SPOLM 2009

ISSN 2175-6295 Rio de Janeiro- Brasil, 05 e 06 de agosto de 2009

062/2009 - RESOLUÇÃO DE UM PROBLEMA DE CORTE UNIDIMENSIONAL PARA MINIMIZAÇÃO DO NÚMERO DE OBJETOS PROCESSADOS E O *SETUP* VIA ALGORITMO GENÉTICO

Julliany Sales Brandão
Instituto Politécnico do Rio de Janeiro – UERJ
Nova Friburgo, Brasil
jbrandao@iprj.uerj.br

Alessandra Martins Coelho
Instituto Federal de Educação, Ciência e Tecnologia
do Sudeste de Minas – Campus Rio Pomba
ale_contatos@ig.com.br

João Flávio V. Vasconcellos
Instituto Politécnico do Rio de Janeiro – UERJ
Nova Friburgo, Brasil
jflavio@iprj.uerj.br

Wagner Figueiredo Sacco
Instituto Politécnico do Rio de Janeiro – UERJ
Nova Friburgo, Brasil
wfsacco@iprj.uerj.br

Resumo

Este artigo faz um estudo sobre a aplicação da metaheurística Algoritmo Genético na resolução do Problema de Corte unidimensional para minimizar o número de objetos processados e o *setup*. O modelo do problema proposto, a função objetivo e os métodos utilizados para sua resolução, são apresentados. O método proposto, denominado SingleGA10, é utilizado para minimizar o número de objetos processados e *setup* simultaneamente. Os resultados obtidos pelo SingleGA10 são comparados com os métodos SHP, Kombi234, ANLCP300 e Symbio10, verificando a capacidade de encontrar soluções viáveis e competitivas. Os resultados computacionais mostram que esta nova abordagem, a qual utiliza apenas um algoritmo genético para resolver esses dois objetivos conflitantes, apresenta bons resultados.

Palavras-Chave: Problema de Corte unidimensional; Algoritmos Genéticos; *Setup*; Algoritmos Evolutivos

Abstract

This paper is a study about the use of genetic algorithms metaheuristics in solving one-dimensional cutting problems, to minimize setup and the number of processed objects. The proposed method, the objective function, and the methods used are presented. The

proposed method, called SinigleGA10, is used to minimize the numbers of processed objects, and setup, simultaneously. The results were compared with the following methods: SHP, Kombi234, ANLCP300, and Symbio10, verifying its capacity to find feasible and competitive solutions. The computational results reveal that this new approach, which uses only one genetic algorithm to solve both conflicting problems, presents good results.
Keywords: One-dimensional Cutting Problem; Genetic Algorithms; Setup; Evolutive Algorithms

1. PROBLEMA DE CORTE

Com os recentes avanços computacionais, tornou-se importante e de grande relevância estudar os problemas de corte e empacotamento bem como modelos de otimização para o controle e planejamento de sistemas produtivos os quais vêm sendo estimulados por diversas indústrias, como a de vidro, papel, têxtil, química, entre outras, que visam tornar seus processos mais eficientes devido a competitividade entre as empresas, redução dos desperdícios, custos e a eficiência na entrega.

O problema de corte consiste em decidir sobre a maneira de se cortar um conjunto de peças retangulares com demandas pré-estabelecidas, a partir de uma placa retangular padrão com a finalidade de minimizar custo, tempo, perda de material envolvidos no processo de fabricação ou uma combinação desses [1]. Os itens são dispostos no objeto para a realização de cortes ao longo da produção. A essa disposição dos itens dá-se o nome de padrão de corte. Observa-se, nesse caso, a importância da geometria, uma vez que as formas e dimensões dos itens e objetos determinarão os possíveis padrões de corte.

Dentre os trabalhos pioneiros publicados sobre o tema destaca-se Kantorovich, cujo primeiro trabalho foi produzido em 1939, e publicado em 1960. Sua formulação procurava minimizar as perdas (*trim loss)* em um problema de corte unidimensional [2]. Problemas semelhantes foram tratados em [3], [4] e [5]. No entanto, todos os métodos propostos adequavam-se apenas a problemas pequenos [6]. Foi com os trabalhos de [7] e [8], respectivamente, no estudo do problema de corte, através do processo de geração de colunas, que houve um considerável avanço nos estudos dos problemas de corte.

Para solucionar um problema de corte necessita-se dos padrões de corte e da freqüência de padrões, ou seja, da quantidade de vezes que esses padrões serão executados.

Embora o objetivo geral seja a minimização de perdas, encontra-se várias modelagens do problema, quais sejam, a maximização de lucros, a redução de objetos utilizados, do tempo de produção e/ou uma combinação desses.

O custo de preparação da máquina, ou seja, do seu tempo de preparação (*setup*), também é um fator relevante no processo de corte. Assim, torna-se interessante avaliar os efeitos da minimização do número de objetos processados (minimização da entrada) e do número de padrões de corte (*setups*), objetivos esses parcialmente conflitantes, para uma avaliação mais geral do custo. Haessler [9] foi o primeiro a tratar o problema de corte unidimensional não-linear desse modo.

Para a maioria das formulações do problema de corte não se é conhecido até o momento, um método que produza uma solução em tempo polinomial. Nesse caso justifica-se a utilização de heurísticas ou metaheurísticas, gerando, para esses, boas soluções num curto intervalo de tempo.

Assim sendo, implementou-se, nesse trabalho, um algoritmo genético com o objetivo de minimizar o número de objetos processados e o *setup*, baseado em uma abordagem recente apresentada por [10]. A grande diferença entre esses dois métodos é quantidade de algoritmos genéticos utilizados. O Symbio10 utiliza dois algoritmos genéticos numa relação de benefício mútuo para resolver o problema de minimização proposto, enquanto o SingleGA10 utiliza apenas um algoritmo genético.

Este artigo está organizado como segue: a seção 2 apresenta formalmente sobre o problema de corte para reduzir o número de objetos processados e o *setup*. Os conceitos básicos de algoritmos genéticos são expostos na seção 3. A seção 4 versa sobre o método proposto. A seção 5 apresenta os testes e resultados computacionais. Por fim, a seção 6 expõe as conclusões desse trabalho.

2. PROBLEMA DE CORTE UNIDIMENSIONAL

Segundo a tipologia de [11], o problema de corte unidimensional está classificado como (1/V/I/R) e na tipologia de [12] é identificado como SSSCSP (*Single Stock Size Cutting Stock Problem*). O SSSCSP inclui o clássico problema de corte unidimensional [8] e [13] cujos os objetos grandes são idênticos, ou seja, de padrões de corte com mesmo tamanho geram itens de tamanhos distintos e menores considerados fracamente heterogêneos.

Um modelo matemático formal para resolver o problema de minimização do número de objetos processados e *setup* é descrito em (1).

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

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

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

onde:

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

n = número de possíveis padrões de corte;
m = número de itens diferentes;
c_1 = custo de cada bobina;
α = custo de troca do padrão de corte;
x_j = número de bobinas processadas com o padrão de corte *j;*
a_{ij} = número de itens do tipo i no padrão *j;*
d_i = número de itens *i* demandados;

3. ALGORITMOS GENÉTICOS

Algoritmos genéticos são algoritmos de busca e otimização de soluções baseado nos mecanismos genéticos e evolucionários dos seres vivos, como a seleção natural e a sobrevivência do mais apto, introduzidos por Charles Darwin no clássico "*The Origin of Species*" (1859). Eles trabalham com uma população de indivíduos, onde cada um representa uma solução possível para um dado problema. A cada indivíduo é dado um *fitness*, isto é, um valor que quantifica adaptabilidade do indivíduo ao meio (problema tratado). Os indivíduos com maiores *fitness* possuem maiores oportunidades de serem escolhidos para a reprodução através do cruzamento e com isso difundir suas características pelas gerações seguintes propiciando que áreas mais promissoras de busca sejam exploradas levando o algoritmo genético, na maioria dos casos, à convergência pela solução ótima do problema.

Para melhor entendimento da implementação alguns conceitos precisam ser apresentados. Um gene é a representação de cada parâmetro de acordo com o alfabeto utilizado (binário, inteiro ou real). Enquanto genoma é o conjunto de genes para formar um

indivíduo. População é o conjunto de indivíduos no espaço de busca e os operadores genéticos são seleção, recombinação e mutação. A seleção é a escolha dos indivíduos privilegiando os mais aptos para permanecer e multiplicar a população. Os métodos de seleção mais utilizados são: elitismo, diversidade e torneio. A recombinação é responsável por recombinar as características dos pais durante a reprodução permitindo que os seus descendentes herdem essas características. Os mais conhecidos são: ponto(s) de corte e uniforme. E por fim, a mutação é a possibilidade de um gene qualquer do indivíduo apresentar características diferentes das existentes nos pais após uma operação de recombinação.

4. MÉTODO PROPOSTO – SINGLEGA10

O SingleGA10, foi baseado no Symbio10 de [10]. No entanto, o Symbio10 utiliza-se de conceitos de simbiose [14] mais especificamente numa relação mutualística entre dois algoritmos genéticos [15] que se evoluem de maneira benéfica, enquanto o método do presente estudo utiliza-se de apenas um algoritmo genético para resolver o mesmo objetivo.

No SingleGA10 existem duas espécies de população: a de soluções e a de padrões. A população de padrões, gerada aleatoriamente, é estática, ou seja, não sofre qualquer espécie de evolução.

O gene da população de solução é representado por dois elementos, onde o primeiro elemento refere-se à quantidade de vezes (frequência) com que o padrão indicado pelo segundo elemento será processado. O segundo elemento da população de solução serve apenas como um ponteiro para a população de padrões.

A soma das frequências equivale ao número de objetos processados e a quantidade de padrões distintos, o número de *setup*.

O número máximo de genes do indivíduo solução, ou seja, o tamanho máximo do genoma é igual ao número de *setups* que é a quantidade de itens diferentes. No entanto, se o maior item a ser cortado apresentar um comprimento inferior ou igual a 50% do comprimento da bobina adota-se o seguinte procedimento experimental adaptado do utilizado por [10]:

```
se (Quantidade de itens > 30) entao
        GenesSolucoes = 16
senao
    se (Quantidade de itens > 15) entao
        GenesSolucoes = 12
    senao
        GenesSolucoes = 8
    fim-se;
fim-se.
```

O tamanho do genoma do padrão é igual ao maior inteiro menor ou igual ao resultado da divisão entre o tamanho da bobina padrão pelo item de menor tamanho do padrão. Com isso, segundo [10], garante-se que é possível a formação de um padrão usando apenas o menor item, fazendo com que o algoritmo cubra todas as maneiras de resolver o problema. No entanto, isto traz implicações para o maior padrão, que não poderá utilizar todos os genes disponíveis, pois, o total de suas peças contidas no padrão ultrapassará o comprimento da bobina. Na tentativa de solucionar esse problema, os itens são adicionados ao padrão da esquerda para a direita, se somente se, o padrão possuir espaço livre o suficiente para acomodar o item. Abordagem inspirada nos trabalhos de [10].

Os indivíduos formados por genes da população de soluções serão chamados de População de soluções e/ou indivíduos soluções, e os indivíduos formados por genes da população de padrões serão chamados de população de padrões e/ou indivíduos padrões.

A população de soluções (indivíduos formados por genes da população de soluções) foi formada por 600 indivíduos e a população de padrões (indivíduos formados por genes da

população de padrões) por 400, valores esses definidos experimentalmente após a realização de simulações com diversos valores. Todos os indivíduos (soluções e padrões de corte) foram gerados aleatoriamente sem nenhum direcionamento.

A estrutura de seleção adotada foi o elitismo com *steady state*. O *steady state* utilizado foi o disponível no pacote do GALib (Wall, 1996) da classe de algoritmos genéticos GASteadyStateGA. Segundo o tutorial do GALib [16], esse método gera, a cada geração, uma população temporária de indivíduos por clonagem e os inserem na população da geração atual, removendo os piores indivíduos. A taxa de substituição adotada foi de 25%, ou seja, os 75% melhores indivíduos da população de soluções foram selecionados para permanecer na população. Esse valor foi ajustado de modo experimental. A escolha do elitismo se deve ao fato desse recurso aumentar o desempenho do algoritmo genético, uma vez que ele garante que a melhor solução encontrada até o momento será mantida nas próximas gerações.

O cálculo do *fitness* (*Fs*) do indivíduo solução para atender o objetivo de minimizar ao mesmo tempo, o número de objetos processados e o *setup*, baseado e descrito em [10] é realizado do seguinte modo:

$$F_s = c_1 \sum_{j=1}^{n} x_j + \alpha \sum_{j=1}^{n} \delta(x_j) + \sum_{j=1}^{n} \tau(x_j) + \rho \qquad (2)$$

onde:

τ é a perda relativa e pode ser calculada por $\tau(x_j) = \dfrac{t_j}{w \sum_{j=1}^{n} x_j}$ sendo t_j o desperdício do padrão *j*.

ρ são penalidades caso a solução seja inviável. É proporcional a soma das infactibilidades, ou seja, o valor que falta para atingir a demanda do item multiplicado por 1000. Esse valor 1000 foi escolhido de modo experimental.

Os valores c_1 e α são os custos de objetos processados e *setup*, respectivamente, tratados de maneira explícita na função objetivo, não necessitando quaisquer outras modificações para alcançar objetivos distintos, ou seja, o custo já participa diretamente do processo de minimização da função objetivo.

Durante o cálculo da função objetivo faz-se necessário uma interpretação dos padrões, ou seja, verificar quais os genes ativos do padrão. Os genes ativos serão aqueles que cabem na bobina-mestre sem ultrapassar o seu tamanho.

O operador de recombinação (*crossover*) utilizado foi o uniforme, o qual consiste em sortear aleatoriamente um valor entre 0 e 1. Se o número sorteado for menor ou igual a 0,7 a recombinação é realizada com o indivíduo que possuir o maior *fitness*, caso contrário, realizará com o indivíduo de menor *fitness*. A taxa de recombinação adotada foi de 30%.

O operador de mutação adotado consiste em sortear aleatoriamente a posição no genoma do gene a ser mutado e em seguida determinar aleatoriamente qual dos seus elementos (freqüência ou indice_padrao) sofrerá mutação.

Caso o elemento escolhido do gene seja a freqüência, será sorteado um valor entre os limites mínimo e máximo. Caso contrário, será sorteado um indivíduo da população de padrão e um ponteiro será criado [10].

Os limites máximo e mínimo para a frequência são necessários para restringir a região de soluções do problema. Ao limite inferior é atribuído o valor zero e para o limite máximo o valor da maior demanda para um único item, ou seja, a maior demanda do pedido. Isso possibilita que um item esteja contido em apenas um padrão e tenha toda a sua demanda atendida.

Para o índice do padrão, elemento que compõe o gene da população de solução, os limites ficam entre zero e o quantidade de indivíduos padrões -1. A taxa de mutação é o quociente entre 1 e número diferentes de itens (1/*m).*

O procedimento adotado pode ser representado pelos passos a seguir:

Procedimento SingleGA10

1 Gere os indivíduos da população de padrões aleatoriamente;
2 Gere os indivíduos da população de soluções aleatoriamente;
3 Calcule a função objetivo dos indivíduos soluções;
4 Selecione os indivíduos soluções genitores;
5 Utilize operadores de recombinação e mutação para gerar novas soluções;
6 Evolua a população;
7 Se algum critério de parada for satisfeito PARE a execução do algoritmo senão retorne ao 3.

Fim-Procedimento SingleGA10.

Os critérios de parada adotados pelo método proposto foram:

- número máximo de gerações: 1000;
- tempo máximo de execução: 500s;
- convergência: 500 gerações, ou seja, se o algoritmo não melhorar a solução por 500 gerações ele pára.

5. TESTES E RESULTADOS COMPUTACIONAIS

O SingleGA10 foi implementado na linguagem *C++* no compilador Bloodshed DevC++, num microcomputador Pentium Dual Core 1.73 533MHz, 2G de memória RAM.

Para a realização dos testes computacionais foram utilizadas instâncias geradas aleatoriamente pelo gerador de problemas de corte unidimensional, CUTGEN1, desenvolvido por [17]. Gerou-se 18 classes caracterizadas pelos diferentes valores dos parâmetros de entrada do CUTGEN1 para aferir a qualidade do método proposto. Os parâmetros e a semente adotados foram os mesmos em todos os métodos.

Os resultados obtidos com o SingleGA10 foram comparados com aqueles encontrados pela heurística Kombi234 proposta por [18], a heurística SHP proposta por [9] e implementada por [19], a heurística ANLCP300 desenvolvida por [19] e a metaheurística Symbio10 proposta por [10], para efeito de avaliação da qualidade do método proposto para a resolução da formulação (1). Exceto o SHP, todos os demais métodos comparados foram implementados por seus respectivos autores. Esses métodos são descritos a seguir:

I. SHP: programado na linguagem FORTRAN, utilizando o compilador g77 para Linux, num microcomputador AMD AthlonXP 1800 MHz com 512 de memória RAM. Basea-se em uma técnica exaustiva, na qual, a cada iteração, calculam-se alguns parâmetros de aspiração, fazendo uma busca por padrões de corte que satisfaçam tais parâmetros até que se atenda toda a demanda [9];

II. KOMBI234: código em MODULA-2 em MS-DOS 6.0 utilizando um IBM 486/66. Baseia-se na combinação de padrões visando diminuir o *setup* de uma solução inicial. Ele tem por base o fato das somas das freqüências dos padrões resultantes serem as mesmas das freqüências dos padrões originais, mantendo assim, o número de objetos processados constante, com uma possível redução de *setup*, ou seja, reduz-se o número de padrões para realização da mesma demanda original [18];

III. ANLCP300: programado na linguagem FORTRAN, utilizando o compilador g77 para Linux, num microcomputador AMD AthlonXP 1800 MHz com 512 de memória RAM. Esse método busca minimizar o número de objetos processados e o *setup* através da aplicação, do método lagrangiano aumentado num problema com função objetivo, proposta por [9], suavizada. Para esse modelo não-linear, o ANLCP300 adapta o método de geração de colunas

de [7] e [8] e para a obtenção de uma solução viável utiliza uma simples heurística de arredondamento [19];

IV. Symbio10**:** implementado na linguagem FORTRAN, utilizando compilador Microsoft FORTRAN Power Station num micro-computador AMD SEMPRON 2300+ 1533MHz com 640 MB de memória RAM. É um algoritmo genético que, baseado num processo simbiótico entre duas populações de espécies diferentes, gera seus próprios padrões em conjunto com soluções para o problema [10];

No SingleGA10, o custo de objetos processados foi $c_1 = 1$ e *setup* $\alpha = 10$. A Tabela 1 apresenta a média dos números de objetos processados e do *setup* para os 100 problemas de cada uma das classes.

Tabela 1: Média dos números de objetos processados e do *setup* para $c_1 = 1$ e *setup* $\alpha = 10$

	SHP		**KOMBI234**		**ANLCP300**		**Symbio10**		**SingleGA10**	
Classe	**objeto**	***setup***	**objeto**	***setup***	**objeto**	***setup***	**objeto**	***setup***	**objeto**	***setup***
1	14,17	3,95	11,49	3,4	14,3	3,02	14,84	1,85	**15,56**	**1,97**
2	116,47	5,94	110,25	7,81	121,48	4,5	117,8	4,68	**127,36**	**3,78**
3	25,29	5	22,13	4,84	25,14	4,84	28,23	4,47	**32,5**	**4,18**
4	255,33	7,31	215,93	7,28	224,86	7,28	239,72	9,36	**269,73**	**7,1**
5	46,89	6,87	42,96	10,75	45,66	7,02	61,42	8,23	**71,14**	**7,79**
6	433,59	10,81	424,71	25,44	432,72	10,92	518,85	14,08	**623,58**	**12,36**
7	55,84	8,84	50,21	7,9	51,54	5,54	53,75	5,21	**54,23**	**5,35**
8	515,76	9,76	499,52	9,96	495,94	8	514,65	7,76	**518,21**	**6,07**
9	108,54	17,19	93,67	15,03	114,52	10,58	101,62	10,01	**104,63**	**9,39**
10	1001,59	19,37	932,32	19,28	969,2	13,96	975,72	16,04	**1032**	**13,34**
11	202,8	32,2	176,97	28,74	231,84	20	201,44	21,56	**249,01**	**24,02**
12	1873,05	37,25	1766,2	37,31	1861,2	23,68	1917,4	31,52	**2462,6**	**34,58**
13	69,97	9,38	63,27	8,97	70,54	5,64	67,32	6,68	**67,63**	**6,56**
14	643,55	9,85	632,12	10,32	634,02	7,92	650,52	8,21	**652,43**	**7,22**
15	136,05	18,03	119,43	16,88	127,38	9,92	128,87	12,99	**130,16**	**12,9**
16	1253,55	19,63	1191,8	19,91	1194,74	13,88	1251,9	16,34	**1301,9**	**15,33**
17	256,01	34,39	224,68	31,46	297,58	22,6	247,62	25,62	**270**	**25,86**
18	2381,54	38,23	2342,4	38,28	2430,04	27,44	2422,41	32,18	**2751,5**	**33,76**

O custo total é o custo da função objetivo e seu cálculo é realizado pela expressão representada a seguir:

$$custo_{total} = c_1 \times obj_{media} + \alpha \times setup_{media} \qquad (3)$$

sendo:

obj_{media} a média do número de objetos processados entre os 100 problemas de cada classe;

$setup_{media}$ a média do número de *setup* entre os 100 problemas de cada classe.

Tabela 2: Médias do custo total para $c_1 = 1$ e *setup* $\alpha = 10$

Classe	**SHP**	**KOMBI234**	**ANLCP300**	**Symbio10**	**SingleGA010**
1	53,67	45,49	44,5	33,34	**35,26**
2	175,87	188,35	166,48	164,6	**165,16**
3	75,29	70,53	73,54	72,93	**74,3**
4	328,43	288,73	297,66	333,32	**340,73**
5	115,59	150,46	115,86	143,72	**149,04**
6	541,69	679,11	541,92	659,65	**747,18**
7	144,24	129,21	106,94	105,85	**107,73**
8	613,36	599,12	575,94	592,25	**578,91**
9	280,44	243,97	220,32	201,72	**198,53**

10	1195,29	1125,12	1108,8	1136,12	**1165,38**
11	524,8	464,37	431,84	417,04	**489,21**
12	2245,55	2139,3	2098	2232,6	**2808,36**
13	163,77	152,97	126,94	134,12	**133,23**
14	742,05	735,32	713,22	732,62	**724,63**
15	316,35	288,23	226,58	258,77	**259,16**
16	1449,85	1390,9	1333,54	1415,3	**1455,24**
17	599,91	539,28	523,58	503,82	**528,6**
18	2763,84	2725,2	2704,44	2744,21	**3089,09**

O custo total médio foi calculado para aferir a qualidade e o desempenho do método proposto tratando, ao mesmo tempo, os dois objetivos para efeito de comparação do valor da função objetivo com os demais métodos. A Tabela 2 traz a média do custo total para $c_1 = 1$ e *setup* $\alpha = 10$. E, para melhor compreender esses valores, a Tabela 3 apresenta a variação percentual do custo total do método proposto, em relação aos demais métodos em comparação nesse trabalho, o que torna mais fácil a análise de desempenho do método. A expressão utilizada para o cálculo da variação do custo total está representada pela equação a seguir:

$$100 \times \left(\frac{custoTotal_{SingleGA} - custoTotal_{metcomparado}}{custoTotal_{SingleGA}}\right) \qquad (4)$$

Tabela 3: Variação percentual do custo total do SingleGA10, em relação aos demais métodos para $c_1 = 1$ e *setup* $\alpha = 10$

Classe	SHP	KOMBI234	ANLCP300	Symbio10
1	-52,2121	-29,01304594	-26,20533182	5,445263755
2	-6,48462	-14,04093001	-0,799224994	0,339065149
3	-1,33244	5,074024226	1,022880215	1,843876178
4	3,609896	15,26135063	12,64050715	2,174742465
5	22,44364	-0,952764359	22,26247987	3,569511541
6	27,50207	9,110254557	27,47129206	11,71471399
7	-33,8903	-19,93873573	0,733314768	1,745103499
8	-5,95084	-3,491043513	0,513033114	-2,304330552
9	-41,2582	-22,88822848	-10,97567118	-1,606810054
10	-2,56654	3,454667147	4,855068733	2,51076902
11	-7,27499	5,077574048	11,72707017	14,75235584
12	20,04052	23,82386873	25,29447792	20,50164509
13	-22,9228	-14,81648277	4,721158898	-0,668017714
14	-2,40399	-1,475235637	1,574596691	-1,102631688
15	-22,0674	-11,2170088	12,57138447	0,150486186
16	0,370386	4,421263847	8,362881724	2,74456447
17	-13,4904	-2,020431328	0,949678396	4,687854711
18	10,52899	11,77984455	12,45188713	11,16445296

Os resultados obtidos pelo SingleGA10 foram melhores que o SHP em 12 classes e em mais uma, na classe 16, apresentou um diferença de apenas 0,4%. Superou o Kombi234

em 10 classes, o ANLCP300 em 3 classes, e nesse último, em mais 5 classes apresentou uma diferença inferior a 2%. E em relação ao Symbio10 foi superior em 4 classes e em 7 a diferença apresentada foi inferior a 3%.

O SingleGA10 foi melhor que o Kombi234 e o SHP a maioria das classes enquanto em relação ao ANLCP300 e Symbio10 o mesmo não ocorreu. Porém, devido as pequenas diferenças das variações percentuais, analisou-se graficamente o comportamento do SingleGA10 com o ANLCP300 (Gráfico 1) e o Symbio10 (Gráfico 2), e notou-se que o desempenho praticamente se igualam em 15 classes com o ANLCP300 e 16 quando comparadas com o Symbio10. As maiores diferenças apresentadas pelo método proposto em relação a todos os outros, se encontram na classe 12 e 18.

Gráfico 1: Representação gráfica das médias do custo total do SingleGA10 com o ANLCP300

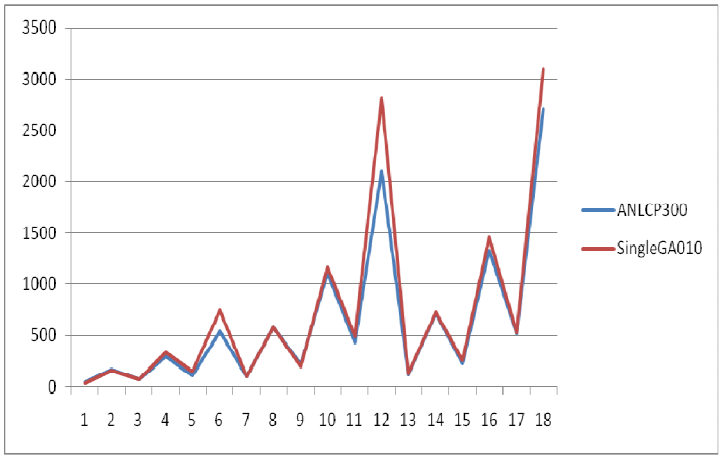

Gráfico 2: Representação gráfica das médias do custo total do SingleGA10 com o Symbio10

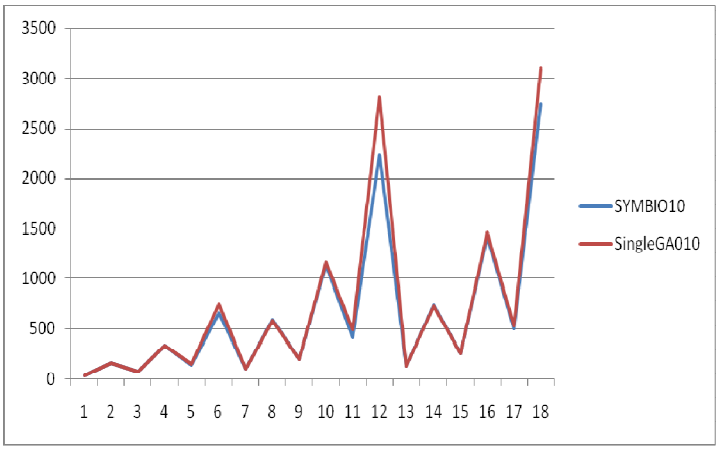

Tabela 4: Tempo médio em segundos dos métodos

Classe	SHP	Kombi234	ANLCP300	Symbio05	SingleGA10
1	0,01	0,14	1,12	18,54	22,35
2	0,08	1,14	3,17	37,88	80,62
3	0,17	1,74	0,89	33,25	99,5
4	0,21	16	1,15	68,11	236,96
5	0,27	38,03	0,64	58,29	409,03
6	0,31	379,17	0,91	158,04	480,74
7	0,01	0,07	18,1	19,62	50,19
8	0,02	0,2	11,78	48,48	55,65

9	0,04	3,37	105,49	38,75	167,13
10	0,06	3,25	106,59	127,25	286,31
11	0,22	36,26	216,97	117,25	486,48
12	0,32	76,31	376,04	426,08	493,27
13	0,01	0,08	7,43	17,66	11,34
14	0,02	0,13	2,02	31,19	16,59
15	0,03	1,81	60,02	41,12	72,57
16	0,04	2,6	39,87	133,9	41,42
17	0,16	50,93	267	153,45	175,21
18	0,24	70,94	538,92	388,89	258,42

O tempo computacional médio dos métodos, em segundos, de cada uma das 18 classes encontra-se na Tabela 4. Esse tempo tem caráter apenas ilustrativo e informativo. Ele não pode ser utilizado para efeito comparativo, pois cada método utiliza plataformas e linguagens de implementação diferentes. O Symbio05 é uma variação do método de [10]. Segundo o autor a diferença para o Symbio10 é irrelevante.

6. CONCLUSÕES E PERSPECTIVAS

Este trabalho realizou um estudo sobre o Problema de Corte unidimensional com o objetivo de minimizar o número de objetos processados e o *setup* e o implementou através da aplicação de um novo algoritmo genético que sozinho conseguiu resolver esses objetivos, geralmente, conflitantes.

A contribuição deste trabalho foi utilizar uma nova abordagem capaz de solucionar o problema de minimização proposto. Bem como, prover um estudo comparativo do seu desempenho com os métodos SHP, Kombi234, ANLCP300 e o Symbio10. Além da possibilidade de trabalhar com os dois custos, (c_1 e α), diretamente na função objetivo, que é uma vantagem do método sobre o Kombi234 e o SHP.

Verificou-se que o método proposto foi superior ao Kombi234, SHP na maioria das classes testadas, e obteve bons resultados em algumas classes quando comparados com o ANLCP300 e o Symbio10.

As análises gráficas realizadas mostram o bom desempenho do método proposto e deixam mais evidente que são pequenas as variações percentuais para a maioria das classes quando comparado com os outros métodos, validando-se o novo método. No entanto, sua grande desvantagem em relação aos demais é o tempo computacional. Acredita-se que as escolha dos parâmetros e os operadores genéticos adotados possam ser os responsáveis por esse alto custo computacional, principalmente o operador de seleção que seleciona apenas os mais aptos, prejudicando assim, a diversidade da população.

E, baseado nos resultados computacionais apresentados, pode-se afirmar que o novo método é competitivo e promissor no ambiente de Problemas de Corte unidimensional.

Como trabalhos futuros visa-se estudar estudar outros custos de *setup* e também o comportamento de outros operadores genéticos, principalmente o de seleção e o de recombinação para obter melhores resultados num menor espaço de tempo e acrescentar uma restrição à demanda para evitar que os números de objetos processados sejam muito elevados. Depois disso, busca-se adaptar o novo método para tratar o problema de corte unidimensional proposto como um problema multiobjetivo, utilizando também os conceitos e técnicas de paralelização com o objetivo de torná-los mais eficientes.

7. REFERÊNCIAS BIBLIOGRÁFICAS

[1] **Carneiro, S. A.**, 1994 Problema de Corte via Algoritmo Genético. Dissertação (Mestrado) Universidade Federal do Espírito Santo, UFES, Brasil.

[2] **Kantorovich, L. V.**, 1960. Mathematical Methods of Organizing and Planning Production. Management Science, vol. 6, pp 366-422.

[3] **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.

[4] **Metzger, R. W.,** 1958 Stock Slitting. Elementary Mathematical Programming, Wiley.

[5] **Eilon, S.,** 1960. Optimizing the shearing of steel bars. Journal of Mechanical Engineering Science,vol. 2, pp: 129-142.

[6] **Dowsland, K. and Dowsland, W., 1992.** Packing Problems. European Journal of Operational Research, vol. 56, pp: 2-14, 1992.

[7] **Gilmore, P. C. & Gomory, R. E.**, 1961. A linear programming approach to the cutting stock problem. Operations Research, vol. 9, pp. 849–859.

[8] **Gilmore, P. C. & Gomory, R. E.,** 1963. A linear programming approach to the cutting stock problem. Operations Research, vol. 11, pp. 863–888.

[9] **Haessler, R.,** 1975. Controlling cutting pattern changes in one-dimensional trim problems. Operations Research, vol. 23, pp. 483–493.

[10] **Golfeto, R. R., Moretti, A. C., & SallesNeto, L. L**., 2007a. Algoritmo genético simbiótico aplicado ao problema de corte unidimensional. In Anais do XXXIX Simpósio Brasileiro de Pesquisa Operacional.

[11] **Dyckhoff, H., 1990**. A typology of cutting and packing problems. European Journal of Operation Research, vol. 444, pp. 145–159.

[12] **Wäscher, G., Haubner, H., Schumann, H**., 2007 An improved typology of cutting and packing problems. European Journal of Operational Research 183, pp 1109-1130.

[13] **Wäscher, G., Gau, T., 1996**. **Heuristics for the integer one-dimensional cutting stock problem**: a computational study. *OR Spektrum*, 18, 131-144.

[14] **Allaby, M. Dictionary of Ecology**., 1998. New York: Oxford University Press.

[15] **Pianka, E. R.,** 1994. Evolutionary Ecology. New York: HarperCollins College Publisher.

[16] **Wall, M.,** 1996. A C++ Library of Genetic Algorithm Components, Mechanical Engineering Department Massachusetts Institute of Technology.

[17] **Gau, T. e Wascher, G**., 1995. CUTGEN1: A Problem Generator for the Standard One-dimensional Cutting Stock Problem. European Journal of Operational Research, vol. 84, pp: 572-579.

[18] **Foerster, H. & Wascher, G.**, 2000. Pattern reduction in one-dimensional cutting-stock problems.International Journal of Prod. Res., vol. 38, pp. 1657–1676.

[19] **SallesNeto, L. L. & Moretti, A. C**., 2005. Modelo não-linear para minimizar o número de objetos processados e o *setup* num problema de corte unidimensional. In Anais do XXXVII Simpósio Brasileiro de Pesquisa Operacional, pp. 1679–1688.