PODes
Revista Eletrônica Pesquisa Operacional para o Desenvolvimento

Rio de Janeiro, v.2, n.3, p. 203-215, setembro a dezembro de 2010

ALGORITMO GENÉTICO SIMBIÓTICO PARA MINIMIZAR O NÚMERO DE OBJETOS PROCESSADOS E O *SETUP* DADO O NÚMERO MÁXIMO DE PILHAS ABERTAS NUM PROBLEMA DE CORTE DE ESTOQUE

Rodrigo Rabello Golfeto
Universidade Federal Fluminense
Departamento de Engenharia de Produção
Avenida dos Trabalhadores, 420, Vila Santa Cecília, Volta Redonda, RJ, Brasil, 27255-970
rodrigo.golfeto@gmail.com

Antônio Carlos Moretti
Universidade Estadual de Campinas
Instituto de Matemática, Estatística e Computação Científica
Cidade Universitária Zeferino Vaz, s/n, Barão Geraldo, Campinas, SP, Brasil, 13084-790
moretti@imec.unicamp.br

Luiz Leduíno de Salles Neto
Universidade Federal de São Paulo
Departamento de Ciência e Tecnologia
Avenida Mário Covas, 610, Vila Nair, São José dos Campos, SP, Brasil
luiz.leduino@unifesp.br

Resumo

Este trabalho apresenta um algoritmo genético simbiótico que, dado o número máximo de pilhas abertas, busca minimizar o número de objetos processados e o *setup* num problema de corte unidimensional. O algoritmo genético simbiótico pode gerar seus próprios padrões de corte através de um processo de simbiose entre duas populações distintas, soluções e padrões. Trabalhando com os dois objetivos na função de aptidão e com a relação simbiótica entre as duas populações, o método proposto obtém bons resultados quando comparado a outros métodos descritos na literatura.

Palavras-Chaves: Problema de Corte de Estoque; Algoritmo Genético; Simbiose.

Abstract

This paper presents a genetic symbiotic algorithm with the aim to minimize the number of processed objects and the setups in a one-dimensional cutting stock problem given a maximum number of open stacks. The genetic symbiotic algorithm can generate its owns cutting patterns joint with the solutions for the problem, through an symbiotic process between two distinct populations, solutions and cutting patterns. Working with two objectives in the fitness function and with the symbiotic relationship between the two populations, the proposed method obtains good results when compared to other methods in the literature.

Keywords: Cutting Stock Problem; Genetic Algorithm; Symbiosis.

Versão inicial submetida em 13/4/2010. Versão final recebida em 23/7/2010.

1. INTRODUÇÃO

O Problema de Corte de Estoque (PCE) objetiva encontrar a melhor forma de se obter itens demandados a partir do corte de peças maiores, minimizando algum tipo de custo ou maximizando o lucro. Um problema de corte pode ser decomposto em dois subproblemas distintos: gerar os padrões de corte e combiná-los da melhor forma possível, a fim de atingir a demanda exigida.

O primeiro trabalho publicado a cerca deste problema foi realizado por Kantorovich (1960) na década de 30 do século passado, publicado 30 anos mais tarde. Problemas de mesma natureza foram tratados por Paull e Walter (1954), Metzger (1958) e Eilon (1960). Porém, como observado por Dowsland e Dowsland (1992), estes métodos são adequados apenas para problemas pequenos. Com os trabalhos de Gilmore e Gomory (1961,1963) houve um grande avanço no estudo dos problemas de corte. Mais precisamente, o PCE consiste em cortar peças de tamanhos maiores W em menores w_i com objetivo de atender uma demanda d_i onde $W > w_i$. Segundo a tipologia de Wäscher *et al.* (2007) este problema é classificado como SSSCSP (*Single Stock Size Cutting Stock Problem*). Cada combinação diferente de itens que serão cortados de uma bobina específica é chamado de padrão de corte e a cada troca de padrão de corte há um custo para reconfiguração da máquina chamado *setup*. O PCE pode ser formulado da seguinte forma:

$$\text{Minimizar } c_1 \sum_{j=1}^{n} x_i$$

$$\text{Sujeito a } \sum_{j=1}^{n} a_{ij} x_j = d_j, \qquad i = 1, ..., m.$$

$$x_j \in N \qquad j = 1, ..., n.$$

onde,

c_1 é o custo de cada peça mestre; d_j é a demanda do item j, n refere-se ao número de padrões de corte e a_{ij} a quantidade de itens do tipo i no padrão j.

Haessler (1975) foi o primeiro a tratar do problema de corte unidimensional não-linear, isto é, considerando tanto a redução de perda devido a sobras, quanto à minimização do número de *setups* da máquina de corte, que pode ser formulado da forma que se segue:

$$\text{Minimizar } c_1 \sum_{j=1}^{n} x_i + c_2 \sum_{j=1}^{n} \delta(x_j)$$

$$\text{Sujeito a } \sum_{j=1}^{n} a_{ij} x_j = d_j, \qquad i = 1, ..., m.$$

$$x_j \in N \qquad j = 1, ..., n.$$

onde,

c_2 refere-se ao custo de cada troca de padrão de corte;

$$\text{e } \delta(x_j) = \begin{cases} 1 \text{ se } x_i > 0, \\ 0 \text{ se } x_i = 0. \end{cases}.$$

Além do número de objetos processados e do *setup*, outros custos e restrições podem ser relevantes para uma indústria que possui uma linha de corte em sua produção. Um importante objetivo descrito na literatura é a minimização do número máximo de pilhas abertas ou a otimização do número de *setup*s e objetos processados dado um número máximo de pilhas abertas. Pilhas estas que podem ser organizadas segundo as especificações do produto - sua

largura, no caso do problema de corte unidimensional - ou quanto aos seus destinos finais.

Na literatura podem ser encontrados diversos trabalhos que tratam o problema de Minimização de Pilhas Abertas, a saber: Yuen (1991, 1995), Yanasse (1997), Fink e Voß (1999) e Oliveira e Lorena (2002), o Problema Integrado: Armbruster (2002), Pileggi (2005) *et al.* e Yanasse e Lamosa (2007). Entretanto, há apenas um trabalho na literatura que propõe um método para minimização do número de objetos processados e *setup*s dado um número máximo de pilhas abertas desenvolvido por Belov (2003).

Em Belov (2003) as pilhas são organizadas segundo sua largura w_i. Uma pilha é considerada aberta quando o primeiro item w_i é cortado e somente pode ser removida (fechada), quando a ultima peça w_i for cortada.

Um método promissor para se encontrar boas soluções em problemas de otimização são os Algoritmos Genéticos que foram introduzidos por Holland (1975). Esta conhecida metaheurística tem suas bases na Computação Evolutiva onde, segundo Von Zuben (2003), três nomes destacaram-se: Holland (1962), Bremermann (1962) e Fraser (1957). Em linhas gerais estes algoritmos são baseados na teoria da evolução de Darwin na qual os indivíduos mais aptos têm maiores chances de procriar e, por conseguinte, passar parte de suas características para as gerações futuras.

Este trabalho propõe um algoritmo genético com uma relação de simbiose entre populações de espécies distintas, denominadas soluções e padrões, para minimização do número de objetos processados e do *setup* dado um número máximo de pilhas abertas (organizadas pela largura dos itens) num problema de corte de estoque. A Seção 2 trata, brevemente, dos algoritmos genéticos e do processo de simbiose. Na Seção 3 é detalhada a construção computacional realizada. A Seção 4 é dedicada aos testes computacionais. Finalmente, a Seção 5 trata das conclusões.

2. ALGORITMO GENÉTICO SIMBIÓTICO

Os algoritmos genéticos fazem parte da computação evolutiva, uma área onde se busca aplicar a teoria da evolução como paradigma de solução de problemas. No caso específico dos algoritmos genéticos eles estão baseados na teoria da evolução de Darwin, publicada em meados do século XIX, onde os indivíduos mais aptos têm maiores chances de procriar e transmitir suas características para gerações futuras. São conceitos chave para o entendimento de algoritmos genéticos:

Fitness: Medida de adaptabilidade de um indivíduo ao meio;

Genes: Blocos funcionais do DNA.

Genoma: A coleção de genes que forma um indivíduo.

Seleção: É responsável por selecionar os indivíduos mais aptos para o cruzamento. Os principais métodos de seleção são: torneio, elitismo e diversidade. Estes métodos podem ser combinados.

Recombinação: Também conhecido como cruzamento ou *crossover*, é a *mistura* de genes de dois indivíduos para gerar um descendente. Existem vários tipos de recombinação, sendo os mais utilizados: ponto de corte, uniforme e máscara.

Mutação: É a probabilidade de um gene qualquer do indivíduo sofrer uma alteração aleatória após a operação de recombinação.

2.1. SIMBIOSE

Em Allaby (1988), o termo simbiose é definido como um termo geral para descrever

situações onde organismos distintos vivem juntos em estreita associação. Como definido originalmente, o termo engloba todos os tipos de relação de mutualismo e parasitismo. Mutualismo é definido como uma interação entre membros de duas espécies diferentes onde ambos são beneficiados. Pianka (1994) apresenta alguns exemplos de interações entre espécies como, por exemplo, pássaros polinizadores e plantas floríferas, formigas e plantas, pássaros e búfalos.

O Algoritmo Genético Simbiótico (AGS), também chamado de algoritmo cooperativo (Potter, 1997; Kim *et al.*, 2000), basicamente tem como objetivo dividir o problema estudado em n subproblemas utilizando *n* diferentes espécies para sua representação computacional. Dividindo o problema em n populações distintas, obtém-se boas soluções para o problema utilizando estruturas simples, que trabalhando em conjunto, tem-se mostrado mais eficientes que estruturas complexas. Kim *et al.* (2001, 2006) propõe um algoritmo evolucionário endosimbiótico para otimização; a idéia básica é incorporar a evolução de células eucarióticas (Margullis, 1981) nos já existentes algoritmos simbióticos. Dessa forma, quando um indivíduo encontra um parceiro com alta aptidão em relação a ele, a combinação inteira passa por um processo de evolução sem haver a troca de parceiros.

Os trabalhos de Eguchi *et al.* (2003), Hirasawa *et al.* (2000) e Mao *et al.* (2000) também simulam relações de simbiose. Entretanto, nesses algoritmos cada indivíduo da população é tratado como se fosse de uma espécie única que desenvolve relações de simbiose com outros indivíduos da população. Em Hirasawa *et al.* (2000) descrevem um exemplo de uma possível relação de simbiose: se o indivíduo *i* está próximo do indivíudo *j* e o *fitness* do indivíduo *i* é maior que o do indivíduo *j* então o indivíduo *i* explora o indivíduo *j*. Tsujimura *et al.* (2001) apresenta um AGS para o problema de *job shop scheduling* e Chang *et al.* (2001) para um problema de localização de facilidades.

3. CONSTRUÇÃO COMPUTACIONAL

O método *Symbio* foi introduzido no trabalho de Golfeto *et al.* (2007a) através da aplicação de um algoritmo genético simbiótico para o problema de corte de estoque unidimensional, paralelamente um estudo para o caso multi-objetivo do mesmo problema também foi desenvolvido resultando em dois diferentes trabalhos (Golfeto *et al.*, 2007b; 2007c). Ambas as abordagens tiveram estudos mais aprofundados publicados nos seguintes trabalhos Golfeto *et al.* (2009a, 2009b).

Além disso, uma aplicação do método *Symbio* visando apenas à redução do número de *setups*, ou seja, partindo de uma solução inicial otimizada em relação ao número de objetos processados, também foi realizada com grande sucesso (Golfeto *et al.*, 2009c).

A estrutura computacional do método *Symbio* se baseia no trabalho de Khalifa *et al.* (2006) que apresenta uma solução para o problema de corte utilizando algoritmos genéticos. Neste trabalho os genes são processados aos pares, sendo que o primeiro gene representa a quantidade de vezes que o padrão representado pelo segundo gene será processado. No método proposto neste artigo este segundo gene representará um indivíduo na população de padrões que, como pode ser visto na Figura 2, serve apenas como um ponteiro para população de padrões, assim, no caso do indivíduo em destaque da Figura 2, há três padrões distintos (37, 11 e 32) sendo executados, respectivamente, duas, quatro e cinco vezes cada, perfazendo um total de onze objetos processados.

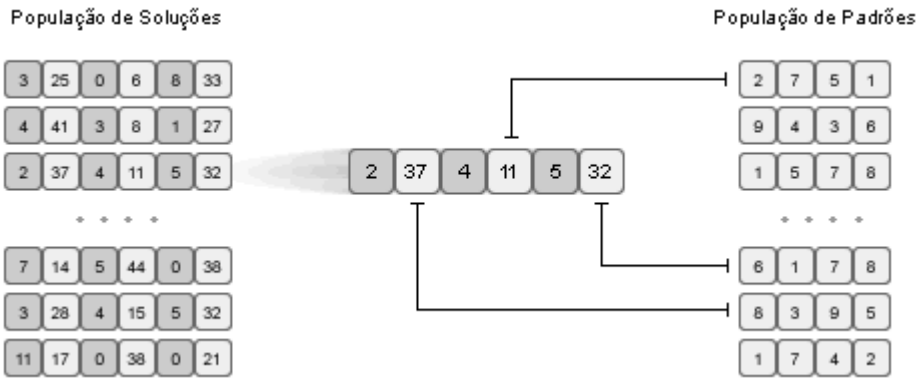

Figura 2: Estrutura dos genes.

Vale ressaltar que a estrutura simbiótica utilizada é idêntica a encontrada no trabalho de Golfeto *et al.* (2009a), já citado anteriormente.

Entretanto, nesta construção, é adicionada uma regra para aceitação de padrões, através da qual soluciona-se o problema de sequenciamento, onde um padrão só pode ser aceito se o seu processamento não implicar no estouro do número máximo de pilhas abertas. Dessa forma, se o processamento de um padrão implicar na transgressão da restrição do número máximo de pilhas a sua freqüência é imediatamente alterada para zero. Vale ainda ressaltar que a ordem na qual os padrões serão processados é a mesma em que eles são interpretados.

Nas próximas subseções é detalhada a estrutura de cada população. Neste ponto é importante estabelecer que os indivíduos da primeira população são chamados de soluções e os da segunda população de padrões.

Em nossa construção considera-se um único critério de parada, sendo ele a convergência do algoritmo durante 1.000 gerações consecutivas.

3.1. INDIVÍDUOS SOLUÇÕES

Estão listadas abaixo as configurações gerais da população de soluções:

Tamanho da população: 1000 indivíduos, apesar do grande custo computacional em lidar-se com uma população deste tamanho, ela apresenta uma grande diversidade, o que é importantíssimo para evitar convergências prematuras do algoritmo;

Tipo de seleção: elitismo, são selecionados os 10% melhores indivíduos da população; este valor foi ajustado de forma experimental, e representa um bom balanceamento entre preservação de indivíduos e geração de novos;

Taxa de recombinação: 90%;

Tipo de recombinação: uniforme, 70% de chances para o melhor *fitness*;

Taxa de mutação: calcula-se a probabilidade para que dois genes sofram mutação e o substituímos de forma aleatória; caso seja um gene de freqüência, será sorteado um valor entre os limites superior e inferior (descritos abaixo), senão será sorteado um indivíduo da população de padrões e um ponteiro será criado;

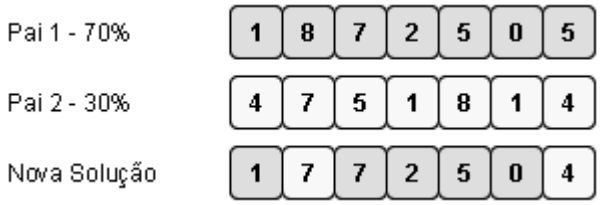

Figura 3: Recombinação Uniforme.

Figura 4: Mutação de um Indivíduo-Solução.

Deve ser estimado o número máximo de *setups* que um problema poderá ter, para com isso determinar o tamanho da cadeia de DNA dos indivíduos. Adota-se *m* * *2*, ou seja, o número de genes do indivíduo será o dobro do número de *setups*.

Para os genes ímpares, a freqüência, deve-se definir o limite superior e inferior para restringir a região de soluções do problema. Para o limite inferior atribui-se o valor zero e para o limite superior define-se o valor da maior demanda para um único item, ou seja, o valor do limite superior será exatamente igual a maior demanda do pedido. Com isso, é possível que um item apareça em apenas um padrão e tenha toda sua demanda atendida.

Como o objetivo é minimizar tanto o número de objetos processados quanto ao número de *setups*, o cálculo do *fitness* (F_s) do indivíduo-solução é feito da seguinte forma:

$$\mathrm{F_s}(i) = c_1 \sum_{j=1}^{n} x_j + c_2 \sum_{j=1}^{n} \delta(x_j) + \sum_{j=1}^{n} \tau(x_j) + \rho$$

onde,

$\tau\ (x_j)$: é a perda relativa do padrão *j*, calculada como, $\tau(x_j) = \dfrac{t_j}{W \sum_{j=1}^{n} x_j}$;

ρ : refere-se a penalidade caso a solução não seja factível.

Vale notar que os valores c_1 e c_2 são tratados explicitamente, não sendo necessário nenhuma outra modificação para atingir diferentes objetivos.

O parâmetro τ tem duas importantes funções: a primeira é determinar se o algoritmo encontrou um ótimo local, já que uma pequena mudança na perda relativa acarretará numa melhora no *fitness*, prolongando sua execução; e a segunda é um fator de diferenciação, visto que quando há duas ou mais soluções com o mesmo *fitness* escolhe-se apenas uma para entrar na elite. Esta estratégia foi adotada com o intuito de aumentar a diversidade da população de soluções.

3.2. INDIVÍDUOS PADRÕES

Estão listadas abaixo as configurações gerais da população de padrões, que são significativamente diferentes em comparação a de soluções:

Tamanho da população: 600 indivíduos; valores pequenos desta população se mostram ineficazes para diversas classes de problemas, já que não conseguem obter uma

grande diversidade, populações muito grandes ampliaram demais o espaço de busca prejudicando o desempenho do método proposto;

Tipo de seleção: elitismo, são selecionados os 66% melhores indivíduos; este valor também foi obtido de forma experimental;

Taxa de recombinação: 34%;

Tipo de recombinação: 2 pontos, são sorteados, aleatoriamente, dois pontos de corte para que seja feita a recombinação dos indivíduos;

Taxa de mutação: 90% de um único gene sofrer mutação, que é feita sorteando-se um item para entrar no lugar do gene que sofreu a mutação;

A Figura 5 mostra como a recombinação é realizada na população de padrões.

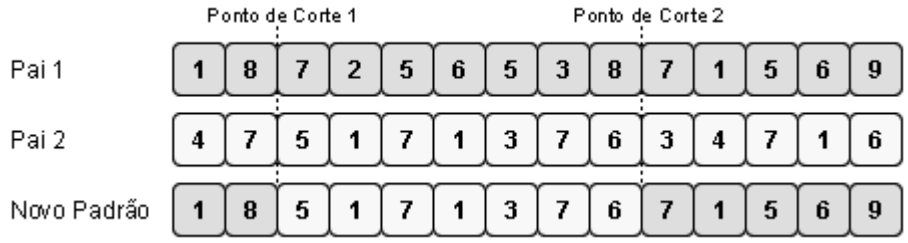

Figura 5: Recombinação Dois-Pontos.

O dimensionamento da cadeia de DNA dos padrões será igual ao maior inteiro menor ou igual ao quociente entre o comprimento da bobina padrão pelo comprimento da menor peça encontrada no pedido. Assim garante-se que é possível formar um padrão utilizando apenas o item de menor comprimento. Isto se faz necessário para que o algoritmo possa cobrir todas as maneiras para resolução do problema.

Porém, com isso, a maior dos padrões não poderá utilizar todos os genes disponíveis, pois o comprimento total das peças contidas no padrão excederá o comprimento da peça mestre. Para solucionar este problema, adicionam-se itens ao padrão da esquerda para direita, tendo como referencial o vetor que dará origem ao padrão, se e somente se o padrão tiver comprimento livre suficiente para acomodar o item.

A Figura 5 apresenta como a mutação é realizada na população de padrões. Um gene é desativado quando seu respectivo item excede o tamanho do objeto e, caso contrário, é ativado. Por exemplo, o quinto gene, que corresponde ao item 5, estava desativado antes da mutação, pois a soma dos itens já presentes no padrão impossibilitava sua entrada. Entretanto, após a mutação do quarto gene, que passou a corresponder ao item 3 ao invés do item 2, a entrada do quinto item foi possível, modificando seu estado para ativado.

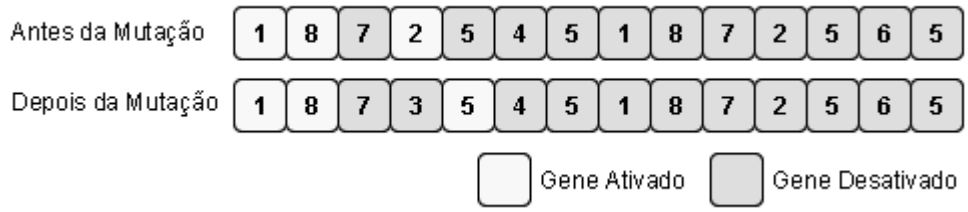

Figura 6: Mutação de um Indivíduo-Padrão.

Para definir o *fitness* de cada padrão (F_p) utiliza-se a elite da população de soluções: para cada indivíduo na elite adicionam-se pontos ao *fitness* desse padrão. Quanto melhor a solução maior será a quantidade de pontos, segundo a seguinte regra:

FAÇA i = 1 **ATÉ** Elite Soluções
  PARA CADA padrão j contido na solução i **FAÇA**
    F_p (j) ← F_p (j) + 1 + (1 / i)
  FIM-PARA
FIM-FAÇA

3.3. BUSCA LOCAL

Com o intuito de aprimorar a convergência do método foi adicionada uma subrotina de busca local. Esta subrotina é especialmente importante quando há um número máximo de pilhas abertas extremamente baixo (por exemplo, dois ou três). A subrotina proposta tem como objetivo reduzir o número de objetos processados, visto que nesses casos as soluções iniciais apresentam uma grande quantidade de itens em excesso.

Dessa forma, seleciona-se, aleatoriamente, dois pares de genes consecutivos e verifica-se o excesso de itens que está sendo produzido. Reduz-se, então, este excesso a zero e resolve-se o problema residual com a heurística FFD (first-fit-decreasing). Em seguida, adiciona-se a solução do problema residual no local antes ocupado pelos genes utilizados para iniciar a busca local.

Em alguns casos se faz necessária a criação de novos indivíduos na população de padrões para que seja possível descrever a solução residual corretamente.

Vale observar que a subrotina proposta não aumenta o número de pilhas abertas, sendo assim não causará nenhuma infactibilidade na solução.

A heurística gulosa FFD consiste em alocar preferencialmente os itens maiores, com a maior freqüência possível em cada padrão. Assim, inicialmente reordena-se os itens de forma que eles estejam em ordem decrescente em relação à largura. Isto porque, e não é difícil se convencer, os itens maiores são os mais complicados de serem alocados e os itens pequenos, em contrapartida, facilmente são "encaixados" quando não é possível alocar os itens grandes.

Pseudo-Código do Método Proposto
1: INICIE os genes da população de soluções aleatoriamente
2: INICIE os genes da população de padrões aleatoriamente
3: SE os itens forem pequenos adicione uma solução homogênea
ENQUANTO (não atingir Geração ou Tempo Limite) **FAÇA**
4: CALCULE a aptidão dos indivíduos-soluções
5: SELECIONE as soluções com melhor aptidão
6: ADICIONE pontos aos padrões utilizados pelas melhores soluções
7: SELECIONE os padrões com melhor aptidão
8: SE a geração atual for múltipla de 50 **FAÇA** uma **Busca Local**
9: UTILIZE os operadores de cruzamento e mutação para gerar novas soluções
10: UTILIZE os operadores de cruzamento e mutação para gerar novos padrões
FIM-ENQUANTO

3.4. PSEUDO-CÓDIGO DA BUSCA LOCAL

1: SELECIONE uma solução da elite
2: SELECIONE um par de genes ativos
3: CALCULE a demanda atendida por este padrão
4: REDUZA o excesso na demanda destes itens

5: RESOLVA este sub-problema com o FFD
6: ADICIONE esta solução ao individuo removendo o padrão antigo

4. TESTES COMPUTACIONAIS

Para aferir a qualidade do método proposto foram realizados testes com as mesmas 9 classes propostas e utilizadas por Belov (2003). Os parâmetros utilizados para gerar as classes foram:

m = {20, 50, 150};
v = {(0, 01; 0, 2), (0, 01; 0, 4), (0, 01; 0, 7), (0, 2; 0, 4), (0, 2; 0, 7)};
s = {[1, 10), [1, 100), [50, 100)}.
onde,
m: é o número de itens distintos;
v: representa os limites, inferior e superior, do tamanho dos itens em relação à bobina mestre;
s: informa o intervalo da demanda que cada item pode assumir.

O algoritmo foi implementado na linguagem FORTRAN 90/95 utilizando o compilador Microsoft FORTRAN Power Station num micro-computador AMD SEMPRON 2300+ 1533 MHz com 640MB de memória RAM. O custos da função objetivo receberam os seguintes valores: c = {1, 10}. Vale observar que o método desenvolvido por Belov (2003) não suporta a alteração de custos, ou seja, os valores de c_1 e c_2 não são definidos de forma explícita na função objetivo, sendo assim o método proposto por Belov é capaz de apresentar apenas uma única solução para o problema.

Na Tabela 2 é apresentado o número de objetos processados conseguido por cada um dos métodos e na Tabela 3 o número de *setups*. Vale notar que se o número de objetos é avaliado o método Symbio apresenta resultados inferiores ao descritos por Belov (2003), entretanto quando avalia-se o número de *setups* a situação é invertida. A primeira coluna das Tabelas 2 e 3 representam o número máximo de pilhas abertas utilizado em cada teste, os testes foram realizados em valores extremos (2 e Infinito) e em valores intermediários (3, 4, 6 e 10).

Tabela 2: Média do Número de Objetos Processados.

c	Classe	1	2	3	4	5	6	7	8	9
2	Symbio	408,6	290,9	578,82	1107,85	999,6	1359,35	119,35	1633	3645,28
	Belov	356,1	262,1	509,5	897,98	763,46	1157,27	91,59	1348,66	2701,61
3	Symbio	395	289,75	563,17	1044,95	970,4	1333,85	113,25	1536	3586,28
	Belov	350,78	262,79	503,8	885,02	747,76	1149,94	88,63	1329,79	2671,06
4	Symbio	397,6	291,1	566,76	1037,75	955,35	1356,25	114,35	1530,05	3556
	Belov	350,03	257,75	502,85	882,12	745,75	1150,21	88,43	1326,59	2665,27
6	Symbio	395,55	292,7	589,94	1036,6	952,5	1346,95	114,6	1541,2	3555,28
	Belov	349,61	257,75	502,7	880,41	744,44	1150,05	88,23	1324,26	2661,87
10	Symbio	395,2	355,55	633,81	1047,8	954,05	1350,65	113,05	1536,2	3528,71
	Belov	349,3	257,7	502,56	878,53	743,88	1150,44	88,09	1323,13	2660,03
Inf	Symbio	399,7	343,55	623,63	1048,25	941,05	1348,7	114,8	1533,3	3507,71
	Belov	349,3	257,65	502,41	878,48	743,59	1150,47	88,09	1322,2	2658,14

Tabela 3: Média do Número de *Setup*s.

c	Classe	1	2	3	4	5	6	7	8	9
2	Symbio	12,85	42,3	39,06	33,75	38,75	39,75	30,95	34,1	136,85
	Belov	23,65	70,05	70,1	61,25	65,95	54,05	45,6	63	184,25
3	Symbio	11	41,6	36,64	30,65	31,8	38,2	25,9	29,45	131,71
	Belov	21,7	67,35	61,35	56,9	59,75	53	41,4	57,2	170,9
4	Symbio	10,8	41,7	37,41	29,7	31,55	37,4	24,75	28,55	129,85
	Belov	20,75	64,2	59,9	54,25	56,6	52,4	39,55	55,65	163,7
6	Symbio	10,45	40,3	32,69	30,15	32,25	38,75	24,8	28,55	126,42
	Belov	20,25	64,85	57,85	51,45	54,8	51,75	38,35	53,15	155,5
10	Symbio	10,65	12,55	21,31	29,7	32,25	38,05	25,15	28,95	125,14
	Belov	20,8	65,45	57,1	50,85	51,9	51,05	37,65	50,4	152,05
Inf	Symbio	10,35	12,55	22,13	30,1	31,9	38,15	24,8	28,55	123
	Belov	20,45	67,45	60,65	51,1	53,5	51,85	39,7	52,9	150,95

Na Tabela 4 pode-se verificar a diferença percentual no tempo de processamento do método Symbio em relação ao método de Belov (2003) para cada uma das classes testadas. O tempo computacional obtido por Belov (2003) é de 600 segundos para cada problema, já o método Symbio apresenta uma média de aproximadamente 1226 segundos. Os percentuais negativos (em laranja) representam que o método Symbio foi mais rápido que o método de Belov (2003). Por exemplo, na classe 2, com no máximo duas pilhas abertas, com $C_1 = 1$ e $C_2 = 5$ o método Symbio teve um tempo computacional 17,96% menor, em média, que o de Belov (2003) – segunda linha, segunda coluna da Tabela 4.

Tabela 4: Diferença Relativa para $C_1 = 1$ e $C_2 = 5$.

Pilhas	1	2	3	4	5	6	7	8	9
2	-0,32	- 17,96	-9,99	6,01	9,16	9,15	-14,23	8,41	19,51
3	-2,02	-16,98	-7,92	2,45	7,92	7,77	-17,89	4,18	20,4
4	-0,48	-13,68	-6,05	2,85	8,2	9,28	-16,8	4,23	20,71
6	-0,68	-15,09	-4,87	4,37	9,36	9,36	-14,78	5,91	21,75
10	-1,07	-28,49	-6,05	5,61	11,15	9,62	-13,58	6,72	21,46
Inf	-0,02	-31,7	-8,86	5,71	8,85	9,2	-16,68	5,63	20,8

Tabela 5: Diferença Relativa para *C1 = 1* e *C2 = 10*.

Pilhas	1	2	3	4	5	6	7	8	9
2	-9,37	-25,84	-19,92	-4,31	-2,52	3,48	-21,68	-0,24	10,34
3	-11,06	-24,62	-16,8	-7,05	-4,23	2,14	-25,94	-3,75	11,95
4	-9,31	-21,3	-14,61	-6,31	-3,12	3,35	-25,23	-3,59	12,84
6	-9,43	-23,23	-15,2	-4,07	-1,35	4,01	-23,13	-1,57	14,29
10	-9,98	-47,26	-21,11	-3,04	1,08	4,23	-21,53	-0,08	14,34
Inf	-9,14	-49,68	-23,81	-2,9	-1,45	3,67	-25,21	-1,75	13,68

5. CONCLUSÕES E PERSPECTIVAS

O presente trabalho apresenta um Algoritmo Genético Simbiótico com o objetivo de minimizar o número de objetos processados e o *setup* dado um número máximo de pilhas abertas. Na literatura foi encontrado apenas uma referência a este problema Belov (2003), embora possam ser encontradas diversas referências que trabalham com *setups* e número de pilhas em separado.

Avaliando os resultados obtidos em relação aos descritos na literatura é possível inferir que os resultados alcançados são complementares aos já obtidos por outros autores.

Agradecimentos: Os autores foram parcialmente suportados pelo MEC-Espanha através do Projeto MTM2007-063432. O segundo autor também é suportado pelo CNPq-Brasil através do Projeto 307907/2007-4.

REFERÊNCIAS BIBLIOGRÁFICAS

Allaby, M. (1988) Dictionary of Ecology, Oxford University Press, New York.

Armbruster, M. (2002) A solution procedure for a pattern sequencing problem as part of a onedimensional cutting stock problem in the steel industry, European Journal of Operational Research, vol. 141, Issue 2, pp: 328-340.

Belov, G. and Scheithauer (2003) G. Preprint MATH-NM-15-2003, TU Dresden.

Bremermann, H. J. (1962) Optimization through evolution and recombination", Spartan Books, pp: 93-106.

Chang, M., Ohkura, K., Ueda, K. and Sugiyama, M. (2002) A symbiotic evolutionary for dynamic facility layout problem. In Proceedings of the Evolutionary Computation, 1745-1750.

Dowsland, K. and Dowsland, W. (1992). Packing Problems, European Journal of Operational Research, vol. 56, pp: 2-14.

Eguchi, T., Hirasawa, K. e Hu, J. (2003) "Symbiotic Evolutional Models in Multiagent Systems", The 2003 Congress on Evolutionary Computation, vol. 2, pp: 739-746.

Eilon, S. (1960) Optimizing the shearing of steel bars", Journal of Mechanical Engineering Science, vol. 2, pp: 129-142.

Fink, A. and Voß, S. (1999) Applications of modern heuristic search methods to pattern sequencing problems", Computers and Operations Research, vol. 26, Issue 1, pp: 17-34.

Fraser, A. S. (1957) Simulation of genetic systems by automatic digital computers: I. Introduction", Austral. J. Biol. Sci., vol. 10, pp: 484-491.

Gilmore, P.C. e Gomory, R.E. (1961) A Linear Programming Approach to the Cutting Stock Problem, Operations Research, vol. 9, pp: 849-859.

Gilmore, P.C. e Gomory, R.E. (1963). A Linear Programming Approach to the Cutting Stock Problem", Operations Research, vol. 11, pp: 864-888.

Golfeto, R.R., Moretti, A.C. and Salles Neto, L.L.. (2007a) Algoritmo Genético Simbiótico Aplicado ao Problema de Corte de Estoque Unidimensional, XXXIX Simpósio Brasileiro de Pesquisa Operacional.

Golfeto, R.R., Moretti, A.C. and Salles Neto, L.L. (2007b) Algoritmo Genético Simbiótico Aplicado ao Problema de Corte de Estoque Unidimensional Multi-Objetivo, XXXIX Simpósio Brasileiro de Pesquisa Operacional.

Golfeto, R.R., Moretti, A.C. and Salles Neto, L.L. (2007c) Algoritmo Genético Simbiótico Co-Evolucionário Aplicado ao Problema de Corte Multi-Objetivo, XXXIX Simpósio Brasileiro de Pesquisa Operacional.

Golfeto, R.R., Moretti, A.C. and Salles Neto, L.L. (2009a) A Genetic Symbiotic Algorithm Applied To One-Dimensional Cutting Stock Problem, Revista Pesquisa Operacional, 365-382.

Golfeto, R.R., Moretti, A.C. and Salles Neto, L.L. (2009b) A Genetic Symbiotic Algorithm Applied To One-Dimensional Cutting Stock Problem with Multiple Objectives, International Journal of Advanced Modeling and Optimization, 4753-501.

Golfeto, R.R., Moretti, A.C. and Salles Neto, L.L., (2009c) Post-Optimization Of The Number of Setups In The Cutting Stock Problem, Anais do XLI Simpósio Brasileiro de Pesquisa Operacional.

Haessler, R. (1975) Controlling Cutting Pattern Changes in One-Dimnesional Trim Problems, Operations Research, vol. 23, pp. 483-493.

Hirasawa, K., Ishikawa, I., Hu, J., Jin, C. e Murata J. (2000) Genetic Symbiosis Algorithm, Proceedings of the Congress on Evolutionary Computation, vol. 2, pp: 02-xxvi.

Holland, J.H. (1962) Outline for a logical theory of adaptive systems, J. Assoc. Comput. Mach., vol. 3, pp. 297-314.

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

Kantorovich, L. V. (1960). Mathematical Methods of Organizing and Planning Production', Management Science, vol. 6, pp 366-422.

Khalifa, Y., Salem, O. e Shahin, A. (2006). Cutting Stock Waste Reduction Using Genetic Algorithms. Proceedings of the 8th Conference on Genetic and evolutionary computation, pp. 1675-1680.

Kim, Y. K., Kim, J. Y. and Kim, Y. (2000) A coevolutionary algorithm for balancing and sequecing in mixed model assembly lines. Applied Intelligence, 13, pp: 247-258.

Kim, J. Y., Kim Y. and Kim Y. K. (2001) An endosymbiotic evolutionary algorithm for optimization. Applied Intelligence, 15, pp: 117-130.

Kim, Y. K., Kim, J. Y. and Kim, Y. (2006). An endosymbiotic evolutionary algorithm for the integration of balancing and sequecing in mixed-model U-lines. European Jounal of Operational Research, 168, 838-852.

Mao, J., Hirasawa, K., Hu, J. e Murata, J., (2000) Genetic Symbiosis Algorithm for Multiobjective Optimization Problem, Proceedings of the IEEE Interenational Workshop on Robot and Human Interactive Communication, pp: 137-142.

Metzger, R. W. (1958). Stock Slitting, Elementary Mathematical Programming, Wiley.

Oliveira, A. C. M. and Lorena, L. A. N. (2002). 2-Opt Population Training for Minimization of Open Stack Problem", Lecture Notes in Computer Science, vol. 2507, pp: 497-504.

Paull, A. E. e Walter, J. R., (1954). The trim problem: an application of linear programming to the manufacture of news-print paper, Presented at Annual Meeting of Econometric Society, Montreal, pp. 10-13.

Pianka, E. R. (1994) Evolutionary Ecology, HarperCollins College Publisher, New York.

Pileggi, G.C.F., Morabito, R. and Arenales M.N. (2005.). Abordagens para otimização integrada dos problemas de geração e seqüenciamento de padrões de corte: caso unidimensional, Pesquisa Operacional, vol. 25, no.3, pp: 417-447.

Potter, M. A. (1997.) The design and analysis of a computational model of cooperative coevolution. Ph.D. Dissertation, George Mason University.

Tsujimura, Y., Mafune, Y. and Mitsuo, G. (2001) Efects of symbiotic evolution in genetic algorithms for job-shop scheduling, IEEE,.

Wäscher, G., Haussner H. and Schumann. (2007). An improved typology of cutting and packing problem. European Journal of Operational Research, 183, pp: 1109-1130.

Von Zuben, F. J., "Computação Evolutiva: uma abordagem pragmática", ftp://ftp.dca.fee.unicamp.br/pub/docs/vonzuben/tutorial/tutorialEC.pdf, UNICAMP, 2003.

Yanasse, H.Y. (1997). On a pattern sequencing problem to minimize the maximum number of open stacks, European Journal of Operational Research, vol. 100, Issue 3, pp: 454-463.

Yanasse, H.Y. and Lamosa, M.J.P. (2007) An integrated cutting stock and sequencing problem, European Journal of Operational Research, vol. 183, Issue 3, pp: 1353-1370.

Yuen, B.J. (1991) Heuristics for sequencing cutting patterns, European Journal of Operational Research, vol. 55, pp: 183-190.

Yuen, B.J. (1995). Improved heuristics for sequencing cutting patterns, European Journal of Operational Research, vol. 87, pp: 57-64.