UM ALGORITMO GRASP PARA O SALBP-2

Dayan de Castro Bissoli
Universidade Federal do Espírito Santo (UFES)
Av. Fernando Ferrari, 514 – Goiabeiras – 29075910 – Vitória/ES – Brasil
dayan.bissoli@ufes.br

André Renato Sales Amaral
Universidade Federal do Espírito Santo (UFES)
Av. Fernando Ferrari, 514 – Goiabeiras – 29075910 – Vitória/ES – Brasil
amaral@inf.ufes.br

RESUMO

O problema de balanceamento de linha de montagem simples (*Simple Assembly Line Balancing Problem* - SALBP) refere-se à atribuição de tarefas com tempos de processamento pré-definidos a estações de trabalho que estão dispostas em uma linha, respeitando-se as restrições de precedência entre as tarefas. O chamado SALBP do tipo dois (SALBP-2) visa minimizar o tempo de ciclo para um número fixo de estações de trabalho. Neste artigo, apresentamos um algoritmo GRASP para o SALBP-2. O algoritmo se mostrou eficaz, gerando soluções com valores iguais ou próximos àqueles das melhores soluções conhecidas para um grande número de instâncias do SALBP-2.

PALAVRAS CHAVE. Balanceamento de linha de montagem, Número fixo de estações de trabalho, GRASP, Área de classificação principal (Metaheuristicas, Otimização Combinatória, PO na Administração & Gestão da Produção).

ABSTRACT

The simple assembly line balancing problem (SALBP) refers to the assignment of tasks with predefined processing times to workstations that are arranged to a line, respecting the precedence constraints between tasks. SALBP type two (SALBP-2) aims minimize the cycle time for a fixed number of workstations. In this article, we present a GRASP for SALBP-2. The algorithm is effective, with values equal or close to those of the best known solutions for a great number of SALBP-2 instances.

KEYWORDS. Assembly line balancing, Fixed number of workstations, GRASP, Área de classificação principal (Metaheuristics, Combinatorial Optimization, OR in Administration & Production Management).

1. Introdução

Na produção industrial de grandes volumes de produtos padronizados, ou mesmo na produção de pequenos volumes customizados, sistemas de produção que seguem um fluxo direcionado são chamados de *linhas de montagem*. Com a necessidade da tomada de decisões no planejamento da produção a médio prazo surge, dentre outros, o *problema de balanceamento de linhas de montagem* (Becker e Scholl, 2006; Silva et al., 2015).

Segundo (Askin e Standridge, 1993), dois parâmetros são particulares à grande maioria das variantes deste problema:

a) A existência de *precedência* entre as tarefas de montagem: o início da execução de certas tarefas está condicionado ao término de outras;
b) O *tempo de ciclo* da linha: representa o intervalo de tempo entre a saída de dois produtos consecutivos em uma linha cadenciada.

Mais precisamente, uma linha de montagem consiste em m estações de trabalho dispostas ao longo de uma esteira rolante ou um equipamento mecânico similar de manejo de materiais, onde peças são movidas consecutivamente entre as estações $k(k = 1, ..., m)$, até chegar ao final da linha. Em cada estação, algumas operações são realizadas repetidamente, conforme seu *tempo de ciclo*. Como há várias operações sendo realizadas simultaneamente, surge a necessidade de distribuir a carga de trabalho o mais uniformemente possível entre as estações de trabalho, buscando seu balanceamento. Dessa forma, problemas que possuem o objetivo de otimizar processos relacionados com a fabricação de produtos através de linhas de montagens são conhecidos como problemas de balanceamento de linhas de montagem (*assembly line balancing problem* - ALBP) (Becker e Scholl, 2006; Blum, 2011).

Becker e Scholl (2006) afirmam que a fabricação de um produto em uma linha de montagem requer o particionamento do volume total de trabalho entre um conjunto de operações elementares denominadas tarefas $T = \{1, \ldots, n\}$. A execução de uma tarefa j leva um tempo t_j e exige máquinas específicas e/ou profissionais especialistas. Devido às condições tecnológicas e organizacionais, restrições de precedência entre as tarefas devem ser observadas.

A existência de precedência entre as tarefas de montagem pode ser facilmente visualizada por um grafo de precedência. Esse grafo contém um nó para cada tarefa, pesos com nós para os tempos das tarefas e arcos para as restrições de precedência. A Figura 1 apresenta um grafo de precedência com $n = 10$ tarefas possuindo tempos entre 2 e 9 unidades de tempo. Em relação às restrições de precedência, pode-se exemplificar a situação da Tarefa 5, que para ser processada, requer que sejam anteriormente processadas as Tarefas 1 e 4 (antecessores diretos) e 3 (antecessor indireto). Por outro lado, a Tarefa 5 deve ser concluída antes de seu sucessor direto (6) e indiretos (8 e 9). (Boysen et al., 2007).

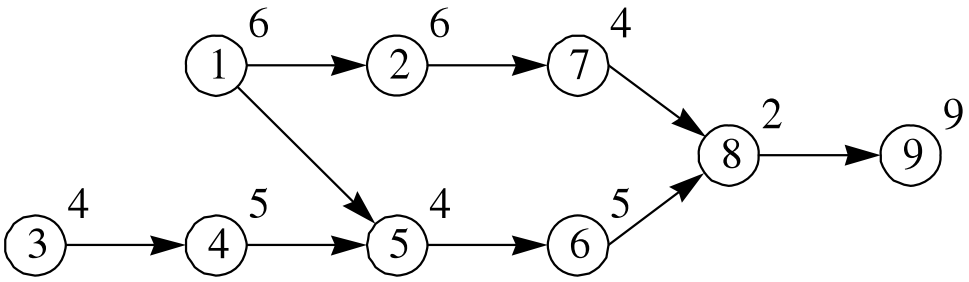

Figura 1. Grafo de precedência

O presente estudo aborda o problema de balanceamento de linha de montagem simples (*simple assembly line balancing problem* - SALBP) – em sua versão de número 2, que tem como objetivo de minimizar o tempo de ciclo, dado o número fixo de estações de trabalho (Becker e Scholl, 2006).

Este trabalho é organizado da seguinte forma: Na próxima seção realiza-se a descrição do problema de balanceamento de linhas de montagem, seguida da apresentação do método proposto para solucionar o problema. O Capítulo 4 apresenta os resultados obtidos, e as conclusões são discutidas na seção seguinte.

2. CARACTERÍSTICAS DO PROBLEMA

2.1. Balanceamento de linha de montagem

De acordo com Peinado e Graeml (2007), o balanceamento da linha de montagem consiste na atribuição de tarefas às estações de trabalho que formam a linha de forma que todas as estações demandem aproximadamente o mesmo tempo para a execução da tarefa. Isto minimiza o tempo ocioso de mão-de-obra e de equipamentos. Em uma linha de montagem, o trabalho flui de uma estação para outra conforme Figura 2.

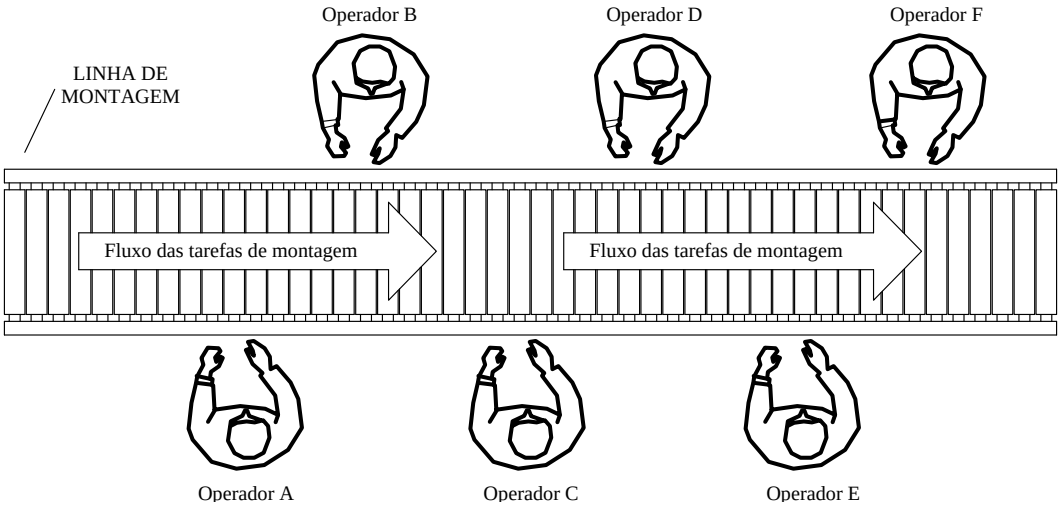

Figura 2. Fluxo de operações em uma linha de montagem.

Uma extensa revisão sobre o problema de balanceamento de linha é apresentada em Becker e Scholl (2006) e Scholl e Becker (2006). Os autores classificaram as principais

características do problema de balanceamento de linha de montagem (ALBP em inglês, *Assembly Line Balancing Problem*) considerando as suas diversas restrições e diferentes objetivos, como mostrado na Figura 3.

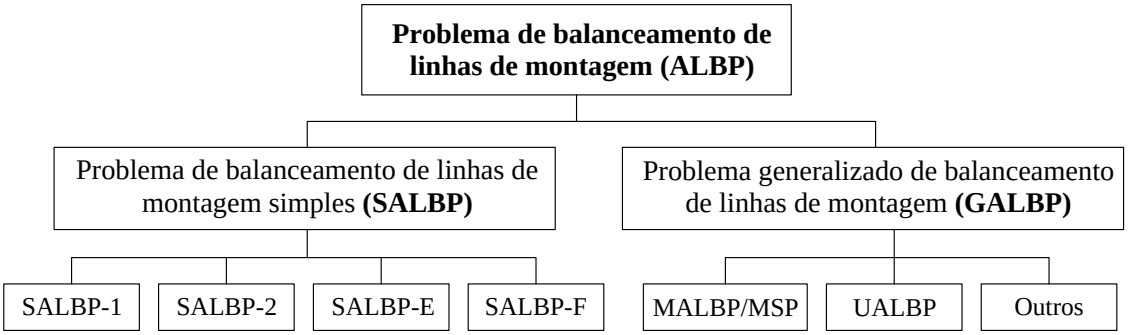

Figura 3. Classificação dos problemas de balanceamento de linha de montagem.

O ALBP é dividido em duas classes: Problema de Balanceamento da Linha de Montagem Simples (SALBP em inglês, *Simply Assembly Line Balancing Problem*) e Problema de Balanceamento da Linha de Montagem Geral (GALBP em inglês, *General Assembly Line Balancing Problem*).

O problema generalizado de balanceamento de linhas de montagem (GALBP em inglês, *generalized assembly line balancing problem*) resolve as situações nas linhas de montagem de maneira mais realista considerando, por exemplo, as linhas de montagem com *layout* em U e com estações de trabalho paralelas. O Problema de Balanceamento da Linha de Montagem Simples (SALBP em inglês, *simple assembly line balancing problem*), consiste na atribuição de um conjunto de tarefas a um conjunto de estações, com a finalidade de minimizar o número de estações ou o tempo de ciclo da linha.

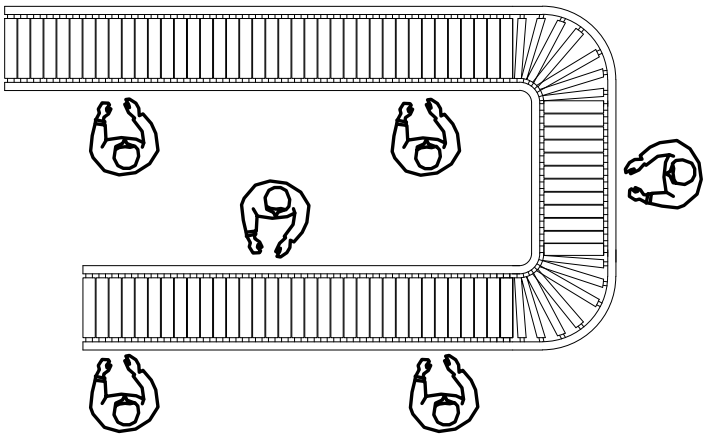

Figura 4. Linha de montagem em formato U.

A maioria das pesquisas no balanceamento de linha de montagem tem sido dedicada à modelagem e resolução do problema de balanceamento de linha de montagem simples (SALBP), que possui as seguintes características principais (Becker e Scholl, 2006):

- produção em massa de um produto homogêneo;

- ritmo da linha com tempo de ciclo fixo c;
- tempos de operações t_j determinísticos (e integrais);
- não há restrições de atribuição além das restrições de precedência;
- *layout* em linha com m estações em apenas um lado;
- todas as estações são equipadas com a mesma quantidade de máquinas e trabalhadores;
- maximizar a eficiência da linha $E = t_{sum}/(m \times c)$ com tempo total de tarefa sendo $t_{sum} = \sum_{j=1}^{n} t_j$.

O SALBP é relevante para a simples montagem de produtos em linha reta no qual apenas as restrições de precedência entre as tarefas são consideradas. São definidos quatro tipos dessa classificação (Becker e Scholl, 2006; Baybars, 1986):

- SALBP-1: consiste na atribuição de tarefas aos postos de trabalho de tal modo que o número de estações (m) é minimizado para uma dada taxa de produção (tempo de ciclo fixo, c), o que é equivalente a minimizar o tempo ocioso total.;
- SALBP-2: busca minimizar o tempo de ciclo (maximizar a taxa de produção), sujeito a restrições de precedência para um determinado número (m) estações disponíveis;
- SALBP-E: é a versão mais geral do problema, que visa maximizar a eficiência da linha (E), e, simultaneamente minimizar c e m, considerando suas inter-relações;
- SALBP-F é um problema de viabilidade que se baseia em estabelecer se há ou não um balanceamento de linha viável para uma dada combinação entre c e m.

Na seção seguinte, explana-se a variação SALBP-2, objeto de estudo deste trabalho.

2.2. SALBP-2

O SALBP-2 pode ser formulado matematicamente como se segue. Seja x_{is} uma variável binária definida como 1 se e somente se a tarefa $i \in T$ é atribuída à estação de trabalho $1 \leq s \leq m$. Uma instância (T, G, m) consiste em três componentes. $T = \{1, ..., n\}$ é um conjunto de n tarefas. Cada tarefa $i \in T$ tem o tempo de processamento pré-definido como $t_i > 0$. Sem perda de generalidade, os tempos de processamento são, daqui em diante, considerados valores inteiros. Além disso, é dado um grafo de precedência acíclico orientado $G = (T, A)$, sendo T seu conjunto de nós e A seu conjunto de arcos. Finalmente, m representa um número pré-definido de estações de trabalho que estão ordenadas de 1 a m. Um arco $a_{i,j} \in A$ indica que a tarefa i deve ser processada antes da tarefa j. Dada uma tarefa $j \in T$, $P_j \subset T$ denota o conjunto de tarefas que devem ser processadas antes de j. Uma solução viável é obtida através da atribuição de cada tarefa a exatamente uma estação de trabalho, de modo que as restrições de precedência entre as tarefas sejam satisfeitas. Conforme Blum (2011), o SALBP-2 pode ser expresso como um problema de programação inteira (PI) da seguinte maneira:

$$\text{Minimizar } z \tag{1}$$

$$\text{Sujeito a: } \sum_{s=1}^{m} x_{is} = 1 \forall i \in T \tag{2}$$

$$x_{is} \leq \sum_{s'=1}^{s} x_{js'} \forall i \in T, s = 1, ..., m, j \in P_i \tag{3}$$

$$\sum_{i \in T} t_i x_{is} \leq z, s = 1, ..., m \quad (4)$$

$$x_{is} \in 0, 1 \forall i \in T, s = 1, ..., m \quad (5)$$

$$z > 0 \quad (6)$$

A função objetivo (1) minimiza o tempo de ciclo $z > 0$. A restrição (2) garante que cada tarefa $i \in T$ seja atribuída a uma única estação de trabalho $1 \leq s \leq m$. A restrição (3) reflete os relacionamentos de precedência entre as tarefas. Mais especificamente, se a tarefa $i \in T$ é atribuída à estação de trabalho $1 \leq s \leq m$, todas as tarefas $j \in P_i$ devem ser atribuídas às estações de trabalho $1 \leq s' \leq m$, sendo $s' \leq s$. A restrição (4) garante que a soma dos tempos de processamento das tarefas atribuídas às estações de trabalho $1 \leq s \leq m$ não excedam o tempo de ciclo z.

Diferentes abordagens foram realizadas para o SALBP-2. No que se refere a métodos exatos, até o presente momento, o trabalho que conseguiu melhores resultados foi o proposto por Klein e Scholl (1999), que desenvolveram um método *branch & bound* com uma nova técnica de enumeração, a que chamaram o método do limitante inferior local, que é complementado por uma série de regras delimitadoras e de dominância. O atual estado da arte para o SALBP-2 encontra-se no método heurístico *iterative beam search* (IBS) proposto por Blum (2011), que encontrou valores melhores que os das melhores soluções conhecidas para um conjunto de instâncias clássicas para o problema.

Considerando que as versões do SABLP são *NP-Hard* (Scholl, 1999), o presente trabalho realiza uma abordagem heurística para o problema, onde implementou-se a metaheurística GRASP (*greedy randomized adaptive search procedure*), proposto por (Feo e Resende, 1995) que é descrita na seção seguinte.

3. Algoritmo GRASP para o SALBP-2

De acordo com Feo e Resende (1995), o GRASP é um processo iterativo, onde a cada iteração, são realizadas duas fases: construção e busca local. A fase de construção gera uma solução viável, cuja vizinhança é explorada pela busca local. A melhor solução encontrada durante a execução do algoritmo GRASP é retornada como resultado.

Algoritmo 1 Pseudocódigo genérico do GRASP

```
1:  GRASP ( ListSize, MaxIter, RandomSeed )
2:  MIN ← ∞;
3:  Para 0 até maxIter faça
4:     x ← ConstruçãoGrasp();
5:     x ← BuscaLocal(x);
6:     Se x é viável e f(x) < MIN então
7:        x* ← x ;
8:        MIN ← f(x) ;
9:     Fim-se;
10: Fim-para;
11: Retorne x*;
```

3.1. Fase de construção

A solução inicial é obtida a partir de uma heurística construtiva que, a cada iteração, insere um novo elemento (tarefa) à solução corrente. Para isso, a fase de construção

GRASP apresenta o α como parâmetro de entrada, pertencente ao intervalo $0 \leq \alpha \leq 1$, que representa a porcentagem relativa ao nível de aleatoriedade na escolha dos elementos a serem inseridos na solução. Assim, dado um conjunto *C* de possíveis candidatos a serem inseridos em uma dada iteração, inicialmente as tarefas são ordenadas topologicamente, utilizando-se da aleatoriedade quando há mais de uma tarefa disponível para ser selecionada na posição i da lista ordenada e, a partir do valor de α, constrói-se a chamada Lista Restrita de Candidatos (LRC), que agrupa os $\lceil(\alpha \times |C|)\rceil$ melhores elementos de *C*.

Quanto mais próximos de zero forem os valores de α, a LRC se torna mais restrita, e consequentemente, leva a uma baixa diversidade de soluções construídas. Já para a escolha de valores de α mais próximos de 1, produz-se um comportamento mais aleatório, o que é importante para a diversificação nas soluções construídas, mas, por outro lado, muitas soluções tendem a apresentar qualidade inferior, tornando mais lento o processo de busca local.

Após finalizar o procedimento anterior, calcula-se um limite inferior para o valor da solução, utilizando o máximo entre $\frac{\sum_{i \in T} t_i}{m}$ e $max_{i \in T} t_i$. A primeira expressão apresenta o valor da distribuição das tarefas mais uniforme possível, e a segunda retorna o custo da tarefa que possui o maior custo.

A partir deste limite inferior, cada tarefa é atribuída a cada máquina de acordo com a ordem topológica, ou seja, essa alocação é realizada enquanto a carga da máquina for menor que o limite inferior da solução calculado. Quando esse valor for extrapolado, escolhe-se a decisão que mais aproxima a carga da máquina do limite inferior calculado. Isso faz com que, ao final da atribuição, todas as tarefas pendentes sejam atribuídas à última máquina.

A Figura 5 (a) apresenta uma solução viável para o problema considerando $m = 4$ e nove tarefas, isto é, $|T| = 9$, e a Figura 5 (b) ilustra a respectiva solução sob uma perspectiva real de operação. A solução construída inicialmente possui $z = 13$. A aplicação da busca local no grafo de precedência possibilitou uma melhora nessa solução, obtendo o valor de $z = 12$, conforme ilustrado na Figura 6.

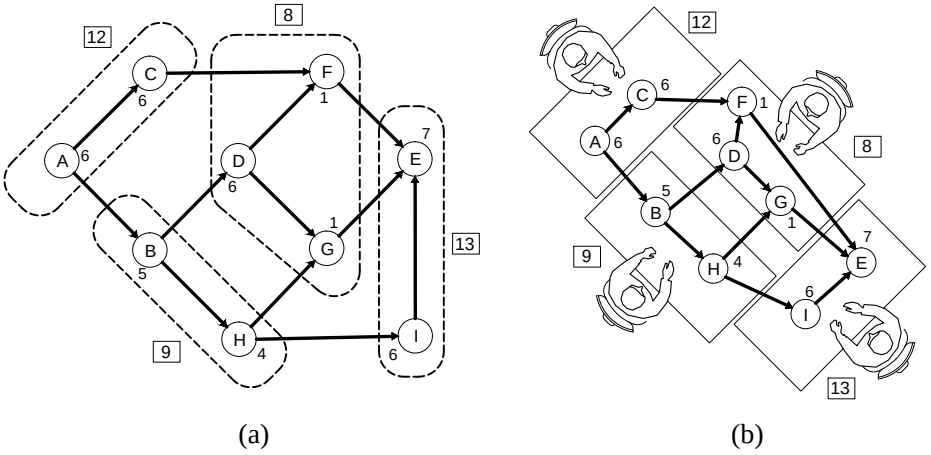

Figura 5. Solução inicial para um grafo de precedência com $m = 4$ e $z = 13$.

3.2. Busca Local

Após gerada uma solução viável inicial, inicia-se o processo de exploração da vizinhança pela busca local. Considerando a máquina que possui o maior tempo de execução da linha de montagem, obtém-se a máquina $m*$ que determina o tempo de ciclo, e então busca-se retirar uma das tarefas t de $m*$, respeitando a ordem de precedência, para atribuir a cada outra máquina, até que encontre um melhor tempo de ciclo. Nesta etapa, utiliza-se o método de melhoramento iterativo *First Improvement Method*, fazendo com que a primeira solução vizinha que melhore a solução seja escolhida como a próxima solução.

O objetivo deste processo é percorrer a vizinhança da solução e realizar, quando possível, trocas entre uma das tarefas da máquina que determina o tempo de ciclo para outra máquina, buscando obter melhora no tempo de ciclo. Quando há mais de uma máquina definindo o tempo de ciclo, de modo a evitar que o algoritmo fique em *loop*, durante a busca local, considera-se que a solução vizinha melhorou a solução corrente se o tempo de ciclo daquela solução vizinha for melhor; ou se o tempo de ciclo da solução vizinha for igual ao da solução corrente, mas a quantidade de máquinas $m*$ for menor.

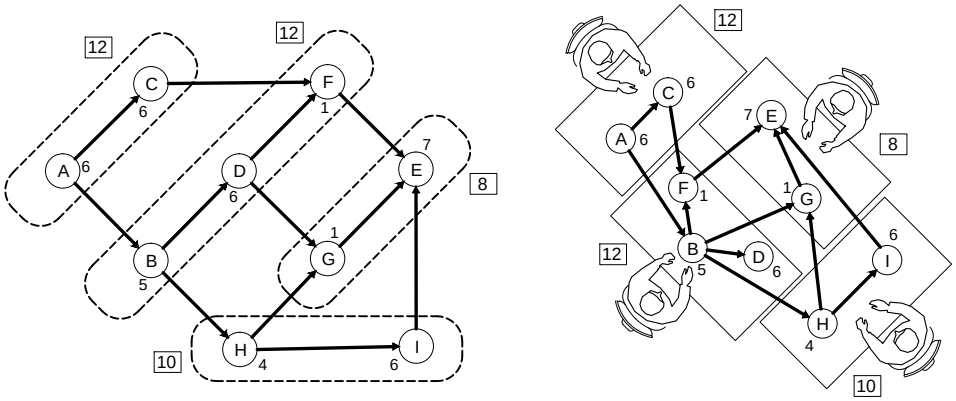

Figura 6. Solução após aplicação da busca local. Tempo de ciclo passou de $z = 13$ para $z = 12$.

4. Experimentos Computacionais

Para ilustrar a eficácia do algoritmo GRASP descrito neste artigo, considera-se um conjunto instâncias tradicionais do SALBP-2 que podem ser obtidas, juntamente com informações sobre suas melhores soluções conhecidas, a partir de um site especialmente dedicado a todos os tipos de problemas de balanceamento de linha de montagem, mantido por Armin Scholl [1]. Cada instância consiste em um grafo de precedência G e um dado número m de máquinas (estações de trabalho). O conjunto de referência consiste em instâncias com um número de tarefas entre 28 e 110 e número de grafos de precedência entre 8 e 9.

O algoritmo GRASP foi implementado em linguagem C e compilado com o compilador GCC. Os testes foram executados em um computador com processador Intel Core i5-2450M com 2.50Ghz, 6GB de memória RAM e sistema operacional Windows 8.1 Pro.

[1] http://alb.mansci.de/

Em relação à parametrização do algoritmo GRASP, o parâmetro α foi definido como 0,8 e o número de iterações como 200000. O algoritmo GRASP foi executado vinte vezes, para cada instância.

Tabela 1. Resultados Computacionais - Parte 1

Grafo	m	BKS	Melhor	Média	t(s)	Desvio
Gunther (35)	6	84	84	84,00	384,82	0,00%
	7	72	72	72,00	447,04	0,00%
	8	63	63	63,00	391,51	0,00%
	9	54	54	54,00	392,52	0,00%
	10	50	50	50,00	373,40	0,00%
	11	48	48	48,00	428,87	0,00%
	12	44	44	44,00	402,59	0,00%
	13	42	42	42,00	380,61	0,00%
	14	40	40	40,00	390,47	0,00%
	15	40	40	40,00	386,73	0,00%
Hahn (53)	3	4787	4787	4787,00	516,26	0,00%
	4	3677	3677	3679,20	542,54	0,00%
	5	2823	2823	2823,00	459,49	0,00%
	6	2400	2400	2400,00	519,86	0,00%
	7	2336	2336	2336,00	483,44	0,00%
	8	1907	1910	1910,00	466,67	0,16%
	9	1827	1866	1866,00	470,83	2,13%
	10	1775	1866	1866,00	497,28	5,13%
Barthold (148)	3	1878	1878	1878,00	1678,69	0,00%
	4	1409	1409	1409,00	1911,81	0,00%
	5	1127	1127	1127,00	2185,75	0,00%
	6	939	939	939,25	2220,84	0,00%
	7	805	805	805,35	2311,64	0,00%
	8	705	705	705,00	2411,23	0,00%
	9	626	627	627,00	2475,40	0,16%
	10	564	564	564,85	2302,42	0,00%
	11	513	513	513,30	2391,00	0,00%
	12	470	470	470,95	2491,10	0,00%
	13	434	435	435,15	2507,40	0,23%
	14	403	404	404,75	2099,80	0,25%
	15	383	383	383,00	3304,38	0,00%
Média						**0,26%**

A Tabela 1 e Tabela 2 apresentam os resultados da seguinte forma: A primeira e segunda coluna apresentam o nome da instância utilizada (e o tamanho do grafo de precedência), e a quantidade de estações de trabalho (m), respectivamente; A coluna BKS (*best known solution*) apresenta a melhor solução conhecida; A coluna Melhor apresenta a melhor solução encontrada pelo algoritmo GRASP em 20 execuções; A média das 20 solu-

ções geradas para cada instância pelo algoritmo GRASP proposto é relacionada na coluna Média; O desvio percentual entre Melhor e BKS é apresentado na coluna Desvio.

O algoritmo GRASP produziu resultados com desvio relativo médio de $0,26\%$ em relação ao BKS para as instâncias apresentadas na Tabela 1. Na Tabela 2, os resultados gerados possuem o valor $2,31\%$ de desvio médio em relação ao BKS. O aumento do desvio, em relação aos resultados apresentados na Tabela 1, é justificado pela complexidade da instância, que possui um grafo de precedência de tamanho 297, enquanto que as demais instâncias possuem esse valor entre 35 e 148.

Tabela 2. Resultados Computacionais - Parte 2

Grafo	m	BKS	Melhor	Média	t(s)	Desvio
Scholl (297)	25	2787	2819	2828,45	5560,27	1,15%
	26	2680	2710	2715,75	5994,80	1,12%
	27	2580	2608	2615,00	7818,06	1,09%
	28	2488	2515	2519,45	8618,69	1,09%
	29	2402	2427	2432,85	9836,09	1,04%
	30	2322	2350	2359,35	11360,92	1,21%
	31	2247	2281	2285,85	9994,28	1,51%
	32	2177	2210	2213,80	10003,78	1,52%
	33	2111	2141	2151,35	10692,81	1,42%
	34	2049	2088	2091,50	8701,78	1,90%
	35	1991	2042	2042,00	6733,14	2,56%
	36	1935	1990	2002,30	6647,88	2,84%
	37	1883	1932	1946,25	6557,36	2,60%
	38	1834	1877	1890,30	6192,44	2,34%
	39	1787	1826	1838,15	6047,86	2,18%
	40	1742	1784	1789,00	6755,82	2,41%
	41	1700	1740	1743,65	7267,19	2,35%
	42	1659	1702	1708,05	7218,60	2,59%
	43	1621	1686	1687,85	7562,08	4,01%
	44	1584	1637	1654,95	7548,90	3,35%
	45	1549	1601	1614,60	6723,73	3,36%
	46	1515	1558	1572,65	6766,25	2,84%
	47	1483	1532	1537,70	6041,00	3,30%
	48	1452	1518	1529,00	6127,37	4,55%
	49	1427	1485	1492,25	7739,46	4,06%
	50	1394	1451	1460,95	6500,81	4,09%
	51	1386	1413	1423,50	8301,56	1,95%
	52	1386	1390	1395,90	10077,67	0,29%
Média						**2,31%**

5. Conclusões

Este trabalho apresentou uma implementação do algoritmo GRASP para o SALBP-2. A fase construtiva deste método gera uma solução viável, um elemento por vez, sendo

que a próxima tarefa a ser atribuída a uma máquina é escolhida aleatoriamente da lista restrita de candidatos (LRC). A LRC é ordenada topologicamente, e todos os candidatos possuem a mesma probabilidade de serem selecionados. Em seguida, é aplicada uma busca local, que consiste em realizar movimentos nas tarefas da máquina que define o tempo de ciclo, e ao encontrar uma melhor atribuição, definir a solução encontrada como a melhor solução. Esse processo é repetido até que o número de iterações seja atingido.

O algoritmo GRASP proposto foi testado em um conjunto de instâncias padrões e comparado com as melhores soluções conhecidas (Blum, 2011). Os resultados computacionais mostram que o algoritmo GRASP atinge os mesmos valores das melhores soluções conhecidas (BKS) ou valores próximos a esses para todas as instâncias testadas.

Agradecimentos

Dayan de Castro Bissoli agradece a Bolsa de Doutorado da Fundação de Amparo à Pesquisa do Espírito Santo (FAPES).

REFERÊNCIAS

Askin, R. G. e Standridge, C. R. (1993). *Modeling and analysis of manufacturing systems*. Wiley, New York.

Baybars, I. (1986). A survey on exact algorithms for the simple assembly line balancing problem. *Management Science*, 32:909–932.

Becker, C. e Scholl, A. (2006). A survey on problems and methods in generalized assembly line balancing. *European Journal of Operational Research*, 168:694–715.

Blum, C. (2011). Iterative beam search for simple assembly line balancing with a fixed number of work stations. *Statistics And Operations Research Transactions*, 35(2):145–164.

Boysen, N., Fliedner, M. e Scholl, A. (2007). A classification of assembly line balancing problems. *European Journal of Operational Research*, 183:674–693.

Feo, T. A. e Resende, M. G. C. (1995). Greedy randomized adaptive search procedures. *Journal of Global Optimization*, 6:109–133.

Klein, R. e Scholl, A. (1999). Maximizing the production rate in simple assembly line balancing - a branch and bound procedure. *European Journal of Operational Research*, 91:367–385.

Peinado, J. e Graeml, A. R. (2007). *Administração da produção: operações industriais e de serviços*. UnicenP, Curitiba.

Scholl, A. (1999). *Balancing and sequencing assembly lines*. Physica,, Heidelberg, 2nd edition.

Scholl, A. e Becker, C. (2006). State-of-the-art exact and heuristic solution procedures for simple assembly line balancing. *European Journal of Operational Research*, 168:666–693.

Silva, G. G. M. P., Tubino, D. F. e Seibel, S. (2015). Linhas de montagem: revisão da literatura e oportunidades para pesquisas futuras. *Production Journal*, 25(1):170–182.