META-HEURÍSTICA *SIMULATED ANNEALING* APLICADA AO PROBLEMA DE CORTE BIDIMENSIONAL NÃO-GUILHOTINADO

Gelinton Pablo Mariano
Mestrando em Informática - Universidade Federal do Espírito Santo
Av. Fernando Ferrari, 514 – Goiabeiras – 29075-910 –Vitória – ES – Brasil
pablo06@gmail.com

André Renato Sales Amaral
Programa de Pós-Graduação em Informática - Universidade Federal do Espírito Santo
Av. Fernando Ferrari, 514 – Goiabeiras – 29075-910 –Vitória – ES – Brasil
amaral@inf.ufes.br

RESUMO

Este trabalho considera o problema de corte bidimensional não-guilhotinado que é o problema de cortar um dado conjunto finito de pequenos retângulos de um grande retângulo de dimensões fixas, de forma a obter o máximo de lucro. Um algoritmo baseado na meta heurística *Simulated Annealing* é proposto para solucionar o problema. O algoritmo aplica movimentos de destruição de parte da solução e de reconstrução a fim de diversificar o espaço de busca. Os resultados computacionais com instâncias da literatura mostraram que o algoritmo é bastante competitivo.

PALAVARAS CHAVE. Corte não-guilhotinado, Meta heurística, *Simulated Annealing*.

ABSTRACT

This paper considers the two-dimensional non-guillotine cutting problem, which is the problem of cutting a given finite set of small rectangles from a large rectangle of fixed dimensions, so as to obtain the maximum profit. An algorithm based on Simulated Annealing is proposed to solve the problem. The algorithm applies moves of destruction and reconstruction of the solution in order to diversify the search space. Computational results on problem instances from the literature have shown that the algorithm is very competitive.

KEYWORDS. Non-guillotine cutting. Metaheuristic. Simulated Annealing.

1. Introdução

O problema de corte bidimensional não-guilhotinado consiste em cortar um dado conjunto finito de pequenos retângulos de um grande retângulo de dimensões fixas obtendo o máximo lucro (Alvarez-Valdes et al., 2007).

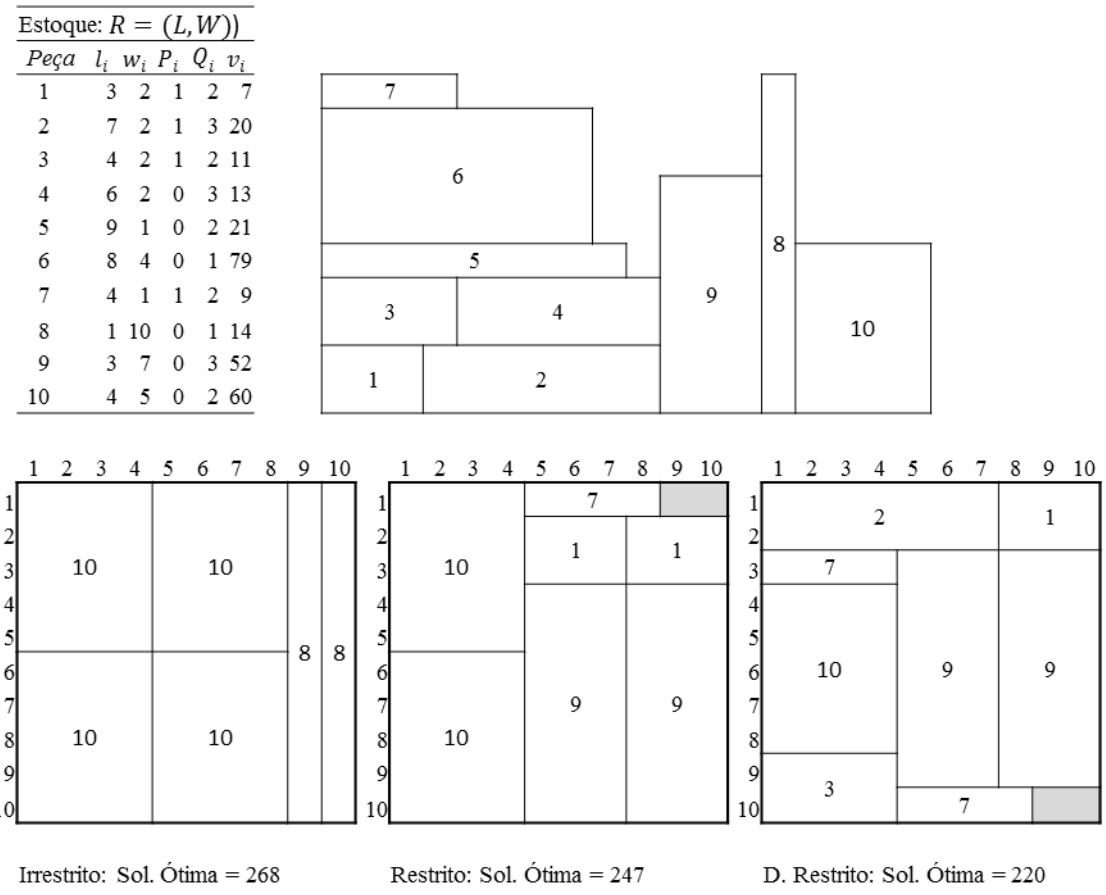
O retângulo grande $R = (L, W)$ possui comprimento L e largura W. Os retângulos menores são denominados *peças*. Cada peça i pode ser definida por três atributos $i = (l_i, w_i, v_i)$, onde l_i representa o comprimento, w_i a largura e v_i o valor da peça.

As peças têm orientação fixa e suas bordas são paralelas às bordas do retângulo *R*. O retângulo *R*, então, deve ser cortado em x_i peças menores, com dimensões correspondentes às peças i, em que $0 \leq P_i \leq x_i \leq Q_i$, onde P_i e Q_i representam a quantidade mínima e máxima, respectivamente, de cada peça. O problema é maximizar $\sum_i v_i x_i$.

Alvarez-Valdes et al. (2007) diferenciam o problema em 3 tipos:

1. Irrestrito: $\forall i, P_i = 0, Q_i = \lfloor L * W / l_i * w_i \rfloor$
2. Restrito: $\forall i,\ P_i = 0;\ \exists i, Q_i < \lfloor L * W / l_i * w_i \rfloor$
3. Duplamente Restrito: $\exists\, i,\ P_i > 0;\ \exists\, j, Q_j < \lfloor L * W / l_j * w_j \rfloor$

No tipo irrestrito, não há limites mínimo e máximo de quantidade para o corte das peças. Entretanto, no tipo restrito, a quantidade de peças de cada tipo possui um limite superior e no duplamente restrito, possui tanto o limite superior quanto o inferior. A Figura 1 mostra o exemplo de um estoque cortado com os três tipos do problema.



Estoque: $R = (L, W)$

Peça	l_i	w_i	P_i	Q_i	v_i
1	3	2	1	2	7
2	7	2	1	3	20
3	4	2	1	2	11
4	6	2	0	3	13
5	9	1	0	2	21
6	8	4	0	1	79
7	4	1	1	2	9
8	1	10	0	1	14
9	3	7	0	3	52
10	4	5	0	2	60

Figura 1 – Exemplo de solução para o problema de corte. Adaptado de Alvarez-Valdes *et al.* (2007).

Os problemas restritos parecem ser mais interessantes para aplicações práticas do que os irrestritos, baseando-se na quantidade de trabalhos relacionados ao primeiro. Diferentes meta heurísticas baseadas em algoritmos genéticos, busca tabu, GRASP, entre outros são encontrados na literatura. Lai e Chan (1997) utilizam o *Simulated Annealing* em que uma solução é dada por uma lista ordenada de peças e essa lista é traduzida para um layout por meio de um algoritmo que coloca as peças em determinadas posições no retângulo *R*. Alvarez-Valdes et al. (2005) utilizam o GRASP e Alvarez-Valdes et al. (2007) utilizam a busca tabu para solucionar o problema. Outros algoritmos heurísticos podem ser encontrados em Bortfeldt e Winter (2009), Chen e Huang (2007), Egeblad e Pisinger (2009), Gonçalves (2007) e Wei et al. (2009).

Na próxima seção é apresentada uma heurística construtiva para o problema. Na seção 3, uma implementação da meta heurística *Simulated Annealing* é descrita. Na seção 4, os experimentos computacionais com os resultados dos testes; e por fim as conclusões.

2. Heurística de construção de uma solução

No processo de construção de uma solução são utilizados um retângulo *R* de onde as peças são cortadas e duas listas (uma lista *P* de tipos de peças a serem cortadas e uma lista *B* de peças que foram cortadas). Inicialmente, a lista *P* contém todos os tipos peças disponíveis com a respectiva quantidade e a lista *B* está vazia. Assim, para cada tipo de peça em *P*, o algoritmo tenta encontrar o primeiro espaço disponível factível em que a peça daquele tipo possa ser cortada de *R*, analisando os pontos candidatos para sua inserção. Cada peça possui dois pontos candidatos: o primeiro é o ponto mais à direita e mais acima; o segundo é o ponto mais à esquerda e mais abaixo em relação a *R*, como mostrado na Figura 2. Encontrado um espaço, a peça é adicionada à *R*, inserida em *B* e decrementada a quantidade do seu tipo em *P*. Neste último caso, se a peça for a única do seu tipo, este tipo é eliminado da lista *P*. O algoritmo tenta adicionar primeiramente as peças do mesmo tipo antes de prosseguir para as demais disponíveis. O processo de construção da solução termina quanto não é possível adicionar mais nenhuma peça presente em *P* ou *P* está vazio. O pseudocódigo deste algoritmo é dado pelo Algoritmo 1.

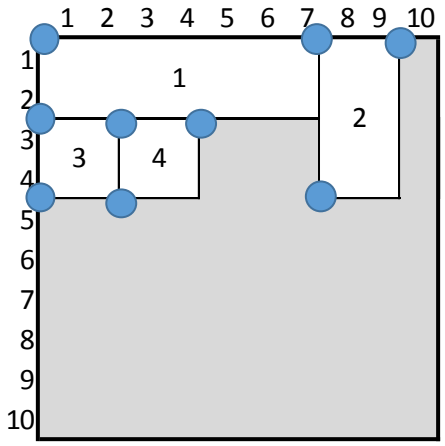


Figura 2 - Exemplo de construção de uma solução com os pontos candidatos em relação à *R*.

As estruturas de duas listas (*B* e *P*) e o grande retângulo *R* foram baseadas no algoritmo construtivo de Alvarez-Valdes et al. (2007), enquanto a ideia de testar os dois pontos de cada peça foi proposta aqui.

Algoritmo 1: Criar Solução

```
1.  criarSolução( Solução nova )
2.      enquanto ( i < |nova.P| )
3.          Tentar adicionar a peça nova.P[i] ao ponto de origem do estoque ou para cada
            peça em B, tentar adicionar a peça nova.P[i] nos pontos candidatos;
4.          se (a peça foi adicionada) então
5.              B.Adiciona(nova.P[i]);
6.              se (nova.P[i].quantidade == 1) então
7.                  Excluir ( nova.P[i] );
8.              senão
9.                  nova.P[i].quantidade − − ;
10.             fim_se
11.         senão
12.             i + +;
13          fim_se
14.     fim_enquanto
15.     retornar nova;
16. fim_algoritmo
```

3. Algoritmo *Simulated Annealing*

O *Simulated Annealing* (SA) foi proposto inicialmente por Kirlpatrick et al. (1983) que faz uma analogia a simulação do processo de recozimento de metais. Inicialmente, o processo começa com uma temperatura elevada e, a cada iteração, uma nova solução é gerada através de um algoritmo de busca até que seja encontrado um equilíbrio de solução naquela temperatura. Então, a temperatura é diminuída, seguindo uma taxa de resfriamento. Nessa nova temperatura, o algoritmo de busca é novamente executado. Esse processo ocorre até que a temperatura alcance um valor de congelamento. Nesse ponto não se obtém mais melhorias e o algoritmo é encerrado.

No SA, uma solução gerada que seja pior que a atual pode ainda ser aceita de acordo com uma certa probabilidade. A Tabela 1 mostra os parâmetros utilizados pelo SA.

O Algoritmo 2 apresenta o pseudocódigo utilizado na implementação do algoritmo SA. Antes de criar uma solução inicial (linha 3), a lista P é ordenada em ordem decrescente de acordo com o critério $\frac{v_i}{l_i * w_i}$. A função *criarSolução( )* na linha 1 foi descrita na seção 2; e a função indicada na linha 11 do pseudocódigo gera as soluções vizinhas como descrito pelo Algoritmo 3. O parâmetro γ indica o percentual da área do retângulo R, ocupada pela solução, que será removida. A priori, as peças a serem removidas são escolhidas aleatoriamente até que o percentual da área definido por γ seja alcançado e, então, depois de removidas, são adicionadas à lista P. Em seguida, as peças restantes são deslocadas para a esquerda e para cima (posição inicial de R). Por fim, a função *criarSolução(nova)* é chamada para preencher os espaços livres, gerando uma nova solução.

Algoritmo 2: Algoritmo SA proposto

```
1.  SA(T, T_c, It_max, α)
2.      Solução atual, nova, melhor;
3.      ordenar lista atual.P em ordem decrescente de v_i / (l_i * w_i)
4.      criarSolução(atual);
5.      melhor = atual;
6.      temp = T;
7.      iterT = 0;
8.      enquanto (temp > T_c) faça
9.          enquanto (iterT > It_max) faça
10.             iterT + +;
11.             nova = trocarSolução(atual);
12.             Δ = f(atual) − f(nova);
13.             se (Δ < 0) então
14.                 atual = nova;
15.                 se (f(nova) > f(melhor) ) então
16.                     melhor = atual;
17.                 fim_se
18.             senão
19.                 Tomar x ∈ [0,1];
20.                 se (x < e^(−Δ/T) ) então
21.                     atual = novo;
22.                 fim_se
23.             fim_se
24.         fim_enquanto
25.         T = α * T;
26.         iterT = 0;
27.     fim_enquanto
28.     retorna melhor;
29. fim_algoritmo
```

Tabela 1 - Descrição dos parâmetros utilizados no SA.

Parâmetro	**Descrição**
T	Temperatura inicial
T_c	Temperatura de congelamento
It_{max}	Número máximo de iterações
α	Taxa de resfriamento
γ	Percentual de remoção da solução

Algoritmo 3: Trocar Solução

1. trocarSolução(Solução $nova$)
2. Remover γ da área de R;
3. Adicionar peças removidas a P;
4. Deslocar peças restantes para esquerda;
5. criarSolucao($nova$);
6. **retornar** $nova$;
7. **fim_algoritmo**

O cálculo da função objetivo é realizado quando se adiciona ou se remove uma peça de R, bastando adicionar ou subtrair seu valor v_i, respectivamente, ao valor da solução total de R.

4. Experimentos Computacionais

Para este experimento, foi utilizado um conjunto de 21 instâncias distintas:

- 12 instâncias propostas por Beasley (1985);
- 2 instâncias propostas por Hadjiconstantinou e Christofides (1995);
- 1 instância por Wang (1983);
- 1 instância de Christofides e Whitlock (1977);
- 5 instâncias de Fekete e Schepers (2000).

Cada instância foi executada 10 vezes com sementes aleatórias em um computador Intel Core 2 Duo de 2,4 GHz com 3 GB de memória RAM com Windows 8.1. Toda a implementação foi desenvolvida na linguagem C++.

A Tabela 2 apresenta os valores dos parâmetros utilizados na implementação do algoritmo SA.

Tabela 2 - Descrição e valores dos parâmetros utilizados no SA.

Parâmetro	Descrição	Valor
T	Temperatura inicial	1000
T_c	Temperatura de congelamento	0,11
It_{max}	Número máximo de iterações	700
α	Taxa de resfriamento	0,98
γ	Percentual de remoção da solução	0,35

A Tabela 3 mostra para cada instância, o valor ótimo da solução (que é conhecido para as instâncias testadas), seguido dos valores das soluções encontradadas por um algoritmo genético (Beasley, 2004), GRASP (Alvarez-Valdes et al., 2005) e Busca Tabu (Alvarez-Valdes et al., 2007). A tabela apresenta também o melhor valor obtido pelo algoritmo SA proposto, o valor médio das 10 execuções e o desvio padrão (D.P.). Por fim, é contabilizado o número de soluções ótimas encontradas por cada método.

Os resultados apresentados na Tabela 3 mostram que para 18 das 21 instâncias testadas, o algoritmo SA atingiu os valores ótimos correspondentes. O valor encontrado para a instância 15 apresentou uma pequena diferença de 0,36% do valor ótimo, para a instância 16 uma diferença de 1,09% e para a instância 19 uma diferença de 0,96%.

A Tabela 4 apresenta os tempos médios, em segundos, para os métodos considerados. Esses tempos são apresentados apenas para referência já que os métodos foram executados em computadores diferentes e portanto não podem ser diretamente comparados. Observa-se que os tempos de execução máximos observados para o algoritmo SA ocorrem para as instâncias 17 e 19 e são de cerca de 23 segundos.

Tabela 3 – Melhores resultados obtidos com o SA.

Problema				Solução				SA		
#	L	W	Q_i	Ótimo	Beasley	GRASP	TABU	Melhor	Média	D.P.[1]
1	10	10	5	164	164	164	164	164	164	0
2	10	10	7	230	230	230	230	230	230	0
3	10	10	10	247	247	247	247	247	247	0
4	15	10	5	268	268	268	268	268	268	0
5	15	10	7	358	358	358	358	358	358	0
6	15	10	10	289	289	289	289	289	289	0
7	20	20	5	430	430	430	430	430	430	0
8	20	20	7	834	834	834	834	834	834	0
9	20	20	10	924	924	924	924	924	924	0
10	30	30	5	1452	1452	1452	1452	1452	1452	0
11	30	30	7	1688	1688	1688	1688	1688	1688	0
12	30	30	10	1865	1801	1865	1865	1865	1865	0
13	30	30	7	1178	1178	1178	1178	1178	1178	0
14	30	30	15	1270	1270	1270	1270	1270	1270	0
15	70	40	20	2726	2721	2726	2726	2716	2715	2,11
16	40	70	20	1860	1720	1860	1860	1840	1826	9,66
17	100	100	15	27718	27486	27589	27718	27718	27596,9	45,36
18	100	100	30	22502	21976	21976	22502	22502	22028,6	166,33
19	100	100	30	24019	23743	23743	24019	23789	23747	14,81
20	100	100	33	32893	31269	32893	32893	32893	32893	0
21	100	100	29	27923	26332	27923	27923	27923	27923	0
Nº Sol. Ótim. (de 21)					13	18	21	18		

[1] D.P.: Desvio Padrão.

Tabela 4 – Tempos de execução.

Problema				Tempo			
#	L	W	Q_i	Beasley	GRASP	TABU	SA
1	10	10	5	0,02	0	0,06	0,002
2	10	10	7	0,16	0	0	0
3	10	10	10	0,53	0	0	0,004
4	15	10	5	0,01	0	0	0
5	15	10	7	0,11	0	0	0
6	15	10	10	0,43	0	0	0,059
7	20	20	5	0,01	0	0	0
8	20	20	7	3,25	0,77	0,16	0,002
9	20	20	10	2,18	0	0,05	0,017
10	30	30	5	0,03	0	0	0,007
11	30	30	7	0,6	0,05	0	0,021
12	30	30	10	3,48	0,05	0,06	0,092
13	30	30	7	0,02	0	0	0
14	30	30	15	0,16	0	0	0
15	70	40	20	6,86	0,77	0,11	3,77
16	40	70	20	0,16	0	0	5,435
17	100	100	15	0,53	2,31	0,05	23,825
18	100	100	30	0,01	4,17	2,14	8,709
19	100	100	30	0,11	3,68	3,40	23,076
20	100	100	33	0,43	0	0,66	3,136
21	100	100	29	3,48	0	0	0,821

3. Conclusões

Neste trabalho consideramos o problema de corte bidimensional não guilhotinado. A estratégia de solução proposta para o problema foi baseada em movimentos de destruição aleatória e reconstrução de uma solução, a fim de explorar o espaço de busca por meio de um algoritmo *Simulated Annealing* (SA). O algoritmo SA obteve resultados bastante competitivos quando comparado com os melhores algoritmos da literatura.

Como pesquisa futura, pretende-se testar outros movimentos de organização das peças e reconstrução da solução.

Referências

Alvarez-Valdes, R. Parreño, F. Tamarit, J.M. A GRASP algorithm for constrained two-dimensional non-guillotine cutting problems, Journal of Operational Research Society 56(4), 414-425, 2005.

Alvarez-Valdes, R. Parreño, F. Tamarit, J.M. A tabu search algorithm for a two-dimensional non-guillotine cutting problem, European Journal of Operational Research 183(3),1167-1182, 2007.

Beasley, J.E. An exact two-dimensional non-guillotine cutting tree search procedure, Operations Research, v.33, p 49-64, 1985.

Beasley, J.E. A population heuristic for constrained two-dimensional nonguillotine cutting, European Journal of Operational Research, v.156, p 601-627, 2004.

Bortfeldt, A.; Winter, T. A genetic algorithm for the two-dimensional knapsack problem with rectangular pieces. International Transactions in Operational Research, Oxford, v. 16, p. 685–713, 2009.

Chen, M.; Huang, W. A two-level search algorithm for 2D rectangular packing problem, Computers & Industrial Engineering, New York, v. 53, p. 123–136, 2007.

Christofides, N.; Whitlock, C. An algorithm for two-dimensional cutting problems, Operations Research 25 (1977) 30-44.

Egeblad, J.; Pisinger, D. Heuristic approaches for the two- and three-dimensional knapsack packing problem. Computers and Operations Research, New York, v. 36, p.1026–1049, 2009.

Fekete, S.P.; Schepers, J. On more-dimensional packing III: Exact Algorithms, Report 97290 available from the first author at Department of Mathematics, Technical University of Berlin, Germany (1997), revised (2000).

Gonçalves, J. F. A hybrid genetic algorithm-heuristic for a two-dimensional orthogonal packing problema. European Journal of Operational Research, Amsterdam, v. 183, p. 1212-1229, 2007.

Hadjiconstantinou, E.; Christofides, N. An exact algorithm for general, orthogonal, two-dimensional knapsack problems, European Journal of Operational Research 83 (1995) 39-56.

Lai, K.K.; Chan, J.W.M. Developing a simulated annealing algorithm for the cutting stock problem, Computers and Industrial Engineering 32 (1997) 115-127.

Wang, P.Y. Two algorithms for constrained two-dimensional cutting stock problems, Operations Research 31 (1983) 573-586.

Wei, L.; Zhang, D.; Chen, Q. A least wasted first heuristic algorithm for the rectangular packing problem. Computers & Operations Research, New York, v. 365, p. 1608–1614, 2009.