executados neste algoritmo foram o *crossover* e a mutação. Os operadores genéticos são aplicados no grafo disjuntivo *fuzzy*.

4. Operadores genéticos

O *crossover* é realizado entre dois indivíduos pais da população gerando dois novos indivíduos filhos. Nesta operação genética é sorteado uma tarefa dos indivíduos pais e a ordem de localização que estas tarefas aparecem nos indivíduos filhos é preservada. O restante do cromossomo do filho é preenchido de acordo com a ordem que as outras tarefas aparecem no outro pai.

A mutação é realizada em um indivíduo pai gerando um indivíduo filho. Neste trabalho a mutação utilizada foi a denominada mutação inversiva. Nela escolhe-se aleatoriamente duas sequências de tarefas e faz a troca entre elas. As demais tarefas do cromossomo permanecem na mesma ordem. Os operadores de *crossover* e mutação são aplicados com probabilidades P_c e P_m respectivamente.

5. Métodos de busca local: Na literatura existem diferentes regras de busca local que são adicionadas ao algoritmo genético para melhorar a qualidade das soluções do problema *job shop*. Neste artigo o algoritmo proposto inicialmente cria uma população de soluções factíveis, ordena de acordo com seu valor de *fitness* e então, aplica-se os operadores de *crossover* e mutação para gerar a população da próxima geração.

O objetivo da busca local é melhorar uma solução obtida na população inicial ou pelos operadores de *crossover* e mutação. Por isso a busca local é realizada apenas no melhor indivíduo da população sendo que, este novo indivíduo só é aceito para a próxima geração se melhorar a qualidade da solução corrente, caso contrário ele é descartado. Neste trabalho propomos três técnicas de buscas locais e elas serão descritas a seguir.

A. *Troca de Tarefas Adjacentes no Caminho Crítico (CC)*:

Nesta regra de busca local o caminho crítico de cada solução é calculado. A seguir, dentro deste caminho crítico são identificados os pares de tarefas adjacentes pertencentes a mesma máquina e então é realizada a mudança de direção entre os arcos que ligam estas tarefas adjacentes.

A mudança de direção dos arcos pertencentes ao caminho crítico de cada solução representa a mudança de direção dos arcos disjuntivos (arcos pertencentes à mesma máquina) no gráfico disjuntivo *fuzzy*. É importante lembrar que esta mudança de direção dos arcos pode resultar em um escalonamento infactível. Por esta razão, antes de aceitar a troca dos arcos é necessário verificar a factibilidade do escalonamento. Na Figura 2 ilustramos um exemplo de solução para o problema da Tabela 1.

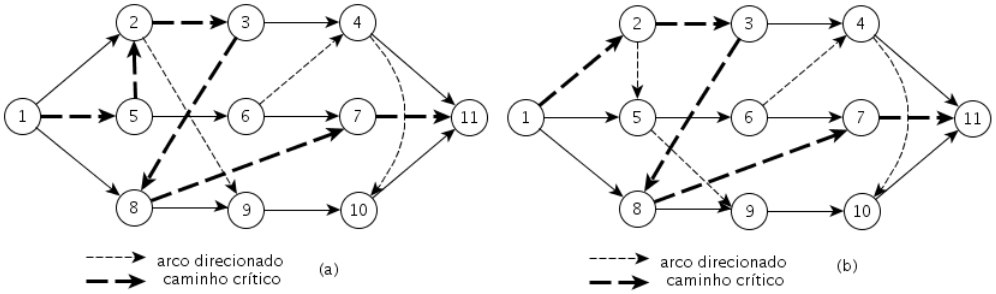

Figura 2: Grafo disjuntivo: (a) antes de aplicar a busca local, (b) depois de aplicar a busca local CC.

Seja V_1 cromossomo que representa uma solução do problema (Figura 2 (a)),

$$V_1 = (1 \rightarrow 5 \rightarrow 2 \rightarrow 3 \rightarrow 8 \rightarrow 6 \rightarrow 4 \rightarrow 9 \rightarrow 7 \rightarrow 10 \rightarrow 11)$$

e seja cc_1 seu caminho critíco representado pelo grafo acíclico (Fig. 2 (a)).

$$cc_1 = (1 \rightarrow 5 \rightarrow 2 \rightarrow 3 \rightarrow 8 \rightarrow 7 \rightarrow 11)$$

O *makespan fuzzy* de V_1 é [11,81 15 17,03] unidades. A busca local CC é realizada nas tarefas adjacentes pertencentes à mesma máquina, no caso os nós 5 e 2. Uma mudança de direção entre os arcos que ligam estes nós (Fig. 2 (b)) é realizada dando origem a uma nova solução V_2.

$$V_2 = (1 \rightarrow 2 \rightarrow 5 \rightarrow 3 \rightarrow 8 \rightarrow 6 \rightarrow 4 \rightarrow 9 \rightarrow 7 \rightarrow 10 \rightarrow 11)$$

Esta nova solução (escalonamento) têm *makespan* de [10,63 13 14,33] unidades e caminho crítico cc_2, representado pelo grafo acíclico (Figura 2 (b)).

$$cc_2 = (1 \rightarrow 2 \rightarrow 3 \rightarrow 8 \rightarrow 7 \rightarrow 11)$$

Como pode ser visto nas Fig. 2 (a) e Fig. 2 (b), a mudança de direção do arco que liga o nó 5 ao nó 2, resultou na inversão da direção entre os arcos que pertencem a m_1. Isto fez com que a solução resultante V_2 obtivesse um valor de *fitness* menor.

B. *Troca de Tarefa na Máquina mais Ociosa (MO)*

Resolver o problema de *job shop* geralmente gera soluções onde o escalonamento tem *gaps* entre as tarefas nas máquinas. Estes *gaps* surgem no escalonamento devido às restrições de precedências impostas às tarefas. Uma grande quantidade de *gaps* numa máquina faz com que ela fique ociosa, o que aumenta o valor de *makespan* do escalonamento.

A busca local MO identifica a máquina ociosa no escalonamento e a tarefa candidata que será remanejada para dentro de um *gap* a fim de obter um melhor escalonamento, reduzindo o valor do *makespan*. Esta regra de troca de tarefas adjacentes na máquina somente será permitida se o escalonamento resultante for factível.

C. *Junção das duas técnicas de buscas locais CC-MO*

A terceira técnica de busca local denominada CC-MO é a junção das duas técnicas de buscas locais descritas acima. Esta técnica aplica primeiramente a busca local de troca de tarefas adjacentes no caminho critíco e posteriormente no cromossomo resultante aplica a busca local de troca de tarefas adjacentes na máquina mais ociosa (com a finalidade de reduzir gaps).

Como saída dos algoritmos é apresentado um conjunto solução com alto grau de otimalidade. Cada um destes indivíduos tem seu valor de *fitness* representado por um número triangular *fuzzy*. Para apresentarmos seu valor *crisp* foi aplicada a *defuzzificação* que é o mapeamento de informações *fuzzy* em valores *crisp*. O método de *defuzzificação* utilizado foi o método da centróide. Um pseudocódigo do algoritmo híbrido aplicado ao problema de escalonamento *job shop* com tempo de processamento incerto é apresentado no Algoritmo 2.

Algoritmo 2: Pseudocódigo do algoritmo híbrido aplicado ao problema de escalonamento *job shop fuzzy*

Entrada: m: número de máquinas, n: número de tarefas, $O_{m\times n}$: matriz com sequência de operações para cada tarefa, $o_{ij} = (m_{ij}, \widetilde{p}_{ij})_{1\leqslant i\leqslant n, 1\leqslant j\leqslant m}$: par ordenado de cada operação composto pela máquina que irá processa-lá e o tempo de processamento da operação, T_{pop}: tamanho da população, N_{it}: número de iterações, N_{form}: número de formigas, ρ: nível de feromônio, β: informação heurística, α: informação de feromônio, q_0: informação heurística, N_{gera}: número de gerações, P_c: probabilidade de *crossover* e P_m: probabilidade de mutação.

- Gera a população inicial através do algoritmo de colônia de formigas;
- Calcular o valor de *makespan* de cada indivíduo;
- Ordena a população utilizando o valor de *makespan* através do método descrito na Seção 2.3;

para $z = 0$ até $z = N_{gera}$, **faça**

- Selecionar 2 indivíduos da população inicial;
- Aplicar *crossover* gerando 2 novos indivíduos para a memória;
- Selecionar 1 indivíduo da população inicial;
- Aplicar mutação gerando 1 novo indivíduo para a memória;
- calcular o valor de *makespan* da memória, através do algoritmo descrito na Seção **??**;
- Aplicar supressão na memória e verificar diversidade;
- Ordena a população utilizando o valor de *makespan* através do método descrito na Seção 2.3
- Selecionar os $x\%$ melhores indivíduos da memória para a próxima geração;
- Atualizar a memória da próxima geração completando com novos indivíduos, se necessário;
- Se MA-ACS(MO), MA-ACS(CC) ou MA-ACS(CC-MO) aplicar busca local no melhor indivíduo da população e atualiza seu valor de *fitness* (Se AG-ACS, este passo não existe).

fim para

Saída: Melhor escalonamento da população e seu valor de *makespan fuzzy*.

5 RESULTADOS NÚMERICOS DAS ABORDAGENS *FUZZY*

Neste capítulo são apresentados e discutidos os resultados obtidos com a aplicação do algoritmo híbrido MA-ACS (CC-MO) proposto neste artigo para resolver o JSSPF. Para comparar a eficiência de MA-ACS (CC-MO) os resultados são comparados com os algoritmos MA-ACS(MO), MA-ACS(CC) e AG-ACS (algoritmo genético com população inicial gerado pelo ACS) implementados. Todos os experimentos foram realizados com os mesmos parâmetros. Na avaliação das abordagens foram utilizadas 10 instâncias com diferentes números de tarefas e máquinas extraídas do OR-Library [10]. As instâncias são caracterizadas pelas seguintes informações: número de tarefas, número de máquinas e uma tabela contendo a sequência de operações de cada tarefa, incluindo o número da máquina que irá processá-la e o tempo de processamento das operações. Como aqui os tempos de processamento das tarefas são considerados parâmetros *fuzzy*, os números *fuzzy* foram gerados seguindo a descrição da Seção 2.1.

Cada experimento foi executado 10 vezes e implementado em Matlab 8 na plataforma Linux, Intel Core 2 Duo 2.0GHz e 4Gb. Os parâmetros adotados na execução dos algoritmos são apresentados a seguir.

- Número de formigas = 15, $\alpha = 0,1$, $\beta = 2$, $\rho = 0,01$, $q_0 = 0,7$; Número de gerações: 500, tamanho da população: 40 indivíduos; Probabilidade de crossover: $0,8$ e Probabilidade de mutação: $0,6$; Supressão: elimina indivíduos de mesmo *makespan*, diversidade: Insere novos indivíduos criados pelo ACS, taxa miníma de diversidade populacional: 50%.

Na Tabela 2 são apresentados todos os resultados encontrados pelos 4 algoritmos, tais como os valores do *makespan fuzzy* encontrados por cada abordagem, a *defuzzificação* pelo método da centróide dos *makespan fuzzy* e os valores dos *makespan crisp* de cada problema extraído do Or-Library. Os tempos de procesassamento que cada algoritmo levou para realizar cada experimento será exibido apenas da versão híbrida mais simples AG-ACS e da mais completa MA-ACS(CC-MO), já que os tempos das outras duas versões estão dentro destes limitantes. Como estamos lidando com um problema *fuzzy* e o critério de parada das abordagens é o número de gerações, o tempo de processamento apresentado corresponde ao tempo total que cada abordagem levou para realizar cada experimento, independentemente de quando a melhor solução foi encontrada.

Tabela 2: Comparação dos resultados para minimizar o *makespan fuzzy*.

Problema e tamanho	Algoritmo	Tempo (seg)	Makespan fuzzy	Deffuzificação	Makespan crisp
ft06 (6x6)	AG-ACS	172	[50,6 55 57,75]	54,4	55
	MA-ACS(MO)		[50,6 55 57,75]	54,4	
	MA-ACS(CC)		[50,6 55 57,75]	54,4	
	MA-ACS(CC-MO)	186	[50,6 55 57,75]	54,4	
la01 (10x5)	AG-ACS	391	[612,7 666 699,3]	659,3	666
	MA-ACS(MO)		[612,72 666 699,3]	659,3	
	MA-ACS(CC)		[612,72 666 699,3]	659,3	
	MA-ACS(CC-MO)	490	[612,7 666 699,3]	659,3	
la08 (15x5)	AG-ACS	887	[793,9 863 906,1]	854,3	863
	MA-ACS(MO)		[793,9 863 906,1]	854,3	
	MA-ACS(CC)		[793,9 863 906,1]	854,3	
	MA-ACS(CC-MO)	910	[793,9 863 906,1]	854,3	
la11 (20x5)	AG-ACS	1588	[1124,2 1222 1283,1]	1209,8	1222
	MA-ACS(MO)		[1124,24 1222 1283,1]	1209,8	
	MA-ACS(CC)		[1124,24 1222 1283,1]	1209,8	
	MA-ACS(CC-MO)	1852	[1124,2 1222 1283,1]	1209,8	
la12 (20x5)	AG-ACS	1507	[955,88 1039 1090,95]	1028,6	1039
	MA-ACS(MO)		[955,88 1039 1090,95]	1028,6	
	MA-ACS(CC)		[955,88 1039 1090,95]	1028,6	
	MA-ACS(CC-MO)	1723	[955,88 1039 1090,95]	1028,6	
la16 (10x10)	AG-ACS	1003	[879,52 956 1003,8]	946,44	945
	MA-ACS(MO)	1130	[879,52 956 1003,8]	946,44	
	MA-ACS(CC)		[882,28 959 1006,95]	949,41	
	MA-ACS(CC-MO)	1152	[870,32 946 993,3]	936,54	
la17 (10x10)	AG-ACS	935	[729,56 793 832,65]	785,07	784
	MA-ACS(MO)		[724,05 787 826,35]	779,13	
	MA-ACS(CC)		[729,56 793 832,65]	785,07	
	MA-ACS(CC-MO)	1011	[722,2 785 824,2]	777,13	
abz6 (10x10)	AG-ACS	918	[894,24 972 1020,6]	962,28	943
	MA-ACS(MO)		[871,24 947 994,35]	937,50	
	MA-ACS(CC)		[871,24 947 994,35]	937,50	
	MA-ACS(CC-MO)	979	[870,32 946 993,3]	936,44	
Orb02 (10x10)	AG-ACS	952	[859,28 934 980,7]	924,66	888
	MA-ACS(MO)		[834,4 907 952,35]	897,93	
	MA-ACS(CC)		[839,04 912 957,6]	902,88	
	MA-ACS(CC-MO)	1097	[828, 900, 945]	891	
la23 (15x10)	AG-ACS	2244	[1004,64 1092 1146,6]	1081,1	1032
	MA-ACS(MO)		[969,68 1054 1106,7]	1043,5	
	MA-ACS(CC)		[966 1050 1102,5]	1039,3	
	MA-ACS(CC-MO)	2802	[966 1050 1102,5]	1039,3	

De acordo com a Tabela 2, a abordagem proposta MA-ACS (CC-MO) mostra desempenho superior sobre AG-ACS, MA-ACS(MO) e MA-ACS (CC) especialmente quando a dimensão do problema é grande ou o problema é muito difícil, como *la16*, *la17*, *orb02* e *la23*, mas quando o tamanho do problema é pequeno, como a instância *ft06* os resultados são semelhantes. Isso se deve ao fato de que a abordagem MA-ACS (CC-MO) combina duas técnicas de buscas locais CC e MO o que melhora ainda mais o algoritmo, pois além de reordenar as tarefas, ela reduz os *gaps* existentes na máquina mais ociosa.

Na execução de cada instância foram verificados quantos indivíduos da população obtiveram seus valores de *makespan fuzzy*

até 80% do melhor indivíduo encontrado pelos algoritmos. A Tabela 3 apresenta a quantidade de indivíduos com 80% de possibilidade de serem ótimos para cada instância testada.

Pela Tabela 3 podemos verificar que por exemplo para a instância *la12*, a abordagem MA-ACS (CC-MO) encontrou 22 indivíduos com valor de *makespan* até 80% do valor do melhor indivíduo encontrado pelos algoritmos, isso para uma população de 40 indivíduos. Isto é um ponto muito positivo pois mostra que o algoritmo encontra não apenas uma solução ótima, mas um conjunto de soluções com alto grau de possibilidade de serem ótimas. Para instâncias com nível mais elevado de dificuldade como por exemplo as instâncias com 10 máquinas e 10 tarefas *abz6*, *orb02* e *la23*, o algoritmo AG-ACS não consegue obter nenhuma solução com alto grau de qualidade em comparado com os resultados obtidos com o algoritmo MA-ACS (CC-MO), que obtêm 15, 15 e 16 indivíduos respectivamente com mais de 80% de possibilidade, evidenciando mais uma vez a qualidade da abordagem proposta.

Tabela 3: Quantidade de indivíduos com $+80\%$ de possibilidade de serem ótimos.

Instância	AG-ACS	MA-ACS(MO)	MA-ACS(MO)	MA-ACS(CC-MO)
ft06 (6×6)	2	2	2	4
la01 (10×5)	16	16	16	16
la08 (15×5)	16	13	16	15
la11 (20×5)	18	20	17	18
la12 (20×5)	13	14	16	22
la16 (10×10)	12	15	10	21
la17 (10×10)	12	15	13	15
abz6 (10×10)	0	15	15	15
orb02(10×10)	0	12	16	15
la23 (15×10)	0	19	16	16

6. CONCLUSÃO

Observando os resultados acima, podemos perceber que os algoritmos propostos possuem a vantagem de lidarem com o problema de *job shop* com parâmetros *fuzzy* na sua íntegra, sem necessidade de métodos de *defuzzificação* ou comparação de números *fuzzy* para encontrar o *makespan fuzzy*. Uma outra vantagem bastante interessante e útil da abordagem proposta neste artigo é que os algoritmos são capazes de encontrar um conjunto de soluções com alto grau de possibilidade de serem ótimas.

AGRADECIMENTOS

Os autores agradecem a Capes e à Fapesp pelo apoio financeiro para a realização deste trabalho.

REFERÊNCIAS

[1] R. E. Bellman. "On a routing problem. Quarterly". *Applied Mathematics*, vol. 16, pp. 87-90, 1958.

[2] T. R. Bonfim. "Escalonamento Memético e Neuro-Memético de Tarefas". Ph.D. thesis, Faculdade de Engenharia Elétrica e Computação - Universidade Estadual de Campinas, Campinas, 2006.

[3] G. Bortolan and R. Degani. "A review of some methods for ranking fuzzy subsets", in: D. Dubois, H. Prade and R. Yager (eds.). *Readings in Fuzzy Sets for Intelligent Systems*, Morgan Kaufmann, San Francisco, CA, pp. 149-158, 1993.

[4] D. Dubois, H. Fargier and H. Prade. "Fuzzy constraints in job-shop scheduling". *Journal of Intelligent Manufacturing*, vol. 6, no. 4, pp. 215-234, 1995.

[5] I. González Rodrígues, C. R. Vela and J. Puente. "A memetic approach to fuzzy job shop based on expectation model". *Proceedings of IEEE International Conference on Fuzzy Systems - FUZZ-IEEE*, London, pp. 692-697, 2007.

[6] F. Hernandes. "Algoritmos para Problemas de Grafos com Incertezas". Ph.D. thesis, Faculdade de Engenharia Elétrica e Computação - Universidade Estadual de Campinas, Campinas, 2007.

[7] F. T. Lin. "Fuzzy job-shop scheduling based on ranking level $(\lambda, 1)$ interval-valued fuzzy numbers". *IEEE Transaction Fuzzy System*, vol. 10, no. 4, pp. 510-522, 2000.

[8] Q. Niu, B. Jiao and X. Gu. "Particle swarm optimization combined with genetic operators for job shop scheduling problem with fuzzy processing time". *Applied Mathematics and Computation*, vol. 205, no. 1, pp. 148-158, 2008.

[9] S. Okada and T. Soper. "A shortest path problem on a network with fuzzy arc lengths". *Fuzzy Sets and Systems*, vol. 109, pp. 129-140, 2000.

[10] N. Tamura. OR-library. Available online at: http://bach.istc.kobe-u.ac.jp/csp2sat/jss/.

[11] L. Zadeh. "Fuzzy Sets". *Information and Control*, vol. 8, pp. 338-353, 1965.