Um algoritmo heurístico para o problema de bin packing com conflitos

Renatha Capua
Instituto de Computação – Universidade Federal Fluminense (UFF)
Rua Passo da Pátria, 156 – CEP 24210-240 – Niterói (RJ), Brasil
rcapua@ic.uff.br

Yuri Frota
Instituto de Computação – Universidade Federal Fluminense (UFF)
Rua Passo da Pátria, 156 – CEP 24210-240 – Niterói (RJ), Brasil
yuri@ic.uff.br

Thibaut Vidal
Dpto de Informática – Pontifícia Universidade Católica do Rio de Janeiro (PUC-Rio)
Rua Marquês de São Vicente, 225, CEP 22451-900, Rio de Janeiro (RJ), Brasil
vidalt@inf.puc-rio.br

Luiz Satoru Ochi
Instituto de Computação – Universidade Federal Fluminense (UFF)
Rua Passo da Pátria, 156 – CEP 24210-240 – Niterói (RJ), Brasil
satoru@ic.uff.br

RESUMO

O Problema de Bin Packing com Conflitos (PBPC) é uma generalização do Problema de Coloração de Vértices e do Problema de Bin Packing. O PBPC consiste em alocar um conjunto de itens a um conjunto de bins idênticos minimizando o número de bins utilizados e respeitando as restrições de conflito entre os itens. Para resolver este problema, foi desenvolvido um método baseado na metaheurística Iterated Local Search. A abordagem proposta utiliza ainda um algoritmo de Busca em Vizinhança Larga que é capaz de encontrar em tempo quadrático a melhor solução em um conjunto exponencial de vizinhos. Testes comparativos utilizando instâncias da literatura demonstram a boa performance do método.

PALAVRAS CHAVE. Bin Packing com Conflitos, Iterated Local Search, Busca em Vizinhança Larga.

Área Principal: MH – Metaheurísticas.

ABSTRACT

The Bin Packing Problem with Conflicts (BPPC) is a generalization of the Vertex Coloring Problem and the Bin Packing Problem. The BPPC consists in assigning a set of items into a set of identical bins, while minimizing the number of bins and respecting conflict constraints between items. To solve this problem, this paper introduces a metaheuristic based on Iterated Local Search. The proposed approach uses a Large Neighborhood Search, able to find, in quadratic time, a best solution in an exponential set of neighbors. Experimental analyses were conducted on instances from the literature, demonstrating the good perfomance of the method.

KEYWORDS. Bin Packing with Conflicts, Iterated Local Search, Large Neighborhood Search.

Main Area: MH – Metaheuristics.

1. Introdução

Diversos problemas reais, como por exemplo, o Problema de transporte de produtos perigosos que combinam materiais tóxicos, explosivos, inflamáveis e corrosivos, possuem restrições quanto a itens que não podem ser alocados juntos. Neste contexto surge um importante problema a ser resolvido, o *Problema de Bin Packing com Conflitos (PBPC)*, onde, um conjunto de itens deve ser alocado a um número mínimo de bins idênticos de uma dada capacidade, respeitando as restrições de conflito entre os itens, que não podem ser alocados ao mesmo bin.

Esse é um problema desafiador na área de otimização, uma vez que combina dois outros problemas importantes e bem estudados na literatura, o *Problema de Coloração de Vértices (PCV)* e o *Problema de Bin Packing (PBP)*. Algumas metodologias desenvolvidas para um problema podem não funcionar apropriadamente para o outro e vice-versa. Portanto, encontrar métodos que resultem em boas soluções para uma ampla variedade de instâncias do PBPC é uma tarefa díficil. Métodos eficientes de resolução para o PBPC podem contribuir ainda, para algoritmos eficientes para estes dois outros problemas.

O PBPC é NP-difícil (Muritiba et al., 2010), portanto não se conhece um algoritmo em tempo polinomial para resolvê-lo. Uma alternativa para sua resolução é o uso de métodos heurísticos, que possibilitam a obtenção de soluções de boa qualidade em um tempo computacional razoável. Neste trabalho é proposta uma abordagem baseada na metaheurística *Iterated Local Search (ILS)*. Segundo Lourenço et al. (2010), o ILS é um método de resolução de simples entendimento e implementação, exigindo poucos parâmetros, porém robusto e efetivo na resolução de diversos problemas, tais como (Stützle, 2006), (Masson et al., 2013), (Ruiz and Stützle, 2007) e (Subramanian et al., 2013). Neste trabalho, o ILS é combinado com um método de Busca em Vizinhança Larga chamado *Ejections Chains (EC)*. Vários outros elementos foram utilizados para obter uma busca eficiente, incluindo estruturas de avaliação dos movimentos em tempo constante $\mathcal{O}(1)$. A heurística resultante produz soluções de alta qualidade para instâncias existentes na literatura.

A principal contribuição deste trabalho é o desenvolvimento de uma nova heurística para resolver o PBPC utilizando técnicas ainda não exploradas para este problema, como ILS e *Ejection Chains*. O restante deste trabalho está assim dividido: a Seção 2 apresenta uma descrição detalhada do problema e uma revisão bibliográfica dos trabalhos existentes na literatura. Na Seção 3, encontram-se os algoritmos heurísticos propostos. A Seção 4 mostra os resultados obtidos pelas heurísticas. Na Seção 5 encontram-se as conclusões e as propostas de trabalhos futuros.

2. Definição do Problema e Aplicações

Dado um conjunto finito de bins de capacidade idêntica Q, um conjunto de itens $\mathcal{V} = \{1, \ldots, n\}$ com pesos $q_1, \ldots, q_n$ e um grafo $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ de conflitos entre os itens, o objetivo do Problema de *Bin Packing* com Conflitos (PBPC) é alocar cada item a um bin minimizando o número de bins utilizados. Esta alocação deve respeitar a capacidade máxima dos bins e a restrição de que dois itens em conflito não podem ser alocados ao mesmo bin.

O PBPC pode ser modelado como um problema de programação linear inteira. Uma formulação, proposta por Gendreau et al. (2004), é apresentada a seguir. Considere a variável binária y_k igual a 1 se o bin k está sendo utilizado e 0, caso contrário. A variável binária x_{ik} indica se o item i está alocado ao bin k ($x_{ik} = 1$) ou não ($x_{ik} = 0$).

Função Objetivo:

$$\text{Minimizar } z = \sum_{k=1}^{n} y_k \tag{1}$$

Sujeito a:

$$\sum_{i=1}^{n} q_i x_{ik} \leq Q y_k \qquad k = 1, \ldots, n \tag{2}$$

$$\sum_{k=1}^{n} x_{ik} = 1 \qquad i = 1, \ldots, n \tag{3}$$

$$x_{ik} + x_{jk} \leq 1 \qquad (i,j) \in \mathcal{E}, k = 1, \ldots, n \tag{4}$$

$$y_k \in \{0,1\} \qquad k = 1, \ldots, n \tag{5}$$

$$x_{ik} \in \{0,1\} \qquad i = 1, \ldots, n, k = 1, \ldots, n \tag{6}$$

O objetivo definido por (1) é minimizar o número de bins utilizados. O conjunto de restrições (2) garante que a capacidade máxima de cada bin seja respeitada. As restrições (3) asseguram que cada item só pode ser alocado a um único bin. O conjunto de restrições (4) garante que dois itens pertencentes ao conjunto de arestas do grafo de conflitos não podem ser alocados no mesmo bin. As restrições (5) e (6) indicam o domínio das variáveis envolvidas na modelagem.

Se o conjunto $\mathcal{E}$ de arestas do grafo de conflitos for vazio, o PBPC é equivalente ao *Problema de Bin Packing Clássico (PBP)*. Se a capacidade de cada bin for infinita, ou seja, maior do que soma do peso de todos os itens, obtém-se o *Problema de Coloração de Vértices (PCV)*. Deste modo, isso significa que o Problema de Bin Packing com Conflitos é NP-difícil mesmo se $\mathcal{E} = \emptyset$ ou se a capacidade de cada bin for infinita. Quando $q_i = 1$ para cada $v_i \in \mathcal{V}$, o PBPC generaliza outro problema chamado *Bounded Vertex Coloring Problem (BVCP)*.

O PBPC possui várias aplicações, uma delas é o problema de *examination scheduling* (Laporte and Desrochers, 1984), onde exames devem ser alocados ao menor número possível de períodos de tempo respeitando a capacidade das salas e, sem conflito com outros exames que possuam estudantes em comum. O trabalho de Jansen (1999) apresenta outra aplicação para o PBPC, a alocação de processos e o balanceamento de carga na área de computação paralela. Neste problema, alguns processos não podem ser executados no mesmo processador devido a questões de aumento de performance ou de tolerância a falhas. Outras aplicações para o PBPC podem ser encontradas na área de transporte e logística (Hamdi-Dhaoui et al., 2014; Minh et al., 2013), neste tipo de problema a solução gerada deve levar em consideração que alguns produtos não podem ser dispostos juntos no mesmo veículo para entrega (Chrisofides et al., 1979), como na entrega de materiais perigosos.

Existem diversos trabalhos na literatura que tratam o PBPC. Nos trabalhos desenvolvidos por Jansen and Öhring (1997) e Jansen (1999) são apresentados algoritmos para casos especiais de grafos de conflitos. Gendreau et al. (2004) avaliaram seis heurísticas para obter limites superiores para o problema e duas estratégias para obter limites inferiores. No trabalho de Muritiba et al. (2010), os autores desenvolveram uma heurística baseada no Algoritmo Genético (Holland, 1975) utilizando como busca local a meta-heurística Busca Tabu (Glover, 1986). Além disso, os autores implementaram um algoritmo de *Branch-and-Price* utilizando a formulação do Problema de Recobrimento de Conjuntos (*Set Covering*). Um algoritmo de *Branch-and-price* também foi desenvolvido por Elhedhli et al. (2011). Mais recentemente, Sadykov and Vanderbeck (2013) desenvolveram um

Branch-and-price para o PBPC. Para resolver o problema de *pricing*, dois casos são considerados de acordo com o tipo de grafo pertencente as intâncias: quando o grafo é de intervalo, foi utlilizado um algoritmo de programação dinâmica. Quando o grafo não possui nenhuma estrutura especial, o subproblema de pricing é o Problema da Mochila com Conflitos, que é NP-difícil, sendo resolvido por um *Branch-and-bound*. Alguns outros trabalhos sobre o PBPC podem ser vistos em (Epstein and Levin, 2008), (Khanafer et al., 2010) e (Maiza and Radjef, 2011).

Com o objetivo de solucionar o problema, propõe-se um algoritmo que utiliza a metaheurística *Iterated Local Search* combinada com mecânismos adicionais de aprimoramento. Na próxima seção é apresentada a heurística proposta para o PBPC.

3. Método Proposto

Neste trabalho, a metaheurística *Iterated Local Search (ILS)* (Lourenço et al., 2010) foi adaptada ao problema tratado como ilustrado no Algoritmo 1. Inicialmente, como apresentado na linha 1, uma solução é gerada pelo procedimento de contrução. Essa solução é viável porém, o número de bins utilizados na realização da alocação não é necessariamente o menor possível.

Para diminuir o número de bins na solução corrente, o método irá iterativamente remover um bin (linha 4), alocando seus itens em outros bins escolhidos aleatoriamente. Essa realocação poderá gerar conflitos com os itens já alocados e excesso na capacidade dos bins restantes. A partir daí, para tentar resolver essas inviabilidades que podem ter sido produzidas na solução, são aplicadas repetidas iterações de Busca Local (BL), do Ejection Chains (EC) e do procedimento de perturbação, linhas 7 a 14.

O algoritmo termina quando nenhuma solução viável é encontrada após It_{MAX} iterações da perturbação ou se um limite inferior K_{LB} for atingido. Nas subseções a seguir, cada um dos componentes do algoritmo é detalhado.

Algorithm 1: ILS para o PBPC $(K_{\text{LB}}, It_{\text{BL}})$

1: $\mathcal{S} \leftarrow$ ConstroiSoluçãoInicial();
2: $K \leftarrow$ Número de bins em $\mathcal{S}$;
3: **enquanto** $K \geq K_{\text{LB}}$ **and** $\mathcal{S}$ é viável **faça**
4: ReduzNúmeroBins($\mathcal{S}$);
5: $K \leftarrow K - 1$;
6: $It_{\text{PERT}} \leftarrow 0$;
7: **enquanto** $It_{\text{PERT}} \leq It_{\text{MAX}}$ **and** $\mathcal{S}$ não é viável **faça**
8: $\mathcal{S} \leftarrow$ BuscaLocal($\mathcal{S}$);
9: $\mathcal{S} \leftarrow$ EjectionChains($\mathcal{S}$);
10: **se** It_{BL} sucessivas BL e EC sem melhora **então**
11: $\mathcal{S} \leftarrow$ Perturbação($\mathcal{S}$);
12: $It_{\text{PERT}} \leftarrow It_{\text{PERT}} + 1$;
13: $It_{\text{BL}} \leftarrow 0$;
14: **fim se**
15: **fim enquanto**
16: **fim enquanto**

3.1. Heurística Construtiva

Algoritmos Construtivos são heurísticas utilizadas para construir uma solução. Foi implementado um algoritmo construtivo proposto por Gendreau et al. (2004), que é uma adaptação do algoritmo de Coffman et al. (1984) chamado *First Fit Decreasing (FFD)*. O procedimento pode ser descrito basicamente nas duas etapas abaixo:

1. Os itens são ordenados em ordem decrescente de acordo com seu peso.
2. Cada item, na ordem definida pela lista ordenada, é alocado ao primeiro bin pertencente a lista de bins disponíveis com espaço residual suficiente e, para o qual não há itens em conflito já alocados.

Esse algoritmo constroi uma solução inicial com K bins.

3.2. Métodos de Busca Local

Os algoritmos de Busca Local (BL) tem por objetivo melhorar a qualidade da solução explorando sua vizinhança na busca por uma solução melhor do que a solução corrente.

Três vizinhanças clássicas são utilizadas na Busca Local: SWAP, SWAP2-1 e RELOCATE. Na vizinhança SWAP é realizada a troca entre um – ou dois – itens de um bin para outro bin diferente. Já na vizinhança RELOCATE, um item é movido para outro bin.

A avaliação dos movimentos se dá para cada par de bins como descrito a seguir. Inicialmente, é definida uma ordem aleatória dos bins. Então, para cada par diferente de bins, a vizinhança obtida utilizando-se os itens pertencentes a esses dois bins é analisada. O melhor movimento obtido entre todos os itens destes dois bins avaliados, e que necessariamente diminui o valor da função objetivo, é então aplicado.

Durante a primeira iteração da BL, movimentos com custo zero também são aceitos. Esses movimentos não melhoram o objetivo do problema mas contribuem para diversificar a busca no espaço de soluções. A Busca Local termina quando nenhuma solução com custo melhor é encontrada.

Para executar todas essas vizinhanças de um modo mais eficiente e rápido, foram desenvolvidas estruturas de dados avançadas. Para cada item i e bin b, o custo de inserção corrente é armazenado e atualizado durante a busca. Baseado nesta informação, os movimentos são avaliados sem a necessidade de enumerar os itens na nova solução para avaliar a existência de conflitos. Um fator de correção é adicionado já que várias realocações combinadas de itens são executadas. Isso pode ser feito em tempo constante. Na média, essas estruturas são capazes de avaliar os movimentos em tempo amortizado $\mathcal{O}(1)$.

3.3. Ejection Chains

As vizinhanças locais são complementadas com uma abordagem da heurística *Ejection Chains (EC)* Glover (1996), obtida pela resolução do subproblema de caminho mínimo. A EC proposta é capaz de encontrar, em tempo polinomial, uma solução de melhor custo em uma vizinhança de tamanho exponencial. Esse procedimento foi aplicado ao PBPC como explicado a seguir.

Primeiro, foi escolhida uma ordem aleatória Π para os bins pertencentes a solução corrente. Então, considere um grafo auxiliar $\mathcal{H} = (\mathcal{V}', \mathcal{A})$ com $\mathcal{V}' = \mathcal{V} \cup \mathcal{V}^{\text{ZERO}}$. Todos os items estão contidos

em $\mathcal{V}$, enquanto, $\mathcal{V}^{\text{ZERO}}$ contém um nó v_k^{ZERO} para cada bin k na solução corrente e um nó v^{SOURCE} que representa a fonte (início) do caminho mínimo. Além disso, considere $\mathcal{K}(i)$ como o bin que contém o vértice i na solução corrente. O conjunto $\mathcal{A}$ contém um arco (i, j) para cada par de nós $i \in \mathcal{V}'$ e $j \in \mathcal{V}'$ tal que o bin $\mathcal{K}(i)$ precede o bin $\mathcal{K}(j)$ na ordem Π, e um arco (v^{SOURCE}, j) para qualquer $j \in \mathcal{V}'$. Os custos dos arcos são definidos do seguinte modo:

$$c_{ij} = \begin{cases} \text{diferença de custo do bin } \mathcal{K}(j) \text{ quando substitui-se o item } j \text{ pelo item } i & \text{se } i \in \mathcal{V} \text{ e } j \in \mathcal{V}, \\ \text{diferença de custo do bin } \mathcal{K}(j) \text{ quando remove-se o item } j & \text{se } i \in \mathcal{V}^{\text{ZERO}} \cup v^{\text{SOURCE}} \text{ e } j \in \mathcal{V}, \\ \text{diferença de custo do bin } \mathcal{K}(j) \text{ quando insere-se o item } i & \text{se } i \in \mathcal{V} \text{ e } j \in \mathcal{V}^{\text{ZERO}}. \end{cases}$$

O caminho mínimo no grafo da fonte até qualquer nó em $\mathcal{V}^{\text{ZERO}}$ é equivalente a uma sequencia combinada de movimentos de inserção e remoção de itens. O movimento é aplicado se o custo é estritamente negativo. A Figura 1 ilustra uma possível solução obtida pela *Ejection Chains* em um grafo com 3 bins e 5 itens. Neste caso, o item 3 é realocado do bin 2 para o bin 3 e o item 1 é realocado do bin 3 para o bin 1, gerando uma sequencia de movimentos combinados. Desta maneira, longas sequencias de itens podem ser realocados, isto leva a uma vizinhança de tamanho exponencial que é explorada aqui, em tempo quadrático.

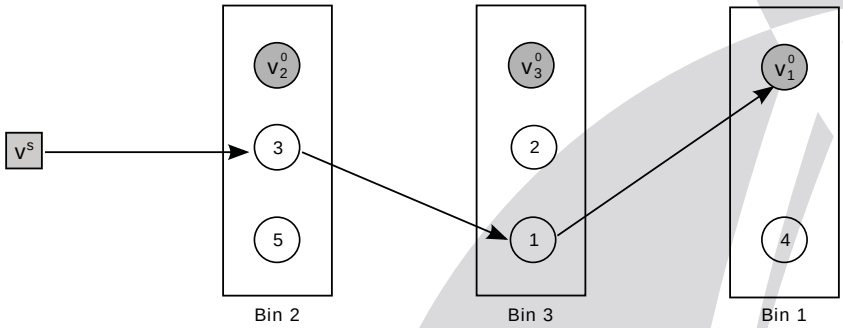

Figura 1: Uma possível solução obtida pela Ejection Chain.

3.4. Procedimento de Perturbação

O algoritmo de perturbação é aplicado se a solução permanecer inviável após It_{BL} iterações consecutivas da Busca Local e do *Ejection Chains*. Durante a perturbação, n_{PERT} itens são realocados aleatoriamente, escolhendo prioritariamente, aqueles itens com conflitos ou pertencentes a bins com violação na capacidade.

4. Experimentos Computacionais

As heurísticas propostas foram implementadas na linguagem C++ utilizando o compilador g++ versão 4.8.2. Os códigos foram executados em um computador Intel Core 2 Quad com o processador Q6600 2.40GHz e 4GB de memória RAM. O sistema operacional utilizado foi o Linux Ubuntu 64 bits.

4.1. Instâncias

A performance do método foi avaliada por seis classes de instâncias do PBPC existentes na literatura: t, u, ta, da, ua e da. As classes de instâncias (t) e (u) foram criadas por Muritiba et al. (2010) de acordo com as instâncias propostas no trabalho de Gendreau et al. (2004). As classes de instâncias (d), (ta), (ua) e (da) foram propostas por Sadykov and Vanderbeck (2013). Todas as classes totalizam 2060 instâncias com uma quantidade de itens variando de 60 a 1000. Algumas intâncias foram obtidas a partir das clássicas instâncias *triplet* (t) e *uniforme* (u) de Falkenauer (1996) para o Problema de *Bin Packing* (PBP). Com relação ao grafo de conflitos, os conjuntos (t), (u) e (d) utilizam grafos de intervalos, enquantos os conjuntos (ta), (ua) e (da) apresentam grafos gerados aleatoriamente (geral).

Devido as restrições de conflito e capacidade, na maioria das instâncias, as soluções ótimas conhecidas incluem poucos itens por bin, com cerca de $1,2$ a 3 itens em média. Isto favorece os métodos de programação matemática baseados em formulações do *set covering* ou *set partitioning*.

4.2. Resultados do ILS

Após experimentos de calibração de parâmetros preliminares, os seguintes valores foram escolhidos: $n_{\text{PERT}} = 5$, e $(It_{\text{LS}}, It_{\text{MAX}}) = (100, 30)$. O limite inferior utilizado para K_{LB} em cada instância foi a soma do seu número de itens dividida pela capacidade dos bins. Essa configuração de testes foi aqui chamada de ILS-*Full*. Para examinar o impacto da busca na qualidade das soluções, foram reportados os resultados de uma configuração simplificada chamada de ILS-*Fast*, onde a vizinhança SWAP2-1 não foi utilizada e $It_{\text{MAX}} = 10$.

As Tabelas 1 e 2 exibem o resultado dos métodos, comparados com a metaheurística de Muritiba et al. (2010) (fase 2, antes da resolução exata), quando os resultados são disponíveis. Para estas instâncias, várias soluções ótimas são fornecidas pelo trabalho de Sadykov and Vanderbeck (2013). Cada coluna exibe o conjunto de classes de instâncias, o Gap (%) para a melhor solução conhecida (BKS), o tempo de CPU (T) dado em segundos e o número de soluções ótimas alcançadas/conhecidas (Num. opt.). Os valores em negrito destacam os melhores resultados obtidos para um determinado conjunto de instâncias.

Pelos resultados, pode-se concluir que o ILS proposto fornece limites superiores de alta qualidade, com um gap médio total de $0,23\%$ para a solução ótima nas intâncias de Muritiba e, com um tempo total médio de 35 segundos. As melhores soluções são geradas com nossa simples abordagem uma vez que o método de Muritiba obtém um gap total médio de $0,30\%$, com um tempo médio de 55 segundos. Para as instâncias de Sadykov and Vanderbeck (2013), nenhuma heurística está disponível para comparação, no entanto, o método também alcança um gap médio total pequeno, de $0,68\%$, em relação as melhores soluções da literatura.

Visivelmente, os grafos de conflito geral levam a um aumento na dificuldade de resolução do problema, como observado em Sadykov and Vanderbeck (2013). Outras classes de instâncias com grafos de conflito esparso (por exemplo, algumas instâncias em t60 ou t120) são igualmente difíceis de se resolver devido ao modo como as restrições de empacotamento foram geradas: "cortando" os bins inicialmente totalmente preenchidos em itens. Essa característica das instâncias leva a soluções ótimas com bins com a capacidade final igual a zero e, mais difíceis de encontrar.

	Muritiba		ILS-*Fast*			ILS-*Full*		
Instâncias	Gap (%)	T(s)	Gap (%)	T(s)	Num. opt.	Gap (%)	T(s)	Num. opt.
t60	**0,45**	37,51	0,95	0,20	81/100	0,85	0,64	83/100
t120	**0,60**	40,00	1,12	0,60	65/100	0,73	2,29	71/100
t249	0,39	51,85	0,93	2,25	59/100	**0,13**	8,80	88/100
t501	0,21	58,93	0,92	9,40	60/100	**0,03**	33,68	94/100
u120	0,10	22,35	0,10	0,76	95/100	**0,04**	2,49	98/100
u250	0,21	52,11	0,17	3,16	81/100	**0,04**	9,90	95/100
u500	0,20	69,67	0,12	14,67	75/100	**0,03**	40,04	92/100
u1000	0,22	107,81	0,08	76,75	74/100	**0,02**	179,74	92/100

Tabela 1: Comparação das metaheurísticas utilizando o $benchmark$ de Muritiba et al. (2010)

.

	Sadykov	ILS-Fast			ILS-Full		
Instâncias	Não opt.	Gap (%)	T(s)	Num. opt.	Gap (%)	T(s)	Num. opt.
ta60	0	0,42	0,24	82/90	**0,27**	0,83	85/90
ta120	0	1,63	1,00	42/90	**1,20**	3,95	49/90
ta249	6	2,07	3,96	15/88	**1,05**	18,55	39/88
ta501	25	2,07	20,31	2/70	**0,61**	88,82	43/70
ua120	0	0,71	1,20	58/90	**0,51**	4,32	67/90
ua250	2	0,70	6,29	50/89	**0,54**	21,24	59/89
ua500	8	0,51	31,04	43/82	**0,36**	92,53	62/82
ua1000	8	0,35	165,51	41/82	**0,24**	435,64	68/82
d120	0	**0,00**	0,47	90/90	**0,00**	1,65	90/90
d250	0	**0,00**	1,91	90/90	0,05	6,25	83/90
d500	0	**0,00**	8,42	90/90	0,02	26,59	86/90
da120	23	1,69	0,81	45/67	**1,26**	3,56	50/67
da250	40	1,91	4,40	36/50	**1,44**	19,54	37/50
da500	41	2,33	22,36	37/49	**1,95**	86,26	37/49

Tabela 2: Comparação das metaheurísticas utilizando as instâncias de Sadykov and Vanderbeck (2013).

5. Conclusões e Trabalhos Futuros

Neste trabalho foi apresentada uma heurística para resolver o PBPC. O método proposto é baseado na metaheurística ILS e possui diversos mecanismos de aprimoramento como *Ejection Chains* e movimentos com custo zero. O método desenvolvido se mostrou competitivo em relação a heurística da literatura proposta por Muritiba et al. (2010). Para as instâncias de Muritiba, o ILS proposto fornece limites superiores com um gap médio total de $0,23\%$ em relação as soluções ótimas, enquanto Muritiba et al. (2010) obteve um gap de $0,30\%$ utilizando sua abordagem heurística. Para as instâncias propostas por Sadykov e Vanderbeck, não estão disponíveis nenhum resultado heurístico, mas o método também alcança um bom gap médio de $0,68\%$ em relação aos melhores resultados conhecidos da literatura.

Como trabalhos futuros propõe-se investigar outros tipos de técnicas de Busca em Vizinhança Larga, incluindo combinar as técnicas atuais com um modelo matemático. Além disso, propõe-se a generalização do método proposto para outras variantes do problema.

Referências

Chrisofides, N.; Mingozzi, A. and Toth, P. (1979). The Vehicle Routing Problem. Chrisofides, N.; Mingozzi, A.; Toth, P. and Sandi, C., editores, *Combinatorial Optimization*, p. 315–338. Wiley Chichester, UK.

Coffman, E.G.; Garey, M.R. and Johnson, D.S. (1984). Approximation Algorithms for Bin-Packing – An Updated Survey. Ausiello, G.; Lucertini, M. and Serafini, P., editores, *Algorithm Design for Computer System Design*, v. 284 de *International Centre for Mechanical Sciences*, p. 49–106. Springer Vienna.

Elhedhli, S.; Li, L.; Gzara, M. and Naoum-Sawaya, J. (2011). A Branch-and-price Algorithm for the Bin Packing Problem with Conflicts. *INFORMS Journal on Computing*, v. 23, n. 3, p. 404–415.

Epstein, L. and Levin, A. (2008). On Bin Packing with Conflicts. *SIAM Journal on Optimization*, v. 19, n. 3, p. 1270–1298.

Falkenauer, E. (1996). A Hybrid Grouping Genetic Algorithm for Bin Packing. *Journal of Heuristics*, v. 2, n. 1, p. 5–30.

Gendreau, M.; Laporte, G. and Semet, F. (2004). Heuristics and Lower Bounds for the Bin Packing Problem with Conflicts. *Computers & Operations Research*, v. 31, n. 3, p. 347–358.

Glover, F. (1986). Future Paths for Integer Programming and Links to Artificial Intelligence. *Computers & Operations Research*, v. 13, n. 5, p. 533–549.

Glover, F. (1996). Ejection Chains, Reference Structures and Alternating Path Methods for Traveling Salesman Problems. *Discrete Applied Mathematics*, v. 65, n. 1-3, p. 223–253.

Hamdi-Dhaoui, K.; Labadie, N. and Yalaoui, A. (2014). The Bi-objective Two-dimensional Loading Vehicle Routing Problem with Partial Conflicts. *International Journal of Production Research*, v. 52, n. 19, p. 5565–5582.

Holland, J.H. (1975). *Adaptation in Natural and Artificial Systems. An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*. The University of Michigan Press, Ann Arbor, MI.

Jansen, K. (1999). An Approximation Scheme for Bin Packing with Conflicts. *Journal of Combinatorial Optimization*, v. 3, p. 363–377.

Jansen, K. and Öhring, S. (1997). Approximation Algorithms for Time Constrained Scheduling. *Information and Computation*, v. 132, n. 2, p. 85–108.

Khanafer, A.; Clautiaux, F. and Talbi, E. (2010). New Lower Bounds for Bin Packing Problems with Conflicts. *European Journal of Operational Research*, v. 206, n. 2, p. 281–288.

Laporte, G. and Desrochers, S. (1984). Examination Timetabling by Computer. *Computers & Operations Research*, v. 11, n. 4, p. 351–360.

Lourenço, H.; Martin, O. and Stützle, T. (2010). Iterated Local Search: Framework and Applications. Gendreau, Michel and Potvin, Jean-Yves, editores, *Handbook of Metaheuristics*, v. 146 de *International Series in Operations Research & Management Science*, p. 363–397. Springer US, 2a edição.

Maiza, M. and Radjef, M. S. (2011). Heuristics for Solving the Bin-Packing Problem with Conflicts. *Applied Mathematical Sciences*, v. 5, n. 35, p. 1739–1752.

Masson, R.; Vidal, T.; Michallet, J.; Penna, P. H. V.; Petrucci, V.; Subramanian, A. and Dubedout, H. (2013). An Iterated Local Search Heuristic for Multi-capacity Bin Packing and Machine Reassignment Problems. *Expert Systems with Applications*, v. 40, n. 13, p. 5266–5275.

Minh, T.; Van Hoai, T. and Nguyet, T. (2013). A Memetic Algorithm for Waste Collection Vehicle Routing Problem with Time Windows and Conflicts. Murgante, B.; Misra, S.; Carlini, M.; Torre, C.; Nguyen, H.; Taniar, D.; Apduhan, B. and Gervasi, O., editores, *Computational Science and Its Applications - ICCSA 2013*, v. 7971 de *Lecture Notes in Computer Science*, p. 485–499. Springer Berlin Heidelberg.

Muritiba, A. E. F.; Iori, M.; Malaguti, E. and Toth, P. (2010). Algorithms for the Bin Packing Problem with Conflicts. *INFORMS Journal on Computing*, v. 22, n. 3, p. 401–415.

Ruiz, R. and Stützle, T. (2007). A Simple and Effective Iterated Greedy Algorithm for the Permutation Flowshop Scheduling Problem. *European Journal of Operational Research*, v. 177, n. 3, p. 2033–2049.

Sadykov, R. and Vanderbeck, F. (2013). Bin Packing with Conflicts: A Generic Branch-and-price Algorithm. *INFORMS Journal on Computing*, v. 25, n. 2, p. 244–255.

Stützle, T. (2006). Iterated Local Search for the Quadratic Assignment Problem. *European Journal of Operational Research*, v. 174, n. 3, p. 1519–1539.

Subramanian, A.; Uchoa, E. and Ochi, L. S. (2013). A Hybrid Algorithm for a Class of Vehicle Routing Problems. *Computers & Operations Research*, v. 40, n. 10, p. 2519–2531.