METAHEURÍSTICA HÍBRIDA APLICADA AO PROBLEMA DE JOB SHOP

Mateus Matos Rizzi
Universidade Federal do ABC – UFABC, Instituto de Ciência e Tecnologia
Av. dos Estados, 5001, Bangú, CEP 09210-580, Santo André, SP, Brasil
m.rizzi@ufabc.edu.br

Antonio Augusto Chaves
Universidade Federal de São Paulo – UNIFESP, Instituto de Ciência e Tecnologia
Rua Talin, 330, CEP 12231-280, São José dos Campos, SP, Brasil
antonio.chaves@unifesp.br

Edson Luiz França Senne
Universidade Estadual Paulista – UNESP, Faculdade de Engenharia de Guaratinguetá
Av. Ariberto Pereira da Cunha, 333, CEP 12516-410, Guaratinguetá, SP, Brasil
elfsenne@feg.unesp.br

Luiz Antonio Nogueira Lorena
Instituto Nacional de Pesquisas Espaciais – INPE
Av. dos Astronautas, 1758, CEP 12227-010, São José dos Campos, SP, Brasil
lorena@lac.inpe.br

RESUMO

Este trabalho tem como objetivo aplicar o método híbrido *Clustering Search* (CS) ao problema de programação das tarefas em máquinas no ambiente de produção *Job Shop*. O objetivo do *Job Shop* é a minimização do tempo de fluxo total de *n* tarefas ser processadas em *m* máquinas, de acordo com um roteiro preestabelecido. O método CS é um método híbrido que procura combinar metaheurísticas e heurísticas de busca local, de tal forma que a busca seja intensificada somente em regiões promissoras do espaço de soluções. Para representar soluções do *Job Shop* utiliza-se o conceito de chaves aleatórias da metaheurística *Biased Random Key Genetic Algorithm* (BRKGA). Para validar o método híbrido proposto são realizados testes computacionais com instâncias disponíveis na literatura.

PALAVARAS CHAVE. Job Shop. Metaheurística híbrida. Chaves Aleatórias.

Área principal (Metaheurísticas)

ABSTRACT

This paper aims to apply the hybrid method Clustering Search (CS) to solve the problem of scheduling tasks on machines in the Job Shop production environment. The objective of Job Shop is to minimize the total time of n tasks to be processed on m machines, according to a predetermined order. The CS method is a hybrid method that seeks to combine metaheuristics and local search heuristics, so that the search is intensified only in promising regions of the solution space. We use the concept of random keys of the metaheuristic Biased Random Key Genetic Algorithm (BRKGA) to represent the Job Shop solutions. Computational tests with instances available in the literature are performed to validate the proposed hybrid method.

KEYWORDS. Job Shop. Hybrid metaheuristic. Random Keys.

Main area (Metaheuristics)

1. Introdução

As metaheurísticas podem ser usadas como uma ferramenta para encontrar respostas eficientes de problemas de otimização. Na literatura, podemos encontrar inúmeros estudos sobre problemas com aplicações reais, tais como: alocação de navios em berços para carregar e descarregar cargas em portos (Araújo e Chaves, 2014), dimensionamento e sequenciamento de lotes baseado num problema real de uma indústria Têxtil (Pimentel *et al*., 2010), processo de seleção para criar reservas ambientais (Vieira, 2005), entre outros.

As metaheurísticas podem ser definidas como métodos de solução que coordenam procedimentos de busca locais com estratégias de mais alto nível, de modo a criar um processo capaz de escapar de mínimos locais e realizar uma busca robusta no espaço de soluções de um problema (Glover e Kochenberger, 2003).

Uma das primeiras metaheurísticas propostas foi o Algoritmo Genético (GA, do inglês *Genetic Algorithm*). Os GAs foram propostos por Holland (1975) aplicando a lógica probabilística da seleção natural como um mecanismo de pesquisa inteligente para encontrar soluções otimizadas. Para fundamentar a proposta, foram utilizados os conceitos da teoria da evolução de Charles Darwin. O GA possui os seguintes processos:

- Avaliação: critério de verificação da aptidão das soluções (considerando que as soluções são os indivíduos da população);
- Seleção: define-se um critério para selecionar indivíduos para reprodução, onde os mais adaptados (melhores soluções) têm uma probabilidade maior de se reproduzir do que soluções ruins;
- Cruzamento: componente responsável por combinar características de diferentes soluções;
- Mutação: permite adicionar variedade à população de soluções;
- Atualização: os novos indivíduos são inseridos na população de soluções.

Recentemente, Bean (1994) propôs o Algoritmo Genético com Chaves Aleatórias (RKGA, do inglês *Random Key Genetic Algorithm*), que faz uso da representação de soluções como vetores de chaves aleatórias (números reais entre 0 e 1 gerados aleatoriamente). Esta representação permite generalizar os componentes do Algoritmo Genético, sendo o programador responsável por definir apenas o método para decodificar a solução de chaves aleatórias em uma solução do problema abordado. Gonçalves e Resende (2004) apresentaram uma adaptação do RKGA na qual se utiliza uma classificação da população em elite e não-elite, selecionando sempre um pai da elite para o processo de cruzamento. Este método foi chamado BRKGA (do inglês *Biased*).

Neste trabalho utiliza-se o conceito de chaves aleatórias no método híbrido *Clustering Search* (CS) proposto por Oliveira *et al*. (2013). O CS é um método híbrido que combina metaheurísticas e heurísticas de busca local, em que a busca é intensificada somente em regiões do espaço de soluções que merecem atenção especial (chamadas regiões promissoras). Assim, será utilizado o BRKGA como gerador de soluções para o CS e os demais componentes do CS também são adaptados para trabalhar com soluções representadas como vetores de chaves aleatórias. O objetivo é obter um método eficiente e que demande pouco esforço de programação para ser aplicado em outros problemas de otimização.

Para avaliar a eficiência do BRKGA+CS propõe-se a aplicação deste método híbrido no problema de *Job Shop* (Giffler e Thompson, 1960). O *Job Shop* é um problema de programação de tarefas, no qual se busca encontrar a melhor sequência de tarefas a serem realizadas em um conjunto de máquinas (cada tarefa possui uma ordem de processamento nas máquinas) de tal forma a minimizar o tempo total. Testes computacionais com instâncias clássicas disponíveis na literatura são realizados para mostrar a robustez do BRKGA+CS.

O restante do artigo está organizado como segue. A Seção 2 apresenta uma descrição dos métodos BRKGA e CS. A Seção 3 contém o método híbrido BRKGA+CS e as aplicações deste método ao problema de *Job Shop*. Os resultados computacionais são apresentados na Seção 4. Por fim, as conclusões são explanadas na Seção 5.

2. Métodos

Nesta seção são descritos os principais conceitos dos métodos de otimização BRKGA e CS utilizados neste trabalho para encontrar boas soluções para o problema de *Job Shop*.

2.1 BRKGA

Algoritmo Genético de Chaves Aleatórias Viciadas (BRKGA, do inglês *Biased Random Key Genetic Algorithm*) foi proposto por Gonçalves e Almeida (2002) e Gonçalves e Resende (2004), e difere do RKGA na forma que compõe as gerações. Essa metaheurística é composta por indivíduos formados por genes que tem um número n de alelos, os quais são gerados aleatoriamente com números reais no intervalo [0, 1]. Estes números são chamados de chaves aleatórias (*random keys*). A parte mais complexa da elaboração do algoritmo é a codificação do problema em n chaves aleatórias, sendo que n esta relacionado ao problema abordado.

Para cada problema é utilizado um algoritmo determinístico chamado decodificador (*decoder*), que transforma um vetor de chaves aleatórias em uma solução viável do problema e retorna um valor de solução (*fitness*) determinado. O restante do algoritmo BRKGA é o mesmo para qualquer problema de otimização.

O BRKGA trabalha com um número p de soluções, cada qual formada por um vetor de n chaves aleatórias, que são enviadas para o *decoder*. A população evolui ao longo de um número de gerações e os valores de *fitness* são utilizados para classificar as soluções em ordem decrescente (do melhor para o pior). Uma constante definida como população elite é utilizada para dividir a população em função de melhores e piores soluções. Sendo o tamanho da população elite definido por p_e, e o restante da população formada por um número $p - p_e$ de soluções. As gerações seguintes são formadas por três operações:

- Copiar a população elite para a nova geração
- Realizar o cruzamento de alelos entre um indivíduo da p_e com um indivíduo da $p - p_e$. Tal troca de alelo é realizada através de um sorteio, no qual o indivíduo da população elite tem uma chance *rhoe* de ser escolhida. A combinação dos pais é realizada com o método de cruzamento uniforme parametrizado, onde o filho herda a i-ésima chave do pai A com probabilidade R(E) > 0,5 e a do pai B com probabilidade 1-R(E).
- Criar um número p_m de mutantes, soluções com chaves aleatórias geradas aleatoriamente.

2.2 Clustering Search

Oliveira *et al*. (2013) propuseram o método híbrido *Clustering Search* (CS), o qual se baseia no conceito de regiões promissoras e aplica heurísticas de busca local nestas regiões para intensificar o processo de convergência para boas soluções. O CS funciona com três componentes principais: gerador de soluções, processo de agrupamento e heurística de busca local.

O gerador de soluções pode ser implementado por qualquer algoritmo de otimização que gera soluções diversificadas do espaço de busca. Geralmente, utiliza-se uma metaheurística para trabalhar como um gerador de soluções, explorando o espaço de busca através de uma manipulação de um conjunto de soluções, de acordo com sua estratégia de busca específica

O processo de agrupamento tem o propósito de definir regiões de soluções (*clusters*) do problema, para definir em quais regiões será aplicado o método de intensificação de busca. Cada *cluster* tem um centro c_i que representa a localização do *cluster* no espaço de soluções. Além disso, os *clusters* possuem um contador de volume, v_i, que representa o número de soluções geradas nessa região. Quanto maior o número de soluções no *cluster*, mais chances a região tem de ser promissora. Se o *cluster* se tornar promissor, o centro desse *cluster* recebe uma busca local. Um *cluster* promissor é encontrado quando o volume de um *cluster* atinge o limitante λ.

A cada interação, o processo de agrupamento precisa identificar semelhanças entre uma solução gerada pela metaheurística e as soluções dos centros de *clusters*. Então, é definida uma distância Δ que retorna um número positivo quantificando a diferença da solução no centro de um *cluster* para a solução gerada pela metaheurística.

A solução deve ser agrupada ao *cluster* mais similar (com menor distância). O processo de assimilação atualiza o centro com características oriundas da solução agrupada, podendo causar uma piora na qualidade do centro. Para prevenir que essa situação ocorra, podem-se utilizar diferentes porcentagens de assimilação. Se a solução agrupada for melhor que o centro utiliza-se uma porcentagem de assimilação alta e caso contrário uma porcentagem baixa. Para realizar esta tarefa pode-se utilizar o método Reconexão por Caminhos (PR, do inglês *Path Relinking*) (*Glover e Martí, 2000*) ou um operador de cruzamento.

Quando um *cluster* se torna promissor é aplicada uma heurística de busca local para intensificar a busca na região representada pelo centro deste *cluster*. Estas heurísticas são específicas para cada problema abordado.

3. BRKGA+CS

O método híbrido BRKGA+CS pode ser dividido em cinco etapas que estão representadas no fluxograma da Figura 1. O conceito de chaves aleatórias do BRKGA também é utilizado nos centros dos *clusters* do CS. Desta forma tem-se um método genérico no qual o programador deve se preocupar apenas com as funções *Decoder* e Busca Local.

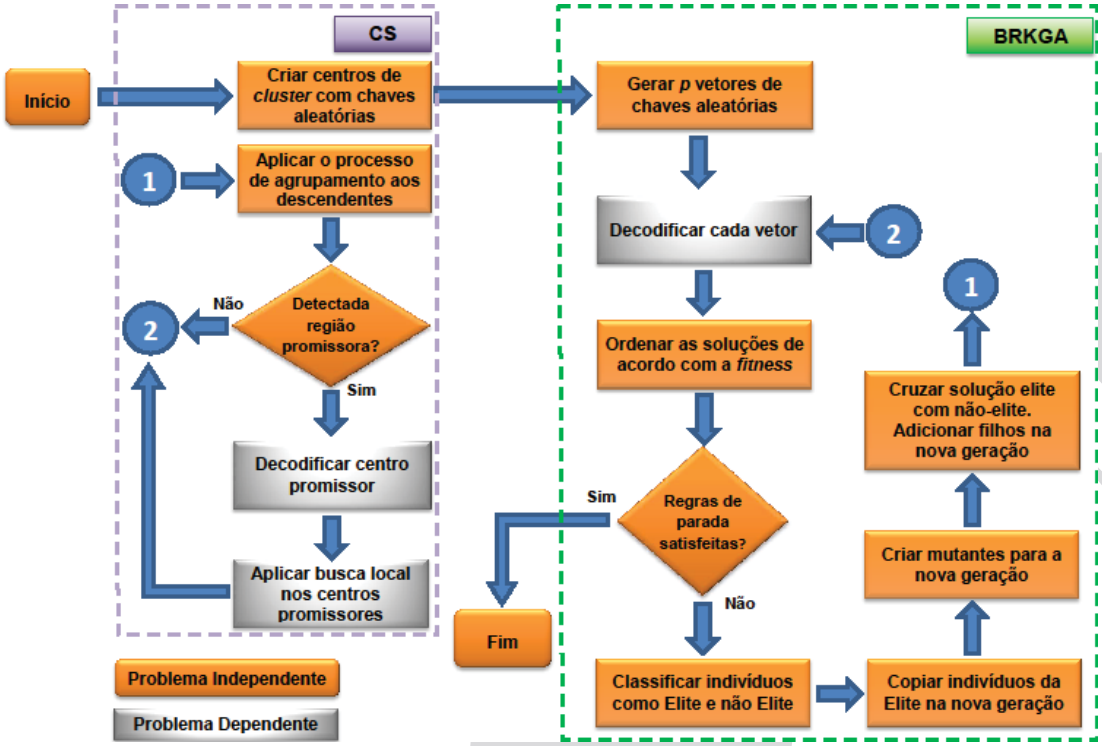

Figura 1: Fluxograma do BRKGA+CS
Fonte: Pomari e Chaves (2014)

A primeira etapa do BRKGA+CS é a criação dos centros dos *clusters* com soluções geradas aleatoriamente. Os centros são representados com vetores de chaves aleatórias. Desta forma, pode-se utilizar a distância euclidiana para calcular a similaridade entre soluções e tornar os componentes do CS (exceto a busca local) independentes do problema de otimização abordado.

A segunda etapa consiste no processo de evolução do BRKGA, que gerará soluções para o processo de agrupamento do CS. A população inicial do BRKGA é gerada e a cada geração copiam-se as soluções da população elite para a nova população, geram-se os filhos por meio do cruzamento uniforme parametrizado e criam-se os mutantes que são inseridos na população. Os filhos criados em cada geração são enviados para o processo de agrupamento.

No processo de agrupamento (terceira etapa) cada filho é agrupado ao *cluster* mais similar de acordo com a medida de distância adotada. A solução agrupada ao *cluster* deve causar uma perturbação em seu centro, de tal forma que alguma informação da nova solução seja adquirida pelo centro do *cluster*. Para tal, pode-se utilizar o método *Path-Relinking* (PR) para analisar o caminho entre o centro do *cluster* e a solução agrupada ou aplicar o cruzamento uniforme parametrizado do BRKGA. Com o PR, o novo centro será a melhor solução obtida neste caminho, mesmo que esta seja pior que o centro atual. Esta estratégia possibilita aumentar a diversidade do método e evita que a intensificação da busca local fique presa em algumas regiões do espaço de soluções.

Quando o número de soluções agrupadas em um *cluster* atinge o limitante λ (quarta etapa), seu centro é decodificado em uma solução do problema e esta é enviada para o componente de busca local do CS (quinta etapa). Após realizar a intensificação da busca nesta região promissora, o centro do *cluster* é atualizado caso a heurística obtenha uma solução melhor que a solução armazenada no centro. O fluxo do algoritmo retorna para o processo evolutivo do BRKGA e o critério de parada do método será o número máximo de gerações do BRKGA.

3.1 BRKGA+CS aplicado ao Job Shop

No ambiente industrial é de extrema importância a otimização dos problemas que surgem no nível operacional, gerando um aumento da produção e também do lucro. Neste contexto tem-se o problema de programação *Job Shop*, que consiste em um número $J = \{1,2,3, ..., j\}$ de tarefas e m máquinas. Cada tarefa é processada nas m máquinas, de acordo com um roteiro preestabelecido. Todas as tarefas estão disponíveis para processamento no instante zero e não é permitida a interrupção do processamento de qualquer tarefa. O objetivo é minimizar o tempo de fluxo total das tarefas.

O *Job Shop* foi estudado em inúmeros trabalhos da literatura, com diversos tipos de métodos heurísticos e exatos. Dentre trabalhos que utilizam métodos heurísticos podemos citar os algoritmos genéticos de Storer *et al*. (1992), Aarts *et al*. (1994), Della Croce *et al*. (1995), Dorndorf e Pesch (1995), e Gonçalves e Beirão (1999), o GRASP de Binato *et al*. (2002) e Aiex *et al*. (2003), o método híbrido algoritmo genético/simulated annealign de Wang e Zheng (2001), e a busca tabu de Nowicki e Smutnicki (1996). Sabuncuoglu e Bayiz (1999) utilizam a técnica *Bean search* para encontrar as soluções ótimas para instâncias com 10 tarefas e 10 máquinas. Recentemente, Gonçalves e Resende (2014) obtiveram as melhores soluções conhecidas para um conjunto de instâncias do *Job Shop* disponíveis na literatura com a utilização do BRKGA em conjunto a metaheurística Busca Tabu.

Gonçalves e Resende (2014) apresentam um exemplo de *Job Shop* com 4 tarefas e 3 máquinas, denotadas por (a, b e c). As matrizes de operações de cada tarefa O e de tempos de processamento nas máquinas P são:

$$O = \begin{bmatrix} a & b & c \\ b & c & a \\ c & b & a \\ b & a & c \end{bmatrix} \quad P = \begin{bmatrix} 2 & 3 & 4 \\ 3 & 2 & 3 \\ 5 & 2 & 3 \\ 2 & 4 & 2 \end{bmatrix}$$

Por exemplo, a primeira linha da matriz O indica que a tarefa 1 é processada nas máquinas a-b-c, nesta ordem, com tempos de processamento de 2, 3 e 4, respectivamente, correspondente aos elementos da primeira linha da matriz P.

Neste trabalho propõe-se utilizar o método BRKGA+CS para resolver o problema *Job Shop*. Desta forma, é necessário desenvolver um decodificador de soluções representadas por chaves aleatórias e heurísticas de buscas locais específicas para o *Job Shop*.

A representação de uma solução do *Job Shop* em um vetor de chaves aleatórias consiste em um vetor de números reais no intervalo [0,1] de tamanho $m * J$ (máquinas x tarefas). Cada posição do vetor de chaves aleatórias representa as tarefas e n_j é o número de operações da tarefa

j. Uma solução é gerada atribuindo aleatoriamente um número entre [0,1] para cada posição do vetor. Um possível vetor de chaves aleatórias para o exemplo anterior é mostrado na Figura 2.

	n_1			n_2			n_3			n_4		
Tarefas	1	1	1	2	2	2	3	3	3	4	4	4
Chaves aleatórias	0.67	0.78	0.49	0.07	0.35	0.87	0.17	0.02	0.93	0.25	0.52	0.42

Figura 2 – Representação do problema *Job Shop* em chaves aleatórias.

Para decodificar este vetor em uma solução do *Job Shop*, deve-se ordenar o vetor de chaves aleatórias em ordem crescente mantendo a relação com as tarefas. Desta forma, têm-se as prioridades das tarefas que serão executadas nas máquinas. Portanto, para o exemplo da Figura 2 a ordem de prioridade é alocar a tarefa 3 em sua primeira máquina (máquina *c*); tarefa 2 é alocada em sua primeira máquina (máquina *b*); tarefa 3 alocada em sua segunda máquina (máquina *b*), e assim por diante (ver Figura 3).

Tarefas	3	2	3	4	2	4	1	4	1	1	2	3
Chaves aleatórias	0.02	0.07	0.17	0.25	0.35	0.42	0.49	0.52	0.67	0.78	0.87	0.93
Operações	3,1	2,1	3,2	4,1	2,2	4,2	1,1	4,3	1,2	1,3	2,3	3,3

Figura 3 – Solução decodificada para o problema *Job Shop*

As tarefas são alocadas às máquinas de acordo com a ordem pré-estabelecida em que cada tarefa deve passar pelas máquinas. Caso a máquina necessária esteja ocupada, a tarefa é alocada depois que a tarefa anterior tenha terminado. Para o exemplo ilustrado, a representação gráfica das operações em máquinas ao longo do tempo (diagrama de Gantt) é mostrada na Figura 4. O tempo de fluxo total das tarefas é 15.

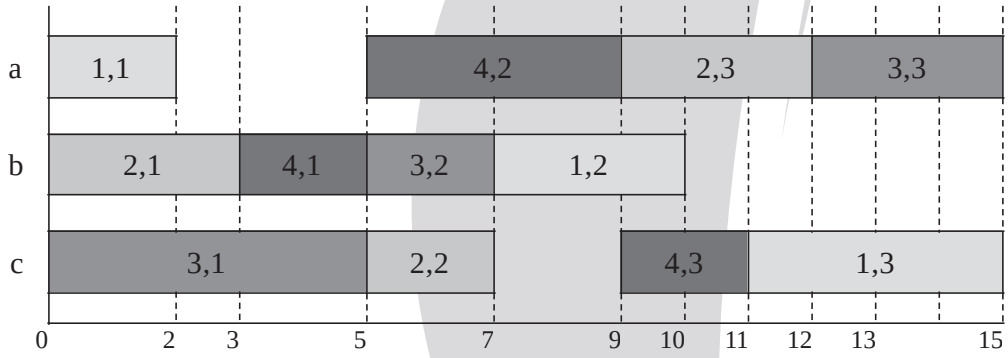

Figura 4 – Diagrama de Gantt para o exemplo de *Job Shop*.

O método BRKGA+CS inicializa gerando aleatoriamente os centros dos *clusters* representados como chaves aleatórias. Para cada solução é gerada um vetor com *n* chaves aleatórias escolhidas com distribuição uniforme no intervalo [0, 1]. Após esta fase inicial, o método entra em um processo iterativo no qual serão executados o processo de evolução do BRKGA e os componentes de agrupamento e busca local do CS.

O BRKGA segue o padrão proposto por Gonçalves e Resende (2014). A população inicial é criada da mesma maneira que os centros dos *clusters*. Esta população é ordenada em termos de qualidade dos indivíduos e dividida em dois grupos (elite e não-elite). A próxima geração do BRKGA é formada pelos p_e indivíduos da elite e por p_m mutantes gerados aleatoriamente. Para completar a população são realizados p - p_e - p_m cruzamentos entre um indivíduo da elite e um indivíduo da não-elite. O método de cruzamento uniforme parametrizado é utilizado para gerar os filhos, sempre priorizando escolher alelos do indivíduo elite (*rhoe* > 0.7). A cada geração a nova população substitui a população antiga.

Os filhos gerados em cada geração do BRKGA são enviados para o processo de agrupamento do CS com objetivo de detectar regiões promissoras do espaço de soluções. Para cada solução calcula-se a distância euclidiana desta aos centros dos *clusters* e agrupa-se ao *cluster* mais similar (com menor distância). Assim, o volume deste *cluster* é aumentado em uma unidade e centro do *cluster* precisa ser atualizado com informações presentes na solução agrupada. Esta atualização é realizada pelo processo de assimilação. Neste trabalho utiliza-se o método de cruzamento uniforme parametrizado para atualizar o centro do *cluster* com alguma informação da solução agrupada (*rhoe* > 0.8).

Após a atualização do centro, analisa-se o volume deste *cluster*. Caso o volume tenha atingido um limitante λ, considera-se que uma grande quantidade de informação está sendo gerada nesta região podendo ser uma região promissora. Assim, o centro do *cluster* é decodificado em uma solução do *Job Shop* e aplica-se uma intensificação da busca nesta solução por meio de heurística de busca local específica para o problema. O centro do *cluster* é atualizado com a melhor solução obtida pela busca local.

Na busca local o centro do *cluster* é decodificado em uma solução do *Job Shop* e são utilizadas duas heurísticas de busca local. A heurística *Two Exchange* (Gonçalves e Resende, 2014) consiste em trocar duas tarefas que estão na mesma máquina e não possuem intervalos ociosos entre elas. Neste trabalho propõe-se uma segunda heurística que consiste em trocar todos os pares de tarefas de cada máquina. Soluções que violem a restrição de ordem das tarefas nas máquinas são penalizadas na função objetivo. As duas heurísticas são executadas na busca local até não existir movimentos de melhoras na solução corrente.

4. Resultados Computacionais

Esta seção apresenta os resultados dos testes computacionais do BRKGA+CS aplicado ao problema de programação *Job Shop*. Para ilustrar a eficiência do método proposto, considera-se, até o momento, 43 instâncias disponíveis na literatura (http://people.brunel.ac.uk/~mastjjb/jeb/orlib/jobshopinfo.html): FT06, FT10 e FT20 (Fischer e Thompson, 1963) e LA01 - LA40 (Lawrence, 1984). Os testes computacionais foram executados em um computador AMD *Dual-Core Processor* E-300 (1.3 GHZ), sob o sistema operacional Windows 8.1. Foram realizadas 20 execuções independentes para cada instância.

Os parâmetros do BRKGA+CS foram calibrados através de testes empíricos realizados sobre um subconjunto de instâncias. Os parâmetros adotados neste trabalho são:

- Tamanho da população (p) = 1000;
- Porcentagem de população elite (p_e) = 0,1;
- Porcentagem de população mutante (p_m) = 0,2;
- Chance de escolher um alelo do pai elite (*rhoe*) = 0,7;
- Número de *clusters* do CS = 20;
- Limitante de *cluster* promissor (λ) = 20;
- Número máximo de gerações do BRKGA = 400.

A Tabela 1 apresenta os resultados do BRKGA+CS para as 43 instâncias testadas. Primeiro tem-se os nomes das instâncias, o número de tarefas (J) e máquinas (m). A coluna *Opt* mostra as soluções ótimas conhecidas na literatura para estas instâncias (Gonçalves e Resende, 2014). As colunas seguintes referem-se ao BRKGA+CS: a melhor solução encontrada em 20

execuções (coluna *Melhor*), a média das soluções obtidas nas 20 execuções (coluna Média), a diferença percentual da melhor solução encontrada em relação à solução ótima (coluna *Gap*) e o tempo de execução do método (coluna *Tempo*). Como critério de parada do BRKGA+CS foi utilizado, além do número máximo de gerações, o fato de o método encontrar o valor da solução ótima disponível na literatura.

Instância				BRKGA+CS			
Nome	*J*	*m*	*Opt*	*Melhor*	*Média*	*Gap (%)*	*Tempo (s)*
FT06	6	6	55	**55**	55,00	0,00	1,70
FT10	10	10	930	936	985,10	0,65	38,14
FT20	20	5	1165	1173	1264,15	0,69	35,38
LA01	10	5	666	**666**	729,25	0,00	6,03
LA02	10	5	655	**655**	704,21	0,00	8,94
LA03	10	5	597	**597**	648,31	0,00	17,40
LA04	10	5	590	**590**	636,26	0,00	14,13
LA05	10	5	593	**593**	624,81	0,00	7,80
LA06	15	5	926	**926**	957,24	0,00	9,00
LA07	15	5	890	**890**	937,86	0,00	11,28
LA08	15	5	863	**863**	909,89	0,00	6,20
LA09	15	5	951	**951**	996,49	0,00	8,00
LA10	15	5	958	**958**	1007,82	0,00	8,00
LA11	20	5	1222	**1222**	1272,39	0,00	9,00
LA12	20	5	1039	**1039**	1102,62	0,00	6,00
LA13	20	5	1150	**1150**	1205,13	0,00	6,00
LA14	20	5	1292	**1292**	1352,25	0,00	6,00
LA15	20	5	1207	**1207**	1274,61	0,00	12,20
LA16	10	10	945	946	1048,33	0,11	22,00
LA17	10	10	784	**784**	853,07	0,00	46,83
LA18	10	10	848	**848**	918,70	0,00	23,35
LA19	10	10	842	**842**	926,83	0,00	33,01
LA20	10	10	902	907	967,69	0,55	31,68
LA21	15	10	1046	1064	1181,58	1,72	17,10
LA22	15	10	927	942	1055,13	1,62	25,00
LA23	15	10	1032	**1032**	1115,30	0,00	53,42
LA24	15	10	935	971	1074,61	3,85	27,50
LA25	15	10	977	998	1119,43	2,15	72,70
LA26	20	10	1218	1229	1343,62	0,90	44,00
LA27	20	10	1235	1293	1416,88	4,70	188,40
LA28	20	10	1216	1258	1394,49	3,45	10,50
LA29	20	10	1157	1225	1378,32	5,88	20,00
LA30	20	10	1355	1367	1503,41	0,89	32,17
LA31	30	10	1784	**1784**	1864,92	0,00	66,69
LA32	30	10	1850	**1850**	1959,25	0,00	64,39
LA33	30	10	1719	**1719**	1829,61	0,00	65,20
LA34	30	10	1721	**1721**	1888,43	0,00	108,50
LA35	30	10	1888	**1888**	2005,92	0,00	108,32
LA36	15	15	1268	1315	1449,94	3,71	74,50
LA37	15	15	1397	1451	1613,55	3,87	79,30
LA38	15	15	1196	1262	1445,18	5,52	76,06
LA39	15	15	1233	1281	1420,96	3,89	76,82
LA40	15	15	1222	1269	1411,10	3,85	92,60
Média			1080,14	1093,23	1182,55	1,12	38,87

Tabela 1. Resultados computacionais do BRKGA+CS aplicado ao *Job Shop*

Observa-se pelos resultados da Tabela 1 que o BRKGA+CS é um método eficiente na resolução do *Job Shop*. O BRKGA+CS encontra a solução ótima em 25 das 43 instâncias utilizadas nos testes. Além disso, a diferença média em relação à solução ótima é de 1,12%. As instâncias com piores resultados são LA27 e LA29 com *gaps* de 4,7% e 5,88%. Em seis instâncias nas quais não se obtêm a solução ótima têm-se *gaps* inferiores a 1%.

5. Conclusão

Neste trabalho foi proposta a aplicação do método híbrido BRKGA+CS para resolver o problema de programação *Job Shop*. Para tal encorpara-se o conceito de chaves aleatórias para representar soluções do *Job Shop*. Esta representação garante que todas as soluções geradas pelo BRKGA e pelo processo de agrupamento do CS são viáveis. Apenas o componente de busca local do CS é responsável por intensificar a busca no espaço de soluções (penalizando possíveis soluções inviáveis).

Desta forma, tem-se um método híbrido para resolver problemas de otimização e que demanda pouco esforço dos programadores, uma vez que é necessário programar apenas o decodificador de chaves aleatórias e as heurísticas de busca local.

Os resultados obtidos mostram que o método híbrido proposto é eficiente na resolução do *Job Shop*. O método foi capaz de encontrar a solução ótima (conhecida na literatura) em 19 instâncias de um conjunto de 43 instâncias testadas. Nas demais, as soluções obtidas são próximas as soluções ótimas. O tempo computacional do método é aceitável, demorando poucos segundos na maioria dos testes.

Como trabalho futuro pretende-se melhorar as heurísticas de buscas locais específicas para o *Job Shop*. Além de realizar a aplicação do BRKGA+CS em outras instâncias do *Job Shop* e a outros problemas de otimização encontrados na literatura.

Agradecimentos

Os autores agradecem ao CNPq (Conselho Nacional de Desenvolvimento Científico e Tecnológico) e a FAPESP (Fundação de Amparo à Pesquisa do Estado de São Paulo) pelo suporte financeiro parcial para a realização deste trabalho.

Referências

Aarts, E.H.L., Van Laarhoven, P.J.M., Lenstra, J.K., Ulder, N.L.J. (1994). A computational study of local search algorithms for job shop scheduling. *INFORMS J. Comput*. 6, 118–125.

Aiex, R.M., Binato, S., Resende, M.G.C. (2003). Parallel GRASP with path-relinking for job shop scheduling. *Parallel Comput.* 29, 393–430.

Araujo, E.J. e Chaves, A.A. (2014). Pareto Clustering Search aplicado ao problema de carregamento de contêiner em navios. Salvador, Brasil. SBPO – Simposio Brasileiro de Pesquisa Operacional.

Bean, J.C. (1994). Genetic algorithms and random keys for sequencing and optimization. ORSA *J. Comput*. 6, 154–160.

Binato, S., Hery, W.J., Loewenstern, D.M., Resende, M.G.C. (2002). A GRASP for job shop scheduling. In: Ribeiro, C.C., Hansen, P. (eds.) *Essays and Surveys in Metaheuristics*. Kluwer Academic, Dordrecht.

Della Croce, F., Tadei, R., Volta, G. (1995). A genetic algorithm for the job shop problem. *Comput. Oper. Res*. 22, 15–24.

Dorndorf, U., Pesch, E. (1995). Evolution based learning in a job shop scheduling environment. *Comput. Oper. Res.* 22, 25–40.

Fisher, H. and Thompson, G.L., (1963). Probabilistic Learning Combinations of Local Job-Shop Scheduling Rules, em: *Industrial Scheduling*, J.F. Muth and G.L. Thompson (eds.), Prentice-Hall, Englewood Cliffs, NJ, 225-251.

Giffler, B.; Thompson, G. (1960). Algorthms for solving production-scheduling problems. *Operations Research*, 487 – 503.

Glover, F. e Kochenberger, G. A. (2003). Handbook of Metaheuristics. Kluwer Academic Publishers, Dordrecht.

Glover, F, e Martí, R.. 2000. Fundamentals of scatter search and path relinking. *Control and Cybernetics* 39: 653-684.

Gonçalves, J.F., Almeida, J. (2002). A hybrid genetic algorithm for assembly line balancing. *J. Heuristics* 8, 629 - 642

Gonçalves, J.F., Resende, M.G.C. (2004). An evolutionary algorithm for manufacturing cell formation. *Comput. Ind. Eng*. 47, 247–273

Gonçalves, J.F., Resende, M.G.C. (2011). Biased Random Key Genetic Algorithms for Combinatorial Optimization, *J Heuristics*, 17: 487–525.

Gonçalves, J.F. Resende, M.G.C. (2014). An extended Akers graphical method with a biased random-key genetic algorithm, *International Transactions in Operational Research*, vol. 21, 215-246.

Gonçalves, J.F., Beirão, N.C. (1999). Um algoritmo genético baseado em chaves aleatórias para sequenciamento de operações. Rev. *Desenvolv. Investig. Oper.* 19, 123–137.

Holland, J.H.. (1975). Adaptation in Natural and Artificial Systems. University of Michigan Press, Ann Arbor, Michigan.

Nowicki, E., Smutnicki, C. (1996). A fast taboo search algorithm for the job shop problem. *Manag. Sci*. 42, 797–813.

Oliveira, A.C.M., Chaves,A.A. e Lorena, L. A.N. (2013). Clustering Search. *Pesquisa Operacional*, v.33, 105 – 121.

Pimentel, C.; Alvelos, F.; Duarte, A.J.S.T. e Carvalho, J.M.V. (2010). A fast heuristic for a lot splitting and scheduling problem for a textile industry. In IFAC MCPL, Coimbra, Portugal.

Pomari, C.; Chaves, A.A. (2014). Algoritmo híbrido com CS e BRKGA aplicado ao problema de alocação de berço. SBPO- Simpósio Brasileiro de Pesquisa Operacional. Salvador – BA.

Sabuncuoglu, I., Bayiz, M. (1999). Job shop scheduling with beam search. *European Journal of Operational Research 118, 390-412.*

Storer, R.H., Wu, S.D., Park, I. (1992). *Genetic algorithms in problem space for sequencing problems*. In: Proceedings of a Joint US-German Conference on Operations Research in Production Planning and Control, pp. 584–597.

Vieira, Sibelius L. (2005).Utilização de metaheurísticas no processo de seleção de reservas ambientais. *Biodivers Conserv* 16:997–1008.

Wang, L., Zheng, D.Z. (2001). An effective hybrid optimization strategy for job-shop scheduling problems. *Comput. Oper. Res.* 28, 585–596.