UNIVERSIDADE FEDERAL FLUMINENSE

RENATHA OLIVA CAPUA

Métodos de Resolução para o Problema de Bin Packing com Conflitos e para o Problema de Bin Packing com Dependências

NITERÓI

2017

UNIVERSIDADE FEDERAL FLUMINENSE

RENATHA OLIVA CAPUA

Métodos de Resolução para o Problema de Bin Packing com Conflitos e para o Problema de Bin Packing com Dependências

Tese de Doutorado submetida ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense como requisito parcial para a obtenção do título de Doutor. Área de concentração: Algoritmos e Otimização.

Orientadores:
Luiz Satoru Ochi
Yuri Abitbol de Menezes Frota

NITERÓI
2017

Métodos de Resolução para o Problema de *Bin Packing* com Conflitos e para o Problema de *Bin Packing* com Dependências

Renatha Oliva Capua

Tese de Doutorado submetida ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense como requisito parcial para a obtenção do título de Doutor em Computação.

Aprovada por:

Prof. Luiz Satoru Ochi, D.Sc. / IC-UFF (Orientador)

Prof. Yuri Abitbol de Menezes Frota, D.Sc. / IC-UFF (Orientador)

Profª. Simone de Lima Martins, D.Sc. / IC-UFF

Prof. Carlos Alberto de Jesus Martinhon, D.Sc. / IC-UFF

Prof. Nelson Maculan Filho, D.Sc. / COPPE-UFRJ

Prof. Haroldo Gambini Santos, D.Sc. / DECOM-UFOP

Niterói - RJ, 01 de setembro de 2017.

Resumo

O Problema de Bin Packing é um dos problemas mais estudados na área de Otimização Combinatória. No entanto, as aplicações práticas do problema acrescentam restrições adicionais à definição clássica dificultando sua resolução utilizando os métodos existentes. Portanto, nesta tese são estudados novos métodos de resoluções para duas variantes do *bin packing*, o Problema de *Bin Packing* com Conflitos (PBPC) e o Problema de *Bin Packing* com Dependências (PBPD). Esses problemas surgem em diversas aplicações reais, tais como no problema de alocação de horários, na alocação de recursos de computação em nuvem, na área de transporte e logística, entre outros.

Para resolver estas duas variantes, foram desenvolvidos métodos heurísticos e exatos. Para o PBPC foi proposta uma abordagem baseada na metaheurística *Iterated Local Search*. Esse método é combinado com várias classes de vizinhanças locais e de larga escala. Foram introduzidos ainda procedimentos de avaliação em $O(1)$ dos movimentos clássicos da busca local, variantes polinomiais do *ejection chains* e da vizinhança *assignment*, além de uma vizinhança adaptativa baseada no problema de cobertura de conjuntos, e finalmente um uso controlado de movimentos de *custo-0* como mecanismo de diversificação da busca. O método desenvolvido produz soluções de alta qualidade em instâncias da literatura. Experimentos computacionais foram realizados para medir a respectiva contribuição de cada vizinhança proposta. Para o PBPD, é apresentado uma definição formal do problema com um modelo matemático de programação linear inteira mista. Um método exato é proposto para resolver o problema, mais especificamente, um algoritmo de *branch-and-price*. Para testar essa abordagem são apresentadas novas instâncias para o problema criadas a partir das instâncias do PBPC existentes na literatura.

Palavras-chave: *Bin Packing* com Conflitos, *Bin Packing* com Dependências, *Iterated Local Search*, Busca em Vizinhança Larga, *Branch-and-price*

Abstract

The Bin Packing Problem is one of the most studied problems in the Combinatorial Optimization area. However, its practical applications add additional constraints to the classical definition. Therefore, this thesis aimed to study new resolution methods for two bin packing variants, the Bin Packing Problem with Conflicts (BPPC) and the Bin Packing Problems with Dependencies (BPPD). These problems have many real applications, such as in timetabling problems, in cloud computing management resources, in transportation and logistics areas, among others.

To solve these two variants, heuristic and exact methods were developed. For the PBPC, an approach based on the metaheuristic Iterated Local Search was proposed. This method is combined with several classes of local and large-scale neighborhoods. We introduce $O(1)$ evaluation procedures for classical local-search moves, polynomial variants of ejection chains and assignment neighborhoods, an adaptive set covering-based neighborhood, and finally a controlled use of 0-cost moves to further diversify the search. The overall method produces solutions of high quality on the classical benchmark instances of the literature. Extensive computational experiments are conducted to measure the respective contribution of each proposed neighborhood. For PBPD, a formal definition of the problem is presented with a mathematical formulation. An exact method is proposed to solve the problem, more specifically a Branch-and-price algorithm. To test this approach, new instances for the problem are created from the PBPC benchmark instances.

Keywords: Bin Packing with Conflicts, Bin Packing with Dependencies, Iterated Local Search, Large Neighborhood Search, Branch-and-price

Glossário

BL	:	Busca Local;
B&B	:	*Branch-and-Bound*;
B&C	:	*Branch-and-Cut*;
B&P	:	*Branch-and-Price*;
BVCP	:	*Bounded Vertex Coloring Problem*;
DFS	:	*Depth First Search* ou Busca em Profundidade;
EC	:	*Ejection Chains*;
FFD	:	*First Fit Decreasing*;
GC	:	Geração de Colunas;
ILS	:	*Iterated Local Search*;
PBP	:	Problema de *Bin Packing*;
PBPC	:	Problema de *Bin Packing* com Conflitos;
PBPD	:	Problema de *Bin Packing* com Dependências;
PCC	:	Problema de Cobertura de Conjuntos;
PCV	:	Problema de Coloração de Vértices;
PLIM	:	Programação Linear Inteira Mista;
PLM	:	Problema Linear Mestre;
PMD	:	Problema da Mochila com Dependências;
PPC	:	Problema de Particionamento de Conjuntos;
PRM	:	Problema de Realocação de Máquinas;
VNS	:	*Variable Neighborhood Search*;

Sumário

Lista de Tabelas

Capítulo 1

Introdução

O *Problema de Bin Packing (PBP)* está entre os problemas mais estudados na área de otimização combinatória. A grande quantidade de trabalhos existentes na literatura que tratam deste problema demonstram a relevância de se obter soluções eficientes para sua resolução. Dentre os fatores que levaram a isso, está principalmente a simplicidade da definição do problema, de fácil entendimento, e as diversas aplicações práticas existentes em que pode ser utilizado.

Dependendo da aplicação em que está sendo empregado, o PBP possui restrições adicionais que devem ser incorporadas ao problema clássico e que trazem determinadas características que dificultam sua resolução utilizando os métodos existentes. Esse fator motiva o contínuo estudo de novas técnicas que possam contribuir na resolução dessas outras variantes do *bin packing*. Neste contexto, esta tese aborda duas variantes do problema de *bin packing*, uma pouco estudada na literatura e uma outra que é proposta neste trabalho, o *Problema de Bin Packing com Conflitos (PBPC)* e o *Problema de Bin Packing com Dependências (PBPD)*, respectivamente.

O problema de *bin packing* com conflitos é encontrado em diversos problemas reais, como por exemplo, no problema de transporte de produtos perigosos que combinam materiais tóxicos, explosivos, inflamáveis e corrosivos, e que possuem restrições quanto a itens que não podem ser transportados juntos no mesmo carregamento. No PBPC, 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*.

Já o problema de *bin packing* com dependências é um problema novo, formalmente definido nesta tese, mas que é relacionado com três outros problemas encontrados na literatura. Assim como no problema de *bin packing* clássico, neste problema considera-se um conjunto de itens

e *bins* idênticos, onde todos os itens devem ser alocados ao menor número possível de *bins*. A diferença é que no PBPD, cada item pode estar associado a uma ou mais vizinhanças (um conjunto de outros itens). A alocação de um item a um *bin* implica em alocar junto no mesmo *bin* ao menos um outro item pertencente a cada uma de suas vizinhanças, caso exista.

Particularmente, esses dois tipos de restrições (conflitos e dependências) aparecem juntos em uma aplicação proposta pelo Google, o problema de realocação de máquinas [Roadef, 2012,], onde tarefas devem ser alocadas a processadores otimizando o uso de recursos heterogêneos em aplicações de computação em nuvem. O estudo de cada característica se torna importante para um melhor entendimento desse problema geral.

Esta tese apresenta abordagens heurísticas, híbridas e exatas para resolver esses problemas. Para o PBPC são utilizadas técnicas ainda não exploradas para este problema, como a metaheurística *Iterated Local Search* e a busca em vizinhanças largas como *Ejection Chains*, *Assignment*, *Grenade* e Cobertura de conjuntos. Para o PBPD são empregadas técnicas que misturam a abordagem heurística com programação matemática, mais especificamente foi proposta uma formulação matemática para o problema e foi desenvolvido um algoritmo de *branch-and-price*.

As principais contribuições desta tese são enumeradas a seguir:

1. Um método ILS simples e eficiente para o PBPC;
2. Novas vizinhanças *ejection chains*, *assignment* e *grenade* ajustadas para o PBPC;
3. Avaliações de movimentos eficientes em $O(1)$ amortizado para vizinhanças largas (ou de grande escala) para o PBPC;
4. Uma investigação da contribuição de todas as vizinhanças propostas para o PBPC;
5. Formalização de uma nova variante do problema de bin packing, o PBPD;
6. A elaboração de um modelo matemático para o PBPD;
7. A elaboração de novas instâncias para o PBPD;
8. Um método de resolução que utiliza técnicas de programação matemática baseado no algoritmo de *branch-and-price*.

O restante deste trabalho está dividido do seguinte modo: no Capítulo 2 é apresentada a definição formal do problema de *bin packing* com conflitos e do problema de *bin packing* com dependências, além disso, é feita uma revisão bibliográfica dos trabalhos existentes na

literatura para os dois problemas. No Capítulo 3 é apresentado um método de resolução proposto para o problema de *bin packing* com conflitos e os resultados computacionais encontrados utilizando-se as instâncias descritas na literatura. No Capítulo 4 encontram-se o método exato proposto para o problema de *bin packing* com dependências, os resultados computacionais utilizando-se o método proposto e a descrição das instâncias de testes utilizadas. No Capítulo 5 encontram-se as conclusões deste trabalho e algumas sugestões de trabalhos futuros.

Capítulo 2

Variantes do Bin Packing Considerados neste Trabalho

Neste capítulo é feita uma descrição formal das variantes do problema de *bin packing* estudados nesta tese. Além disso, é feita uma revisão dos trabalhos existentes na literatura para os dois problemas. Primeiro, o problema de *bin packing* com conflitos é apresentado e a seguir, o problema de *bin packing* com dependências.

2.1 Problema de *Bin Packing* com Conflitos

Neste problema, considere um conjunto finito de *bins* de capacidade idêntica Q, um conjunto de itens $\mathcal{V} = \{1, \ldots, n\}$ com pesos $w_1, \ldots, w_n$ e um grafo $G = (V, E)$ de conflitos entre os itens, onde qualquer aresta $(i, j) \in E$ representa um conflito entre os itens i e j. 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:

Função Objetivo:

$$\text{Minimizar } z = \sum_{k=1}^{n} y_k \tag{2.1}$$

Sujeito a:

$$\sum_{i=1}^{n} w_i x_{ik} \leq Q y_k \qquad k = 1, \ldots, n \tag{2.2}$$

$$\sum_{k=1}^{n} x_{ik} = 1 \qquad i = 1, \ldots, n \tag{2.3}$$

$$x_{ik} + x_{jk} \leq 1 \qquad (i,j) \in E, k = 1, \ldots, n \tag{2.4}$$

$$y_k \in \{0,1\} \qquad k = 1, \ldots, n \tag{2.5}$$

$$x_{ik} \in \{0,1\} \qquad i = 1, \ldots, n, k = 1, \ldots, n \tag{2.6}$$

Neste modelo, 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 ($x_{ik} = 0$). O objetivo definido por 2.1 é minimizar o número de *bins* utilizados. O conjunto de restrições 2.2 garante que a capacidade máxima de cada *bin* seja respeitada. As restrições 2.3 asseguram que cada item só pode ser alocado a um único *bin*. O conjunto de restrições 2.4 garantem que dois itens pertencentes ao conjunto de arestas do grafo de conflitos não podem ser alocados no mesmo *bin*. As restrições 2.5 e 2.6 indicam o domínio das variáveis envolvidas na modelagem.

O PBPC possui diversas aplicações importantes, na área de transporte e logística, certas restrições de compatibilidade são impostas no transporte de diversas classes de produtos, incluindo alimentos, medicamentos e materiais perigosos. Combinações de produtos inflamáveis, explosivos ou tóxicos não devem ser transportados conjuntamente na mesma carga [Hamdi-Dhaoui et al., 2014, Minh et al., 2013], levando as decisões de atribuição item-veículo difíceis de serem realizadas durante o planejamento operacional. Uma aplicação também surge no problema de programação de horários para exames (*timetabling examination*), onde busca-se uma tabela de horários que respeite as capacidades das salas e evite conflitos [Laporte e Desrochers, 1984]. Nenhum estudante, por exemplo, deve fazer mais de um exame ao mesmo tempo. Finalmente, outra aplicação surge no campo da computação paralela, onde deve-se atribuir um conjunto de tarefas a um número mínimo de processadores, sujeito a certas restrições existentes na alocação de algumas tarefas conflitantes em algumas máquinas [Jansen, 1999, Masson et al., 2013].

Do ponto de vista metodológico, o problema também é de alta relevância, incluindo dois tipos diferentes de restrições: capacidade e conflito. Cada uma dessas classes de restrições, sozinha, leva a um subproblema NP-difícil: *Problema de Bin Packing (PBP)* ou *Problema de Coloração de Vértices (PCV)*. Se o conjunto E de arestas do grafo de conflitos for vazio, tem-se que o PBPC é equivalente ao PBP. No entanto, se a capacidade de cada *bin* for infinita, ou seja, maior do que soma do peso de todos os itens, obtém-se o PCV. Tem-se ainda

que, quando $w_i = 1$ para cada $v_i \in \mathcal{V}$, o PBPC generaliza outro problema chamado *Bounded Vertex Coloring Problem (BVCP)*.

Existem diversos trabalhos na literatura que tratam o PBPC. Os primeiros trabalhos dedicados a estudar o problema de *bin packing* com conflitos e suas formulações foram apresentados em [Jansen e Öhring, 1997] e [Jansen, 1999]. As restrições de conflito também possuem uma longa história no domínio dos problemas de escalonamento de horários [Laporte e Desrochers, 1984]. Diversos artigos que consideram algoritmos aproximativos podem ser encontrados em [Epstein e Levin, 2008] e [Epstein et al., 2008]. Nenhum esquema de aproximação polinomial limitada pode ser obtido para o PBPC com grafo de conflito geral [Jansen, 1999]. Como tal, a maioria dos resultados de aproximação referem-se a estruturas de grafos específicos, tais como grafos perfeitos, grafos de intervalo e grafos bipartidos, com taxas de aproximação de 2,5, 2,33333 e 1,75, respectivamente [Epstein e Levin, 2008].

Mais recentemente, [Gendreau et al., 2004] investigaram algumas heurísticas (sem garantia de desempenho) para o problema. Os autores propuseram seis heurísticas construtivas diferentes, bem como dois limites inferiores, e um primeiro conjunto de instâncias para o problema baseados nas instâncias de [Falkenauer, 1996] para o problema de *bin packing*. Infelizmente, essas instâncias não estão atualmente disponíveis, e, portanto, foram posteriormente re-geradas nos trabalhos de [Muritiba et al., 2010, Elhedhli et al., 2011].

Nos últimos anos, um esforço significativo tem sido dedicado a pesquisa de algoritmos de programação matemática para o problema. [Khanafer et al., 2010] propuseram novos limites inferiores baseados no conceito de soluções duais *data-dependent* para o problema. Além disso, nada menos do que três artigos recentes propuseram procedimentos exatos de *branch-and-price* [Muritiba et al., 2010, Elhedhli et al., 2011, Sadykov e Vanderbeck, 2013]. Esses artigos usam diferentes regras de *branching*, colunas iniciais e sub-procedimentos para resolver o problema de *pricing* (um problema de mochila com conflitos). Quando os conflitos formam um grafo de intervalo, o problema de *pricing* pode ser resolvido de modo eficiente através de programação dinâmica [Sadykov e Vanderbeck, 2013], enquanto procedimentos de *branch-and-bound* podem ser utilizados em casos mais gerais. Esses métodos exatos podem resolver muitos conjuntos de instâncias, com até 1000 itens em alguns casos. Ainda assim, as instâncias com um grande número de itens por *bins*, por exemplo o conjunto (*da*) de [Sadykov e Vanderbeck, 2013], permanecem desafiadoras. O esforço computacional também pode variar significativamente de uma instância para outra, e torna-se impraticável para grandes conjuntos de dados com grafos arbitrários.

Metaheurísticas para o PBPC têm recebido, até esta data, menos atenção do que algoritmos de programação matemática. Uma metaheurística populacional avançada foi proposta em [Muritiba et al., 2010], e usada para gerar bons limites superiores e colunas iniciais. O método é uma combinação complexa de uma busca tabu baseada em classe de vizinhanças de impasse de [Morgenstern, 1996], com um algoritmo genético usando o operador *crossover* de [Galinier e Hao, 1999]. Sadykov e Vanderbeck [Sadykov e Vanderbeck, 2013] também introduzem uma heurística de *diving* baseada em uma exploração incompleta da árvore de *branch-and-bound*. Este método é uma boa alternativa entre abordagens de solução exata e heurísticas, pois combina os benefícios da programação matemática para instâncias do problema com poucos itens por *bins*, com um tempo de CPU menor. De modo geral, metaheurísticas para o PBPC ainda merecem mais investigações.

Finalmente, como o PBPC é uma generalização tanto do problema de *bin packing* quanto do problema de coloração de vértices, os resultados das pesquisas na área de metaheurísticas para estes dois subproblemas podem ser uma boa fonte de inspiração. A literatura sobre estes dois subproblemas é extensa, e pode-se referir aos trabalhos de [Lewis et al., 2012, Malaguti e Toth, 2010, Quiroz-Castellanos et al., 2015, Delorme et al., 2016] e ainda o trabalho de [Lewis, 2016] para algumas revisões detalhadas. Algumas das melhores metaheurísticas atuais para o problema de *bin packing* incluem o algoritmo genético de agrupamento de [Falkenauer, 1996], a extensão do método Perturbation-MBS proposto por [Fleszar e Charalambous, 2011] e também o algoritmo genético de agrupamento com a transmissão controlada de genes de [Quiroz-Castellanos et al., 2015]. As heurísticas baseadas em população tendem a ser muito frequentemente utilizadas, devido à sua capacidade de explorar áreas muito diversas do espaço de busca. Para o problema de coloração de vértices, algumas das metaheurísticas mais avançadas incluem a abordagem TabuCol de [Hertz e de Werra, 1987], o algoritmo React-PartialCol de [Blöchliger e Zufferey, 2008], os algoritmos híbridos evolutivos de [Galinier e Hao, 1999, Malaguti et al., 2008], e a metaheuristica de colônia de formigas desenvolvida no trabalho de [Dowsland e Thompson, 2008].

2.2 Problema de *Bin Packing* com Dependências

Conflitos são apenas um dos tipos de restrições que devem ser levadas em consideração na resolução de problemas de alocação de recursos e empacotamento. Na aplicação formulada pelo Google [Roadef, 2012,] por exemplo, conflitos aparecem em conjunto com restrições de dependência, que expressam o fato de que algumas tarefas devem ser executadas em

conjunto com outras no mesmo processador. Porém, enquanto o problema de *bin packing* com restrições de conflitos é bem conhecido na literatura, o problema incorporando restrições de dependência permanece pouco explorado até então, mesmo considerando o fato desse tipo de restrição possuir uma estrutura básica que pode ser facilmente aplicada a diversos problemas complexos. Desta forma, nesta tese, essa restrição será isolada para criar uma nova variante do problema de *bin packing* com dependências. Isso irá contribuir para entender melhor como essa classe de restrições pode ser resolvida de modo eficiente e mais tarde ser generalizado para aplicações reais que contam ainda com a presença de restrições adicionais.

Deste modo, o *Problema de Bin Packing com Dependências (PBPD)* pode ser definido com segue. Considere um conjunto finito de *bins* de capacidade idêntica Q, um conjunto de itens $\mathcal{V} = \{1, \ldots, n\}$ com pesos $w_1, \ldots, w_n$. Cada item $i \in \mathcal{V}$ pode ter um conjunto de vizinhanças $N(i) = \{N_1(i), \ldots, N_{L_i}(i)\}$, onde cada vizinhança é composta por um conjunto de itens.

Assim como no PBPC, o objetivo do PBPD é alocar todos os itens a *bins* utilizando o número mínimo possível de *bins*. Porém, esta alocação deve respeitar, além das restrições de capacidade, as restrições de vizinhanças, onde uma solução é viável se, para cada item i, e para cada vizinhança $N_l(i)$, existe pelo menos um item $j \in N_l(i)$ alocado ao mesmo *bin* que em que i está alocado. Um modelo de programação linear inteira mista (PLIM) para o problema é dado por:

$$(F1) \quad \min \sum_{k=1}^{n} y_k \tag{2.7}$$

$$\text{s.a.} \sum_{i=1}^{n} w_i x_{ik} \leqslant Q y_k \qquad k = 1, \ldots, n \tag{2.8}$$

$$\sum_{k=1}^{n} x_{ik} = 1 \qquad i = 1, \ldots, n \tag{2.9}$$

$$x_{ik} \leqslant \sum_{j \in N_l(i)} x_{jk} \qquad \forall i \in 1, \ldots, n, \forall k \in 1, \ldots, n, \forall l \in 1, \ldots, L_i \tag{2.10}$$

$$y_k \in \{0, 1\} \qquad k = 1, \ldots, n \tag{2.11}$$

$$x_{ik} \in \{0, 1\} \qquad i = 1, \ldots, n, k = 1, \ldots, n \tag{2.12}$$

Neste modelo, a cada variável binária x_{ik} é atribuído o valor 1, se e somente se, o item i é alocado ao *bin* k, e cada variável binária y_k recebe o valor 1, se e somente se, o *bin* k está sendo utilizado. A função objetivo (2.7) minimiza a soma do número de *bins* utilizados.

As restrições (2.8) garantem que o número de itens alocados a cada *bin* não ultrapasse sua capacidade máxima. O conjunto de restrições (2.9) assegura que cada item deve ser atribuído a somente um *bin*. As restrições (2.10) garantem que pelo menos um vizinho de cada vizinhança de um item deve ser alocado ao mesmo *bin* em que este foi alocado. As restrições (2.11) e (2.12) definem o domínio das variáveis de decisão do problema.

Embora o PBPD não tenha sido diretamente estudado e formulado na literatura, existem dois problemas relacionados ao PBPD que pertencem à classe de problemas da mochila: o *Problema da Mochila 1-vizinho* (*1-neighbour knapsack problem*) e o *Problema da Mochila Todos-vizinhos* (*all-neighbour knapsack problem*).

O problema da mochila 1-vizinho e o problema da mochila todos-vizinhos pertencem a uma classe de problemas chamados de Problema da Mochila Limitado (*Constrained Knapsack Problems*). Em ambos, existe uma mochila que suporta um peso máximo igual a k, um grafo $G = (V, E)$, onde cada vértice $i \in V$ representa um objeto ou item que pode ser levado na mochila, com peso w_i e um lucro p_i. O objetivo de cada um dos problemas é encontrar um conjunto de vértices com máximo lucro e um peso máximo de k. No problema da mochila 1-vizinho, se um item possui um vizinho, ele pode ser selecionado somente se pelo menos um dos seus vizinhos também for selecionado. Já no problema da mochila todos-vizinhos, quando um item é selecionado, todos os seus vizinhos também são selecionados. Problemas da mochila limitados possuem aplicações em diversas áreas tais como escalonamento, gerenciamento de ferramentas, estratégias de investimento, armazenamento de banco de dados e formação de redes (*network formation*) [Borradaile et al., 2012].

O PBPD foi formulado baseado apenas no conceito do problema da mochila 1-vizinho. Já que, embora a variante do problema da mochila todos-vizinhos seja relevante no estudo do problema da mochila, quando inserido no contexto do problema de *bin packing*, onde todos os vizinhos devem ser alocados juntos no mesmo *bin*, este se reduz ao problema de *bin packing* clássico, no qual todos os itens interdependentes são agrupados em um único item grande. Por esta razão, torna-se desnecessário definir um novo problema com a estrutura "todos-vizinhos".

No trabalho de [Borradaile et al., 2012], os autores desenvolveram um algoritmo polinomial para resolver o problema da mochila 1-vizinho considerando os valores dos pesos e dos lucros uniformes e o grafo não-direcionado. Os autores mostram que o problema é fortemente NP-difícil (*NP-difícil in the strong sense*) quando o problema é direcionado e uniforme. Este trabalho apresenta ainda algoritmos aproximados para o problema com peso e lucro arbitrários com grafos direcionados e não-direcionados.

Para o problema da mochila todos-vizinhos, [Borradaile et al., 2012] demonstram que o problema uniforme e direcionado é NP-completo. Para o problema geral e não-direcionado, os autores apresentam uma redução para o problema da mochila 0-1, que é resolvido em tempo polinomial.

No PBPD proposto, foram introduzidos não apenas uma única restrição 1-vizinho por item, mas um conjunto de restrições baseado em diferentes vizinhanças, com a condição de que pelo menos um item em cada vizinhança deva ser selecionado em conjunto. Essa extensão do conceito de 1-vizinho é necessária para cobrir aplicações adicionais relacionadas com o problema de realocações de máquinas.

O Problema de Realocação de Máquinas (PRM) - *Machine Reassignment Problem* foi proposto pelo Google como um desafio no Roadef 2012 [Roadef, 2012,]. O objetivo do PRM é otimizar a utilização de um conjunto de máquinas heterogêneas M por um conjunto de processadores P. Inicialmente cada processo $p \in P$ está alocado a uma determinada máquina no entanto, eles podem ser movidos entre as máquinas. Esse movimento deve respeitar restrições de capacidade, dependência e conflito. O problema considera também um conjunto de recursos R onde um processo $p \in P$ pode requerer $R_{p,r}$ unidades de cada recurso $r \in R$. A capacidade de uma máquina $m \in M$ com relação a um recurso $r \in R$ é chamado $C_{m,d}$. No PRM, a utilização de um recurso em uma máquina não pode exceder sua capacidade. A utilização U de um recurso r em uma máquina m é definida como $U(m,r) = \sum_{p \in P|M(p)=m} R(p,r)$, onde $M(p) = m$ significa que o processo p está executando na máquina m. Os processos são particionados em um conjunto s de serviços, e cada processo pertence somente a um serviço $s \in S$ e todos os processos do mesmo serviço devem executar em máquinas diferentes. Seja L, o conjunto de localidades, onde cada localidade $l \in L$ é um conjunto de máquinas, o número de localidades distintas onde os processos pertencentes ao mesmo serviço devem executar, devem respeitar uma dada restrição de espalhamento. Adicionalmente, as máquinas são agrupadas também em vizinhanças denotadas por N. Um serviço pode depender de outros serviços, portanto, se um serviço s executar em uma máquina na vizinhança $n \in N$, pelo menos um processo de cada serviço pertence ao mesmo conjunto de serviços de que s depende e deve executar em uma máquina na vizinhança de n.

Destaca-se que neste problema as tarefas pertencem aos serviços, e que cada serviço pode ser dependente de um ou vários serviços, de modo que uma tarefa de cada serviço necessário deva ser incluída juntamente com o item. Isto cria um relacionamento de dependência entre múltiplos serviços, razão pela qual se propõe nesta tese uma generalização com múltiplas restrições 1-vizinho para cada item.

Vários trabalhos na literatura tratam o PRM, no trabalho de [Masson et al., 2013], os autores desenvolvem uma formulação matemática e um algoritmo baseado na metaheurística *Multi-start Iterated Local Search* para o PRM. [Gavranović et al., 2012] propuseram limites inferiores para o problema e uma abordagem baseada no *Variable Neighborhood Search* (VNS), onde eles tentam realocar os processos grandes primeiro. O procedimento de perturbação do VNS de [Gavranović et al., 2012] é feito por uma busca local considerando uma outra função objetivo. Lopes *et al.* [Lopes et al., 2015] desenvolveram quatro metaheurísticas baseadas no ILS, onde duas delas usam uma perturbação baseada em programação inteira. No trabalho de [Portal et al., 2015], os autores propuseram uma heurística baseada na metaheurística *Simulated Annealing*, bem como desenvolveram estruturas de dados especiais para acelerar o tempo de execução. Mais recentemente, [Wang et al., 2016] desenvolveram um método que usa diferentes estruturas de vizinhanças, duas delas já utilizadas na literatura para o Problema de Alocação Generalizado (*Generalized Assignment Problem*), permitindo movimentos inviáveis na busca local e um mecanismo de busca de particionamento da vizinhança.

Capítulo 3

Heurísticas Propostas para o Problema de Bin Packing com Conflitos

O Problema de *Bin Packing* com Conflitos (PBPC) é um problema de otimização combinatória de difícil solução que junta características dos problemas de *bin packing* e coloração de vértices. Neste capítulo é apresentado um algoritmo heurístico híbrido para o problema de *bin packing* com conflitos. O algoritmo proposto utiliza vizinhanças em larga escala combinadas a metaheurística *Iterated Local Search*. A seguir é dada uma noção básica do funcionamento do ILS, e logo após, na Seção 3.2, é apresentada a heurística proposta para o PBPC baseada nessa técnica. Nas próximas seções todos os componentes do algoritmo proposto são detalhados. Ao final do capítulo, são apresentadas ainda as instâncias existentes na literatura para o problema e os resultados obtidos com o método proposto.

3.1 *Iterated Local Search*

No método proposto utiliza-se a metaheurística *Iterated Local Search (ILS)*, que faz uso de várias classes de vizinhanças para melhorar a solução, e um procedimento de perturbação. A metaheurística ILS [Lourenço et al., 2010] tem como ideia principal realizar sucessivas buscas no espaço de soluções tendo como partida um ótimo local no qual foi aplicado um procedimento de perturbação. Os seus componentes principais são: um procedimento para geração de uma solução inicial, um procedimento de busca local, um procedimento de perturbação da solução e um procedimento que define o critério de aceitação de uma solução. Todos os componentes do ILS são vistos no pseudo-código apresentado no Algoritmo 1.

O procedimento de perturbação tem a função de modificar a solução corrente, tentando escapar de ótimos locais. Se as perturbações realizadas forem pequenas, o espaço de soluções

Algoritmo 1 Iterated Local Search

1: $s_0 \leftarrow$ ConstroiSoluçãoInicial;
2: $s^* \leftarrow$ BuscaLocal(s_0);
3: **enquanto** (critério de parada não alcançado) **faça**
4: $s' \leftarrow$ Perturbação(s^*, histórico);
5: $s^{*'} \leftarrow$ BuscaLocal(s');
6: $s^* \leftarrow$ CritérioAceitação(s^*, $s^{*'}$);
7: **fim enquanto**
8: **retorne** s^*;

será pouco explorado não conseguindo escapar de ótimos locais, porém, se as perturbações forem demasiadamente grandes, o algoritmo poderá reiniciar toda a busca. O critério de aceitação é utilizado para decidir a próxima solução que será perturbada.

3.2 ILS para o PBPC

Esta seção descreve o algoritmo heurístico proposto, cujo pseudo-código é apresentado no Algoritmo 2. Inicialmente, uma solução é gerada pelo procedimento de Construção (Linha 1). Essa solução é viável porém, o número de *bins* utilizados na realização da alocação não é necessariamente o menor possível. A seguir, para diminuir o número de *bins* utilizados, o método iterativamente remove um *bin* (Linha 4), alocando seus itens a outros *bins* escolhidos de modo aleatório. Essa realocação poderá gerar conflitos com os itens já alocados e excesso na capacidade dos *bins* da solução. A partir daí, para tentar resolver esse problema, são aplicadas repetidas iterações de Busca Local (BL) com movimentos de custo zero, *Ejection Chains (EC)*, movimentos de *Grenade*, Cobertura de Conjuntos e *Assignment Neighborhood* (Linhas 7 – 17). Usando a terminologia empregada no problema de Coloração de Vértices, e como discutido por [Lewis et al., 2012], este tipo de estratégia explora o espaço de busca completo mas com colorações impróprias. O algoritmo termina quando nenhuma solução viável é encontrada após N_{SHAK} perturbações, ou se um limite inferior trivial de K_{LB} número de *bins* for alcançado. O valor de K_{LB} é dado pela soma de todos os pesos dos itens dividido pelo valor da capacidade dos bins.

Nas subseções a seguir, cada um dos componentes do algoritmo será detalhado.

3.2.1 Solução Inicial

Para construir uma solução inicial, foi implementado o algoritmo construtivo proposto para o PBPC por [Gendreau et al., 2004]. Essa heurística construtiva é uma adaptação do algoritmo

Algoritmo 2 Iterated local search para o PBPC

1: $\mathcal{S}_{\text{BEST}} \leftarrow$ ConstruirSoluçãoInicial();
2: $K_{\text{LB}} \leftarrow$ CalcularLowerBound();
3: **enquanto** NbBins($\mathcal{S}_{\text{BEST}}$) $> K_{\text{LB}}$ **e** $\mathcal{S}_{\text{BEST}}$ é viável **faça**
4: $\quad\mathcal{S}_{\text{BEST}} \leftarrow$ ReduzirNbBins($\mathcal{S}_{\text{BEST}}$);
5: $\quad\mathcal{S} \leftarrow \mathcal{S}_{\text{BEST}}$;
6: $\quad i_{\text{SHAK}} \leftarrow 0$;
7: $\quad$**enquanto** $i_{\text{SHAK}} \leq N_{\text{SHAK}}$ **e** $\mathcal{S}_{\text{BEST}}$ não é viável **faça**
8: $\quad\quad$**para** N_{LS} iterações **faça**
9: $\quad\quad\quad\mathcal{S} \leftarrow$ BuscaLocal($\mathcal{S}$);
10: $\quad\quad\quad\mathcal{S} \leftarrow$ AssignmentNeighborhood($\mathcal{S}$);
11: $\quad\quad\quad\mathcal{S} \leftarrow$ EjectionChains($\mathcal{S}$);
12: $\quad\quad\quad\mathcal{S} \leftarrow$ Grenade($\mathcal{S}$);
13: $\quad\quad$**fim para**
14: $\quad\quad$**se** $\exists\ k \in \mathbb{N}^+$ tal que $i_{\text{SHAK}} = k \times N_{\text{SC}}$ **então** $\mathcal{S} \leftarrow$ CoberturaConjuntos();
15: $\quad\quad$**se** $custo(\mathcal{S}) < custo(\mathcal{S}_{\text{BEST}})$, **então** $\mathcal{S}_{\text{BEST}} \leftarrow \mathcal{S}$ **e** $i_{\text{SHAK}} \leftarrow 0$; **else** $i_{\text{SHAK}} \leftarrow i_{\text{SHAK}} + 1$;
16: $\quad\quad\mathcal{S} \leftarrow$ Perturbação($\mathcal{S}_{\text{BEST}}$);
17: $\quad$**fim enquanto**
18: **fim enquanto**

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 constrói uma solução inicial com K *bins*.

3.2.2 Métodos de Busca Local

O procedimento de busca local tem por objetivo melhorar qualquer inviabilidade contida na solução produzida pela remoção de um *bin* ou pelo operador de perturbação. Para definir o que seria um melhoramento na solução, considere que o custo de cada solução $\mathcal{S}$ é definida pela soma dos custos de seus *bins* $B \in \mathcal{S}$, dada pela Equação (3.1). Nesta equação, $C(B)$ representa o número de conflitos presentes no *bin*, $W(B)$ representa a quantidade total de excesso de peso do *bin*, ω^{C} e ω^{W} são os fatores de penalidade associados a cada unidade de conflito e violação de peso, respectivamente.

$$\Phi(B) = \omega^{\mathrm{C}} C(B) + \omega^{\mathrm{W}} W(B) \tag{3.1}$$

Na busca local são utilizadas três vizinhanças clássicas: Swap, Swap2vs1 e Relocate. Um movimento de Swap troca dois itens pertencentes a *bins* diferentes. Como sugerido pelo nome, no Relocate, um item é movido de um *bin* para outro diferente. Já no Swap2vs1, um par de itens alocados a um mesmo *bin* é trocado com um item pertencente a um *bin* diferente. As vizinhanças são exploradas como detalhadas no Algoritmo 3, onde todos os pares de *bins* são enumerados em uma ordem aleatória para testar os movimentos (Linhas 4–5). Para cada par diferente de *bins*, o melhor movimento obtido entre todos os itens pertencentes aos *bins* avaliados é aplicado (Linha 9). A busca local termina quando nenhuma solução com custo melhor é encontrada.

Essa política de avaliação permite otimizar o esforço computacional necessário para resolver o problema, uma vez que não é necessário reavaliar os movimentos entre pares de *bins* quando a avaliação tiver sido realizada anteriormente sem sucesso, e os *bins* não tiverem sofrido nenhuma alteração desde então. Além disso, pelo menos um *bin* B envolvido em um movimento deve satisfazer a $\Phi(B) > 0$. Essa condição é incluída como um filtro adicional (Linha 4), permitindo economizar uma quantidade significativa de tempo de CPU e reduzir a complexidade computacional.

Algoritmo 3 Busca local com movimentos de *custo-0*.

```
1: It_Loop ← 0
2: enquanto um mínimo local não foi alcançado faça
3:    It_Loop ← It_Loop + 1
4:    para cada bin B escolhido em um ordem aleatória tal que Φ(B) > 0 faça
5:       para cada bin B' ≠ B escolhido em um ordem aleatória faça
6:          se B ou B' foi modificado desde a última avaliação ou It_Loop = 1 então
7:             Δ ← CustoMelhorMovimento(B, B')
8:             se {Δ < 0} ou {Δ ≤ 0 e It_Loop = 1} então
9:                AplicaMelhorMovimento(B, B')
10:            fim se
11:         fim se
12:      fim para
13:   fim para
14: fim enquanto
```

Finalmente, durante a primeira iteração da BL, movimentos com custo zero (*custo-0*) também são aceitos (Linha 8). Esses movimentos não melhoram o objetivo do problema mas contribuem para diversificar a busca no espaço de soluções.

Avaliação eficiente dos movimentos. A eficiência computacional na avaliação dos movimentos realizados durante a busca é muito importante para o bom desempenho do método, pois este é o principal gargalo das recentes heurísticas baseadas em busca local. Qualquer redução significativa no tempo de CPU necessário para avaliar movimentos pode ser traduzida num ganho direto em termos de número de movimentos que podem ser avaliados durante uma execução do algoritmo e, assim, melhorar a qualidade da solução. Para o PBPC, as avaliações de movimento não são triviais, uma vez que requerem calcular o número de conflitos na nova solução. Uma implementação direta do SWAP, por exemplo, entre os *bins* B e B' exigiria $\mathcal{O}(|B| + |B'|)$ operações elementares por avaliação de movimento, onde $|B|$ é o número atual de itens em um *bin* B. Este esforço é devido à avaliação de conflitos entre os itens trocados e os existentes nos *bins*. Esse esforço cresce linearmente com o número médio de itens por *bin* e pode ser responsável pela maioria do tempo de CPU gasto para resolver instâncias do problema com um grande número de itens por *bin*.

Para tratar esta questão, nesta tese propõe-se um mecanismo de pré-processamento e avaliação dos movimentos que realiza as avaliações dos movimentos em $\mathcal{O}(1)$ amortizado. Foi utilizado uma matriz que dá acesso em $\mathcal{O}(1)$ ao número atual de conflitos $\textsc{Conf}[i][B]$ entre um item i e todos os itens em um *bin* B. Seja d_i o grau de i no grafo de conflito. Esta matriz pode ser construída em $\mathcal{O}(Kn + \sum_{i=1}^{n} d_i)$ operações elementares para a solução inicial. Subsequentemente, qualquer modificação da solução pode ser considerada como uma sequência de realocações de itens. Para cada realocação de um item i de um *bin* B para um *bin* B', a matriz CONF é atualizada em $\mathcal{O}(d_i)$ por meio do Algoritmo 4.

Algoritmo 4 Mecanismo de Atualização – Realocação de um item i de um bin B para B'.

1: **para** cada item j em conflito com i **faça**
2: $\quad \textsc{Conf}[j][B] = \textsc{Conf}[j][B] - 1$
3: $\quad \textsc{Conf}[j][B'] = \textsc{Conf}[j][B'] + 1$
4: **fim para**

Essa estrutura de dados pode agora ser utilizada para executar avaliações eficientes dos movimentos na busca local. Considere um movimento de SWAP entre um item i de um *bin* B, com o item j de um *bin* B'. A diferença Δ no número de conflitos antes e depois da

realização do movimento pode ser calculada da seguinte forma:

$$\Delta = \textsc{Conf}[i][B'] + \textsc{Conf}[j][B] - \textsc{Conf}[i][B] - \textsc{Conf}[j][B'] - 2 \times \textsc{temConflito}(i,j) \quad (3.2)$$

Pode-se destacar o termo corretivo $2 \times \textsc{temConflito}(i,j)$, que tem valor 2 se e somente se houver um conflito entre os itens i e j. Este termo vem do fato de que a matriz $\textsc{Conf}[i][B']$ contém o número de conflitos ao mover i para B', mas *antes* da remoção do item j, e vice-versa para a matriz $\textsc{Conf}[j][B]$.

Esta abordagem pode ser generalizada para avaliar em $\mathcal{O}(1)$ qualquer movimento envolvendo a troca de conjuntos de itens limitados S e S' entre o par de *bins* B e B', permitindo assim uma avaliação eficiente de todos os movimentos considerados. Um exemplo de um movimento, que troca três itens de cada *bin*, é exibido na Figura 3.1. Seja $N_{\text{INTER}}(S, S')$ o número de conflitos entre pares de itens em S e S', seja $N_{\text{INTRA}}(S)$ o número de conflitos dentro do conjunto S, e seja $N_{\text{INTRA}}(S')$ o número de conflitos dentro do conjunto S'. A diferença no número de conflitos após aplicar o movimento pode ser obtida a partir da Equação (3.3).

$$\begin{aligned} \Delta = & \sum_{i \in S} (\textsc{Conf}[i][B'] - \textsc{Conf}[i][B]) + \sum_{j \in S'} (\textsc{Conf}[j][B] - \textsc{Conf}[j][B']) \\ & + 2 \times (N_{\text{INTRA}}(S) + N_{\text{INTRA}}(S') - N_{\text{INTER}}(S, S')) \end{aligned} \quad (3.3)$$

O excesso na capacidade dos *bins* também pode ser avaliado com essa estrutura em $\mathcal{O}(1)$. Em geral, isso leva a avaliações dos movimentos em $\mathcal{O}(1)$ mas com um esforço computacional levemente maior quando a solução é atualizada. Em todos os experimentos computacionais, pode-se observar que, em média, centenas de movimentos são testados antes mesmo de um deles ser aplicado. A complexidade necessária para inicializar e atualizar as estruturas de dados permaneceu pequena comparada ao esforço dedicado às avaliações dos movimentos, levando às avaliações em $\mathcal{O}(1)$ amortizadas, como ditas anteriormente.

3.2.3 Vizinhanças Largas

O método ILS híbrido proposto neste trabalho possui uma perturbação e conceitos de busca local simples, mas utiliza quatro vizinhanças largas para melhorar sua performance. As duas primeiras vizinhanças – Assignment e Ejection Chains – exploram restrições do problema que possuem o benefício de serem combinatórias mas são polinomiais. Isto leva a vizinhanças de tamanho exponencial que podem ser exploradas em tempo polinomial. A terceira vizinhança – Grenade – é baseada na enumeração de movimentos grandes.

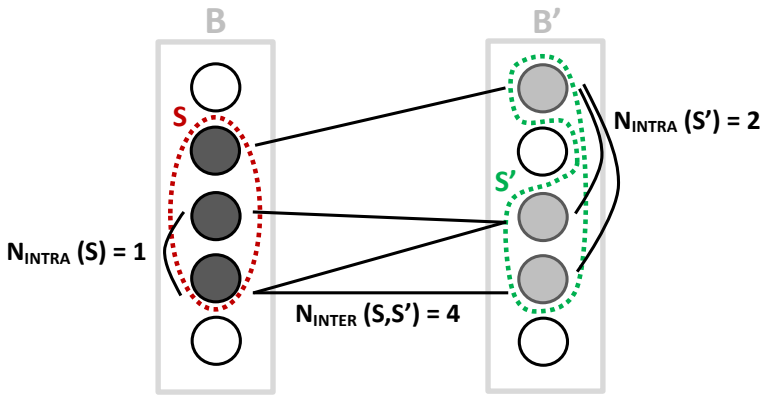


Figura 3.1: Generalização de um movimento envolvendo a troca de três itens em B com três itens em B'.

Finalmente, a quarta vizinhança – Cobertura de Conjuntos – considera o fato que resolvedores de programação inteira eficientes podem resolver a formulação do problema de cobertura de conjuntos na presença de um conjunto limitado de colunas (itens combinados em um *bin*). Esses resolvedores são utilizados para combinar colunas de alta qualidade obtidas de um mínimo local para produzir melhores soluções.

Algumas dessas famílias de vizinhanças são bem conhecidas nos trabalhos de pesquisa operacional, como detalhados em [Deineko e Woeginger, 2000] e [Ahuja et al., 2002]. Nem todos os problemas de otimização combinatória permitem uma busca eficiente em tais vizinhanças de tamanho exponencial. No caso do PBPC, tal busca é possível. Nas próximas seções são descritas como essas vizinhanças largas são adaptadas para o problema, bem como, é fornecida uma descrição de uma forma específica de *ejection chains*, que permite uma exploração em tempo polinomial de um subconjunto de soluções de tamanho exponencial, sob a condição de que a ejeção entre *bins* deva seguir uma ordem pré-definida.

3.2.3.1 Assignment Neighborhood

Dada uma solução corrente, na vizinhança *Assignment Neighborhood* alguns itens são selecionados e removidos dos *bins* onde estão alocados. Esses itens são então reinseridos nos "buracos" formados na solução através da resolução de um problema de atribuição em um grafo bipartido ponderado. Uma metodologia similar foi utilizada por [Gutin, 1999] and [Toth e Tramontani, 2008] nos problemas do caixeiro viajante e de roteamento de veículos.

Durante a execução do ILS para o PBPC, observou-se que somente alguns itens *problemáticos* apresentam conflito ou pertencem a *bins* com excesso de peso. Para melhorar a solução, deve-se buscar alocações mais eficientes para esses itens o que pode incluir movimentos

simultâneos destes com outros itens em conjunto. Deste modo, para realizar essa tarefa, o algoritmo proposto seleciona aleatoriamente um item problemático, chamado de k, bem como N_{ASSIGN} outros itens quaisquer, pelo menos um em cada *bin* da solução corrente. Juntos, estes itens formam um conjunto de vértices V_1'. Considere ainda V_2' uma cópia desse conjunto de vértices, e seja o conjunto de arestas $E' = \{(i,j) \text{ tal que } i \in V_1' \text{ e } j \in V_2'\}$. O custo de cada aresta (i,j) é definido como o custo de inserir um item i no *bin* $B(j)$ do item j (que foi removido de antemão). Finalmente, para favorecer o movimento de k e evitar um possível loop, uma penalidade ϵ é adicionada ao custo da aresta que conecta k a sua cópia. Isto define o grafo bipartido $G' = (V_1', V_2', E')$, ilustrado na Figura 3.2.

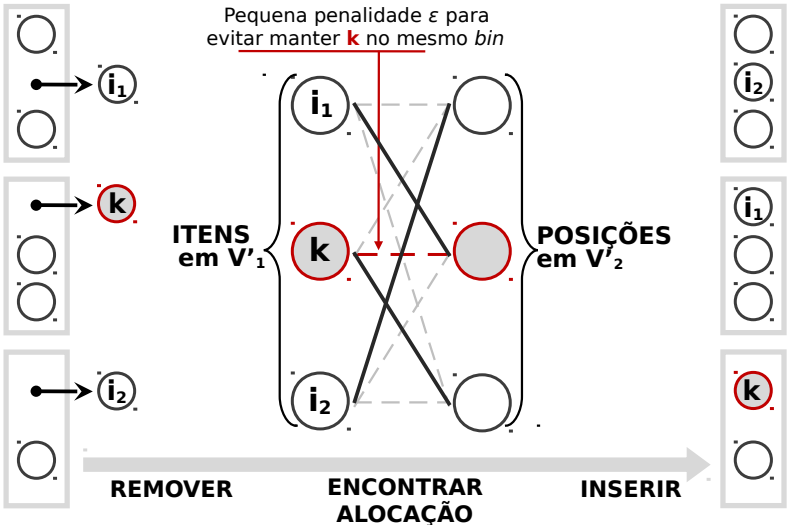


Figura 3.2: *Assignment neighborhood* e seu grafo bipartido G' associado.

Então, encontrar a melhor realocação dos itens aos *bins* é feita resolvendo-se um problema de atribuição em um grafo bipartido. A solução ótima é encontrada em $\mathcal{O}(n^3)$ por uma implementação eficiente do algoritmo Húngaro [Kuhn, 1955]. Qualquer solução ótima não trivial desse modelo de atribuição (diferente de parear cada nó i com sua cópia) corresponde a um movimento na solução do PBPC. Esse movimento é aplicado, levando a uma nova (melhor ou igual) solução incumbente no ILS.

3.2.3.2 Ejection Chains

Ejection Chains (EC) permite explorar um subconjunto diferente de soluções vizinhas obtidas pela realocação em cadeia dos itens (ver, por exemplo, [Thompson e Psaraftis, 1993] e [Glover, 1996]). A principal diferença entre EC e a vizinhança *Assignment neighborhood* é que a escolha dos itens que serão realocados nesta primeira vizinhança não são fixados

a priori. Por outro lado, para permitir um procedimento de exploração da vizinhança em tempo polinomial, as realocações dos itens são restringidas para atender uma ordem pré definida dos *bins*.

O algoritmo de *Ejection Chains* desse trabalho funciona como descrito a seguir. Primeiro, foi escolhida uma ordem aleatória Π para os *bins* pertencentes a solução corrente. Então, considere um grafo auxiliar $G'' = (V'', A'')$ com $V'' = \mathcal{V} \cup \mathcal{V}^{\text{ZERO}} \cup v^{\text{SOURCE}}$, como ilustrado na Figura 3.3.

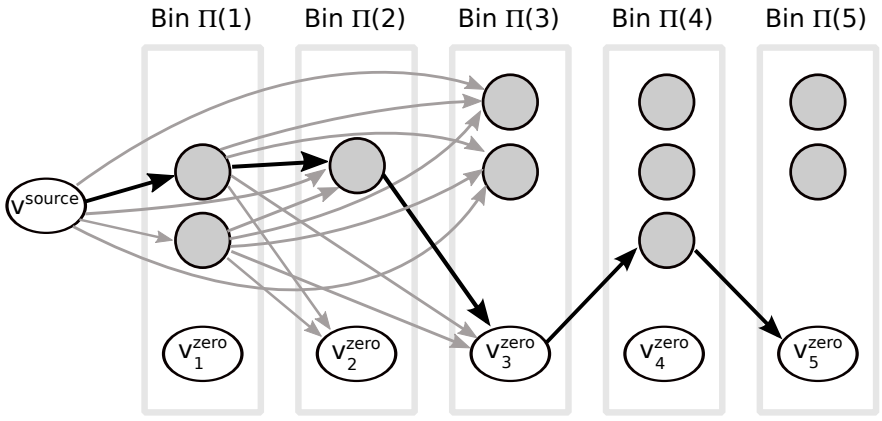


Figura 3.3: Grafo construído durante o *Ejection-chains*, e uma possível solução.

O conjunto $\mathcal{V}$ contém um nó para cada item existente, enquanto $\mathcal{V}^{\text{ZERO}}$ contém um nó v_k^{ZERO} para cada *bin* k da solução corrente. Cada nó em $\mathcal{V}$ irá modelar uma possível substituição de um item por outro, e cada nó em $\mathcal{V}^{\text{ZERO}}$ irá modelar uma possível inserção de um item e o final de uma *ejection chain*. O nó v^{SOURCE} representa a fonte. Seja ainda $B(i)$ o *bin* que contém o vértice i na solução. O conjunto A'' contém um arco (i, j) para qualquer par de nós $i \in \mathcal{V} \cup \mathcal{V}^{\text{ZERO}}$ e $j \in \mathcal{V} \cup \mathcal{V}^{\text{ZERO}}$ tal que o *bin* $B(i)$ precede o *bin* $B(j)$ em Π, e um arco (v^{SOURCE}, j) para qualquer $j \in \mathcal{V}$. O custo dos arcos são definidos como segue:

$$c_{ij} = \begin{cases} \text{diferença de custo do bin } B(j) \text{ quando o item } j \text{ é substituído pelo item } i & \text{se } i \in \mathcal{V} \text{ e } j \in \mathcal{V}, \\ \text{diferença de custo do bin } B(j) \text{ quando o item } j \text{ é removido} & \text{se } i \in \mathcal{V}^{\text{ZERO}} \cup v^{\text{SOURCE}} \text{ e } j \in \mathcal{V}, \\ \text{diferença de custo do bin } B(j) \text{ quando o item } i \text{ é inserido} & \text{se } i \in \mathcal{V} \text{ e } j \in \mathcal{V}^{\text{ZERO}}. \end{cases}$$

O caminho mínimo no grafo que vai da fonte até qualquer nó em $\mathcal{V}^{\text{ZERO}}$ é equivalente a uma sequência combinada de movimentos de inserção e remoção de itens. Para encontrar o caminho mínimo no grafo foi utilizado o Algoritmo de Bellman-Ford [Cormen et al., 2009]. Note que, graças ao uso dos nós em $\mathcal{V}^{\text{ZERO}}$, pode-se provavelmente obter uma coleção de várias *ejection chains* disjuntas. Finalmente, a solução é atualizada caso seja melhorada.

3.2.3.3 Vizinhança *Grenade*

A vizinhança GRENADE foi utilizada anteriormente em instâncias do problema de coloração de vértices [Avanthay et al., 2003], e consiste na enumeração de alguns movimentos grandes. No contexto deste trabalho, considera-se cada item problemático $\mathcal{P}$ (como definido na seção anterior: itens alocados a *bins* com peso acima da sua capacidade ou junto com itens com os quais possui conflito). Para cada possível *bin* $B \neq B(k)$, o método avalia a possibilidade de realocação de k em B juntamente com a realocação de qualquer item i em conflito com o item k de B para outro *bin*. O custo total dessas realocações combinadas é avaliado e o melhor movimento de *grenade* para o item k é aplicado se esse for um movimento de melhora.

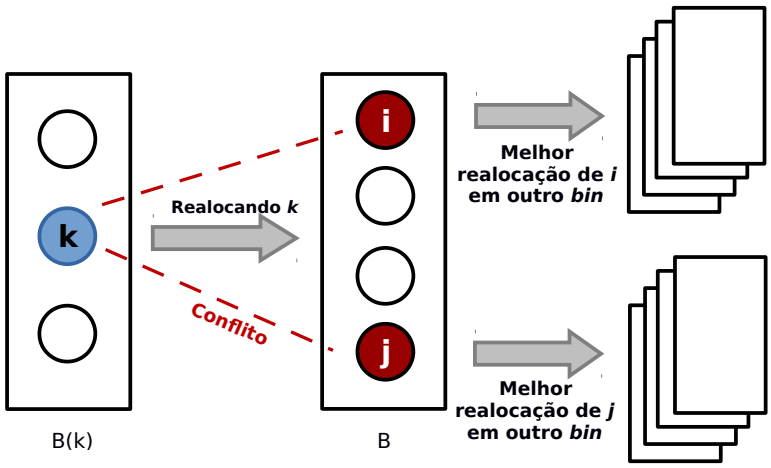


Figura 3.4: Movimento GRENADE. A realocação do item i cria dois conflitos que são resolvidos por outras realocações.

O esquema geral do método é ilustrado na Figura 3.4, e o pseudo-código da exploração da vizinhança é detalhado no Algoritmo 5. Note que a busca consiste de um único laço sobre os itens $k \in \mathcal{P}$, uma descida completa para tentar alcançar o mínimo local desta vizinhança poderia consumir muito tempo.

3.2.3.4 Cobertura de Conjuntos Adaptativo

Formulações do Problema de Cobertura de Conjuntos (e do Particionamento de Conjuntos) tem sido empregadas com sucesso para resolver na otimalidade uma ampla variedade de problemas de otimização combinatória e para gerar vizinhanças largas, como vistos em [Monaci e Toth, 2006], [Muter et al., 2010] e [Subramanian et al., 2013]. Neste trabalho, investigou-se a formulação do problema de cobertura de conjuntos em que o número máximo de *bins* é fixo, e tem por objetivo minimizar a soma dos custos das colunas associadas a

Algoritmo 5 Exploração da vizinhança GRENADE

para cada item $k \in \mathcal{P}$ escolhido aleatoriamente **faça**
  para cada bin $B \neq B(k)$ **faça**
    $\Delta \leftarrow$ Custo de realocar o item k em B
    para cada item $i \in B$ em conflito com k, em uma ordem aleatória **faça**
      $\Delta \leftarrow \Delta +$ Menor custo de realocação de i em um bin diferente de B
    fim para
    AtualizeMelhorMovimentoGrenade(Δ,k)
  fim para
  se MelhorMovimentoGrenade(k) < 0, **então** aplique este movimento
fim para

possíveis violações nas restrições de capacidade, e usá-lo como base para uma vizinhança em larga escala no ILS. Para diminuir o tempo gasto, esse procedimento é utilizado somente apenas algumas vezes durante a busca, após N_{SC} iterações consecutivas da busca local sem melhora.

A formulação do problema de cobertura de conjuntos utilizada é dada nas Equações (3.4–3.7). Considere R como o conjunto de todas as possíveis colunas existentes, onde cada coluna é definida como um subconjunto de itens, e tem associada um custo c_j. O custo c_j é baseado nas mesmas penalidades ponderadas para conflitos e excesso na capacidade dos *bins* como explicitado anteriormente na Equação (3.1). Seja $R_i \subseteq R$ o subconjunto de colunas que contém um item $i \in \mathcal{V}$, o objetivo da formulação é encontrar um conjunto de até K colunas que cubram todos os itens com custo mínimo. Pode-se destacar o fato que colunas de custo positivo (com $c_i > 0$) correspondem a conjuntos de itens com conflito ou excesso de capacidade, e que qualquer solução com custo zero (*custo-0*) corresponderia a uma alocação viável em K *bins*. Uma formulação similar, visando minimizar as penalidade para restrições violadas nos *bins*, é usada em [Monaci e Toth, 2006] e exibida a seguir.

$$\text{Minimizar} \sum_{j \in R} c_j u_j \tag{3.4}$$

Sujeito a:

$$\sum_{j \in R_i} u_j \geqslant 1 \qquad \forall i \in \mathcal{V} \tag{3.5}$$

$$\sum_{j \in R} u_j \leqslant K \tag{3.6}$$

$$u_j \in \{0,1\} \qquad \forall j \in R \tag{3.7}$$

Esta formulação envolve um conjunto de variáveis de decisão de tamanho exponencial $(2^n - 1)$. Para contornar esses problema, o modelo foi restringido a um subconjunto de colunas $R' \subset R$, obtidas de um de mínimo local do ILS. Mais especificamente, antes de cada iteração da perturbação, as colunas da solução corrente são adicionadas no *pool* R', filtrando possíveis colunas repetidas (isso é facilmente implementado por *hashing*), e controlando o tamanho do *pool* até um limite máximo de S_{POOL}. Se o limite for excedido, metade das colunas são descartadas, começando pelas mais antigas. A formulação é resolvida por meio de um resolvedor de programação inteira em um tempo de CPU limitado T_{LIMIT}. Para equilibrar a dificuldade do problema e obter boas soluções com uma formulação que permanece tratável, o tamanho do *pool* de colunas é variável durante a busca. Se o modelo for resolvido na otimalidade durante o tempo limite determinado, então, S_{POOL} é aumentado em 15%. Por outro lado, se o tempo limite for atingido sem encontrar uma solução ótima, então S_{POOL} é reduzido em 15%. A solução ótima fornecida por esse modelo é utilizada como uma nova solução incumbente.

3.2.4 Procedimento de Perturbação

O procedimento de perturbação é aplicado após N_{LS} iterações consecutivas de movimentos de *custo-0*, busca local e de larga escala sem melhora na melhor solução. De fato, os próprios movimentos de *custo-0* permitem significantes modificações na solução, que ajudam a progredir em direção a diferentes áreas do espaço de busca.

Durante a perturbação, S_{SHAK} itens são realocados aleatoriamente, destes tenta-se selecionar ao menos 50% de itens escolhidos aleatoriamente entre os itens com conflitos ou pertencentes a *bins* com violação na capacidade, quando houver. O restante dos itens são escolhidos aleatoriamente. O tamanho do operador de perturbação é um parâmetro importante para a busca. Seu valor foi definido de acordo com o resultado de experimentos computacionais preliminares realizados, como descrito nas próximas seções.

3.3 Experimentos Computacionais

O objetivo desses experimentos é avaliar o desempenho do método proposto para o PBPC, bem como identificar a contribuição relativa de cada vizinhança e o impacto de suas escolhas no projeto do método, colaborando para pesquisas futuras. Os conjunto-testes de instâncias clássicas para o PBPC são divididas em seis conjuntos: (t) e (u) de [Muritiba et al., 2010],

e (ta), (ua), (d), (da) de [Sadykov e Vanderbeck, 2013], totalizando 2060 intâncias com o número de itens variando de 60 a 1000.

As instâncias dos conjuntos (t – *triplet*) e (u – *uniforme*) foram adaptadas das instâncias de [Falkenauer, 1996] para o problema de *bin packing* clássico por [Gendreau et al., 2004]. O procedimento de geração, contudo, continha algumas escolhas aleatórias e as instâncias originais não estão mais disponíveis. Assim, essas instâncias foram geradas novamente em [Muritiba et al., 2010] e [Elhedhli et al., 2011] de acordo com as instruções fornecidas no trabalho de [Gendreau et al., 2004]. Nesta tese foram utilizados os arquivos de [Muritiba et al., 2010], pois isso permite uma comparação com a única metaheurística populacional existente na literatura, bem como com o atual algoritmo do estado da arte de programação matemática – heurística e exato – de [Sadykov e Vanderbeck, 2013].

O conjunto (t) inclui quatro grupos de instâncias com um número de itens variando de 60 a 501. Uma característica que pode-se destacar deste conjunto de instâncias é que, na solução ótima para o problema de *bin packing*, sem as restrições de conflito, os *bins* são preenchidos com exatamente três itens cada. De forma semelhante, o conjunto (u) inclui quatro classes de instâncias, com 120 a 1000 itens. Em ambos os conjuntos, o grafo de conflitos é um grafo de intervalos, com uma densidade que varia de 0% a 90%. Cada conjunto inclui 400 instâncias.

As instâncias dos conjuntos (ta) e (ua) possuem as mesmas características que as instâncias dos conjuntos (t) e (u), mas com um grafo de conflitos arbitrário. Essas instâncias foram geradas por [Sadykov e Vanderbeck, 2013] com densidades das arestas variando de 10% a 90%, totalizando 360 instâncias em cada conjunto. Finalmente, as instâncias dos conjuntos (d) e (da) foram geradas incluindo itens com pesos uniformemente distribuídos entre [500, 2500] em *bins* com capacidade igual a 10000. Essas instâncias incluem entre 120 e 500 itens, com o grafo de conflitos de intervalo (d) ou arbitrário (da). O objetivo dessas instâncias era incluir um maior número médio de itens por *bin*, para obter problemas mais difíceis para os métodos baseados em geração de colunas. Este objetivo foi, no entanto, apenas parcialmente atingido, uma vez que o número médio de itens por *bins* nas melhores soluções conhecidas permanece relativamente pequeno para todos os conjuntos de dados: 2,18, 1,96, 2,91, 2,48, 2,76 e 5,83 para as classes (t), (u), (ta), (ua), (d) e (da), respectivamente. Por esta razão, estes conjuntos de instâncias são ideais para os métodos *branch-and-price* existentes, uma vez que o algoritmo de *pricing* pode gerar rapidamente uma grande quantidade de colunas úteis, levando a um alto desempenho do método, difícil de alcançar por meio de metaheurísticas.

3.3.1 Resultados

O primeiro experimento computacional realizado tem por objetivo comparar o desempenho do ILS com vizinhanças de larga escala proposto com o desempenho das melhores metaheurísticas existentes na literatura, a heurística populacional (PH) de [Muritiba et al., 2010], e o desempenho dos melhores algoritmos heurísticos baseados em programação matemática, a abordagem de *diving* com busca de discrepância limitada (DH-LDS) de [Sadykov e Vanderbeck, 2013]. Para uma comparação precisa, os resultados detalhados destes dois métodos, em cada caso, foram gentilmente fornecidos pelos autores em uma comunicação pessoal. O algoritmo proposto foi codificado em C ++, usando CPLEX 12.5.1, e foi executado em uma única *thread* de uma máquina Xeon X5675 com 3.07 GHz de CPU. Em comparação, os algoritmos PH e DH-LDS foram executados em CPUs Pentium IV 3.0GHz e Xeon X5460 3.16GHz, respectivamente.

Uma calibração preliminar dos parâmetros de busca – tamanho e frequência do operador de perturbação, número de iterações do método, tamanho do subproblema de cobertura de conjuntos – foi realizada com o objetivo de produzir boas soluções em um tempo de CPU comparável com os métodos anteriormente desenvolvidos para instâncias de tamanho médio. Isso levou a um conjunto padrão de parâmetros: $N_{\text{SHAK}} = 50$, $N_{\text{LS}} = 100$, $N_{\text{SC}} = 25$, $S_{\text{POOL}} = 1500$, $T_{\text{LIMIT}} = 20$ e $S_{\text{SHAK}} = 3$. O impacto de qualquer modificação desses parâmetros de busca será investigado em detalhes na Seção 3.3.2. O método proposto foi executado 10 vezes para cada instância com diferentes sementes aleatórias. Esta versão do algoritmo é chamada HILS-*Completo*. Para examinar o impacto da busca utilizando vizinhanças de larga escala na qualidade da solução, também foram relatados os resultados de uma configuração simplificada, ILS-*Simples*, que corresponde ao mesmo algoritmo sem as vizinhanças em larga escala. Os resultados completos desses experimentos são apresentados nas Tabelas A.1 a A.6 (no Apêndice). Essas tabelas exibem os resultados das instâncias agrupadas em grupos de 10 arquivos, uma linha para cada valor de densidade × número de itens. As primeiras três medidas são as médias das 10 instâncias, a última é uma contagem cumulativa:

- **Avg Gap.** A diferença percentual entre a solução média do método em 10 execuções e a melhor solução anteriormente conhecida na literatura (BKS). É calculado como $100(z - z_{BKS})/z_{BKS}$, onde z é o valor da solução encontrada pelo método e z_{BKS} é o valor do BKS;

- **Best Gap.** A diferença percentual entre a melhor solução encontrada pelo método em 10 execuções, e o BKS;

- **T(s).** O tempo de CPU (até o fim da busca);
- **Opt.** O número de instâncias para as quais a solução ótima conhecida foi encontrada pelo menos uma vez pelo método.

Para cada grupo de instâncias, os melhores resultados são destacados em negrito. As duas últimas colunas indicam o número médio de *bins* nas melhores soluções conhecidas da literatura, e o número de soluções ótimas conhecidas para o grupo. Finalmente, foram incluídas duas tabelas que apresentam as mesmas medidas com um nível de agregação mais elevado: por classe de instância × número de itens na Tabela 3.1, e por classe de instância × densidade na Tabela 3.2.

Inst.		HILS-*Completo*				ILS-*Simples*				PH			DH-LDS			BKS	
Classe	n	Avg Gap	Best Gap	T(s)	Opt	Avg Gap	Best Gap	T(s)	Opt	Gap	T(s)	Opt	Gap	T(s)	Opt	Avg Bins	Opt
t	60	**0,00**	**0,00**	1,26	90	0,42	0,22	0,72	86	0,50	41,68	81	**0,00**	0,20	90	33,40	90
	120	0,01	**0,00**	5,17	90	0,63	0,39	2,34	76	0,66	44,44	66	0,03	1,36	89	66,11	90
	249	0,08	0,05	17,27	86	0,35	0,15	5,09	76	0,44	57,61	58	**0,00**	7,81	90	135,83	90
	501	0,01	**0,00**	23,88	90	0,26	0,10	13,19	66	0,23	65,48	60	**0,00**	50,43	90	275,69	90
u	120	0,03	**0,00**	3,67	90	0,06	0,02	1,22	89	0,11	24,58	85	**0,00**	0,76	90	70,38	90
	250	0,01	**0,00**	17,57	90	0,13	0,03	4,67	86	0,19	49,81	71	**0,00**	3,34	90	143,72	90
	500	0,01	0,01	37,56	89	0,37	0,20	16,47	57	0,18	66,44	65	**0,00**	18,74	90	286,03	90
	1000	0,01	0,01	90,07	88	0,53	0,34	65,07	38	0,20	105,34	56	**0,00**	116,71	90	571,88	90
ta	60	**0,00**	**0,00**	1,85	90	0,09	0,06	0,78	89	–	–	–	**0,00**	0,16	90	21,87	90
	120	0,05	**0,00**	23,34	90	1,18	1,01	3,41	53	–	–	–	0,05	1,46	88	41,39	90
	249	0,49	0,32	75,17	64	0,94	0,77	12,88	50	–	–	–	**0,29**	23,67	67	83,80	88
	501	0,40	0,35	128,30	45	0,57	0,53	32,87	45	–	–	–	**0,15**	271,33	48	167,61	70
ua	120	**0,00**	**0,00**	3,26	90	0,52	0,38	1,37	73	–	–	–	**0,00**	0,73	90	49,27	90
	250	0,02	**0,00**	28,70	89	0,55	0,48	6,83	60	–	–	–	0,02	6,21	87	100,64	89
	500	0,08	0,05	154,09	74	0,38	0,35	26,62	62	–	–	–	**0,01**	63,10	80	200,77	82
	1000	0,15	0,06	399,37	68	0,27	0,23	55,74	68	–	–	–	**0,01**	516,75	79	400,03	82
d	120	**0,00**	**0,00**	3,17	90	0,04	**0,00**	1,01	90	–	–	–	–	–	–	61,82	90
	250	**0,00**	**0,00**	11,57	90	0,08	0,03	2,77	87	–	–	–	–	–	–	127,93	90
	500	**0,00**	**0,00**	23,35	90	0,10	0,02	8,96	87	–	–	–	–	–	–	252,79	90
da	120	0,80	0,18	12,91	63	0,98	0,39	1,18	61	–	–	–	**0,12**	6,02	63	23,63	67
	250	1,14	0,52	25,50	45	1,32	0,73	4,88	42	–	–	–	**0,04**	59,60	49	44,66	50
	500	1,83	1,38	53,31	38	1,85	1,41	18,12	37	–	–	–	**0,10**	541,48	47	84,12	49

Tabela 3.1: Resultados para cada conjunto de instâncias e tamanho n.

Dois fatores são determinantes para o desempenho de uma método em otimização: a qualidade de suas soluções, e a escalabilidade do método, ou seja, a rapidez com que o tempo de CPU cresce para instâncias grandes do problema. A partir destes resultados, podem ser feitas as seguintes observações.

Em termos de qualidade da solução, o HILS-*Completo* fornece soluções de qualidade superior quando comparado a melhor metaheurística da literatura (PH) e uma qualidade

Inst.		HILS-*Completo*				ILS-*Simples*				PH			DH-LDS			BKS	
Classe	ρ	Avg Gap	Best Gap	T(s)	Avg Opt	Avg Gap	Best Gap	T(s)	Opt	Gap	T(s)	Opt	Gap	T(s)	Opt	Avg Bins	Opt
t	0	**0,00**	**0,00**	0,93	40	1,44	0,75	0,97	33	**0,00**	0,00	40	–	–	–	77,50	40
	10	**0,00**	**0,00**	1,05	40	1,26	0,69	1,12	33	2,08	134,03	2	**0,00**	11,93	40	77,55	40
	20	**0,00**	**0,00**	1,77	40	0,72	0,50	1,67	32	1,20	131,98	9	**0,00**	21,93	40	77,73	40
	30	0,21	0,12	19,99	36	0,45	0,34	7,88	32	0,71	115,05	18	**0,06**	24,15	39	78,33	40
	40	**0,00**	**0,00**	8,78	40	0,23	0,07	4,09	36	0,14	24,55	36	**0,00**	19,30	40	94,10	40
	50	**0,00**	**0,00**	11,35	40	0,29	0,12	5,00	32	**0,00**	12,73	40	**0,00**	18,69	40	117,93	40
	60	**0,00**	**0,00**	13,10	40	0,28	0,08	5,87	35	**0,00**	21,98	40	**0,00**	14,18	40	141,53	40
	70	**0,00**	**0,00**	15,85	40	0,27	0,10	6,89	30	**0,00**	16,10	40	**0,00**	10,31	40	164,33	40
	80	**0,00**	**0,00**	16,94	40	0,15	0,03	7,36	37	**0,00**	10,18	40	**0,00**	8,57	40	187,78	40
	90	**0,00**	**0,00**	18,22	40	0,08	0,02	8,17	37	**0,00**	4,15	40	**0,00**	5,51	40	210,58	40
u	0	**0,00**	**0,00**	0,56	40	**0,00**	**0,00**	0,26	40	0,31	75,95	18	–	–	–	188,53	40
	10	**0,00**	**0,00**	0,60	40	**0,00**	**0,00**	0,28	40	0,11	57,33	31	**0,00**	29,43	40	188,53	40
	20	**0,00**	**0,00**	0,60	40	**0,00**	**0,00**	0,27	40	0,10	57,78	32	**0,00**	31,63	40	188,53	40
	30	0,05	**0,00**	2,65	40	0,12	0,07	1,32	38	0,27	76,15	25	**0,00**	37,26	40	188,53	40
	40	0,09	0,02	58,14	37	0,39	0,21	20,12	28	0,79	91,95	14	**0,00**	57,83	40	194,03	40
	50	**0,00**	**0,00**	41,99	40	0,62	0,40	22,43	21	0,10	46,80	34	**0,00**	50,78	40	236,80	40
	60	**0,00**	**0,00**	44,72	40	0,36	0,17	27,02	28	0,06	61,93	34	**0,00**	34,85	40	284,80	40
	70	0,01	**0,00**	62,13	40	0,45	0,23	35,54	25	0,06	64,40	32	**0,00**	28,84	40	330,43	40
	80	**0,00**	**0,00**	68,64	40	0,35	0,19	42,03	21	0,03	59,70	36	**0,00**	23,78	40	376,73	40
	90	**0,00**	**0,00**	55,48	40	0,16	0,07	47,72	29	**0,00**	37,88	39	**0,00**	19,57	40	423,68	40
ta	10	**0,00**	**0,00**	1,52	40	0,74	0,63	1,42	31	–	–	–	**0,00**	51,24	40	77,73	40
	20	**0,00**	**0,00**	2,46	40	0,57	0,56	1,84	31	–	–	–	**0,00**	42,81	40	77,78	40
	30	0,33	**0,00**	16,99	40	0,76	0,53	6,73	31	–	–	–	0,11	72,27	36	77,80	40
	40	0,28	**0,27**	34,60	31	0,28	0,27	13,58	31	–	–	–	**0,27**	112,75	28	78,03	40
	50	**0,42**	**0,42**	74,85	21	0,42	0,42	25,38	21	–	–	–	**0,42**	120,84	21	78,03	40
	60	**0,06**	**0,06**	93,88	32	**0,06**	**0,06**	20,04	32	–	–	–	**0,06**	80,08	32	78,40	36
	70	**0,00**	**0,00**	107,48	32	0,10	**0,00**	16,37	32	–	–	–	**0,00**	86,29	32	78,50	32
	80	0,44	0,33	82,58	24	1,66	1,44	16,72	11	–	–	–	**0,12**	9,40	38	79,00	40
	90	0,59	0,43	100,13	29	1,66	1,41	10,27	17	–	–	–	**0,13**	91,72	26	82,75	30
ua	10	0,01	**0,00**	2,46	40	0,02	**0,00**	1,14	40	–	–	–	**0,00**	74,71	40	187,58	40
	20	0,01	**0,00**	0,81	40	0,01	**0,00**	0,48	40	–	–	–	**0,00**	74,44	40	188,35	40
	30	**0,00**	**0,00**	0,71	40	**0,00**	**0,00**	0,40	40	–	–	–	**0,00**	69,08	40	187,90	40
	40	**0,00**	**0,00**	3,69	40	0,06	0,05	1,79	39	–	–	–	**0,00**	69,72	40	186,78	40
	50	**0,00**	**0,00**	4,07	40	0,05	0,05	1,72	39	–	–	–	**0,00**	66,68	40	186,93	40
	60	0,02	0,01	13,52	39	0,18	0,17	8,54	35	–	–	–	**0,00**	81,51	40	187,03	40
	70	0,10	0,06	75,84	33	0,54	0,49	39,83	19	–	–	–	**0,03**	106,85	38	187,48	39
	80	0,18	0,13	306,37	23	1,08	0,83	86,13	5	–	–	–	**0,00**	171,81	34	187,75	34
	90	0,26	**0,04**	909,73	26	1,91	1,64	63,73	6	–	–	–	0,07	605,47	24	189,33	30
d	10	**0,00**	**0,00**	0,00	30	**0,00**	**0,00**	0,00	30	–	–	–	–	–	–	43,90	30
	20	**0,00**	**0,00**	5,22	30	0,06	**0,00**	2,52	30	–	–	–	–	–	–	58,67	30
	30	0,01	**0,00**	9,31	30	0,13	0,04	3,13	28	–	–	–	–	–	–	87,67	30
	40	**0,00**	**0,00**	10,70	30	0,17	0,03	3,57	29	–	–	–	–	–	–	114,30	30
	50	**0,00**	**0,00**	13,59	30	0,02	**0,00**	4,05	30	–	–	–	–	–	–	147,63	30
	60	**0,00**	**0,00**	16,14	30	0,11	0,05	4,82	28	–	–	–	–	–	–	174,07	30
	70	**0,00**	**0,00**	17,90	30	0,11	0,01	5,81	29	–	–	–	–	–	–	204,40	30
	80	**0,00**	**0,00**	19,58	30	0,03	**0,00**	6,50	30	–	–	–	–	–	–	235,07	30
	90	**0,00**	**0,00**	21,82	30	0,04	**0,00**	7,83	30	–	–	–	–	–	–	261,93	30
da	10	**0,00**	**0,00**	0,00	30	**0,00**	**0,00**	0,00	30	–	–	–	0,04	92,49	29	43,90	30
	20	**0,00**	**0,00**	0,00	30	**0,00**	**0,00**	0,00	30	–	–	–	0,09	84,50	29	43,80	30
	30	0,38	0,38	3,29	28	0,38	0,38	0,22	28	–	–	–	**0,00**	36,71	30	44,17	30
	40	0,55	0,27	18,54	26	0,69	0,55	4,09	24	–	–	–	**0,00**	79,81	28	44,03	28
	50	1,97	1,06	67,05	11	2,40	1,46	10,05	7	–	–	–	**0,00**	81,00	24	44,43	24
	60	2,30	1,38	74,63	0	2,65	1,59	12,14	0	–	–	–	**0,04**	384,41	2	45,50	3
	70	3,12	1,77	47,54	3	3,27	1,94	15,25	3	–	–	–	**0,04**	543,40	3	49,77	3
	80	2,43	1,36	35,15	8	2,52	1,62	16,01	8	–	–	–	**0,21**	348,24	7	60,30	8
	90	0,55	**0,01**	28,96	10	0,57	0,05	14,78	10	–	–	–	0,34	170,71	7	81,33	10

Tabela 3.2: Resultados para cada conjunto de instâncias e densidade ρ.

ligeiramente inferior que o algoritmo de *diving* baseado em geração de colunas (DH-LDS). A magnitude das diferenças depende dos conjuntos de instâncias. Para os conjuntos (t) e (u), que são comuns aos três algoritmos, HILS-*Completo* encontra um gap médio de 0,02%, em comparação com 0,29% e 0,00% do PH e DH-LDS, respectivamente. Para os conjuntos (ta) e (ua), a diferença da qualidade da solução também é pequena, e ambos os métodos retornam soluções quase ótimas com um gap médio abaixo de 0,15%. Para as instâncias (da), o DH-LDS produz soluções de maior qualidade (um gap de 1,17% em média) do que HILS-*Completo*, embora ao custo de um tempo computacional maior (10×) para as instâncias grandes. Ainda assim, HILS-*Completo* atingiu 1819 das 1917 soluções ótimas conhecidas, e um total de 9 novos limites superiores melhores também são encontrados para algumas instâncias (da) em aberto.

Observe que o DH-DLS executa uma busca truncada na árvore de *branch-and-price*, que perde sua garantia de otimalidade, mas muitas vezes consegue encontrar soluções quase ótimas quando o tamanho do problema permite. No entanto, à medida que n cresce, seu tempo de CPU se deteriora muito rapidamente como observado em [Sadykov e Vanderbeck, 2013]: "uma desvantagem de nossa heurística primitiva é que o tempo de execução aumenta rapidamente com o número de itens". Como tal, as metaheurísticas permanecem essenciais para as instâncias do PBPC onde os métodos de programação matemática se tornam impraticáveis (grande número de itens ou itens/*bin*), abrindo o caminho para pesquisas sobre métodos híbridos que buscam tirar proveito tanto da busca local quanto da programação matemática em uma tempo melhor controlado.

Para avaliar a escalabilidade dos métodos em maiores detalhes, a Figura 3.5 exibe o tempo de CPU do DH-DLS e do HILS-*Completo* para cada conjunto de instâncias e tamanho existente. O tempo de CPU do DH-DLS é inicialmente pequeno para instâncias pequenas, mas tende a aumentar em um fator 10 cada vez que o tamanho da instância duplica. Esse efeito é especialmente observado no conjunto de instâncias (da), com uma proporção maior de itens por *bin* e com o grafo de conflitos arbitrários. Para confirmar estas observações, ajustou-se o tempo de CPU como uma lei de potência $f(n) = \alpha \times n^{\beta}$ do número de itens n para cada classe de instância (regressão de mínimos quadrados de uma função afim no grafo log-log). O tempo de CPU observado no HILS-*Completo* cresce em $\mathcal{O}(n^{1,49})$. Em contraste com o tempo do DH-DLS, que cresce a uma taxa mais rápida do que a cúbica para alguns conjuntos de instâncias, com leis de potência na forma n^{β} onde $2{,}45 \leq \beta \leq 3{,}62$.

Analisou-se também o impacto de algumas características da instância sobre o desempenho do HILS-*Completo*. A Figura 3.6 exibe os *boxplots* do gap percentual para cada combinação

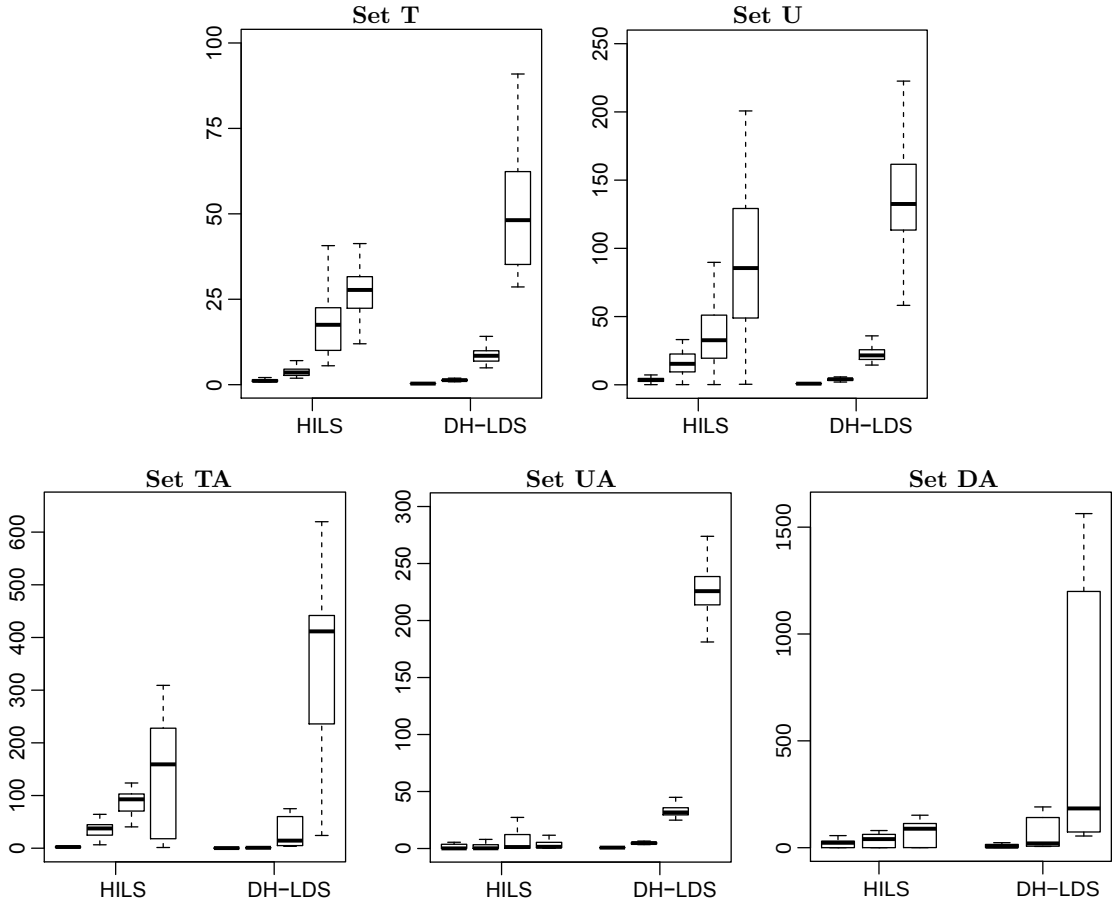


Figura 3.5: tempo de CPU, em segundos, do HILS-*Completo* e do DH-LDS. Um *boxplot* para cada grupo × tamanho da instância em termos de número de itens.

de classe × número de itens. Para os conjuntos de dados (t), (u) e (d) com o grafo de conflitos de intervalo, a solução ótima foi atingida pelo HILS-*Completo* em quase todas as instâncias (1063/1070). Como consequência, os *boxplots* são representados como uma única linha com um gap de 0,00% e qualquer outro valor de gap aparece como um *outlier*. Os outros conjuntos de dados, especialmente (ta) e (da) com grafo de conflitos arbitrário, são mais difíceis, de modo que o gap de 0,00% não representa mais o comportamento médio do método. A dificuldade das instâncias tende a crescer com seu tamanho, embora esse efeito seja contrabalançado pela forma como o "gap" é definido: em pequenas instâncias com poucos *bins* na solução ótima, um *erro* de um *bin* se traduz diretamente em um gap grande (por exemplo, 5% se o BKS possui 20 *bins*) levando a alguns *outliers* para problemas pequenos.

A classe da instância e seu tamanho não são os únicos fatores que afetam o desempenho dos métodos. A densidade do grafo de conflitos também é importante. Esse efeito é ilustrado na Figura 3.7, que exibe os *boxplots* do gap percentual em função da densidade. Geralmente,

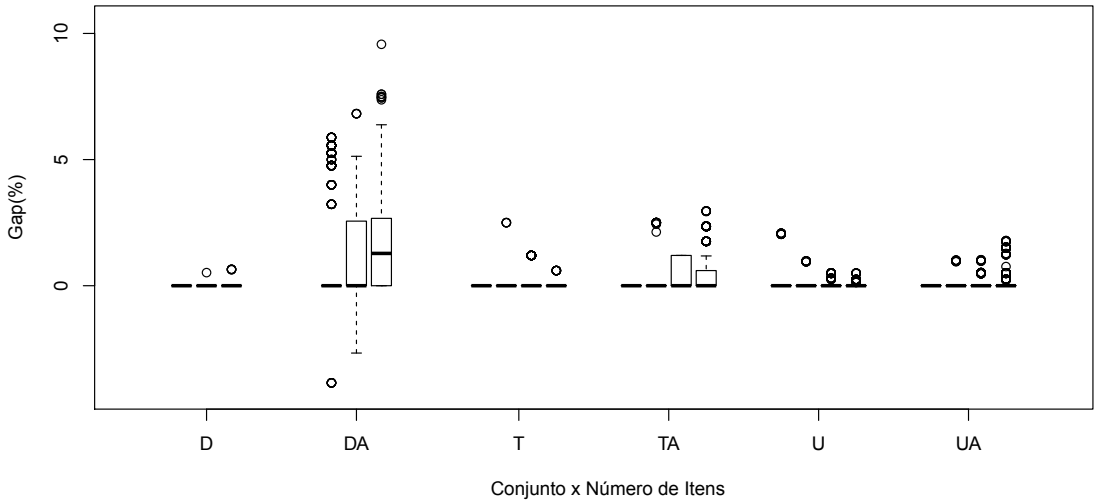


Figura 3.6: HILS-*Completo* GAP para cada grupo de instância.

as instâncias mais difíceis não são aquelas com baixa densidade (equivalente ao problema de *bin packing*), nem aqueles com alta densidade (equivalente a um problema de coloração com restrições bem apertadas), mas aquelas que combinam os efeitos de restrições de *bin packing* e conflito em um nível de densidade entre 50% e 80%. HILS-*Completo* parece funcionar muito bem para instâncias do problema com baixa densidade (até 20%). Nestes casos, a BKS foi sistematicamente alcançada e até mesmo melhorada, como destacado pelos dois *outliers* localizados abaixo de 0,00%.

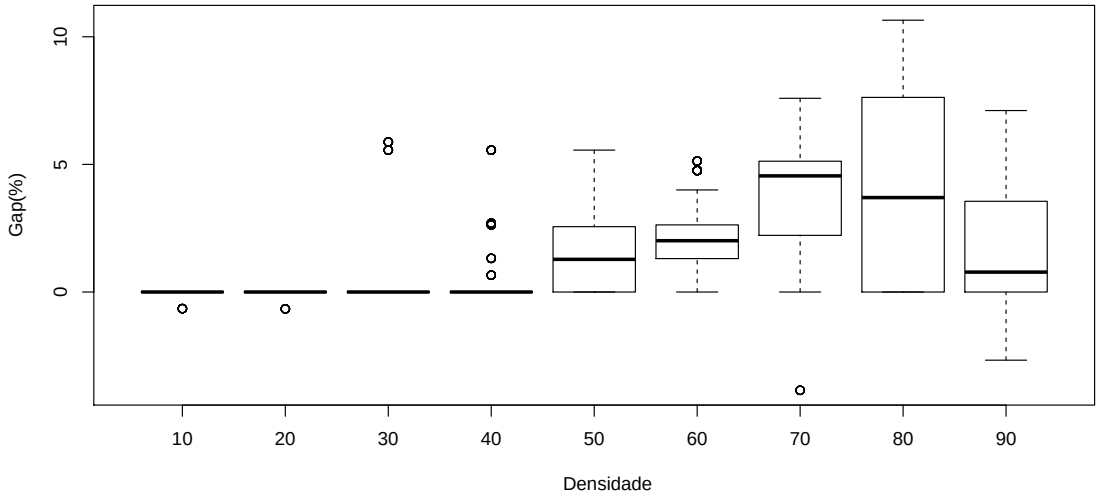


Figura 3.7: Gap do HILS-*Completo* por densidade do grafo de conflitos.

Foram comparados os resultados do ILS-*Simples* e do HILS-*Completo* para avaliar o impacto das vizinhanças em larga escala. Observou-se diferenças significativas na qualidade da solução, confirmada por um teste pareado de Wilcoxon (p-valor $\leq 10^{-64}$). As vizinhanças

de larga escala têm um impacto positivo na qualidade da solução, mas consomem um tempo de CPU adicional (49,85 segundos em média por instância com as vizinhanças em larga escala e 12,53 segundos sem). Observe que experimentos paralelos também foram realizados para investigar se um aumento no número de iterações do ILS-*Simples* poderia obter algum benefício deste tempo de CPU adicional para alcançar melhores soluções, mas a melhoria na qualidade foi pequena quando o número de iterações foi duplicado (ver Seção 3.3.2).

Finalmente, o tempo gasto por cada componente do método foi analisado. Os resultados desta análise podem ser vistos na Figura 3.8, que exibe a porcentagem do tempo de CPU de cada um dos principais componentes do método proposto. Essa métrica foi agregada por classe de instância × número de itens. O esforço da busca parece estar bem distribuído entre seus componentes, isto é, nenhuma vizinhança específica consome a maior parte do tempo. No geral, a vizinhança de *assignment*, a busca local e os movimentos de custo zero são os componentes que consomem maior parte do tempo. Em contraste, *ejection chains* e a vizinhança *grenade* usam menos de 6% e 4% do tempo de CPU, respectivamente. O esforço computacional dos movimentos de *custo-0* se refere principalmente ao fato deste ser utilizado no primeiro laço da busca local e, portanto, envolver muitas avaliações de movimentos que são posteriormente conhecidos como de não-melhora e, portanto, podados em subsequentes iterações da busca local. De acordo com as discussões anteriores, também observou-se que a vizinhança baseada na cobertura de conjuntos requer uma quantidade maior de tempo de CPU para instâncias difíceis que possuem um grafo de conflito arbitrário, como em (ta) e (da).

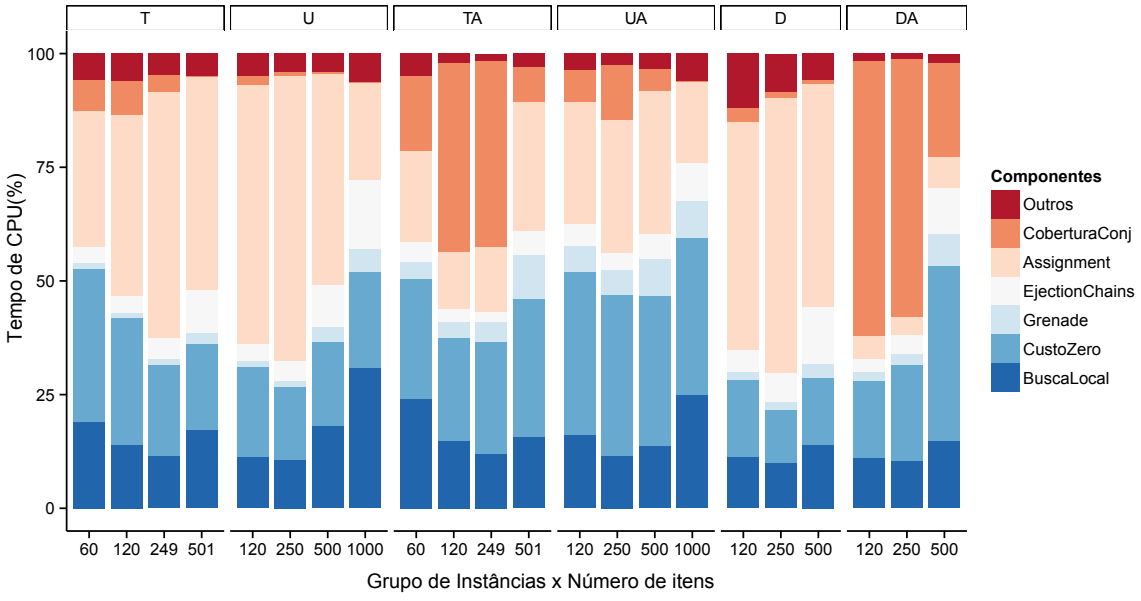


Figura 3.8: Tempo de execução de cada componente do HILS-*Completo*.

No geral, a partir destes experimentos pode-se afirmar que o HILS-*Completo* é um método alternativo importante para o BPPC, produzindo soluções de alta qualidade com um gap total de 0,22% do BKS, e superando a melhor metaheurística anteriormente existente para o problema com um tempo de CPU mais controlado do que os métodos de programação matemática existentes. Possuir dois métodos competitivos (perto da otimalidade com o DH-LDS e com boa escalabilidade com o HILS) é muito positivo para trabalhos futuros, que podem explorar novas hibridizações, bem como possíveis conceitos de população. Estas opções foram mantidas à parte neste trabalho, a fim de se concentrar melhor nos aspectos de vizinhança. Uma vez que essas vizinhanças são o ponto central na concepção do método, é importante investigar os componentes da busca mais promissores e medir o impacto de alguns parâmetros-chave do método. Este é o assunto de uma análise detalhada na próxima seção.

3.3.2 Análise de Sensibilidade – Parâmetros da Busca e das Vizinhanças

Esta seção analisa a sensibilidade do método a uma mudança em seus parâmetros, e avalia a contribuição de seus componentes principais e suas vizinhanças. A partir da configuração padrão HILS-*Completo*, variou-se um parâmetro/fator de cada vez (abordagem OFAT) e testaram-se as configurações resultantes. Esses experimentos são resumidos da Tabela 3.3 a 3.5. Cada linha apresenta o desempenho de uma configuração alternativa do método (indexado de A a R), em termos de gap médio e tempo, nos seis conjuntos de instâncias. Para manter um esforço computacional razoável, uma única execução foi realizada para cada configuração nas 2060 instâncias.

Impacto dos parâmetros gerais da busca. A primeira investigação realizada diz respeito ao operador de perturbação e ao critério de parada, estes parâmetros são recorrentes em todas as metaheurísticas baseadas na metaherística *Iterated Local Search*. Nas duas primeiras configurações alternativas, a periodicidade do operador de perturbação foi modificada, isto é, o número N_{LS} de movimentos de custo zero e fases de busca local realizadas antes da perturbação. Este parâmetro é dividido pela metade na configuração A e duplicado na configuração B. Uma vez que o produto $N_{\text{SHAK}} \times N_{\text{LS}}$ controla o número total de descidas da busca local antes da terminação, este é mantido constante, ajustando-se N_{SHAK}. As configurações C e D tem por objetivo investigar o impacto de execuções mais longas e curtas, respectivamente, duplicando ou reduzindo o valor de N_{SHAK} sem afetar N_{LS}. Finalmente, as configurações E, F e G investigam valores alternativos para o operador de perturbação, o que pode envolver executar uma perturbação mais forte para uma exploração melhor do

espaço de busca ($S_{\text{SHAK}} = 5$) ou uma perturbação mais fraca para permitir uma intensificação adicional ($S_{\text{SHAK}} = 1$ ou 2).

	N_{SHAK}	N_{LS}	S_{SHAK}	(t)		(u)		(ta)		(ua)		(d)		(da)	
				Gap	Tempo	Gap	Tempo	Gap	Tempo	Gap	Tempo	Gap	Tempo	Gap	Tempo
Padrão	50	100	3	**0,02**	9,97	0,01	34,44	0,23	52,22	0,05	190,40	**0,00**	11,78	1,16	31,45
A. Mais Frequente	100	50	3	**0,02**	13,76	0,01	42,16	0,22	80,40	0,04	250,58	**0,00**	13,81	1,16	57,97
B. Menos Frequente	25	200	3	0,04	10,96	0,02	32,58	0,26	46,08	0,10	73,39	**0,00**	13,05	1,40	22,70
C. Exe. Longa	100	100	3	**0,02**	21,03	**0,00**	67,83	0,21	97,67	**0,03**	382,53	**0,00**	25,35	**1,09**	59,95
D. Exe. Curta	25	100	3	0,04	5,65	0,02	20,18	0,28	34,69	0,10	50,05	0,01	6,45	1,47	15.60
E. Perturb. Forte	50	100	5	0,03	10,93	0,01	37,71	0,24	56,82	0,09	111,00	**0,00**	12,87	1,17	31,18
F. Perturb. Fraca 1	50	100	1	**0,02**	10,56	0,01	31,57	**0,18**	76,25	0,06	166,64	**0,00**	12,47	1,42	25,78
G. Perturb. Fraca 2	50	100	2	0,03	11,03	0,01	32,81	0,24	62,01	0,07	132,53	**0,00**	12,73	1,29	30,31

Tabela 3.3: Impacto de algumas variações dos parâmetros da perturbação e de terminação do algoritmo.

As alterações nos parâmetros realizadas nas configurações A, B, E, F e G são em geral prejudiciais para o desempenho do método. Parece que a execução de uma perturbação mais forte, ou recorrer a este operador com mais frequência tem apenas um pequeno impacto sobre o desempenho, mas o processo inverso teria um impacto negativo maior. A configuração C permite uma execução mais longa e, assim, leva naturalmente a melhores soluções. Esta melhoria da qualidade da solução, no entanto, permanece pequena em comparação com o tempo de CPU adicional utilizado.

Contribuição das vizinhanças em larga escala. O segundo experimento tem como objetivo avaliar o papel de cada vizinhança larga, bem como os movimentos de *custo-0*, que são críticos para a diversificação da busca. Assim, nas configurações de H a K, cada uma das quatro vizinhanças largas foram desativadas. Na configuração L, todas as vizinhanças de larga escala foram desativadas. Finalmente, não são aplicados movimentos de *custo-0* na configuração N.

Todos os componentes da busca parecem contribuir positivamente para a qualidade final da solução. Por ordem de grandeza, a maior perda ocorre ao desativar os movimentos *custo-0*, pois contribuem muito significativamente para a diversidade da busca. Na segunda posição, desativar todos as vizinhanças de larga escala (como em ILS-*Simples*) também leva a uma grande perda de qualidade da solução, mas também a um ganho de tempo computacional. Foram testadas longas execuções do algoritmo (configuração M, através de um aumento quádruplo de N_{SHAK}) sem as vizinhanças em larga escala, e isto não foi suficiente para recuperar a qualidade da solução do método completo.

	(t)		(u)		(ta)		(ua)		(d)		(da)	
	Gap	Tempo	Gap	Tempo	Gap	Tempo	Gap	Tempo	Gap	Tempo	Gap	Tempo
Padrão	**0,02**	9,97	**0,01**	34,44	**0,23**	52,22	**0,05**	190,40	**0,00**	11,78	**1,16**	31,45
H. Sem Assign	0,03	5,21	**0,01**	24,38	0,27	43,48	0,10	61,89	**0,00**	4,73	1,29	28,37
I. Sem Ejection Chains	**0,02**	9,99	**0,01**	30,92	0,24	56,67	0,07	108,31	**0,00**	11,58	1,23	28,87
J. Sem Grenade	**0,02**	9,84	0,02	34,99	0,25	46,49	0,07	70,97	0,01	11,58	1,21	29,51
K. Sem Cobertura Conj.	0,37	9,28	0,03	33,16	0,54	33,76	0,30	46,34	**0,00**	11,88	1,40	13,75
L. Sem Vizinhanças Largas	0,49	4,50	0,24	18,66	0,72	11,35	0,43	18,98	0,07	3,97	1,36	7,39
M. Sem Vizinhanças Largas ++	0,39	50,21	0,15	65,62	0.65	27,61	0,39	47,52	0,05	15,56	1,22	16,33
N. Sem custo-0	0,10	9,52	0,02	32,63	0,31	57,97	0,07	172,05	**0,00**	10,15	3,16	17,06

Tabela 3.4: Análise de sensibilidade quando algumas vizinhanças são desativadas.

A vizinhança em larga escala baseada no modelo de PI para o problema de cobertura de conjuntos tem o maior impacto na busca, especialmente para os conjuntos de instâncias (t) e (ua). Conforme observado na Figura 3.8, nessas instâncias o problema de cobertura de conjuntos é convenientemente resolvido com um tempo de CPU pequeno. Isso permite usar um conjunto maior de colunas, aumentando assim as chances de melhoria da solução. Finalmente, as últimas três vizinhanças: *ejection chains*, *assignment* e *grenade* têm um impacto mais moderado na qualidade da solução. Suas contribuições só são observadas em instâncias mais difíceis: (ta), (ua) e (da). No entanto, note que a remoção destas três vizinhanças juntas é mais prejudicial (compare as configurações K e L). As três vizinhanças oferecem diferentes formas de melhoria e diversificação nas soluções, que não são críticas quando tomadas uma a uma, mas muito mais significativas quando consideradas como um todo.

Impacto dos parâmetros da cobertura de conjuntos. Os últimos parâmetros do método referem-se à vizinhança de cobertura de conjuntos. A calibração preliminar realizada no método levou a um limite de tempo $T_{\text{LIMIT}} = 20$ segundos para o resolvedor, bem como a um mecanismo adaptativo para o tamanho do conjunto de colunas (*pool*), a partir de um tamanho de $S_{\text{POOL}} = 1500$ colunas. Assim, foram testados o impacto deste mecanismo adaptativo nas configurações de O a Q, avaliando três valores do tamanho de *pool* fixado em 1000, 1500 e 2000 colunas, juntamente com um tempo de 10, 20 e 60 segundos, respectivamente, para a resolução do problema de cobertura de conjuntos. Finalmente, a última configuração R considera uma possível substituição da cobertura de conjuntos por uma formulação de particionamento de conjuntos definida, usando uma igualdade na Equação (3.5).

Nestes experimentos, pode-se observar que o tamanho do *pool* é um parâmetro muito sensível. Um valor pequeno leva a problemas de cobertura de conjuntos simples que não resultarão em melhorias na solução, mesmo quando resolvido até a otimalidade, enquanto um

	Adapt	S_{POOL}	T_{LIMIT}	(t)		(u)		(ta)		(ua)		(d)		(da)	
				Gap	Tempo	Gap	Tempo	Gap	Tempo	Gap	Tempo	Gap	Tempo	Gap	Tempo
Padrão	Sim	1500	20	**0,02**	9,97	**0,01**	34,44	0,23	52,22	**0,05**	190,40	**0,00**	11,78	**1,16**	31,45
O. Pool Fixo –	Não	1000	10	0,05	9,80	0,02	34,14	0,33	41,57	0,16	44,97	**0,00**	11,84	1,30	16,58
P. Pool Fixo	Não	1500	20	0,04	9,85	0,02	35,18	0,26	54,75	0,14	48,92	**0,00**	11,80	1,20	24,84
Q. Pool Fixo +	Não	2000	60	0,03	11,42	0,02	34,29	**0,20**	95,66	0,11	71,90	**0,00**	11,93	1,28	58,21
R. Particionamento Conj.	Sim	1500	20	**0,02**	10,19	**0,01**	33,85	0,22	52,82	0,07	122,83	**0,00**	12,13	1,20	27,71

Tabela 3.5: Análise de Sensibilidade nos parâmetros da vizinhança de cobertura de conjuntos.

valor maior abre o caminho para mais oportunidades de melhoria, mas com o risco de não resolver o modelo matemático no tempo determinado. Para as instâncias do conjunto (da), um tamanho de *pool* intermediário de 1500 parece ser apropriado, enquanto para outras instâncias, por exemplo, (ta) e (ua), um tamanho de *pool* maior com 2000 colunas leva a soluções de melhor qualidade mas com um tempo computacional adicional. Claramente, a melhor configuração do parâmetro depende das características das instâncias. Isto leva a utilização do mecanismo adaptativo no método, que possui um desempenho melhor globalmente do que cada uma das configurações estáticas. Finalmente, a utilização do modelo de particionamento de conjuntos ao invés do modelo de cobertura de conjuntos não conduziu a uma diferença significativa na qualidade da solução.

Capítulo 4

Um Abordagem Exata para o Problema de Bin Packing com Dependências

O Problema de *Bin Packing* com Dependências (PBPD) possui uma estrutura diferente da apresentada pelo problema de *bin packing* com conflitos, no sentido que uma simples realocação de um item em outro *bin* necessita de realocações simultâneas de possivelmente muitos outros itens em conjunto. Esta tarefa pode ser difícil de ser elaborada e computacionalmente custosa em abordagens baseadas em metaheurísticas sem a inclusão de novas vizinhanças avançadas que sejam baseadas na realocação de cadeias de itens inter-dependentes. Deste modo, para produzir resultados iniciais para o PBPD, neste trabalho considerou-se o uso de resolvedores de programação inteira que estão disponíveis, tais como o CPLEX, para implementar a formulação compacta da Seção 2.2. Porém, os resultados obtidos por essa formulação são limitados a instâncias de pequeno e médio porte, como serão vistos na Seção 4.4.2, contendo os experimentos computacionais realizados.

Como uma alternativa a esses métodos, considera-se ainda neste trabalho o desenvolvimento de outra abordagem de programação matemática, que se beneficia do desenvolvimento recente de *frameworks Branch-and-Price* (B&P) e onde as dependências podem ser mais facilmente integradas. De modo geral, o uso desse tipo de abordagem tem apresentado bons resultados para diversas variantes do problema de *bin packing* [Delorme et al., 2016, Sadykov e Vanderbeck, 2013, Muritiba et al., 2010]. O método desenvolvido nesta tese utiliza o algoritmo de *branch-and-price* para resolver o PBPD, onde uma heurística de *pricing* baseada no algoritmo de Busca em Profundidade (ou *Depth First Search (DFS)*) foi implementada. O método desenvolvido utiliza uma implementação do *branch-and-price* chamada BaPCod [Vanderbeck, 2011, Sadykov e Vanderbeck, 2013].

Este capítulo está organizado do seguinte modo: inicialmente é feita uma pequena revisão sobre os métodos exatos usados na abordagem proposta. Logo após, são apresentados os modelos matemáticos desenvolvidos para o problema. A seguir, é descrito como o *branch-and-price* foi aplicado ao PBPD. Existe ainda uma descrição das instâncias criadas para o problema e, por fim, são apresentados os resultados experimentais obtidos com o algoritmo desenvolvido.

4.1 Revisão de Conceitos

Esforços na tentativa de desenvolver métodos para a resolução de problemas de programação linear inteira e de programação linear inteira mista têm sido realizados ao longo dos anos. A partir da década de noventa, o uso de abordagens que utilizam o *branch-and-price* começaram a ser mais amplamente investigados.

Branch-and-price é uma técnica que combina o algoritmo de *Branch-and-Bound*(B&B) com o método de geração de colunas para resolver problemas com uma grande quantidade de variáveis. De modo que em cada nó da árvore de *branch-and-bound*, a relaxação linear é resolvida por um método de geração de colunas. O sucesso desse método no tratamento de problemas com muitas variáveis se deve ao fato que somente um subconjunto de variáveis é considerado explicitamente, já que de qualquer forma a maioria das variáveis será igual a zero na solução ótima do problema e a geração explícita de todas as variáveis consumiria muito tempo e memória [Barnhart et al., 1998]. Essa abordagem é também frequentemente utilizada para gerar e explorar melhores limites inferiores para alguns problemas já que nem sempre é possível obter o ótimo.

Para realizar a geração de colunas, o problema é decomposto retirando-se parte de suas restrições e tratando-as em um problema auxiliar, reduzindo assim o número de restrições do problema resultante. Métodos de decomposição podem ser aplicados dividindo o problema principal em um ou mais subproblemas. A decomposição de Dantzig-Wolfe é uma técnica importante e bem sucedida para realizar essa reformulação.

As seções seguintes fornecem uma breve revisão sobre todos esses conceitos que foram estudados para aplicar a abordagem proposta.

4.1.1 Decomposição de Dantzig-Wolfe

Métodos de decomposição foram criados a partir da ideia de dividir para conquistar. Desta forma, problemas maiores podem ser divididos ou decompostos em outros problemas menores

que podem ser resolvidos separadamente. O método de decomposição de Dantzig-Wolfe, proposto por [Dantzig e Wolfe, 1960], foi introduzido inicialmente com o intuito de resolver problemas de programação linear de grande escala mas também pode ser aplicado a problemas de programação inteira. Esse método se baseia na propriedade que as soluções de um certo domínio convexo limitado podem ser representadas como uma combinação linear convexa de seus pontos extremos.

Partindo dessa propriedade, para entender como a decomposição é realizada, considere um problema de programação linear escrito da seguinte forma:

$$\min \quad cx \tag{4.1}$$

s.a.

$$Ax = b \tag{4.2}$$

$$Dx \leqslant d \tag{4.3}$$

$$x \geqslant 0 \tag{4.4}$$

Supondo que as restrições 4.3 e 4.4 do problema definam um poliedro limitado e não vazio P e, seja X o conjunto de seus pontos extremos, tem-se que uma solução $x \in P$ pode ser escrita como uma combinação convexa dos pontos extremos de P do seguinte modo:

$$x = \sum_{q \in X} q\lambda_q \tag{4.5}$$

$$\sum_{q \in X} \lambda_q = 1 \tag{4.6}$$

$$\lambda_q \geqslant 0 \qquad \forall q \in X \tag{4.7}$$

Portanto, o problema linear pode ser reescrito substituindo as variáveis x como dado a seguir:

$$\min \quad \sum_{q \in X} cq\lambda_q \tag{4.8}$$

s.a.

$$\sum_{q \in X} Aq\lambda_q = b \tag{4.9}$$

$$\sum_{q \in X} \lambda_q = 1 \tag{4.10}$$

$$\lambda_q \geqslant 0 \qquad \forall q \in X \tag{4.11}$$

Essa reformulação é chamada de *Problema Linear Mestre (PLM)* e contém menos restrições mas uma grande quantidade de variáveis. Na prática, ainda é um problema computacionalmente custoso de ser resolvido, no entanto, pode-se utilizar o procedimento de Geração de Colunas na sua resolução. Este procedimento é descrito a seguir.

4.1.2 Geração de Colunas

Geração de Colunas (GC) é uma técnica utilizada para resolver problemas lineares que contêm muitas variáveis (colunas), tais como os comumente produzidos pela técnica de decomposição apresentada na seção anterior. Na geração de colunas, ao invés de resolver o problema mestre utilizando todas as suas colunas, parte-se de um subconjunto restrito dessas, em um processo onde novas colunas podem ser adicionadas iterativamente quando necessário. Essas colunas que irão alimentar o método podem ser obtidas resolvendo-se subproblemas (*pricing*). A abordagem geral está resumida na Figura 4.1.

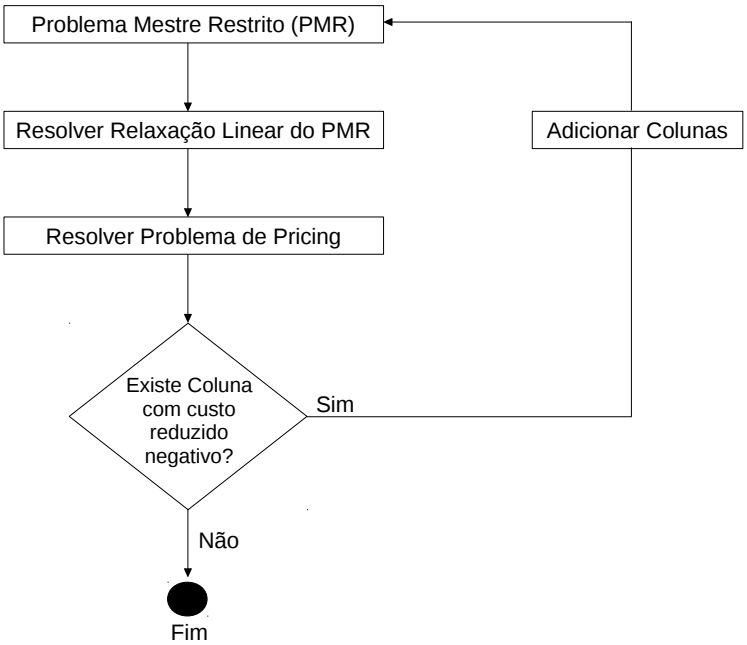


Figura 4.1: Estrutura esquemática do algoritmo de Geração de Colunas.

Devido à grande quantidade de colunas resultante da decomposição de Dantzig-Wolfe, a maioria dessas não são incluídas no problema linear mestre, resultando no que chamamos de

Problema Mestre Restrito (PMR). Iterativamente, através da resolução da relaxação linear do PMR, valores duais para as restrições do PMR são encontrados, esses valores são utilizados para definir o problema de *pricing*: o subproblema é resolvido afim de encontrar colunas com custo reduzido negativo. Se estas forem encontradas, elas são adicionadas ao PMR e o problema será resolvido novamente. O processo termina quando não houver mais colunas com custo reduzido negativo para o PMR, neste caso, a solução do PMR será ótima também para o problema mestre.

4.1.3 Branch-and-Price

Finalmente, *Branch-and-Price* (B&P) refere-se a um método *branch-and-bound* que é combinado com um procedimento de geração de colunas. Essa geração de colunas é utilizada para resolver uma reformulação do problema original tal como a reformulação de Dantzig-Wolfe, com o objetivo de obter limites inferiores melhores que a formulação original. Mesmo que o esforço necessário para obter cada limite inferior seja maior, uma relaxação mais forte também permite cortar partes maiores da árvore de busca usando os limites, resultando possivelmente em um método geral mais eficiente. A reformulação de Dantzig-Wolfe permite ainda reduzir a simetria em diversos casos, o que ajuda também na resolução.

Combinar a geração de colunas com o algoritmo de *branch-and-bound* leva a alguns desafios específicos. Primeiro, gerar (*pricing*) as variáveis necessárias para o problema mestre usando o procedimento de geração de colunas no nó raiz normalmente é insuficiente para garantir que todas as colunas necessárias sejam consideradas durante a busca em toda a árvore. Devido às restrições de ramificação, outras colunas podem ser necessárias em níveis mais baixos da árvore de busca. Por esse motivo, a geração de colunas é realizada em cada nó da árvore de *branch-and-bound* para certificar que todas as colunas necessárias entraram no problema mestre e que os limites gerados pelo algoritmo de geração de colunas são válidos. Diferentes algoritmos de *branch-and-bound* podem ter políticas distintas sobre quais colunas serão reutilizadas de níveis anteriores. Além disso, a ramificação nas variáveis λ do PLM é geralmente impraticável, pois a imposição de $\lambda = 0$ no *branch* equivale a eliminar tal coluna, e não existe uma maneira simples de proibir a reaparição da mesma variável (re-pricing) nas resoluções subsequentes do problema de *pricing*. Por essa razão, outras regras de ramificação geralmente devem ser implementadas [Vanderbeck, 2011].

A seguir, nas próximas seções são detalhados como esse método foi utilizado para resolver o PBPD.

4.2 Modelos Matemáticos para o PBPD

Uma formulação matemática para o PBPD, chamada aqui de $F1$, foi descrita anteriormente na Seção 2.2. Nesta formulação, que por questões de clareza será apresentada novamente a seguir, considere a variável x_{ik} igual a 1 se o item i foi alocado ao *bin* k e 0, caso contrário; a variável y_k é igual a 1 se o *bin* k foi utilizado na solução. w_i expressa o peso do item i, Q a capacidade dos *bins*, $N(i)$ o conjunto de vizinhanças do item i e $N_l(i)$, o conjunto de itens que pertencem a vizinhança l do item i.

$$(F1) \quad \min \sum_{k=1}^{n} y_k \tag{4.12}$$

$$\text{s.a.} \sum_{i=1}^{n} w_i x_{ik} \leqslant Q y_k \qquad k = 1, \ldots, n \tag{4.13}$$

$$\sum_{k=1}^{n} x_{ik} = 1 \qquad i = 1, \ldots, n \tag{4.14}$$

$$x_{ik} \leqslant \sum_{j \in N_l(i)} x_{jk} \qquad \forall i \in 1, \ldots, n, \forall k \in 1, \ldots, n, \forall l \in 1, \ldots, L_i \tag{4.15}$$

$$y_k \in \{0, 1\} \qquad k = 1, \ldots, n \tag{4.16}$$

$$x_{ik} \in \{0, 1\} \qquad i = 1, \ldots, n, k = 1, \ldots, n \tag{4.17}$$

Em $F1$, existe uma grande quantidade de simetria, uma vez que existem diversas soluções nas quais o conjunto de itens está alocado exatamente com a mesma configuração mas em *bins* diferentes. Para acelerar o tempo de computação e melhorar o comportamento do modelo matemático, reduzindo por exemplo a simetria, foram adicionadas as seguintes desigualdades no modelo:

$$y_k \geqslant y_{k+1} \qquad \forall k = 1, \ldots, n-1 \tag{4.18}$$

$$x_{ik} = 0 \qquad \forall i = 1, \ldots, n-1, \forall k = i+1, \ldots, n \tag{4.19}$$

$$x_{ik} \leqslant y_k \qquad \forall k = 1, \ldots, n, \forall i = 1, \ldots, n \tag{4.20}$$

O conjunto de desigualdades em (4.18) foi adicionada para reduzir o tamanho da árvore de enumeração do *branch-and-bound*. Esse conjunto de desigualdades impõe que os *bins* sejam utilizados de acordo com a ordem de seus índices. O conjunto de desigualdades em (4.19) elimina diversas soluções simétricas considerando que há uma solução ótima onde o item 1 foi

alocado ao *bin* 1, o item 2, ao *bin* 1 ou 2, e assim sucessivamente. Para melhorar a relaxação linear, foi adicionado o conjunto de restrições (4.20), que determina que itens com valores inteiros não podem ser alocados a *bins* com valores fracionários. Essas desigualdades já são bem conhecidas e utilizadas para o problema de *bin packing* [Delorme et al., 2016].

Mesmo com a adição dos cortes, o modelo $F1$ não é resolvido de maneira eficiente utilizando-se os resolvedores de programação matemática para diversas instâncias do PBPD. Portanto, adicionalmente a $F1$, outra formulação alternativa para o PBPD pode ser obtida. Tal formulação, chamada de $F2$, é baseada na formulação do problema de particionamento de conjuntos e pode ser vista nas Equações de (4.21) a (4.23) a seguir.

$$(F2) \qquad \min \quad \sum_{j \in D} \lambda_j \tag{4.21}$$

$$\text{s.a.} \quad \sum_{j \in D} a_{ij} \lambda_j = 1 \qquad \forall i \in \mathcal{V} \tag{4.22}$$

$$\lambda_j \in \{0, 1\} \qquad \forall j \in D \tag{4.23}$$

Considere D um conjunto composto por todos os sub-conjuntos de itens que podem ser alocados em um *bin* e que respeitam a definição de dependência entre vizinhanças do problema. A função objetivo em (4.21) minimiza o número de conjuntos selecionados, ou seja, o número de *bins* que serão utilizados. O conjunto de restrições (4.22) garante que cada item i deve estar associado a um conjunto que estará na solução. Cada variável binária λ_j será igual a 1 se o subconjunto de itens j é selecionado para fazer parte da solução e 0, caso contrário. Associado a cada item i e cada sub-conjunto j existe ainda uma constante a_{ij} que será igual a 1 se o item i está no conjunto j.

Tal formulação acima envolve um número exponencial de variáveis λ. No entanto, ela possui uma alternativa para sua resolução, que é a utilização do algoritmo de *branch-and-price* para resolvê-la. A seguir será fornecida uma descrição de como essa abordagem foi aplicada ao PBPD.

4.3 *Branch-and-Price* aplicado ao PBPD

No caso do *branch-and-price* desenvolvido para o PBPD, a decomposição de Dantzig-Wolfe é aplicada à formulação $F1$ e, assim como no problema de *bin packing* clássico, resulta no problema de particionamento de conjuntos como problema mestre. A formulação resultante

para o problema mestre é a mesma já apresentada em $F2$. O subproblema de *pricing* obtido, de modo análogo, é o *Problema da Mochila com Dependências (PMD)*, que pode ser formulado por $F3$ como:

$$
\begin{aligned}
(F3) \quad & \max \sum_{i=1}^{n} \Phi_i x'_i && && (4.24)\\
& \text{s.a.} \sum_{i=1}^{n} w_i x'_i \leqslant Q && && (4.25)\\
& x'_i \leqslant \sum_{j \in N_l(i)} x'_j && \forall i \in 1,\ldots,n, \forall l \in 1,\ldots,L_i && (4.26)\\
& x'_i \in \{0,1\} && i = 1,\ldots,n && (4.27)
\end{aligned}
$$

Para o problema de *pricing*, considere Φ_i, onde $i \in V$, os valores duais relativos às restrições 4.22. A seguir é descrito como o problema de *pricing* foi tratado.

4.3.1 Pricing

Para resolver o problema de *pricing*, neste trabalho foi desenvolvido um algoritmo heurístico. Se este algoritmo falhar em encontrar colunas com custo reduzido, o subproblema é resolvido de modo exato. Os dois métodos são detalhados nas próximas seções.

4.3.1.1 Pricing usando heurística

Foi proposto um algoritmo heurístico para resolução do problema de *pricing*. Essa abordagem utiliza o algoritmo de Busca em Profundidade (ou *Depth First Search – DFS*) para auxiliar na escolha dos itens que devem ser incluídos na mochila. O pseudo-código é apresentado nos Algoritmos 6 e 7.

No Algoritmo 6, considere como entrada o conjunto D de itens. Cada item possui um status que indica se o item foi fixado com 1, com 0 ou se é livre. Seja V, o conjunto de todos os itens, w_i o peso de cada item, $Q_{restante}$ o espaço disponível na mochila e, S o conjunto de itens já alocados na mochila.

Enquanto a mochila não está cheia, o algoritmo tenta inserir itens na mochila. Inicialmente, na linha 2, um item i é escolhido para ser alocado na mochila. Essa escolha é realizada de forma gulosa, de acordo com o status do item, o número de dependências que o item resolve se alocado na mochila e o seu benefício. O benefício é calculado pelo valor do custo reduzido do item dividido pelo seu peso. Se há espaço para alocar o item na mochila, o mesmo é

Algoritmo 6 Heurística para o *Pricing*

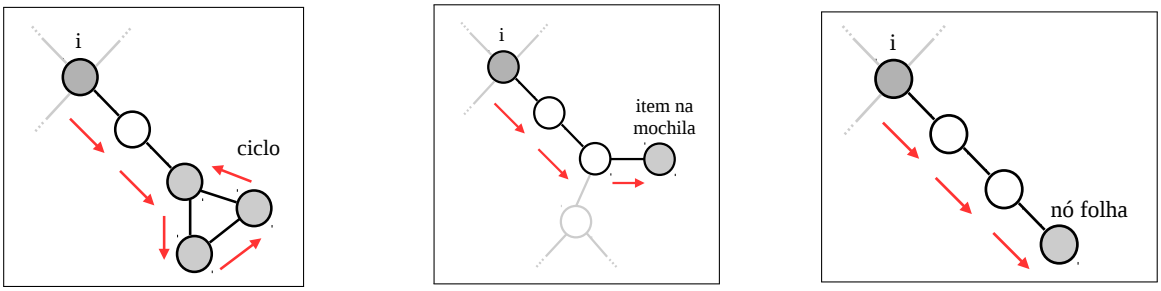
enquanto ($Q_{restante} > 0$) e ($|S| < |V|$) **faça**
  Escolha um item $i \in V - S$;
  se $w_i < Q_{restante}$ **então**
    DFS(i,$Q_{restante}$,G,S);
    $S = S + i$;
    se DFS encontrou um nó folha, um ciclo ou item já adicionado **então**
      Adiciona itens da árvore de DFS;
    fim se
  fim se
fim enquanto

Algoritmo 7 DFS(i,$Q_{restante}$,G,S)

Marca i como visitado;
$pre(i) + +$;
se ($w_i < Q_{restante}$) e ($i.status \neq 0$) **então**
  se $Adj(i) = \emptyset$ **então**
    encontrou um nó folha;
  senão
    Ordena $Adj(i)$ pelos custos reduzidos;
    para $v \in Adj(i)$ **faça**
      se $v \in S$ **então**
        encontrou um item já adicionado na mochila;
        break;
      senão
        se v não foi visitado **então**
          Insere aresta (i, v) na árvore de DFS;
          DFS(v,$Q_{restante}$,G,S);
        senão
          se $pos(v)$ indefinido **então**
            encontrou um ciclo;
            break;
          fim se
        fim se
      fim se
    fim para
  fim se
fim se
$pos(i) + +$;

alocado. Além disso, usa-se o algoritmo de DFS (linha 4), para tentar alocar outros itens juntos na mochila e tentar resolver as restrições de dependência.

O algoritmo de DFS (Algoritmo 7) percorre o grafo examinando seus nós, o item i é considerado como o nó raiz. A busca pela árvore tem por objetivo encontrar um caminho na árvore que forme um ciclo que inclua o item i (linha 18) ou outro item já adicionado na mochila (linha 10), ou um nó folha (linha 5), que não possua nós dependentes, como vistos na Figura 4.2. Desta forma, se alguma destas estruturas for encontrada, o conjunto formado pelo nó i e a árvore de DFS irá formar um subconjunto de itens viável para o PBPD.



(a) Caminho que forma um ciclo.

(b) Caminho que inclui um item já alocado.

(c) Caminho que inclui um nó folha.

Figura 4.2: Tipos de caminhos na árvore que são buscados pelo DFS.

4.3.1.2 Pricing usando programação inteira

Para a resolução exata do problema de *pricing* uma opção mais direta e simples é utilizar o modelo matemático existente para o problema da mochila com dependências. Isto é feito com o uso de um resolvedor de programação matemática, tal como o CPLEX, que é baseado no algoritmo de *branch-and-cut*. Essa solução é eficiente, mas para a resolução do problema de *pricing* em instâncias grandes ainda pode utilizar uma significativa quantidade de tempo.

4.3.2 BapCode

O desenvolvimento de um método B&P exigia um grande desenvolvimento para formular e resolver o problema, escolhendo estratégias de *branching*, incluindo estabilização e outras técnicas de aprimoramento que permitiam uma resolução eficiente do problema estudado. Com o criação de *frameworks* B&P, e do BapCod [Vanderbeck, 2011, Sadykov e Vanderbeck, 2013], este desenvolvimento foi facilitado devido a disponibilização desse *framework* de caixa preta que gerencia esses passos críticos.

O BaPCod utiliza um esquema de *branching* de [Vanderbeck, 2011] e a heurística primal baseada em uma exploração parcial da árvore de *branch-and-price* de [Joncour et al., 2010].

4.4 Experimentos Computacionais

As abordagens propostas neste capítulo foram codificadas em C++ e testes experimentais foram conduzidos para analisar a performance e o comportamento dos métodos. Todos os testes foram executados em uma máquina Intel i7 com 3,4 GHz e 16 GB de memória RAM utilizando o sistema operacional Linux 64 bits. Como resolvedor matemático foi utilizado o ILOG CPLEX versão 12.5.1. Todos os testes utilizaram apenas uma única *thread*. Na implementação da formulação $F1$ foi utilizada ainda a biblioteca ILOG Concert.

Na próxima seção serão descritas como as instâncias utilizadas nos testes foram geradas. E a seguir, serão exibidos os resultados encontrados.

4.4.1 Instâncias

Como não haviam instâncias para o PBPD, para a realização dos testes foram geradas instâncias inspiradas no problema da Mochila 1-vizinho como descritas a seguir.

Como ponto de partida para gerar o conjunto de problemas teste foram utilizadas algumas das instâncias existentes na literatura para o PBPC. Mais especificamente, foram utilizadas 40 instâncias do conjunto (u – *uniforme*) e 40 instâncias do conjunto (t – *triplet*) ambas propostas por [Muritiba et al., 2010], com *bins* de capacidade 150 e número de itens iguais a $n_u = \{120, 250, 500, 1000\}$ e $n_t = \{60, 120, 249, 501\}$ respectivamente. Além dessas, foram utilizadas outras 40 instâncias do conjunto (d) (com mais itens por *bin*) criadas por [Sadykov e Vanderbeck, 2013] com capacidade de 10000 e número de itens $n_d = \{120, 250, 500, 1000\}$.

Essas instâncias do PBPC foram tratadas da seguinte forma: primeiro, os grafos de conflitos existentes foram removidos e a capacidade dos *bins* foi dobrada afim de evitar inviabilidades. Depois, para cada instância do PBPC considerada, uma solução viável do *bin packing* foi produzida. Nesta solução de referência foram adicionadas as dependências entre os itens tomando-se cuidado para não criar nenhuma inviabilidade e garantindo que essa solução seja viável para o problema considerando as dependências.

A solução viável do problema de *bin packing* foi obtida do seguinte modo: foi criado um algoritmo onde a cada iteração de um loop seleciona-se aleatoriamente um item entre os ainda não alocados a nenhum *bin*. Esse item é alocado ao primeiro *bin* que tenha menos de

80% de seu espaço ocupado e cuja capacidade residual seja menor ou igual ao peso do item que será inserido. Se não houver nenhum *bin* já utilizado com espaço suficiente, um novo *bin* é adicionado a solução e o item alocado a esse *bin*.

Para adicionar as dependências na solução de referência, os itens foram separados nas vizinhanças de acordo com a alocação destes nos *bins*. As características do grafo criado levaram em considerações três parâmetros: α – porcentagem de itens sem vizinhanças; β – número de itens em cada vizinhança; *ciclo* – existência ou não de ciclo no grafo de dependências. O procedimento para criar as vizinhanças é descrito abaixo.

1. α% dos itens são escolhidos para serem "sem vizinhança": um item é escolhido aleatoriamente em cada *bin* da solução de referência e marcado como "sem vizinhança" até que o valor de α seja atingido;

2. Para cada item i que não está marcado como "sem vizinhança", selecione β itens para pertencer a sua vizinhança, começando com pelo menos um item escolhido aleatoriamente entre os itens pertencentes ao *bin* onde o item está alocado na solução de referência.

Após alguns testes empíricos, foram considerados os valores de $\alpha = \{30\%, 50\%\}$ e $\beta = \{2, 3, 5, 10, 20\}$. Foram criadas instâncias com ciclo entre os itens no grafo de dependências e sem ciclo. Essas configurações combinadas geraram 240 instâncias, 120 sem ciclo e 120 com ciclo.

4.4.2 Resultados

Foram realizados diversos testes para verificar a qualidade dos métodos propostos. Serão apresentados os resultados encontrados na implementação do modelo matemático proposto $F1$ (solucionado pelo *branch-and-cut* do CPLEX com configuração padrão) e do *branch-and-price* com a heurística de *pricing* proposta (solucionado pelo BaPCode e usando o CPLEX como resolvedor).

Cada instância considerada foi executada com um tempo limite de 1 hora, tanto para o modelo matemático $F1$ quanto para o *branch-and-price* proposto para o PBPD.

Devido a quantidade de instâncias existentes, os resultados encontrados foram divididos em 6 tabelas, nas três primeiras (Tabelas 4.1, 4.2 e 4.3) são exibidos os resultados para as instâncias que não possuem ciclo no grafo de dependências e nas três últimas (Tabelas 4.4, 4.5 e 4.6), os resultados vistos são para as instâncias com ciclo no grafo de dependências. Cada

tabela possui 40 instâncias pertencentes a um determinado conjunto, dentre os conjuntos (u), (t) e (d) existentes inicialmente para o problema de *bin packing* com conflitos.

Em todas estas tabelas, as quatro primeiras colunas indicam as características de cada instância: **Inst.** denota o problema teste, **n** indica o número de itens que a instância possui, $\boldsymbol{\alpha}$ é o valor do parâmetro α utilizado para gerar a instância, $\boldsymbol{\beta}$ denota o valor do parâmetro β utilizado. Nas outras colunas, $\mathbf{LB_0}$ indica o valor da relaxação linear no nó raiz, **LB** é o melhor valor do limite inferior (ou *lower bound*) obtido, **UB** é o valor do melhor limite superior (ou *upper bound*), $\mathbf{T_0}$ indica o tempo de CPU em segundos gasto pela relaxação linear, **T** indica o tempo total de CPU em segundos gasto pelo método considerado, **GAP** denota a diferença percentual entre o valor do melhor limite inferior e o melhor limite superior obtido. **NNós** indica o número de nós que foram processados na árvore de B&B e **NCol** indica o número de colunas geradas.

Em alguns casos a execução do método proposto atingiu o tempo limite predefinido, essa informação é exibida nas colunas marcadas com o valor "TL". As colunas referentes ao dado que não pôde ser obtido no tempo limite foi marcada com o símbolo "-". Para cada instância o melhor limite inferior obtido é destacado em negrito.

Considerando os resultados das Tabelas 4.1 – 4.6, observa-se que a geração de colunas produz limites inferiores de boa qualidade, que permitem ao algoritmo de B&P resolver a maioria das instâncias no nó raiz. Das 240 instâncias existentes, o B&P apresentou um GAP melhor ou igual ao B&C em 177 delas, onde 87 em instâncias que não possuem ciclo no grafo de dependências e em 90 que possuem ciclo. O tempo utilizado pelo B&P é, em geral, menor que o tempo gasto pelo B&C, a única exceção se dá para as instâncias do conjunto (d), que possuem mais itens por *bin*, o que torna o subproblema mais difícil de ser resolvido pelo *branch-and-price*.

De modo geral, pode-se perceber ainda que quanto maior o número de itens maior a dificuldade das abordagens propostas em encontrar uma solução dentro do tempo limite. A seguir analisaremos o comportamento dos métodos para cada conjunto de instâncias individualmente.

Para as instâncias do conjunto (u) sem ciclo, com os resultados exibidos na Tabela 4.1, não foi possível obter nem o resultado da relaxação linear do modelo e nem o resultado do B&P dentro do tempo limite para algumas instâncias que possuem 1000 itens e o valor de α

igual a 30. A resolução do modelo $F1$ com o CPLEX permitiu provar a otimalidade em 5 instâncias desse conjunto enquanto utilizando-se o B&P é possível provar em 34 destes.

Na Tabela 4.2, para as instâncias do conjunto (t) sem ciclo, utilizando o modelo, o CPLEX não conseguiu provar a otimalidade em nenhuma das instâncias, obtendo valores para o GAP entre 2,33% e 9,09%. No entanto, o valor obtido pela relaxação linear do modelo no nó raiz (LB_0) chegou ao mesmo valor de LB em 20 das 40 instâncias. Já utilizando-se o B&P, provou-se a otimalidade em 29 instâncias. Atingindo o tempo limite sem nenhuma solução em apenas 6 instâncias.

Na Tabela 4.3, para o conjunto (d) sem ciclo, de modo geral, tanto a implementação do modelo quanto o B&P tiveram dificuldade em resolver as instâncias com número de itens acima de 500. Para quase todas as outras instâncias desse conjunto, a resolução do modelo $F1$ permitiu provar a otimalidade. O B&P teve um comportamento similar porém em um tempo maior que o modelo.

Já para as instâncias que apresentam ciclo no grafo de dependências, cujos resultados são exibidos nas Tabelas 4.4, 4.5 e 4.6, o modelo teve um comportamento bem similar aos apresentados com as instâncias sem ciclo.

Na Tabela 4.4, para as instâncias do conjunto (u), a implementação do modelo $F1$ provou a otimalidade em 5 instâncias, destas, 3 foram resolvidas diretamente no nó raiz. Destacam-se neste caso, as instâncias I13C e I14C, onde os valores do GAP alcançado foram respectivamente de 3,77% e 3,85%, enquanto na sua versão sem o ciclo o GAP (exibido em 4.1) havia sido de 70,52% e 74,23%, respectivamente. Já o B&P conseguiu provar a otimalidade em 33 instâncias. Além disso, o tempo de execução gasto pelo B&P foi menor que o tempo gasto pelo B&C na maioria das instâncias.

Os resultados obtidos para as instâncias do conjunto (t) com ciclo são exibidos na Tabela 4.5. Nestas instâncias, o modelo $F1$ conseguiu provar a otimalidade apenas em uma instância atingindo o tempo limite para quase todas as instâncias, enquanto o B&P conseguiu provar a otimalidade em 31 instâncias.

Na Tabela 4.6, para as instâncias do conjunto (d), as duas abordagens tiveram um comportamento similar. Para as instâncias com até 250 itens, os métodos foram capazes de provar a otimalidade em quase todas no nó raiz. Já para a maioria das instâncias com mais de 250 itens, as abordagens desenvolvidas tiveram dificuldade em encontrar um valor para o limite inferior e superior, dentro do tempo limite.

				Formulação F1							Branch-and-Price					
Inst.	**n**	α	β	$\mathbf{LB_0}$	**LB**	**UB**	**Gap**	$\mathbf{T_0(s)}$	**T(s)**	**NNós**	**LB**	**UB**	**GAP**	**T(s)**	**NNós**	**NCol**
I1	120	30	2	23,59	24	25	4,00%	4,04	TL	56020	24	24	**0,00%**	229,16	1	700
I2	120	30	3	24,02	25	25	**0,00%**	10,45	450,54	10908	25	25	**0,00%**	412,68	1	900
I3	120	30	5	22,65	23	24	4,17%	23,33	TL	65994	23	23	**0,00%**	20,20	1	636
I4	120	30	10	24,28	25	25	**0,00%**	22,19	57,54	1557	25	25	**0,00%**	16,22	1	714
I5	120	30	20	24,51	25	25	**0,00%**	33,92	172,83	5904	25	25	**0,00%**	11,33	1	733
I6	120	50	2	23,74	24	24	**0,00%**	2,70	1665,31	54446	24	24	**0,00%**	29,73	1	576
I7	120	50	3	23,79	24	25	4,00%	7,32	TL	102912	24	24	**0,00%**	13,22	1	632
I8	120	50	5	24,32	25	25	**0,00%**	16,53	62,2	11116	25	25	**0,00%**	12,08	1	678
I9	120	50	10	24,93	25	26	3,85%	12,79	TL	161509	25	25	**0,00%**	9,22	1	713
I10	120	50	20	22,90	23	24	4,17%	16,30	TL	173636	23	23	**0,00%**	8,83	1	886
I11	250	30	2	49,28	50	52	3,85%	109,63	TL	95719	50	50	**0,00%**	1522,16	1	1184
I12	250	30	3	49,51	50	52	3,85%	193,35	TL	11717	50	50	**0,00%**	2124,51	1	1520
I13	250	30	5	50,71	51	173	70,52%	313,82	TL	4948	51	51	**0,00%**	183,82	1	1464
I14	250	30	10	49,71	50	194	74,23%	656,14	TL	4411	50	50	**0,00%**	82,28	1	1621
I15	250	30	20	50,31	51	52	1,92%	1967,52	TL	10883	51	51	**0,00%**	60,39	1	1606
I16	250	50	2	50,41	51	52	1,92%	88,00	TL	14424	51	51	**0,00%**	129,18	1	1259
I17	250	50	3	50,51	51	52	1,92%	136,83	TL	23749	51	51	**0,00%**	71,25	1	1291
I18	250	50	5	51,39	52	53	1,89%	207,27	TL	21184	52	52	**0,00%**	45,78	1	1373
I19	250	50	10	52,46	53	54	1,85%	328,40	TL	176836	53	53	**0,00%**	43,22	1	1514
I20	250	50	20	50,10	51	52	1,92%	861,49	TL	1591	51	51	**0,00%**	70,90	1	2908
I21	500	30	2	98,79	-	-	-	1420,33	TL	-	-	-	-	TL	-	-
I22	500	30	3	100,42	-	-	-	1777,97	TL	-	-	-	-	TL	-	-
I23	500	30	5	100,72	-	-	-	TL	TL	-	101	101	**0,00%**	2009,29	1	3055
I24	500	30	10	101,91	-	-	-	TL	TL	-	102	102	**0,00%**	565,25	1	3206
I25	500	30	20	102,56	-	-	-	TL	TL	-	103	103	**0,00%**	606,00	1	5556
I26	500	50	2	102,54	-	-	-	1946,29	TL	-	103	103	**0,00%**	541,93	1	2469
I27	500	50	3	103,44	-	-	-	2239,25	TL	-	104	104	**0,00%**	291,08	1	2603
I28	500	50	5	101,99	-	-	-	TL	TL	-	102	102	**0,00%**	253,32	1	2845
I29	500	50	10	97,84	-	-	-	TL	TL	-	98	98	**0,00%**	271,20	1	3308
I30	500	50	20	100,53	-	-	-	TL	TL	-	101	101	**0,00%**	270,13	1	3513
I31	1000	30	2	199,21	-	-	-	TL	TL	-	-	-	-	TL	-	-
I32	1000	30	3	-	-	-	-	TL	TL	-	-	-	-	TL	-	-
I33	1000	30	5	-	-	-	-	TL	TL	-	-	-	-	TL	-	-
I34	1000	30	10	-	-	-	-	TL	TL	-	-	-	-	TL	-	-
I35	1000	30	20	-	-	-	-	TL	TL	-	199	199	**0,00%**	3548,23	1	7062
I36	1000	50	2	199,25	-	-	-	TL	TL	-	200	200	**0,00%**	3000,16	1	5264
I37	1000	50	3	197,10	-	-	-	TL	TL	-	198	198	**0,00%**	2247,70	1	5618
I38	1000	50	5	-	-	-	-	TL	TL	-	202	202	**0,00%**	2173,11	1	5907
I39	1000	50	10	198,94	-	-	-	TL	TL	-	200	200	**0,00%**	2458,24	1	6608
I40	1000	50	20	198,46	-	-	-	TL	TL	-	199	199	**0,00%**	2837,10	1	7293

Tabela 4.1: Resultados para as instâncias do conjunto (u) sem ciclo.

				Formulação F1							Branch-and-Price					
Inst.	**n**	α	β	$\mathbf{LB_0}$	**LB**	**UB**	**Gap**	$\mathbf{T_0(s)}$	**T(s)**	**NNós**	**LB**	**UB**	**GAP**	**T(s)**	**NNós**	**NCol**
I41	60	30	2	10,00	10	11	9,09%	0,51	TL	177969	11	11	**0,00%**	45,78	1	303
I42	60	30	3	10,00	10	11	**9,09%**	1,23	TL	224605	10	11	**9,09%**	306,41	1	641
I43	60	30	5	10,00	10	11	**9,09%**	0,26	TL	266338	10	11	**9,09%**	133,93	1	1909
I44	60	30	10	10,00	10	11	9,09%	0,88	TL	274782	10	10	**0,00%**	9,43	1	463
I45	60	30	20	10,00	10	11	9,09%	0,78	TL	1524456	10	10	**0,00%**	4,80	1	477
I46	60	50	2	10,00	10	11	**9,09%**	0,17	TL	529040	10	11	**9,09%**	168,99	1	1098
I47	60	50	3	10,00	10	11	9,09%	0,41	TL	807823	10	10	**0,00%**	56,40	1	1306
I48	60	50	5	10,00	10	11	9,09%	0,42	TL	738551	10	10	**0,00%**	9,86	1	432
I49	60	50	10	10,00	10	11	9,09%	0,70	TL	510725	10	10	**0,00%**	16,27	1	1239
I50	60	50	20	10,00	10	11	9,09%	0,56	TL	754884	10	10	**0,00%**	7,12	1	779
I51	120	30	2	20,00	20	21	4,76%	5,08	TL	15671	21	21	**0,00%**	369,57	1	642
I52	120	30	3	20,00	20	21	**4,76%**	8,37	TL	41657	-	-	-	TL	-	-
I53	120	30	5	20,00	20	21	**4,76%**	13,59	TL	58041	20	21	**4,76%**	585,10	1	4404
I54	120	30	10	20,00	20	21	4,76%	21,20	TL	178714	20	20	**0,00%**	154,83	1	3165
I55	120	30	20	20,00	20	21	4,76%	36,96	TL	253743	20	20	**0,00%**	25,52	1	1017
I56	120	50	2	20,00	20	21	**4,76%**	4,30	TL	38625	20	21	**4,76%**	595,91	1	3541
I57	120	50	3	20,00	20	21	4,76%	7,77	TL	46192	20	20	**0,00%**	87,37	1	1504
I58	120	50	5	20,00	20	21	4,76%	9,27	TL	65876	20	20	**0,00%**	60,47	1	1289
I59	120	50	10	20,00	20	21	4,76%	25,79	TL	106664	20	20	**0,00%**	56,26	1	2041
I60	120	50	20	20,00	20	21	4,76%	18,87	TL	229002	20	20	**0,00%**	23,30	1	1005
I61	249	30	2	41,50	-	-	-	72,28	TL	-	-	-	-	TL	-	-
I62	249	30	3	41,50	42	44	**4,55%**	114,47	TL	5221	-	-	-	TL	-	-
I63	249	30	5	41,50	42	43	2,33%	444,93	TL	12613	42	42	**0,00%**	691,08	1	1833
I64	249	30	10	41,50	42	43	2,33%	727,13	TL	17699	42	42	**0,00%**	343,92	1	2006
I65	249	30	20	41,50	42	43	2,33%	1807,75	TL	11031	42	42	**0,00%**	305,98	1	4532
I66	249	50	2	41,50	42	43	2,33%	140,70	TL	29599	42	42	**0,00%**	487,81	1	1515
I67	249	50	3	41,50	42	43	2,33%	99,41	TL	32001	42	42	**0,00%**	305,91	1	1544
I68	249	50	5	41,50	42	43	2,33%	225,08	TL	140432	42	42	**0,00%**	174,77	1	1720
I69	249	50	10	41,50	42	43	2,33%	375,46	TL	10182	42	42	**0,00%**	132,68	1	1881
I70	249	50	20	41,50	42	43	2,33%	738,50	TL	17818	42	42	**0,00%**	111,00	1	2209
I71	501	30	2	83,50	-	-	-	2022,20	TL	-	-	-	-	TL	-	-
I72	501	30	3	83,50	-	-	-	TL	TL	-	-	-	-	TL	-	-
I73	501	30	5	83,50	-	-	-	2604,94	TL	-	-	-	-	TL	-	-
I74	501	30	10	83,50	-	-	-	TL	TL	-	84	84	**0,00%**	1955,85	1	4044
I75	501	30	20	83,50	-	-	-	TL	TL	-	84	84	**0,00%**	1180,15	1	4042
I76	501	50	2	83,50	-	-	-	1165,93	TL	-	84	84	**0,00%**	2866,86	1	3099
I77	501	50	3	83,50	-	-	-	2406,68	TL	-	84	84	**0,00%**	1405,31	1	3281
I78	501	50	5	83,50	-	-	-	TL	TL	-	84	84	**0,00%**	866,34	1	3327
I79	501	50	10	83,50	-	-	-	TL	TL	-	84	84	**0,00%**	701,71	1	3743
I80	501	50	20	83,50	-	-	-	TL	TL	-	84	84	**0,00%**	1180,41	1	9597

Tabela 4.2: Resultados para as instâncias do conjunto (t) sem ciclo

				Formulação F1							Branch-and-Price					
Inst.	**n**	α	β	$\mathbf{LB_0}$	**LB**	**UB**	**Gap**	$\mathbf{T_0(s)}$	**T(s)**	**NNós**	**LB**	**UB**	**GAP**	**T(s)**	**NNós**	**NCol**
I81	120	20	2	8,60	9	9	**0,00%**	10,08	5,86	1	9	9	**0,00%**	530,12	1	1008
I82	120	20	3	8,92	9	9	**0,00%**	11,68	18,76	1	9	9	**0,00%**	600,60	1	2033
I83	120	20	5	8,62	9	9	**0,00%**	17,52	25,65	1	9	9	**0,00%**	887,46	1	5468
I84	120	20	10	9,27	10	10	**0,00%**	21,39	24,54	1	10	10	**0,00%**	175,77	1	1133
I85	120	20	20	8,41	9	9	**0,00%**	30,48	30,47	1	9	9	**0,00%**	100,36	1	1195
I86	120	50	2	9,31	10	10	**0,00%**	4,44	8,02	1	10	10	**0,00%**	271,52	1	1069
I87	120	50	3	9,24	10	10	**0,00%**	8,54	13,19	1	10	10	**0,00%**	349,08	1	2074
I88	120	50	5	9,04	10	10	**0,00%**	10,06	16,47	1	10	10	**0,00%**	206,96	1	1733
I89	120	50	10	9,01	10	10	**0,00%**	16,25	14,72	1	10	10	**0,00%**	144,70	1	1509
I90	120	50	20	8,69	9	9	**0,00%**	15,10	17,76	1	9	9	**0,00%**	87,07	1	1719
I91	250	20	2	18,37	19	19	**0,00%**	287,57	128,17	1	19	19	**0,00%**	3068,90	1	2544
I92	250	20	3	18,23	19	19	**0,00%**	173,52	334,67	1	-	-	-	TL	-	-
I93	250	20	5	18,91	19	20	**5,00%**	325,52	TL	24579	-	-	-	TL	-	-
I94	250	20	10	19,06	20	20	**0,00%**	893,33	989,95	1	20	20	**0,00%**	2409,17	1	8179
I95	250	20	20	18,29	19	19	**0,00%**	2018,40	2321,01	1	19	19	**0,00%**	1281,58	1	6226
I96	250	50	2	19,43	20	20	**0,00%**	128,36	149,32	1	20	20	**0,00%**	1228,79	1	3174
I97	250	50	3	18,38	19	19	**0,00%**	333,28	207,64	1	19	19	**0,00%**	1555,43	1	5863
I98	250	50	5	18,73	19	19	**0,00%**	158,07	193,51	1	19	19	**0,00%**	826,78	1	3024
I99	250	50	10	19,26	20	20	**0,00%**	545,62	413,76	1	20	20	**0,00%**	692,49	1	3600
I100	250	50	20	19,24	20	20	**0,00%**	696,43	830,79	1	20	20	**0,00%**	917,01	1	7365
I101	500	20	2	36,60	-	-	-	2053,14	TL	-	-	-	-	TL	-	-
I102	500	20	3	37,97	-	-	-	2320,55	TL	-	-	-	-	TL	-	-
I103	500	20	5	37,72	-	-	-	TL	TL	-	-	-	-	TL	-	-
I104	500	20	10	37,10	-	-	-	TL	TL	-	-	-	-	TL	-	-
I105	500	20	20	38,49	-	-	-	TL	TL	-	-	-	-	TL	-	-
I106	500	50	2	37,36	-	-	-	1636,02	TL	-	-	-	-	TL	-	-
I107	500	50	3	36,76	37	38	**2,63%**	2494,32	TL	9472	-	-	-	TL	-	-
I108	500	50	5	37,88	-	-	-	TL	TL	-	-	-	-	TL	-	-
I109	500	50	10	37,01	-	-	-	TL	TL	-	-	-	-	TL	-	-
I110	500	50	20	37,12	-	-	-	TL	TL	-	38	38	**0,00%**	3017,99	1	8775
I111	1000	20	2	-	-	-	-	TL	TL	-	-	-	-	TL	-	-
I112	1000	20	3	-	-	-	-	TL	TL	-	-	-	-	TL	-	-
I113	1000	20	5	-	-	-	-	TL	TL	-	-	-	-	TL	-	-
I114	1000	20	10	-	-	-	-	TL	TL	-	-	-	-	TL	-	-
I115	1000	20	20	-	-	-	-	TL	TL	-	-	-	-	TL	-	-
I116	1000	50	2	-	-	-	-	TL	TL	-	-	-	-	TL	-	-
I117	1000	50	3	-	-	-	-	TL	TL	-	-	-	-	TL	-	-
I118	1000	50	5	-	-	-	-	TL	TL	-	-	-	-	TL	-	-
I119	1000	50	10	-	-	-	-	TL	TL	-	-	-	-	TL	-	-
I120	1000	50	20	-	-	-	-	TL	TL	-	-	-	-	TL	-	-

Tabela 4.3: Resultados para as instâncias do conjunto (d) sem ciclo.

				Formulação F1							Branch-and-Price					
Inst.	**n**	α	β	**LB_0**	**LB**	**UB**	**Gap**	**T_0(s)**	**T(s)**	**NNós**	**LB**	**UB**	**GAP**	**T(s)**	**NNós**	**NCol**
I1C	120	30	2	23,59	24,00	24	**0,00%**	6,63	700,35	16304	24	24	**0,00%**	147,46	1	538
I2C	120	30	3	24,02	25,00	25	**0,00%**	10,25	30,37	975	25	25	**0,00%**	93,46	1	660
I3C	120	30	5	22,65	23,00	24	4,17%	13,16	TL	45044	23	23	**0,00%**	19,71	1	661
I4C	120	30	10	24,28	25,00	25	**0,00%**	23,78	39,01	1	25	25	**0,00%**	12,21	1	676
I5C	120	30	20	24,51	25,00	25	**0,00%**	29,29	23,35	1	25	25	**0,00%**	12,06	1	775
I6C	120	50	2	23,74	24,00	25	4,00%	8,34	TL	166401	24	24	**0,00%**	20,78	1	584
I7C	120	50	3	23,79	24,00	25	4,00%	6,80	TL	149253	24	24	**0,00%**	13,86	1	617
I8C	120	50	5	24,32	25,00	25	**0,00%**	9,84	7,92	1	25	25	**0,00%**	11,71	1	628
I9C	120	50	10	24,93	25,00	26	3,85%	13,57	TL	206954	25	25	**0,00%**	9,79	1	720
I10C	120	50	20	22,90	23,00	24	4,17%	17,47	TL	199896	23	23	**0,00%**	9,66	1	931
I11C	250	30	2	49,28	50,00	52	3,85%	59,89	TL	13023	50	50	**0,00%**	2205,88	1	1367
I12C	250	30	3	49,51	50,00	52	3,85%	187,62	TL	19546	50	50	**0,00%**	177,38	1	1238
I13C	250	30	5	50,71	51,00	53	3,77%	336,15	TL	3179	51	51	**0,00%**	111,65	1	1310
I14C	250	30	10	49,71	50,00	52	3,85%	662,10	TL	11490	50	50	**0,00%**	70,23	1	1550
I15C	250	30	20	50,31	51,00	53	3,77%	323,36	TL	1019	51	51	**0,00%**	66,71	1	1663
I16C	250	50	2	50,41	51,00	52	1,92%	84,42	TL	29080	51	51	**0,00%**	74,64	1	1289
I17C	250	50	3	50,51	51,00	52	1,92%	115,52	TL	27622	51	51	**0,00%**	52,74	1	1284
I18C	250	50	5	51,39	52,00	53	1,89%	201,19	TL	11968	52	52	**0,00%**	41,33	1	1312
I19C	250	50	10	52,46	53,00	54	1,85%	351,01	TL	12948	53	53	**0,00%**	39,86	1	1425
I20C	250	50	20	50,10	51,00	52	1,92%	921,94	TL	33083	51	51	**0,00%**	66,88	1	2755
I21C	500	30	2	98,79	-	-	-	1324,72	TL	-	-	-	-	TL	-	-
I22C	500	30	3	100,42	-	-	-	1588,13	TL	-	101	101	**0,00%**	2779,41	1	2791
I23C	500	30	5	100,72	-	-	-	3022,69	TL	-	101	101	**0,00%**	1008,80	1	2906
I24C	500	30	10	101,88	-	-	-	TL	TL	-	102	102	**0,00%**	579,88	1	3223
I25C	500	30	20	-	-	-	-	TL	TL	-	103	103	**0,00%**	529,61	1	5043
I26C	500	50	2	102,54	-	-	-	2397,16	TL	-	103	103	**0,00%**	481,00	1	2449
I27C	500	50	3	103,44	-	-	-	1384,62	TL	-	104	104	**0,00%**	294,16	1	2536
I28C	500	50	5	101,99	-	-	-	TL	TL	-	102	102	**0,00%**	345,27	1	4481
I29C	500	50	10	97,84	-	-	-	TL	TL	-	98	98	**0,00%**	264,76	1	3350
I30C	500	50	20	100,53	-	-	-	TL	TL	-	101	101	**0,00%**	285,53	1	3446
I31C	1000	30	2	199,21	-	-	-	TL	TL	-	-	-	-	TL	-	-
I32C	1000	30	3	-	-	-	-	TL	TL	-	-	-	-	TL	-	-
I33C	1000	30	5	-	-	-	-	TL	TL	-	-	-	-	TL	-	-
I34C	1000	30	10	205,43	-	-	-	TL	TL	-	-	-	-	TL	-	-
I35C	1000	30	20	-	-	-	-	TL	TL	-	-	-	-	TL	-	-
I36C	1000	50	2	199,25	-	-	-	TL	TL	-	200	200	**0,00%**	2285,01	1	5147
I37C	1000	50	3	197,10	-	-	-	TL	TL	-	198	198	**0,00%**	1907,66	1	5423
I38C	1000	50	5	-	-	-	-	TL	TL	-	202	202	**0,00%**	3183,93	1	9307
I39C	1000	50	10	199,22	-	-	-	TL	TL	-	200	200	**0,00%**	2329,80	1	6448
I40C	1000	50	20	198,46	-	-	-	TL	TL	-	-	-	-	TL	-	-

Tabela 4.4: Resultados para as instâncias do conjunto (u) com ciclo.

				Formulação F1							Branch-and-Price				
Inst.	**n**	α	β	$\mathbf{LB_0}$	**LB**	**UB**	**Gap**	$\mathbf{T_0(s)}$	**T(s)**	**NNós**	**LB**	**UB**	**GAP**	**T(s)**	**NNós**	**NCol**
I41C	60	30	2	10,00	10,00	11	9,09%	0,20	TL	163585	11	11	**0,00%**	49,14	1	295
I42C	60	30	3	10,00	10,00	11	**9,09%**	1,26	TL	264586	10	11	**9,09%**	205,50	1	1163
I43C	60	30	5	10,00	10,00	11	9,09%	1,88	TL	533018	10	10	**0,00%**	81,50	1	1386
I44C	60	30	10	10,00	10,00	11	9,09%	1,27	TL	555672	10	10	**0,00%**	15,93	1	680
I45C	60	30	20	10,00	10,00	10	**0,00%**	0,84	271,33	119996	10	10	**0,00%**	8,97	1	791
I46C	60	50	2	10,00	10,00	11	9,09%	0,28	TL	660171	10	10	**0,00%**	56,75	1	814
I47C	60	50	3	10,00	10,00	11	9,09%	0,19	TL	1136760	10	10	**0,00%**	27,09	1	608
I48C	60	50	5	10,00	10,00	11	**9,09%**	1,08	TL	546379	10	11	**9,09%**	55,86	1	2029
I49C	60	50	10	10,00	10,00	11	9,09%	0,20	TL	529160	10	10	**0,00%**	11,05	1	703
I50C	60	50	20	10,00	10,00	11	9,09%	0,84	TL	1002576	10	10	**0,00%**	4,88	1	479
I51C	120	30	2	20,00	20,00	21	**4,76%**	7,45	TL	34397	20	21	**4,76%**	3315,17	1	1815
I52C	120	30	3	20,00	20,00	21	**4,76%**	11,05	TL	45010	20	21	**4,76%**	1080,61	1	3270
I53C	120	30	5	20,00	20,00	21	**4,76%**	22,21	TL	67648	20	21	**4,76%**	473,04	1	4615
I54C	120	30	10	20,00	20,00	21	4,76%	17,21	TL	230571	20	20	**0,00%**	256,82	1	5807
I55C	120	30	20	20,00	20,00	21	4,76%	30,76	TL	77769	20	20	**0,00%**	52,57	1	2378
I56C	120	50	2	20,00	20,00	21	4,76%	5,69	TL	59631	20	20	**0,00%**	124,72	1	1046
I57C	120	50	3	20,00	20,00	21	4,76%	7,62	TL	53390	20	20	**0,00%**	126,19	1	1997
I58C	120	50	5	20,00	20,00	21	4,76%	16,38	TL	91429	20	20	**0,00%**	88,86	1	1768
I59C	120	50	10	20,00	20,00	21	4,76%	13,08	TL	119902	20	20	**0,00%**	28,06	1	925
I60C	120	50	20	20,00	20,00	21	4,76%	34,57	TL	181732	20	20	**0,00%**	48,03	1	2347
I61C	249	30	2	41,50	42,00	43	**2,33%**	90,65	TL	12479	-	-	-	TL	-	-
I62C	249	30	3	41,50	42,00	43	2,33%	114,71	TL	12875	42	42	**0,00%**	1196,84	1	2417
I63C	249	30	5	41,50	42,00	43	2,33%	343,54	TL	14887	42	42	**0,00%**	582,44	1	1725
I64C	249	30	10	41,50	42,00	43	2,33%	610,92	TL	20026	42	42	**0,00%**	315,78	1	1910
I65C	249	30	20	41,50	-	-	-	1272,38	TL	-	42	42	**0,00%**	294,35	1	4576
I66C	249	50	2	41,50	42,00	43	2,33%	141,90	TL	16635	42	42	**0,00%**	435,25	1	1596
I67C	249	50	3	41,50	42,00	43	2,33%	139,33	TL	27851	42	42	**0,00%**	257,25	1	1575
I68C	249	50	5	41,50	42,00	43	2,33%	204,56	TL	46536	42	42	**0,00%**	181,87	1	1699
I69C	249	50	10	41,50	42,00	43	2,33%	427,69	TL	51312	42	42	**0,00%**	121,23	1	1867
I70C	249	50	20	41,50	42,00	43	2,33%	983,52	TL	29156	42	42	**0,00%**	100,59	1	2113
I71C	501	30	2	83,50	-	-	-	2393,56	TL	-	-	-	-	TL	-	-
I72C	501	30	3	83,50	-	-	-	2331,82	TL	-	-	-	-	TL	-	-
I73C	501	30	5	83,50	-	-	-	2997,38	TL	-	-	-	-	TL	-	-
I74C	501	30	10	83,50	-	-	-	TL	TL	-	84	84	**0,00%**	2047,86	1	3768
I75C	501	30	20	83,50	-	-	-	TL	TL	-	84	84	**0,00%**	1139,30	1	3970
I76C	501	50	2	83,50	84,00	86	2,33%	1085,43	TL	2038	84	84	**0,00%**	2210,31	1	3133
I77C	501	50	3	83,50	-	-	-	3077,22	TL	-	84	84	**0,00%**	1098,46	1	3154
I78C	501	50	5	83,50	-	-	-	TL	TL	-	84	84	**0,00%**	814,45	1	3375
I79C	501	50	10	83,50	-	-	-	TL	TL	-	84	84	**0,00%**	727,25	1	3797
I80C	501	50	20	83,50	-	-	-	TL	TL	-	84	84	**0,00%**	620,59	1	4149

Tabela 4.5: Resultados para as instâncias do conjunto (t) com ciclo

				Formulação F1							Branch-and-Price					
Inst.	**n**	α	β	$\mathbf{LB_0}$	**LB**	**UB**	**Gap**	$\mathbf{T_0(s)}$	**T(s)**	**NNós**	**LB**	**UB**	**GAP**	**T(s)**	**NNós**	**NCol**
I81C	120	20	2	8,60	9,00	9	**0,00%**	6,11	6,46	1	9	9	**0,00%**	638,98	1	1872
I82C	120	20	3	8,92	9,00	9	**0,00%**	11,66	19,57	1	9	9	**0,00%**	409,24	1	1102
I83C	120	20	5	8,62	9,00	9	**0,00%**	16,37	18,75	1	9	9	**0,00%**	401,47	1	2482
I84C	120	20	10	9,27	10,00	10	**0,00%**	24,11	23,71	1	10	10	**0,00%**	163,81	1	1200
I85C	120	20	20	8,41	9,00	9	**0,00%**	30,20	27,45	1	9	9	**0,00%**	133,52	1	1601
I86C	120	50	2	9,31	10,00	10	**0,00%**	5,33	6,90	1	10	10	**0,00%**	208,65	1	921
I87C	120	50	3	9,24	10,00	10	**0,00%**	9,87	9,35	1	10	10	**0,00%**	308,18	1	1949
I88C	120	50	5	9,04	10,00	10	**0,00%**	17,29	17,14	1	10	10	**0,00%**	177,66	1	1716
I89C	120	50	10	9,01	10,00	10	**0,00%**	16,11	16,48	1	10	10	**0,00%**	252,16	1	3304
I90C	120	50	20	8,69	9,00	9	**0,00%**	46,70	27,83	1	9	9	**0,00%**	103,26	1	1666
I91C	250	20	2	18,37	19,00	19	**0,00%**	223,77	156,69	1	-	-	-	TL	-	-
I92C	250	20	3	18,23	19,00	19	**0,00%**	215,18	486,56	1	19	19	**0,00%**	2263,97	1	2859
I93C	250	20	5	18,91	19,00	20	5,00%	501,15	TL	46215	19	19	**0,00%**	1667,78	1	2963
I94C	250	20	10	19,06	20,00	20	**0,00%**	807,92	762,13	1	20	20	**0,00%**	3124,50	1	10513
I95C	250	20	20	18,29	19,00	19	**0,00%**	1674,64	2029,66	1	19	19	**0,00%**	720,37	1	3803
I96C	250	50	2	19,43	20,00	20	**0,00%**	111,53	116,15	1	20	20	**0,00%**	1294,32	1	3265
I97C	250	50	3	18,38	19,00	19	**0,00%**	329,35	189,49	1	19	19	**0,00%**	3019,43	1	11393
I98C	250	50	5	18,73	19,00	19	**0,00%**	155,30	233,46	1	19	19	**0,00%**	685,67	1	2992
I99C	250	50	10	19,26	20,00	20	**0,00%**	431,93	424,04	1	20	20	**0,00%**	646,45	1	3463
I100C	250	50	20	19,24	20,00	20	**0,00%**	612,05	1134,41	1	20	20	**0,00%**	542,44	1	4108
I101C	500	20	2	36,60	-	-	-	1572,64	TL	-	-	-	-	TL	-	-
I102C	500	20	3	-	-	-	-	TL	TL	-	-	-	-	TL	-	-
I103C	500	20	5	37,72	-	-	-	TL	TL	-	-	-	-	TL	-	-
I104C	500	20	10	37,10	-	-	-	TL	TL	-	-	-	-	TL	-	-
I105C	500	20	20	38,49	-	-	-	TL	TL	-	-	-	-	TL	-	-
I106C	500	50	2	37,36	38,00	38	**0,00%**	1247,52	2528,09	1	38	38	**0,00%**	3358,78	1	5471
I107C	500	50	3	36,76	36,76	498	**92,62%**	2005,42	TL	1	-	-	-	TL	-	-
I108C	500	50	5	37,88	-	-	-	TL	TL	-	-	-	-	TL	-	-
I109C	500	50	10	37,01	-	-	-	TL	TL	-	38	38	**0,00%**	3459,62	1	8017
I110C	500	50	20	-	-	-	-	TL	TL	-	-	-	-	TL	-	-
I111C	1000	20	2	74,57	-	-	-	TL	TL	-	-	-	-	TL	-	-
I112C	1000	20	3	-	-	-	-	TL	TL	-	-	-	-	TL	-	-
I113C	1000	20	5	-	-	-	-	TL	TL	-	-	-	-	TL	-	-
I114C	1000	20	10	-	-	-	-	TL	TL	-	-	-	-	TL	-	-
I115C	1000	20	20	-	-	-	-	TL	TL	-	-	-	-	TL	-	-
I116C	1000	50	2	-	-	-	-	TL	TL	-	-	-	-	TL	-	-
I117C	1000	50	3	75,60	-	-	-	TL	TL	-	-	-	-	TL	-	-
I118C	1000	50	5	-	-	-	-	TL	TL	-	-	-	-	TL	-	-
I119C	1000	50	10	-	-	-	-	TL	TL	-	-	-	-	TL	-	-
I120C	1000	50	20	-	-	-	-	TL	TL	-	-	-	-	TL	-	-

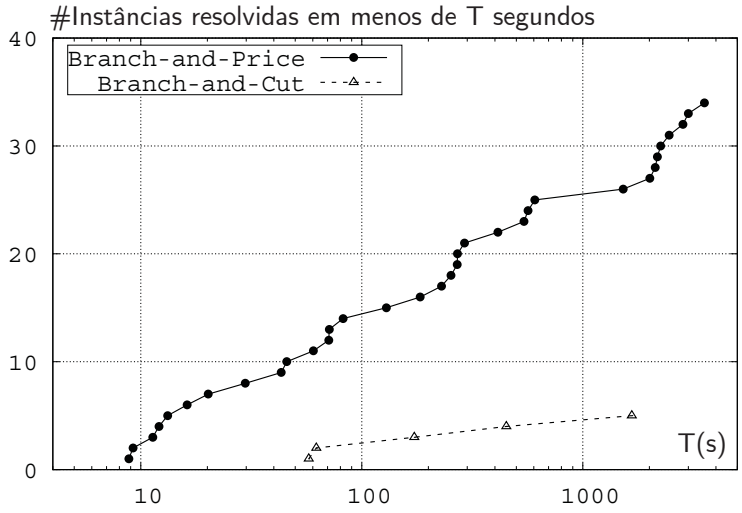
Tabela 4.6: Resultados para instâncias do conjunto (d) com ciclo.

Finalmente, a Figura 4.3 resume as observações anteriores. Nesta figura o eixo x representa o tempo de execução de cada solução encontrada pelo método, enquanto o eixo y representa o número de instâncias do conjunto ((u), (t) e (d), sem ciclos) que foram resolvidas *para provar a otimalidade* dentro deste tempo, para cada método. Nota-se que quanto mais alto o gráfico na figura, melhor é o método. Isso significa que mais instâncias foram resolvidas em menos tempo. Da mesma forma, a Figura 4.4 fornece os mesmos resultados para os conjuntos de instâncias com ciclo.

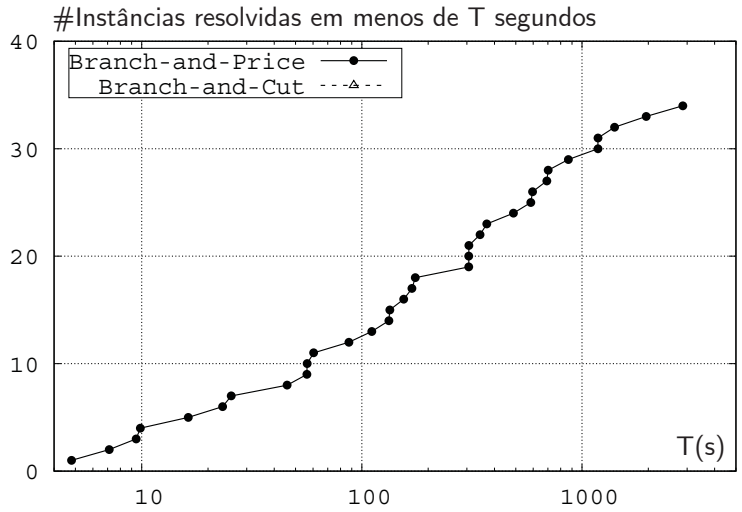
Na Figura 4.3, pode-se observar claramente um comportamento diferente dos algoritmos B&P e B&C, dependendo do conjunto de instâncias. Para as instâncias dos conjuntos (t) e (u), cujos *bins* contém poucos itens, o problema de *pricing* é resolvido de forma muito eficiente, e o B&P obtém mais soluções em menos tempo do que o B&C. No conjunto (t), em particular, a diferença é óbvia, uma vez que o *branch-and-cut* não resolve até a otimalidade nenhuma instância, enquanto o *branch-and-price* consegue resolver 25/40 instâncias em menos de 1000 segundos. No conjunto (d), a tendência é inversa, mesmo que menos dramática: o B&C visivelmente supera o algoritmo B&P, provavelmente devido ao fato de que o número médio de itens por *bin* nessas instâncias é maior, tornando o problema de *pricing* mais difícil de resolver.

Finalmente, considerando agora as instâncias com ciclos, as mesmas observações anteriores podem ser vistas na Figura 4.4. No conjunto (u), o B&C pôde resolver 5/40 instâncias até a otimalidade, em comparação com 23/40 do B&P, enquanto que no conjunto (t), o método B&C resolveu apenas uma instância, e o B&P foi capaz de resolver quase todo o conjunto (36/40). Mais uma vez, o comportamento inverso é observado para o conjunto (d), embora as diferenças sejam menores: o gráfico que representa o algoritmo B&P é apenas um pouco mais baixo do que o do algoritmo B&C, e mais uma instância do problema é resolvida por B&P.

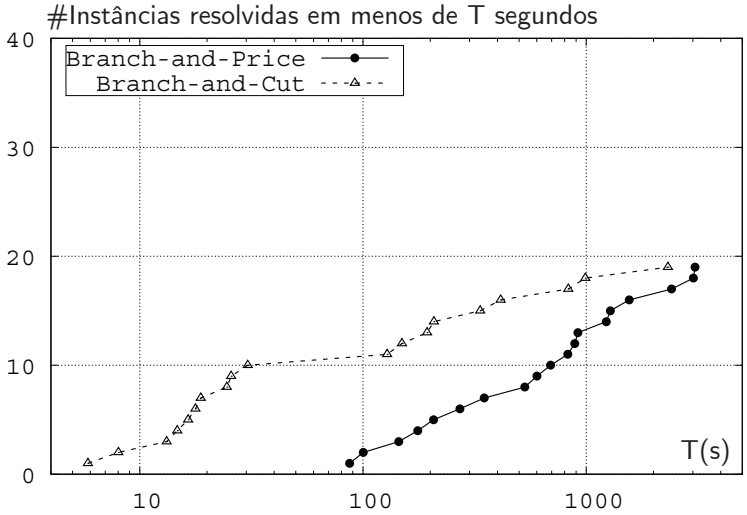
Em geral, os dois métodos parecem complementares e podem ser eficientes em condições diferentes. Para problemas com mais itens por *bins*, o *branch-and-cut* deve ser o método de escolha, enquanto o *branch-and-price* deve ser escolhido nos outros casos.



(a) Conjunto (u) sem ciclo.

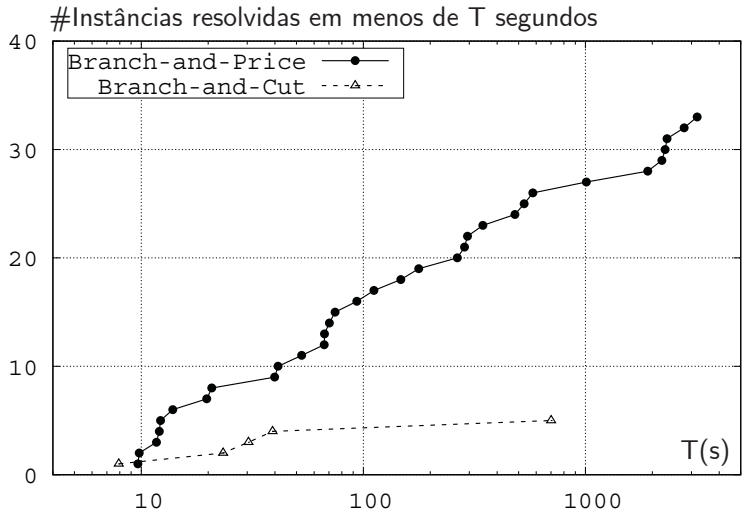


(b) Conjunto (t) sem ciclo.

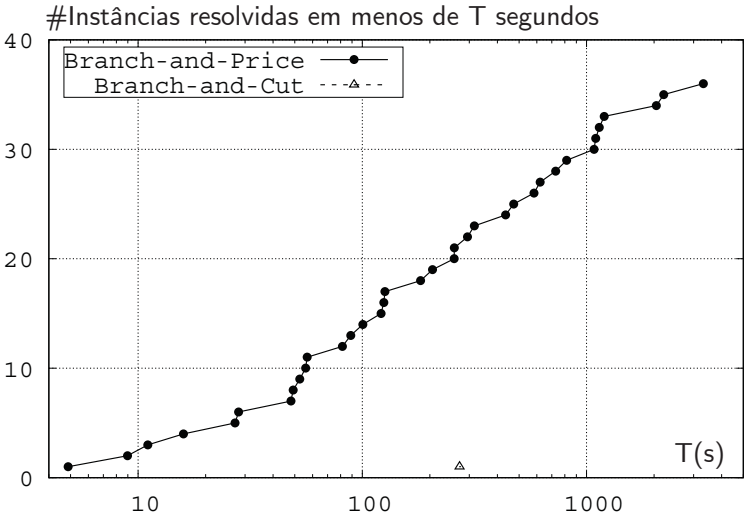


(c) Conjunto (d) sem ciclo.

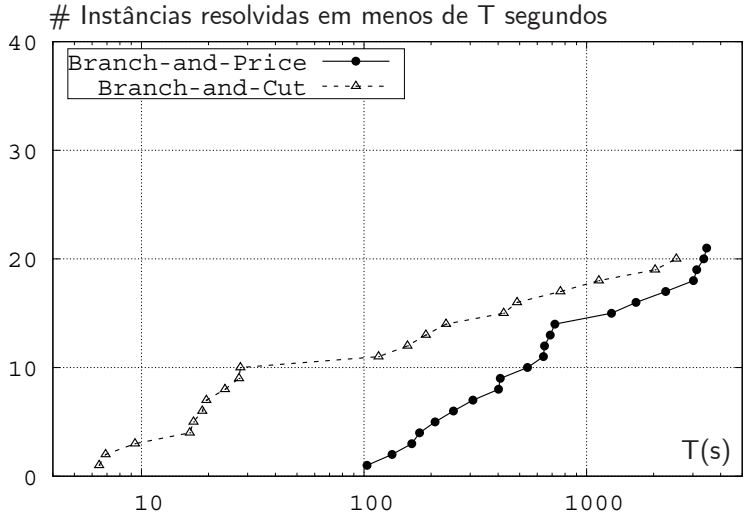
Figura 4.3: Número de instâncias resolvidas em uma dada quantidade de tempo, para os conjuntos (u), (t) e (d), sem ciclo.



(a) Conjunto (u) com ciclo.



(b) Conjunto (t) com ciclo.



(c) Conjunto (d) com ciclo.

Figura 4.4: Número de instâncias resolvidas em uma dada quantidade de tempo, para os conjuntos (u), (t) e (d) com ciclo.

Capítulo 5

Conclusões e Trabalhos Futuros

Nesta tese foram apresentados métodos heurísticos e exatos para resolver respectivamente, o Problema de *Bin Packing* com Conflitos (PBPC) e o Problema de *Bin Packing* com Dependências (PBPD).

Para o PBPC, foi desenvolvido um método baseado no ILS com várias classes de vizinhanças locais e em larga escala. O objetivo foi investigar a utilização de estruturas de vizinhanças mais inteligentes estudando como estas podem contribuir para alcançar soluções de alta qualidade. O método desenvolvido produz soluções de alta qualidade nas instâncias clássicas do PBPC, com uma diferença média para o BKS de 0,22%. Esta qualidade de solução encontrada é superior quando comparada a melhor metaheurística anteriormente conhecida da literatura, o algoritmo genético de [Muritiba et al., 2010], e ligeiramente inferior, com relação à qualidade da solução, comparada à heurística de *diving* com busca de discrepância limitada (DH-LDS) de [Sadykov e Vanderbeck, 2013]. No entanto, essas instâncias utilizadas são ideais para métodos que utilizam decomposições matemáticas e *pricing*, como o DH-LDS, pois contêm poucos itens por bin (2,87 em média). Por outro lado, o ILS proposto é escalável quando o tamanho das instâncias aumenta, com um esforço computacional de $\mathcal{O}(n^{1,49})$. Em comparação, o tempo de CPU de DH-LDS tende a aumentar em um fator de dez cada vez que o tamanho do problema dobra.

Foram realizadas extensivas análises de sensibilidade para medir a contribuição de cada vizinhança, bem como o impacto de seus parâmetros chave. Observou-se que os movimentos de *custo-0* são críticos para o desempenho da busca, bem como a vizinhança baseada no problema de cobertura de conjuntos. As outras vizinhanças em larga escala (*ejection chains*, *assignment* e *grenade*) têm um impacto individual menor, mas contribuem significativamente para a busca quando utilizadas em conjunto.

Muitas perspectivas de pesquisa ainda estão abertas sobre essa variante do problema de *bin packing*. A existência de dois métodos competitivos deve inspirar novos métodos hibridizados que visam reunir as qualidades de ambas as abordagens em um tempo mais controlado, bem como possíveis conceitos de população. Assim, a pesquisa futura poderia ser direcionada para o desenvolvimento de novas *matheuristicas*, com o objetivo de aproveitar a força da programação matemática e a busca em vizinhança para este problema. Como o conjunto de instâncias atuais são agora resolvidas até quase a otimalidade, novos conjuntos de testes maiores (em termos de número total de itens e número de itens por *bin*) também devem ser investigados.

O PBPC generaliza outros três problemas bem conhecidos da literatura, o Problema de *Bin Packing* Clássico, o Problema de Coloração de Vértices e o *Bounded Vertex Coloring Problem*. Propõe-se como trabalhos futuros investigar ainda a aplicabilidade do método proposto, através da realização de testes e adaptações dos algoritmos, para esses três outros problemas.

Com relação ao Problema de *Bin Packing* com Dependências, neste trabalho foi realizada uma descrição formal deste problema, até então inédito na literatura. Para sua resolução, foram desenvolvidos métodos exatos: um modelo de programação inteira desenvolvido para o problema e um *branch-and-price*.

No modelo matemático foram incorporadas algumas desigualdades já utilizadas para outras variantes do *bin packing*. Estas tem por objetivo diminuir a simetria e melhorar a relaxação linear do modelo. O *branch-and-price* por sua vez foi implementado utilizado o *framework* BaPCode criado por Sadykov e Vanderbeck [Sadykov e Vanderbeck, 2013]. Foram incorporados nessa ferramenta as especificidades que o PBPD possui, como as formulações e os algoritmos para resolver o problema de *pricing*. O subproblema de *pricing* tratado foi o problema da mochila com dependências, que foi resolvido utilizando-se um modelo matemático e uma heurística baseada no algoritmo de busca em profundidade.

Para testar os métodos desenvolvidos, foram criadas 240 instâncias para o PBPD que foram criadas a partir das instâncias existentes para o problema de *bin packing* com conflitos. De modo geral, os resultados da execução do *branch-and-price*, quando comparados com a implementação do modelo matemático, são relativamente superiores para a maioria dos conjuntos de instâncias do problema. O *branch-and-price* conseguiu provar a otimalidade de 167 das 240 instâncias existentes. Enquanto a implementação do modelo alcançou o ótimo em 50 instâncias. Com relação ao tempo de execução, o *branch-and-price* obteve os resultados em

menos tempo, à exceção das instâncias pertencentes ao conjunto (d), que possui um número maior de itens por *bin*.

Como esse foi um trabalho inicial sobre o problema, ainda existe uma gama grande de métodos a se investigar para o PBPD. Inicialmente, propomos o aprimoramento da heurística de *pricing*, usando técnicas heurísticas e matheuristiscas na maioria das iterações, e completando com a quantidade mínima necessária de execuções do *pricing* exato.

Como visto na seção dos resultados apresentados, a resolução de instâncias do PBPD com uma maior quantidade de itens ainda é um desafio para métodos exatos. Como trabalho futuro, consideramos que a elaboração de uma metaheurística para o PBPD poderia colaborar no sentido de resolver esse tipo de instância.

Finalmente, as técnicas desenvolvidas para o PBPC e para o PBPD aliada aos novos conhecimentos obtidos nesse trabalho podem contribuir para a resolução eficiente de variantes mais avançadas dos problemas de *bin packing*, tal como o problema de realocação de máquinas proposto pelo Google. Neste caso, o tamanho do problema se torna também um desafio, já que as instancias práticas envolvem milhões de tarefas. Existem todas essas promissoras linhas de pesquisa para o futuro.

Referências

[Ahuja et al., 2002] Ahuja, R. K., Ergun, O., Orlin, J. B., e Punnen, A. P. (2002). A survey of very large-scale neighborhood search techniques. *Discrete Applied Mathematics*, 123(1-3):75–102.

[Avanthay et al., 2003] Avanthay, C., Hertz, A., e Zufferey, N. (2003). A variable neighborhood search for graph coloring. *European Journal of Operational Research*, 151(2):379–388.

[Barnhart et al., 1998] Barnhart, C., Johnson, E. L., Nemhauser, G. L., Savelsbergh, M. W. P., e Vance, P. H. (1998). Branch-and-price: column generation for solving huge integer programs. *Operations Research*, 46(3):316–329.

[Blöchliger e Zufferey, 2008] Blöchliger, I. e Zufferey, N. (2008). A graph coloring heuristic using partial solutions and a reactive tabu scheme. *Computers & Operations Research*, 35(3):960–975.

[Borradaile et al., 2012] Borradaile, G., Heeringa, B., e Wilfong, G. (2012). The knapsack problem with neighbour constraints. *Journal of Discrete Algorithms*, 16:224–235.

[Coffman et al., 1984] Coffman, E.G., J., Garey, M., e Johnson, D. (1984). Approximation algorithms for bin-packing – An updated survey. Em Ausiello, G., Lucertini, M., e Serafini, P., editores, *Algorithm Design for Computer System Design*, volume 284 de *International Centre for Mechanical Sciences*, pp. 49–106. Springer Vienna.

[Cormen et al., 2009] Cormen, T. H., Leiserson, C. E., Rivest, R. L., e Stein, C. (2009). *Introduction to Algorithms, Third Edition*. The MIT Press, 3rd edição.

[Dantzig e Wolfe, 1960] Dantzig, G. B. e Wolfe, P. (1960). Decomposition principle for linear programs. *Operations Research*, 8(1):101–111.

[Deineko e Woeginger, 2000] Deineko, V. G. e Woeginger, G. J. (2000). A study of exponential neighborhoods for the travelling salesman problem and for the quadratic assignment problem. *Mathematical Programming*, 87(3):519–542.

[Delorme et al., 2016] Delorme, M., Iori, M., e Martello, S. (2016). Bin packing and cutting stock problems: mathematical models and exact algorithms. *European Journal of Operational Research*, 255(1):1–20.

[Dowsland e Thompson, 2008] Dowsland, K. e Thompson, J. (2008). An improved ant colony optimisation heuristic for graph colouring. *Discrete Applied Mathematics*, 156(3):313–324.

[Elhedhli et al., 2011] Elhedhli, S., Li, L., Gzara, M., e Naoum-Sawaya, J. (2011). A branch-and-price algorithm for the bin packing problem with conflicts. *INFORMS Journal on Computing*, 23(3):404–415.

[Epstein e Levin, 2008] Epstein, L. e Levin, A. (2008). On bin packing with conflicts. *SIAM Journal on Optimization*, 19(3):1270–1298.

[Epstein et al., 2008] Epstein, L., Levin, A., e Stee, R. (2008). Two-dimensional packing with conflicts. *Acta Informatica*, 45(3):155–175.

[Falkenauer, 1996] Falkenauer, E. (1996). A hybrid grouping genetic algorithm for bin packing. *Journal of Heuristics*, 2(1):5–30.

[Fleszar e Charalambous, 2011] Fleszar, K. e Charalambous, C. (2011). Average-weight-controlled bin-oriented heuristics for the one-dimensional bin-packing problem. *European Journal of Operational Research*, 210(2):176–184.

[Galinier e Hao, 1999] Galinier, P. e Hao, J. (1999). Hybrid evolutionary algorithms for graph coloring. *Journal of combinatorial optimization*, 3(4):379–397.

[Gavranović et al., 2012] Gavranović, H., Buljubašić, M., e Demirović, E. (2012). Variable neighborhood search for google machine reassignment problem. *Electronic Notes in Discrete Mathematics*, 39:209–216.

[Gendreau et al., 2004] Gendreau, M., Laporte, G., e Semet, F. (2004). Heuristics and lower bounds for the bin packing problem with conflicts. *Computers & Operations Research*, 31(3):347–358.

[Glover, 1996] Glover, F. (1996). Ejection chains, reference structures and alternating path methods for traveling salesman problems. *Discrete Applied Mathematics*, 65(1-3):223–253.

[Gutin, 1999] Gutin, G. (1999). Exponential neighbourhood local search for the traveling salesman problem. *Computers & Operations Research*, 26(4):313–320.

[Hamdi-Dhaoui et al., 2014] Hamdi-Dhaoui, K., Labadie, N., e Yalaoui, A. (2014). The bi-objective two-dimensional loading vehicle routing problem with partial conflicts. *International Journal of Production Research*, 52(19):5565–5582.

[Hertz e de Werra, 1987] Hertz, A. e de Werra, D. (1987). Using tabu search techniques for graph coloring. *Computing*, 39(4):345–351.

[Jansen, 1999] Jansen, K. (1999). An approximation scheme for bin packing with conflicts. *Journal of Combinatorial Optimization*, 3:363–377.

[Jansen e Öhring, 1997] Jansen, K. e Öhring, S. (1997). Approximation algorithms for time constrained scheduling. *Information and Computation*, 132(2):85–108.

[Joncour et al., 2010] Joncour, C., Michel, S., Sadykov, R., Sverdlov, D., e Vanderbeck, F. (2010). Column generation based primal heuristics. *Electronic Notes in Discrete Mathematics*, 36:695 – 702.

[Khanafer et al., 2010] Khanafer, A., Clautiaux, F., e Talbi, E.-G. (2010). New lower bounds for bin packing problems with conflicts. *European Journal of Operational Research*, 206(2):281–288.

[Kuhn, 1955] Kuhn, H. W. (1955). The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97.

[Laporte e Desrochers, 1984] Laporte, G. e Desrochers, S. (1984). Examination timetabling by computer. *Computers & Operations Research*, 11(4):351–360.

[Lewis, 2016] Lewis, R. (2016). *A guide to graph colouring: algorithms and applications*. Springer International Publishing.

[Lewis et al., 2012] Lewis, R., Thompson, J., Mumford, C., e Gillard, J. (2012). A wide-ranging computational comparison of high-performance graph colouring algorithms. *Computers & Operations Research*, 39(9):1933–1950.

[Lopes et al., 2015] Lopes, R., Morais, V. W. C., Noronha, T. F., e Souza, V. a. a. (2015). Heuristics and matheuristics for a real-life machine reassignment problem. *International Transactions in Operational Research*, 22(1):77–95.

[Lourenço et al., 2010] Lourenço, H., Martin, O., e Stützle, T. (2010). Iterated Local Search: framework and applications. Em Gendreau, M. e Potvin, J.-Y., editores, *Handbook of Metaheuristics*, volume 146 de *International Series in Operations Research & Management Science*, pp. 363–397. Springer US, 2nd edição.

[Malaguti et al., 2008] Malaguti, E., Monaci, M., e Toth, P. (2008). A metaheuristic approach for the vertex coloring problem. *INFORMS Journal on Computing*, 20(2):302–316.

[Malaguti e Toth, 2010] Malaguti, E. e Toth, P. (2010). Survey on vertex coloring problems. *International Transactions in Operational Research*, 17(1):1–34.

[Masson et al., 2013] Masson, R., Vidal, T., Michallet, J., Penna, P. H. V., Petrucci, V., Subramanian, A., e Dubedout, H. (2013). An iterated local search heuristic for multi-capacity bin packing and machine reassignment problems. *Expert Systems with Applications*, 40(13):5266–5275.

[Minh et al., 2013] Minh, T., Van Hoai, T., e Nguyet, T. (2013). A memetic algorithm for waste collection vehicle routing problem with time windows and conflicts. Em Murgante, B., Misra, S., Carlini, M., Torre, C., Nguyen, H.-Q., Taniar, D., Apduhan, B., e Gervasi, O., editores, *Computational Science and Its Applications - ICCSA 2013*, volume 7971 de *Lecture Notes in Computer Science*, pp. 485–499. Springer Berlin Heidelberg.

[Monaci e Toth, 2006] Monaci, M. e Toth, P. (2006). A set-covering-based heuristic approach for bin-packing problems. *INFORMS Journal on Computing*, 18(1):71–85.

[Morgenstern, 1996] Morgenstern, C. (1996). Distributed coloration neighborhood search. Em Johnson, D. e Trick, M., editores, *Discrete Mathematics and Theoretical Computer Science*, volume 26, pp. 335–358. American Mathematical Society, Providence, RI.

[Muritiba et al., 2010] Muritiba, A., Iori, M., Malaguti, E., e Toth, P. (2010). Algorithms for the bin packing problem with conflicts. *INFORMS Journal on Computing*, 22(3):401–415.

[Muter et al., 2010] Muter, I., Birbil, S. I., e Sahin, G. (2010). Combination of metaheuristic and exact algorithms for solving set covering-type optimization problems. *INFORMS Journal on Computing*, 22(4):603–619.

[Portal et al., 2015] Portal, G. M., Ritt, M., Borba, L. M., e Buriol, L. S. (2015). Simulated annealing for the machine reassignment problem. *Annals of Operations Research*, pp. 1–22.

[Quiroz-Castellanos et al., 2015] Quiroz-Castellanos, M., Cruz-Reyes, L., Torres-Jimenez, J., Gómez, C., Huacuja, H. J. F., e Alvim, A. C. (2015). A grouping genetic algorithm with controlled gene transmission for the bin packing problem. *Computers & Operations Research*, 55:52–64.

[Roadef, 2012,] Roadef, 2012. ROADEF/EURO challenge 2012: Machine reassignment. `http://challenge.roadef.org/2012/en/index.php`. Acessado: 06.09.2016.

[Sadykov e Vanderbeck, 2013] Sadykov, R. e Vanderbeck, F. (2013). Bin packing with conflicts: a generic branch-and-price algorithm. *INFORMS Journal on Computing*, 25(2):244–255.

[Subramanian et al., 2013] Subramanian, A., Uchoa, E., e Ochi, L. S. (2013). A hybrid algorithm for a class of vehicle routing problems. *Computers & Operations Research*, 40(10):2519–2531.

[Thompson e Psaraftis, 1993] Thompson, P. e Psaraftis, H. (1993). Cyclic transfer algorithms for multi-vehicle routing and scheduling problems. *Operations Research*, 41(5):935–946.

[Toth e Tramontani, 2008] Toth, P. e Tramontani, A. (2008). *The vehicle routing problem: latest advances and new challenges*, volume 43 de *Operations Research/Computer Science Interfaces*, chapter An integer linear programming local search for capacitated vehicle routing problems, pp. 275–295. Springer US, Boston, MA.

[Vanderbeck, 2011] Vanderbeck, F. (2011). Branching in branch-and-price: a generic scheme. *Mathematical Programming*, 130(2):249–294.

[Wang et al., 2016] Wang, Z., Lü, Z., e Ye, T. (2016). Multi-neighborhood local search optimization for machine reassignment problem. *Computers & Operations Research*, 68:16–29.

APÊNDICE A – Resultados Detalhados para o PBPC

Inst.		HILS-*Completo*				ILS-*Simples*				PH			DH-LDS			BKS	
n	d	Avg Gap	Best Gap	T(s)	Opt	Avg Gap	Best Gap	T(s)	Opt	Gap	T(s)	Opt	Gap	T(s)	Opt	#Bins	Opt
60	0	**0,00**	**0,00**	0,62	10	4,15	2,50	0,81	5	**0,00**	0,00	10	–	–	–	20,0	10
	10	**0,00**	**0,00**	0,84	10	3,15	2,00	0,87	6	4,00	121,00	2	**0,00**	0,10	10	20,2	10
	20	**0,00**	**0,00**	1,61	10	0,45	**0,00**	1,03	10	0,50	121,00	9	**0,00**	0,31	10	20,9	10
	30	**0,00**	**0,00**	1,53	10	**0,00**	**0,00**	1,02	10	**0,00**	84,70	10	**0,00**	0,21	10	21,7	10
	40	**0,00**	**0,00**	1,12	10	**0,00**	**0,00**	0,69	10	**0,00**	12,10	10	**0,00**	0,28	10	24,8	10
	50	**0,00**	**0,00**	1,05	10	**0,00**	**0,00**	0,67	10	**0,00**	12,10	10	**0,00**	0,28	10	30,7	10
	60	**0,00**	**0,00**	1,07	10	0,22	**0,00**	0,59	10	**0,00**	0,00	10	**0,00**	0,22	10	36,3	10
	70	**0,00**	**0,00**	1,11	10	**0,00**	**0,00**	0,53	10	**0,00**	24,20	10	**0,00**	0,19	10	42,6	10
	80	**0,00**	**0,00**	1,28	10	**0,00**	**0,00**	0,51	10	**0,00**	0,00	10	**0,00**	0,15	10	49,0	10
	90	**0,00**	**0,00**	1,71	10	**0,00**	**0,00**	0,60	10	**0,00**	0,00	10	**0,00**	0,10	10	54,4	10
120	0	**0,00**	**0,00**	2,26	10	1,60	0,50	2,67	8	**0,00**	0,00	10	–	–	–	40,0	10
	10	**0,00**	**0,00**	2,42	10	1,90	0,75	3,16	7	2,50	121,40	0	**0,00**	1,58	10	40,0	10
	20	**0,00**	**0,00**	3,33	10	2,43	2,00	4,29	2	2,50	121,00	0	**0,00**	1,66	10	40,0	10
	30	0,10	**0,00**	12,96	10	0,75	0,75	4,88	7	0,75	96,80	7	0,25	3,12	9	40,9	10
	40	**0,00**	**0,00**	2,39	10	0,14	**0,00**	1,60	10	0,23	24,20	9	**0,00**	1,31	10	48,3	10
	50	**0,00**	**0,00**	2,89	10	0,29	**0,00**	1,44	10	**0,00**	0,10	10	**0,00**	1,40	10	60,9	10
	60	**0,00**	**0,00**	3,50	10	**0,00**	**0,00**	1,28	10	**0,00**	12,10	10	**0,00**	1,13	10	74,4	10
	70	**0,00**	**0,00**	4,65	10	0,11	**0,00**	1,53	10	**0,00**	0,10	10	**0,00**	0,90	10	84,9	10
	80	**0,00**	**0,00**	6,19	10	0,02	**0,00**	1,41	10	**0,00**	24,20	10	**0,00**	0,70	10	96,6	10
	90	**0,00**	**0,00**	8,22	10	**0,00**	**0,00**	1,49	10	**0,00**	0,10	10	**0,00**	0,50	10	109,0	10
249	0	**0,00**	**0,00**	0,37	10	**0,00**	**0,00**	0,16	10	**0,00**	0,00	10	–	–	–	83,0	10
	10	**0,00**	**0,00**	0,50	10	**0,00**	**0,00**	0,24	10	1,20	126,90	0	**0,00**	6,22	10	83,0	10
	20	**0,00**	**0,00**	1,44	10	**0,00**	**0,00**	1,04	10	1,20	126,70	0	**0,00**	9,95	10	83,0	10
	30	0,69	0,48	48,11	6	0,89	0,60	16,73	5	1,20	117,10	1	**0,00**	14,54	10	83,2	10
	40	**0,00**	**0,00**	7,86	10	0,52	0,19	3,78	8	0,33	48,60	7	**0,00**	10,24	10	98,4	10
	50	**0,00**	**0,00**	12,27	10	0,33	0,17	4,27	8	**0,00**	12,40	10	**0,00**	9,29	10	124,8	10
	60	**0,00**	**0,00**	19,09	10	0,53	0,21	4,43	8	**0,00**	49,00	10	**0,00**	7,42	10	150,1	10
	70	**0,00**	**0,00**	22,57	10	0,55	0,18	4,72	7	**0,00**	24,80	10	**0,00**	5,67	10	174,7	10
	80	**0,00**	**0,00**	21,30	10	0,24	**0,00**	5,02	10	**0,00**	0,40	10	**0,00**	4,03	10	200,2	10
	90	**0,00**	**0,00**	22,27	10	0,07	**0,00**	5,63	10	**0,00**	12,60	10	**0,00**	2,95	10	225,1	10
501	0	**0,00**	**0,00**	0,45	10	**0,00**	**0,00**	0,22	10	**0,00**	0,00	10	–	–	–	167,0	10
	10	**0,00**	**0,00**	0,43	10	**0,00**	**0,00**	0,22	10	0,60	166,80	0	**0,00**	39,84	10	167,0	10
	20	**0,00**	**0,00**	0,68	10	**0,00**	**0,00**	0,32	10	0,60	159,20	0	**0,00**	75,82	10	167,0	10
	30	0,07	**0,00**	17,35	10	0,15	**0,00**	8,88	10	0,89	161,60	0	**0,00**	78,74	10	167,5	10
	40	**0,00**	**0,00**	23,75	10	0,28	0,10	10,28	8	**0,00**	13,30	10	**0,00**	65,38	10	204,9	10
	50	**0,00**	**0,00**	29,20	10	0,53	0,32	13,63	4	**0,00**	26,30	10	**0,00**	63,80	10	255,3	10
	60	**0,00**	**0,00**	28,74	10	0,39	0,10	17,18	7	**0,00**	26,80	10	**0,00**	47,96	10	305,3	10
	70	**0,00**	**0,00**	35,07	10	0,43	0,23	20,76	3	**0,00**	15,30	10	**0,00**	34,50	10	355,1	10
	80	**0,00**	**0,00**	38,99	10	0,34	0,13	22,50	7	**0,00**	16,10	10	**0,00**	29,38	10	405,3	10
	90	**0,00**	**0,00**	40,67	10	0,24	0,07	24,98	7	**0,00**	3,90	10	**0,00**	18,50	10	453,8	10
Average		0,02	0,01	10,80	396,00	0,52	0,27	4,90	84,25	0,41	47,07	76,25	0,01	14,95	89,75	122,73	400

Tabela A.1: Resultados computacionais detalhados para as instâncias do grupo (t).

Inst.		HILS-*Completo*				ILS-*Simples*				PH			DH-LDS			BKS	
n	d	Avg Gap	Best Gap	T(s)	Opt	Avg Gap	Best Gap	T(s)	Opt	Gap	T(s)	Opt	Gap	T(s)	Opt	#Bins	Opt
120	0	**0,00**	**0,00**	0,00	10	**0,00**	**0,00**	0,00	10	**0,00**	2,30	10	–	–	–	48,3	10
	10	**0,00**	**0,00**	0,00	10	**0,00**	**0,00**	0,00	10	**0,00**	2,20	10	**0,00**	0,61	10	48,3	10
	20	**0,00**	**0,00**	0,01	10	**0,00**	**0,00**	0,01	10	**0,00**	0,50	10	**0,00**	0,61	10	48,3	10
	30	0,12	**0,00**	1,16	10	0,36	0,20	0,84	9	0,40	24,60	8	**0,00**	1,05	10	48,3	10
	40	0,13	**0,00**	2,43	10	0,17	**0,00**	1,78	10	0,41	36,30	8	**0,00**	0,81	10	52,8	10
	50	**0,00**	**0,00**	3,05	10	**0,00**	**0,00**	1,60	10	0,17	12,20	9	**0,00**	0,88	10	64,3	10
	60	**0,00**	**0,00**	4,81	10	**0,00**	**0,00**	1,73	10	**0,00**	48,40	10	**0,00**	0,72	10	75,8	10
	70	**0,00**	**0,00**	5,02	10	**0,00**	**0,00**	1,45	10	**0,00**	12,10	10	**0,00**	0,56	10	87,5	10
	80	**0,00**	**0,00**	7,19	10	0,03	**0,00**	1,80	10	**0,00**	48,50	10	**0,00**	1,19	10	98,3	10
	90	**0,00**	**0,00**	9,32	10	**0,00**	**0,00**	1,82	10	**0,00**	36,40	10	**0,00**	0,38	10	109,8	10
250	0	**0,00**	**0,00**	0,20	10	**0,00**	**0,00**	0,08	10	0,39	72,80	6	–	–	–	101,4	10
	10	**0,00**	**0,00**	0,20	10	**0,00**	**0,00**	0,09	10	0,19	28,00	8	**0,00**	2,73	10	101,4	10
	20	**0,00**	**0,00**	0,23	10	**0,00**	**0,00**	0,14	10	0,19	26,50	8	**0,00**	2,84	10	101,4	10
	30	0,07	**0,00**	6,24	10	0,11	0,10	2,44	9	0,29	38,40	7	**0,00**	3,42	10	101,4	10
	40	0,05	**0,00**	16,99	10	0,17	**0,00**	6,75	10	0,68	85,80	4	**0,00**	5,17	10	104,8	10
	50	**0,00**	**0,00**	15,24	10	0,32	0,17	5,43	8	0,08	37,50	9	**0,00**	4,53	10	126,0	10
	60	**0,00**	**0,00**	22,31	10	0,22	**0,00**	5,66	10	0,14	60,80	8	**0,00**	3,77	10	152,2	10
	70	**0,00**	**0,00**	40,15	10	0,21	**0,00**	7,78	10	0,11	97,50	8	**0,00**	3,33	10	176,9	10
	80	**0,00**	**0,00**	30,96	10	0,11	0,05	6,29	9	0,05	48,90	9	**0,00**	2,49	10	202,6	10
	90	**0,00**	**0,00**	25,77	10	0,04	**0,00**	7,47	10	**0,00**	24,90	10	**0,00**	1,77	10	226,8	10
500	0	**0,00**	**0,00**	1,19	10	**0,00**	**0,00**	0,37	10	0,44	98,70	2	–	–	–	202,6	10
	10	**0,00**	**0,00**	1,29	10	**0,00**	**0,00**	0,40	10	0,15	56,20	7	**0,00**	14,16	10	202,6	10
	20	**0,00**	**0,00**	1,38	10	**0,00**	**0,00**	0,46	10	0,15	60,80	7	**0,00**	19,00	10	202,6	10
	30	**0,00**	**0,00**	2,26	10	**0,00**	**0,00**	1,26	10	0,20	76,20	6	**0,00**	18,30	10	202,6	10
	40	0,08	0,05	62,39	9	0,27	0,19	32,90	6	0,88	113,10	2	**0,00**	29,29	10	208,9	10
	50	**0,00**	**0,00**	44,39	10	1,00	0,64	15,67	2	0,04	88,40	9	**0,00**	25,30	10	252,2	10
	60	**0,00**	**0,00**	37,60	10	0,29	0,07	18,18	8	0,03	63,30	9	**0,00**	20,99	10	304,1	10
	70	0,03	**0,00**	84,98	10	0,83	0,46	23,79	3	0,12	84,50	6	**0,00**	17,40	10	350,0	10
	80	0,01	**0,00**	60,25	10	0,65	0,33	28,52	1	0,03	40,10	9	**0,00**	13,63	10	398,9	10
	90	**0,00**	**0,00**	43,55	10	0,25	0,07	27,07	7	**0,00**	15,40	10	**0,00**	10,63	10	452,4	10
1000	0	**0,00**	**0,00**	0,85	10	**0,00**	**0,00**	0,61	10	0,40	130,00	0	–	–	–	401,8	10
	10	**0,00**	**0,00**	0,89	10	**0,00**	**0,00**	0,64	10	0,10	142,90	6	**0,00**	100,23	10	401,8	10
	20	**0,00**	**0,00**	0,80	10	**0,00**	**0,00**	0,47	10	0,07	143,30	7	**0,00**	104,06	10	401,8	10
	30	**0,00**	**0,00**	0,94	10	**0,00**	**0,00**	0,76	10	0,17	165,40	4	**0,00**	126,25	10	401,8	10
	40	0,10	0,05	150,74	8	0,94	0,66	39,03	2	1,20	132,60	0	**0,00**	196,06	10	409,6	10
	50	0,02	**0,00**	105,30	10	1,16	0,80	67,05	1	0,10	49,10	7	**0,00**	172,41	10	504,7	10
	60	**0,00**	**0,00**	114,15	10	0,93	0,60	82,52	0	0,05	75,20	7	**0,00**	113,93	10	607,1	10
	70	**0,00**	**0,00**	118,36	10	0,77	0,46	109,15	2	0,03	63,50	8	**0,00**	94,06	10	707,3	10
	80	**0,00**	**0,00**	176,14	10	0,63	0,37	131,50	1	0,02	101,30	8	**0,00**	77,83	10	807,1	10
	90	**0,00**	**0,00**	143,29	10	0,35	0,20	154,54	2	0,01	74,80	9	**0,00**	65,52	10	905,7	10
Average		0,02	0,00	33,55	397,00	0,25	0,13	19,70	77,50	0,18	62,99	73,75	0,00	34,89	90,00	260,06	400

Tabela A.2: Resultados computacionais detalhados para as instâncias do grupo (u).

Inst.		HILS-*Completo*				ILS-*Simples*				PH			DH-LDS			BKS	
n	d	Avg Gap	Best Gap	T(s)	Opt	Avg Gap	Best Gap	T(s)	Opt	Gap	T(s)	Opt	Gap	T(s)	Opt	#Bins	Opt
60	10	**0,00**	**0,00**	1,82	10	0,50	0,50	1,06	9	–	–	–	**0,00**	0,30	10	20,9	10
	20	**0,00**	**0,00**	1,93	10	**0,00**	**0,00**	1,04	10	–	–	–	**0,00**	0,15	10	21,0	10
	30	**0,00**	**0,00**	2,17	10	**0,00**	**0,00**	1,04	10	–	–	–	**0,00**	0,15	10	21,0	10
	40	**0,00**	**0,00**	2,43	10	**0,00**	**0,00**	0,95	10	–	–	–	**0,00**	0,14	10	21,0	10
	50	**0,00**	**0,00**	2,33	10	**0,00**	**0,00**	0,75	10	–	–	–	**0,00**	0,13	10	21,0	10
	60	**0,00**	**0,00**	2,10	10	**0,00**	**0,00**	0,61	10	–	–	–	**0,00**	0,14	10	21,0	10
	70	**0,00**	**0,00**	2,05	10	0,29	**0,00**	0,63	10	–	–	–	**0,00**	0,15	10	21,0	10
	80	**0,00**	**0,00**	1,08	10	**0,00**	**0,00**	0,45	10	–	–	–	**0,00**	0,22	10	22,8	10
	90	**0,00**	**0,00**	0,73	10	**0,00**	**0,00**	0,44	10	–	–	–	**0,00**	0,10	10	27,1	10
120	10	**0,00**	**0,00**	3,25	10	2,45	2,00	4,05	2	–	–	–	**0,00**	1,45	10	40,0	10
	20	**0,00**	**0,00**	5,01	10	2,25	2,25	4,46	1	–	–	–	**0,00**	2,30	10	40,1	10
	30	0,43	**0,00**	13,53	10	2,00	2,00	4,82	2	–	–	–	**0,00**	1,82	10	40,2	10
	40	**0,00**	**0,00**	31,37	10	**0,00**	**0,00**	4,51	10	–	–	–	**0,00**	1,55	10	41,0	10
	50	**0,00**	**0,00**	44,67	10	**0,00**	**0,00**	3,56	10	–	–	–	**0,00**	0,92	10	41,0	10
	60	**0,00**	**0,00**	46,61	10	**0,00**	**0,00**	2,66	10	–	–	–	**0,00**	0,65	10	41,0	10
	70	**0,00**	**0,00**	37,34	10	0,12	**0,00**	2,46	10	–	–	–	**0,00**	0,72	10	41,0	10
	80	0,02	**0,00**	23,06	10	3,02	2,19	2,04	1	–	–	–	0,49	3,10	8	41,2	10
	90	0,02	**0,00**	5,25	10	0,81	0,64	2,14	7	–	–	–	**0,00**	0,60	10	47,0	10
249	10	**0,00**	**0,00**	0,63	10	**0,00**	**0,00**	0,37	10	–	–	–	**0,00**	16,78	10	83,0	10
	20	**0,00**	**0,00**	2,29	10	0,01	**0,00**	1,51	10	–	–	–	**0,00**	15,40	10	83,0	10
	30	0,89	**0,00**	50,48	10	1,04	0,12	19,97	9	–	–	–	0,36	31,53	7	83,0	10
	40	1,08	1,08	76,90	1	1,08	1,08	29,62	1	–	–	–	**0,72**	49,08	4	83,1	10
	50	**1,08**	**1,08**	93,10	1	1,08	1,08	22,07	1	–	–	–	**1,08**	54,57	1	83,1	10
	60	**0,00**	**0,00**	106,95	9	**0,00**	**0,00**	14,97	9	–	–	–	**0,00**	14,61	9	84,0	9
	70	**0,00**	**0,00**	104,54	10	**0,00**	**0,00**	10,88	10	–	–	–	**0,00**	4,68	10	84,0	10
	80	1,14	0,71	92,92	4	2,38	2,38	9,33	0	–	–	–	**0,00**	4,96	10	84,0	10
	90	0,20	**0,00**	148,73	9	2,84	2,30	7,19	0	–	–	–	0,46	21,44	6	87,0	9
501	10	**0,00**	**0,00**	0,40	10	**0,00**	**0,00**	0,20	10	–	–	–	**0,00**	186,45	10	167,0	10
	20	**0,00**	**0,00**	0,60	10	**0,00**	**0,00**	0,36	10	–	–	–	**0,00**	153,38	10	167,0	10
	30	**0,00**	**0,00**	1,78	10	**0,00**	**0,00**	1,09	10	–	–	–	0,06	255,60	9	167,0	10
	40	0,03	**0,00**	27,69	10	0,05	**0,00**	19,25	10	–	–	–	0,36	400,24	4	167,0	10
	50	**0,60**	**0,60**	159,30	0	**0,60**	**0,60**	75,14	0	–	–	–	**0,60**	427,73	0	167,0	10
	60	**0,24**	**0,24**	219,85	3	**0,24**	**0,24**	61,90	3	–	–	–	**0,24**	304,92	3	167,6	7
	70	**0,00**	**0,00**	285,97	2	**0,00**	**0,00**	51,50	2	–	–	–	**0,00**	339,60	2	168,0	2
	80	0,60	0,60	213,26	0	1,25	1,19	55,07	0	–	–	–	**0,00**	29,34	10	168,0	10
	90	2,14	1,71	245,82	0	2,97	2,71	31,30	0	–	–	–	**0,06**	344,74	0	169,9	1
Average		0,24	0,17	57,17	289,00	0,69	0,59	12,48	59,25	–	–	–	0,12	74,16	73,25	78,67	338

Tabela A.3: Resultados computacionais detalhados para as instâncias do grupo (ta).

Inst.		HILS-*Completo*				ILS-*Simples*				PH			DH-LDS			BKS	
n	d	Avg Gap	Best Gap	T(s)	Opt	Avg Gap	Best Gap	T(s)	Opt	Gap	T(s)	Opt	Gap	T(s)	Opt	#Bins	Opt
120	10	**0,00**	**0,00**	0,08	10	**0,00**	**0,00**	0,06	10	–	–	–	**0,00**	0,70	10	49,40	10
	20	**0,00**	**0,00**	0,00	10	**0,00**	**0,00**	0,00	10	–	–	–	**0,00**	0,68	10	49,10	10
	30	**0,00**	**0,00**	0,00	10	**0,00**	**0,00**	0,00	10	–	–	–	**0,00**	0,70	10	49,10	10
	40	**0,00**	**0,00**	1,78	10	0,21	0,21	0,89	9	–	–	–	**0,00**	0,73	10	48,70	10
	50	**0,00**	**0,00**	1,96	10	0,21	0,21	1,08	9	–	–	–	**0,00**	0,78	10	47,70	10
	60	**0,00**	**0,00**	4,09	10	0,43	0,43	1,73	8	–	–	–	**0,00**	0,76	10	47,70	10
	70	**0,00**	**0,00**	6,14	10	1,01	0,83	3,44	6	–	–	–	**0,00**	1,23	10	48,80	10
	80	**0,00**	**0,00**	9,56	10	1,60	1,00	2,86	5	–	–	–	**0,00**	0,60	10	50,20	10
	90	**0,00**	**0,00**	5,75	10	1,22	0,77	2,26	6	–	–	–	**0,00**	0,39	10	52,70	10
250	10	0,05	**0,00**	1,63	10	0,07	**0,00**	0,66	10	–	–	–	**0,00**	4,73	10	100,50	10
	20	0,03	**0,00**	1,49	10	0,04	**0,00**	0,74	10	–	–	–	**0,00**	5,10	10	100,40	10
	30	**0,00**	**0,00**	0,18	10	**0,00**	**0,00**	0,08	10	–	–	–	**0,00**	5,59	10	101,10	10
	40	**0,00**	**0,00**	0,15	10	**0,00**	**0,00**	0,07	10	–	–	–	**0,00**	4,26	10	99,00	10
	50	**0,00**	**0,00**	10,42	10	**0,00**	**0,00**	3,56	10	–	–	–	**0,00**	4,19	10	99,60	10
	60	0,02	**0,00**	14,79	10	0,20	0,20	5,81	8	–	–	–	**0,00**	6,01	10	100,80	10
	70	0,06	**0,00**	34,87	10	0,80	0,80	18,67	2	–	–	–	0,10	9,92	9	100,30	10
	80	**0,00**	**0,00**	130,92	9	1,37	1,10	19,01	0	–	–	–	**0,00**	7,83	9	100,40	9
	90	0,03	**0,00**	63,88	10	2,43	2,23	12,89	0	–	–	–	0,10	8,29	9	103,70	10
500	10	**0,00**	**0,00**	0,87	10	**0,00**	**0,00**	0,29	10	–	–	–	**0,00**	32,26	10	200,30	10
	20	**0,00**	**0,00**	0,83	10	**0,00**	**0,00**	0,64	10	–	–	–	**0,00**	34,28	10	201,00	10
	30	**0,00**	**0,00**	2,01	10	**0,00**	**0,00**	1,06	10	–	–	–	**0,00**	34,34	10	201,70	10
	40	0,02	**0,00**	7,90	10	0,02	**0,00**	2,97	10	–	–	–	**0,00**	33,49	10	200,30	10
	50	**0,00**	**0,00**	2,69	10	**0,00**	**0,00**	1,22	10	–	–	–	**0,00**	36,06	10	199,50	10
	60	0,06	0,05	30,81	9	0,09	0,05	23,02	9	–	–	–	**0,00**	31,30	10	200,60	10
	70	0,28	0,20	147,57	5	0,30	0,30	60,63	3	–	–	–	**0,00**	66,47	9	201,30	9
	80	0,34	0,20	492,58	4	0,90	0,85	77,61	0	–	–	–	**0,00**	128,61	7	199,60	7
	90	0,04	**0,00**	701,52	6	2,11	1,93	72,15	0	–	–	–	0,10	171,07	4	202,60	6
1000	10	**0,00**	**0,00**	7,28	10	0,01	**0,00**	3,54	10	–	–	–	**0,00**	261,14	10	400,10	10
	20	**0,00**	**0,00**	0,93	10	**0,00**	**0,00**	0,54	10	–	–	–	**0,00**	257,69	10	402,90	10
	30	**0,00**	**0,00**	0,65	10	**0,00**	**0,00**	0,46	10	–	–	–	**0,00**	235,71	10	399,70	10
	40	**0,00**	**0,00**	4,92	10	**0,00**	**0,00**	3,25	10	–	–	–	**0,00**	240,39	10	399,10	10
	50	**0,00**	**0,00**	1,22	10	**0,00**	**0,00**	1,03	10	–	–	–	**0,00**	225,69	10	400,90	10
	60	**0,00**	**0,00**	4,40	10	**0,00**	**0,00**	3,58	10	–	–	–	**0,00**	287,98	10	399,00	10
	70	0,05	0,05	114,80	8	0,05	0,05	76,60	8	–	–	–	**0,00**	349,79	10	399,50	10
	80	0,37	0,32	592,42	0	0,47	0,37	245,04	0	–	–	–	**0,00**	550,21	8	400,80	8
	90	0,95	0,18	2867,75	0	1,86	1,66	167,60	0	–	–	–	**0,10**	2242,11	1	398,30	4
Average		0,06	0,03	146,36	321,00	0,43	0,36	22,64	65,75	–	–	–	0,01	146,70	84,00	187,68	343

Tabela A.4: Resultados computacionais detalhados para as instâncias do grupo (ua).

Inst.		HILS-*Completo*				ILS-*Simples*				PH			DH-LDS			BKS	
n	d	Avg Gap	Best Gap	T(s)	Opt	Avg Gap	Best Gap	T(s)	Opt	Gap	T(s)	Opt	Gap	T(s)	Opt	#Bins	Opt
	10	**0,00**	**0,00**	0,00	10	**0,00**	**0,00**	0,00	10	–	–	–	–	–	–	18,4	10
	20	**0,00**	**0,00**	1,12	10	**0,00**	**0,00**	0,84	10	–	–	–	–	–	–	24,8	10
	30	**0,00**	**0,00**	1,65	10	**0,00**	**0,00**	1,05	10	–	–	–	–	–	–	37,3	10
	40	**0,00**	**0,00**	1,87	10	0,23	**0,00**	1,12	10	–	–	–	–	–	–	46,2	10
120	50	**0,00**	**0,00**	2,58	10	**0,00**	**0,00**	1,09	10	–	–	–	–	–	–	62,7	10
	60	**0,00**	**0,00**	3,28	10	**0,00**	**0,00**	1,13	10	–	–	–	–	–	–	73,1	10
	70	**0,00**	**0,00**	4,28	10	0,09	**0,00**	1,24	10	–	–	–	–	–	–	84,3	10
	80	**0,00**	**0,00**	6,01	10	**0,00**	**0,00**	1,27	10	–	–	–	–	–	–	99,4	10
	90	**0,00**	**0,00**	7,73	10	0,01	**0,00**	1,31	10	–	–	–	–	–	–	110,2	10
	10	**0,00**	**0,00**	0,00	10	**0,00**	**0,00**	0,00	10	–	–	–	–	–	–	38,0	10
	20	**0,00**	**0,00**	3,39	10	0,07	**0,00**	2,10	10	–	–	–	–	–	–	51,4	10
	30	**0,00**	**0,00**	5,08	10	0,10	**0,00**	2,37	10	–	–	–	–	–	–	78,4	10
	40	**0,00**	**0,00**	7,11	10	0,19	0,10	2,54	9	–	–	–	–	–	–	97,3	10
250	50	**0,00**	**0,00**	11,92	10	0,02	**0,00**	2,71	10	–	–	–	–	–	–	128,6	10
	60	**0,00**	**0,00**	17,40	10	0,22	0,14	3,05	8	–	–	–	–	–	–	150,2	10
	70	0,01	**0,00**	19,04	10	0,07	**0,00**	3,58	10	–	–	–	–	–	–	179,6	10
	80	**0,00**	**0,00**	19,65	10	0,04	**0,00**	4,08	10	–	–	–	–	–	–	202,5	10
	90	**0,00**	**0,00**	20,51	10	0,04	**0,00**	4,52	10	–	–	–	–	–	–	225,4	10
	10	**0,00**	**0,00**	0,01	10	**0,00**	**0,00**	0,01	10	–	–	–	–	–	–	75,3	10
	20	**0,00**	**0,00**	11,16	10	0,10	**0,00**	4,63	10	–	–	–	–	–	–	99,8	10
	30	0,03	**0,00**	21,19	10	0,28	0,13	5,95	8	–	–	–	–	–	–	147,3	10
	40	**0,00**	**0,00**	23,12	10	0,08	**0,00**	7,05	10	–	–	–	–	–	–	199,4	10
500	50	**0,00**	**0,00**	26,28	10	0,04	**0,00**	8,34	10	–	–	–	–	–	–	251,6	10
	60	**0,00**	**0,00**	27,74	10	0,09	**0,00**	10,26	10	–	–	–	–	–	–	298,9	10
	70	**0,00**	**0,00**	30,38	10	0,17	0,03	12,60	9	–	–	–	–	–	–	349,3	10
	80	**0,00**	**0,00**	33,07	10	0,06	**0,00**	14,14	10	–	–	–	–	–	–	403,3	10
	90	**0,00**	**0,00**	37,22	10	0,08	**0,00**	17,66	10	–	–	–	–	–	–	450,2	10
Average		0,00	0,00	12,70	270,00	0,07	0,01	4,25	88,00	–	–	–	–	–	–	147,51	270

Tabela A.5: Resultados computacionais detalhados para as instâncias do grupo (d).

Inst.		HILS-*Completo*				ILS-*Simples*				PH			DH-LDS			BKS	
n	d	Avg Gap	Best Gap	T(s)	Opt	Avg Gap	Best Gap	T(s)	Opt	Gap	T(s)	Opt	Gap	T(s)	Opt	#Bins	Opt
	10	**0,00**	**0,00**	0,00	10	**0,00**	**0,00**	0,00	10	–	–	–	**0,00**	1,28	10	18,60	10
	20	**0,00**	**0,00**	0,00	10	**0,00**	**0,00**	0,00	10	–	–	–	**0,00**	1,03	10	18,30	10
	30	1,14	1,14	9,46	8	1,14	1,14	0,41	8	–	–	–	**0,00**	1,43	10	18,70	10
	40	0,94	0,56	9,82	9	1,11	1,11	0,35	8	–	–	–	**0,00**	1,76	10	18,70	10
120	50	2,96	1,08	35,81	5	3,74	1,61	1,60	4	–	–	–	**0,00**	8,27	6	18,70	6
	60	1,67	**0,00**	23,64	0	1,86	**0,00**	2,19	0	–	–	–	**0,00**	16,71	0	21,00	0
	70	0,17	**-1,15**	23,81	3	0,37	-0,38	2,18	3	–	–	–	0,00	12,79	3	25,30	3
	80	0,32	**0,00**	11,05	8	0,64	**0,00**	2,11	8	–	–	–	0,32	6,89	7	31,40	8
	90	**0,00**	**0,00**	2,63	10	**0,00**	**0,00**	1,75	10	–	–	–	0,72	4,02	7	42,00	10
	10	**0,00**	**0,00**	0,00	10	**0,00**	**0,00**	0,00	10	–	–	–	**0,00**	14,28	10	37,60	10
	20	**0,00**	**0,00**	0,01	10	**0,00**	**0,00**	0,01	10	–	–	–	0,26	29,66	9	37,80	10
	30	**0,00**	**0,00**	0,01	10	**0,00**	**0,00**	0,01	10	–	–	–	**0,00**	14,90	10	38,10	10
	40	0,45	**0,00**	11,17	10	0,69	0,26	2,16	9	–	–	–	**0,00**	11,58	10	37,80	10
250	50	1,77	1,05	53,99	5	2,18	1,57	5,89	3	–	–	–	**0,00**	23,44	9	38,50	9
	60	2,84	2,30	67,80	0	3,35	2,55	8,42	0	–	–	–	**0,00**	163,30	1	39,20	1
	70	3,54	1,82	38,26	0	3,67	1,82	9,54	0	–	–	–	**0,00**	138,03	0	44,20	0
	80	1,63	0,19	33,04	0	1,76	0,74	9,27	0	–	–	–	**0,00**	89,31	0	54,60	0
	90	0,03	**-0,67**	25,24	0	0,27	-0,40	8,67	0	–	–	–	0,13	51,89	0	74,10	0
	10	**0,00**	**0,00**	0,01	10	**0,00**	**0,00**	0,01	10	–	–	–	0,13	261,90	9	75,50	10
	20	**0,00**	**0,00**	0,01	10	**0,00**	**0,00**	0,01	10	–	–	–	**0,00**	222,82	10	75,30	10
	30	**0,00**	**0,00**	0,38	10	**0,00**	**0,00**	0,25	10	–	–	–	**0,00**	93,81	10	75,70	10
	40	0,26	0,26	34,62	7	0,26	0,26	9,77	7	–	–	–	**0,00**	226,09	8	75,60	8
500	50	1,17	1,05	111,35	1	1,28	1,19	22,65	0	–	–	–	**0,00**	211,30	9	76,10	9
	60	2,40	1,84	132,46	0	2,74	2,23	25,80	0	–	–	–	**0,13**	973,22	1	76,30	2
	70	5,66	4,64	80,54	0	5,78	4,39	34,03	0	–	–	–	**0,13**	1479,39	0	79,80	0
	80	5,33	3,90	61,37	0	5,14	4,11	36,66	0	–	–	–	**0,32**	948,53	0	94,90	0
	90	1,63	0,70	59,02	0	1,45	0,55	33,91	0	–	–	–	**0,16**	456,23	0	127,90	0
Average		1,26	0,69	30,57	146,00	1,39	0,84	8,06	46,67	–	–	–	0,09	202,36	53,00	50,80	166

Tabela A.6: Resultados computacionais detalhados para as instâncias do grupo (da).