UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE CIÊNCIA DA COMPUTAÇÃO

MARCO ANTONIO WISNIEWSKI

Uma Busca Tabu para o Problema de Roteamento de Veículos Capacitados com Restrições de Empacotamento Tridimensionais

Trabalho de Graduação

Prof. Dr. Marcus Ritt
Orientador

Prof[a]. Dr[a]. Luciana S. Buriol
Co-orientadora

Porto Alegre, julho de 2011

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

RESUMO

Este trabalho estuda o problema de roteamento de veículos capacitados com restrições de empacotamento tridimensionais. O 3L-CVRP consiste em encontrar um roteamento de distância mínima para uma frota de veículos através de uma rede de rodovias com o objetivo de entregar produtos a clientes e ao mesmo tempo prover um empacotamento para esses produtos nos veículos respeitando restrições tridimensionais. O problema é altamente relevante para aplicações reais de logística, e sua dificuldade exige uma solução eficiente para que os resultados sejam significantes também em cenários reais.

É feita uma completa revisão bibliográfica das soluções existentes. Propomos um novo algoritmo de empacotamento randomizado baseado na heurística inferior-esquerda e uma nova representação de empacotamento com múltiplas vistas utilizando matrizes dinâmicas. Ao mesmo tempo propõe-se uma solução utilizando a meta-heurística busca tabu com avaliação *first-improvement* da vizinhança para o roteamento. Essa nova estratégia gera resultados comparáveis ou melhores que o estado da arte.

Palavras-chave: 3L-CVRP, roteamento de veículos, empacotamento, busca tabu.

A Tabu Search for the Capacitated Vehicle Routing Problem with Three-Dimensional Loading Constraints

ABSTRACT

This work considers the three-dimensional loading capacitated vehicle routing problem, which consists in finding a shortest routing of a fleet of vehicles through a network so as to deliver goods to customers and, at the same time, providing a packing plan for those goods and vehicles satisfying three-dimensional loading constraints. The studied problem is highly relevant to logistics, and its high complexity demands an efficient solution for the results to be significant in real word scenarios.

A literature review of the current state of the art is presented. A new loading approach based on generating packing permutations is proposed together with a new packing representation based on multiple views dynamic matrices. The routing is done by means of a tabu search algorithm with first improvement evaluation of the neighborhood. These strategies are able to efficiently produce solutions that are comparable to or better than the literature.

Keywords: 3L-CVRP, vehicle routing, container loading, tabu search.

1 INTRODUÇÃO

Os problemas de roteamento e de empacotamento aparecem naturalmente em logística de transportes, e o seu estudo tem aplicações práticas diretas. Quando tentamos otimizar tanto o roteamento de veículos quanto o carregamento dos mesmos com caixas encontramos um problema extremamente difícil de ser tratado. Ambos subproblemas são vastamente estudados em otimização combinatória, mas ainda difíceis de resolver já que são problemas NP-difíceis. Apenas recentemente o problema conjunto começou a ser tratado, devido à dificuldade e ao alto poder de processamento necessários para resolvê-lo. Além disso, restrições vindas diretamente de necessidades logísticas reais tornam o problema ainda mais complicado.

Os problemas de roteamento e carregamento são geralmente estudados em separado. Apesar de os resultados individuais obtidos serem muito bons, em aplicações reais em que precisamos das duas componentes ao mesmo tempo essa estratégia não funciona tão bem, já que não considera as relações que aparecem apenas no problema combinado. Considerar o problema geral nos permite obter uma melhor solução quanto à otimização logística de interesse. A pressão de mercado faz com que as empresas busquem otimizar fortemente as suas cadeias de suprimento, e qualquer ganho se torna decisivo quando consideramos o imenso volume tratado diariamente por empresas ligadas ao transporte.

A literatura que trata esse problema e problemas relacionados é ainda recente e limitada, mas o seu estudo é facilmente justificado devido à grande aplicação prática e à relação direta entre as restrições que tratamos com o que se encontra diariamente na logística, além do interesse acadêmico em explorar os problemas em conjunto.

1.1 Roteamento de Veículos

O problema exemplar em roteamento é o problema do Caixeiro Viajante, que pode ser modelado por um grafo G = (V, A), onde $V = \{v_0, v_1, \ldots, v_n\}$ é o conjunto que representa cidades e cada aresta $a_i \in A$ possui um custo de transporte c_{ij} associado entre as cidades v_i e v_j. O problema consiste em encontrar o circuito de menor custo total que passe por todas as cidades uma única vez.

Uma extensão que surge de situações reais é o roteamento de veículos capacitados (*Capacitated Vehicle Routing Problem - CVRP*). Determinamos o vértice v_0 como o centro de triagem. Lá, temos uma frota homogênea de b veículos todos com a mesma capacidade D. Os vértices $v_1, v_2, \ldots, v_n$ representam n clientes, cada um com uma demanda d_i ($1 \leq i \leq n$). O problema de otimização é encontrar um conjunto de no máximo b circuitos, cada um partindo do centro de triagem e visitando um subconjunto de clientes com as seguintes restrições:

(a) cada cliente é visitado por exatamente um veículo

(b) para cada veículo é designado no máximo um circuito (chamado de rota)

(c) a soma das demandas dos clientes pertencentes a uma rota não excede D

(d) a soma dos custos de transporte de todos os circuitos é minimizado

Os problemas de roteamento são de grande relevância para sistemas de distribuição e transporte, onde o custo relativo à operação dos veículos forma uma grande parte do custo total de operação das empresas. Além disso, o grande volume lidado implica em uma grande economia com qualquer ganho encontrado na solução usada (IORI; MARTELLO, 2010).

1.1.1 Restrições de empacotamento

Uma extensão lógica ao CVRP é modelar de forma mais fiel a demanda de cada cliente. No CVRP cada cliente possui uma demanda representada por um único valor. Isso não é representativo em aplicações reais, onde os veículos devem ser preenchidos com caixas de diferentes dimensões e pesos, que devem respeitar diversas restrições de posicionamento a fim de garantir a estabilidade e possibilidade de descarregamento.

Esses problemas já foram individualmente amplamente estudados, e são variações do problema *Bin Packing*: dada uma capacidade de armazenamento V e uma lista de tamanhos de itens $x_1, \ldots, x_n$, encontre o menor N tal que exista uma N-partição $S_1 \cup \ldots \cup S_N$ de $\{1, \ldots, n\}$ com $\sum_{i \in S_k} x_i \leq V$ para todo $k = 1, \ldots, N$. O problema pode ser estendido para n-dimensões. O caso de três dimensões é o que nos interessa: similarmente, dada uma descrição de um contêiner e de diversas caixas, o objetivo é posicionar as caixas dentro de N contêineres tal que N seja mínimo. Uma variação deste problema é usada neste trabalho, o *Three-Dimensional Strip Packing Problem* (3SPP), onde dada uma largura e altura de um contêiner de comprimento infinito, deve-se empacotar as caixas de forma a minimizar o comprimento utilizado. Ambos problemas são NP-difíceis.

1.2 Estrutura do Trabalho

Este trabalho trata o problema de roteamento de veículos capacitados com restrições de empacotamento tridimensionais (3L-CVRP). Com base no trabalho prévio de Portal et al. (2009) onde foi desenvolvida uma meta-heurística de busca tabu, apresenta-se uma nova abordagem para tratá-lo considerando restrições de orientação, fragilidade, superfície de apoio e ordem de descarregamento conforme os clientes de uma rota. O desempenho do procedimento responsável pelo empacotamento se mostrou crucial, e com base nisso propomos um algoritmo randomizado baseado na heurística inferior-esquerda ajustada para o espaço tridimensional, com o auxílio de uma representação com matriz dinâmica de múltiplas vistas. O roteamento é realizado através de uma busca tabu com uma busca local *first improvement*, o que se mostrou eficiente.

Apresentamos quatro contribuições. Primeiro, é feita uma revisão bibliográfica das soluções propostas encontradas na literatura. Segundo, propõe-se um novo algoritmo para o subproblema de empacotamento de um contêiner (*strip packing*). Terceiro, uma nova representação de um empacotamento (i.e. dos espaços e das distribuições das caixas) de um contêiner é definida. Por fim é proposta uma busca tabu com uma política de movimentos específica para as características do problema em questão.

O trabalho está estruturado da seguinte maneira. O Capítulo 2 consiste de uma revisão bibliográfica do estado da arte do problema estudado, bem como de um rápido comentário a respeito dos subproblemas de empacotamento e de roteamento de veículos. O Capítulo 3 descreve detalhadamente o empacotamento que deve ser resolvido, bem como as soluções e variações propostas com base em características da busca tabu. O Capítulo 4 analisa o subproblema do roteamento dos veículos, geração da solução inicial bem como o algoritmo busca tabu proposto. As instâncias, os métodos de testes, bem como os resultados seguidos de uma análise comparativa são apresentados no Capítulo 5. Finalmente, no Capítulo 6, as conclusões são descritas juntamente com algumas estratégias para a continuação do estudo.

2 TRABALHOS RELACIONADOS

O problema aparece primeiro em Gendreau et al. (2006), motivado por restrições encontradas em transporte de cargas. Também há interesse teórico já que ele generaliza os problemas CVRP e o Bin Packing em três dimensões (3BPP). A situação real modelada é a de um distribuidor que deve definir rotas para a sua frota de veículos a fim de entregar produtos para um número de clientes, sendo cada produto uma caixa retangular com dado tamanho e peso. O objetivo é minimizar o custo total do transporte, definido como distância total percorrida pelos veículos. Todos os pedidos de um cliente devem estar no mesmo veículo. A frota considerada é homogênea: consiste de veículos com o mesmo espaço tridimensional para carregamento e uma capacidade máxima em relação ao peso de suas cargas. A solução inclui um carregamento factível para cada veículo, com as caixas totalmente carregadas sem nenhuma sobreposição (GENDREAU et al., 2006).

Advindas de aplicações reais outras restrições são consideradas. Ao se empilhar caixas, deve ser garantida uma superfície de apoio mínima para assegurar a estabilidade dessas. Além disso, algumas caixas podem ser frágeis, impossibilitando que outras sejam colocadas acima. Por fim, o descarregamento do veículo em cada ponto de sua rota deve ser facilitado: ao visitar um cliente, todas as caixas destinadas a ele devem ser retiradas sem ter de mover produtos com outros destinos.

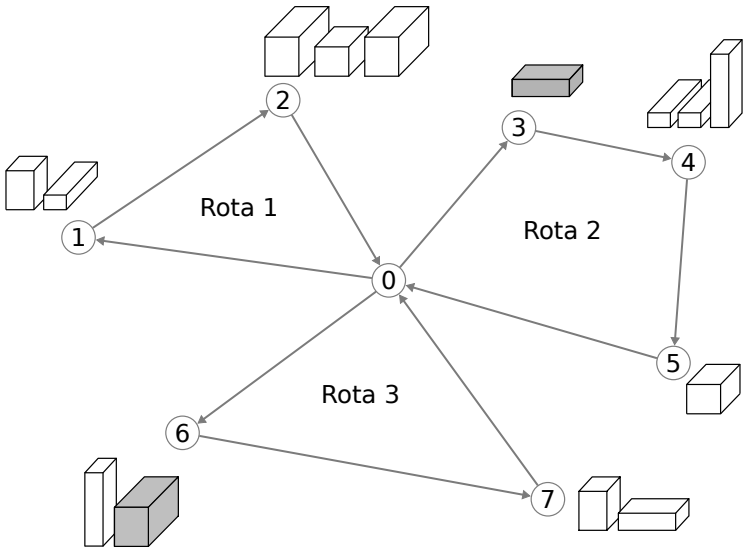


Figura 2.1: Uma instância simples do 3L-CVRP

A Figura 2.1 representa uma instância simples do problema bem como rotas de uma solução possível. A instância é formada por sete clientes mais um depósito (número zero) e uma demanda total de quatorze itens. Uma solução, além das rotas, deve prover um carregamento respeitando restrições para cada rota. As caixas em cinza são frágeis, e não podem servir como apoio para caixas não-frágeis, como exemplo de uma restrição.

Não só o 3L-CVRP é um problema NP-difícil (GENDREAU et al., 2006), como encontrar soluções factíveis para ele é também. Isso pode ser concluído uma vez que o problema de decidir se um conjunto de caixas tridimensionais podem ser carregadas em um contêiner, mesmo sem restrições, já é NP-difícil (MARTELLO; PISINGER; VIGO, 2000).

2.1 Definição do Problema

Seja $G = (V, A)$ um grafo não direcionado, onde $V = \{v_0, v_1, \ldots, v_n\}$ é um conjunto de $n + 1$ vértices que correspondem ao depósito (vértice v_0) e a n clientes (vértices $v_1, \ldots, v_n$) e A é o conjunto completo de arestas conectando todos os pares de vértices. Cada aresta (i, j) possui um custo de transporte $c_{i,j}(1 \leq i, j \leq n)$ associado. Seja v o número de veículos idênticos disponíveis. Cada veículo possui uma capacidade P de peso máxima. L, H e C representam respectivamente a largura, altura e comprimento de cada veículo. Cada cliente $i(i = 1, .., n)$ tem uma demanda de k_i itens tridimensionais $M_{i,k}(k = 1, \ldots, k_i)$ cada um com largura $l_{i,k}$, altura $h_{i,k}$ e comprimento $c_{i,k}$. Todos os itens tem sua orientação fixa em relação a altura, mas podem ser rotacionados em 90 ° no plano $x - z$ (ver Figura 2.2 para definição dos eixos). Finalmente, cada item possui um indicador de fragilidade $f_{i,k}$, que é igual a 1 se o item é frágil e 0 caso contrário. Itens que não são frágeis não podem ser empilhados sobre itens frágeis, mas não há restrição quanto ao empilhamento de itens frágeis.

Uma solução é então um conjunto de no máximo v rotas, cada uma partindo e terminando no depósito (v_0), onde

(i) cada cliente é visitado uma única vez; (ii) nenhum veículo carrega um peso que excede sua capacidade; (iii) para cada veículo existe um carregamento factível dos itens demandados pelos clientes da sua rota; e (iv) o tamanho total do caminho percorrido em todas as rotas é minimizado.

Um carregamento é factível se todas as restrições abaixo são respeitadas:

- Não ha intersecção entre os itens, e todos estão completamente contidos nos veículos (restrições básicas do 3BPP).

- Os itens são carregados respeitando as regras de ortogonalidade e de fragilidade.

- Todos os itens que usam como apoio outros itens possuem uma determinada superfície de apoio que nem sempre é total. Para garantir a estabilidade das caixas, dado um parâmetro a que chamamos de *superfície de apoio mínima*, itens colocados embaixo de um item qualquer M_{ik}, com as suas faces superiores em contato direto com a face inferior de M_{ik}, devem formar uma área de contato total $A \geq ac_{ik}l_{ik}$, com $0 \leq a \leq 1$.

- Em cada ponto pertencente a rota de um veículo, deve ser possível descarregar todas as caixas relativas ao cliente corrente através de uma série de movimentos paralelos ao eixo C. Isso implica que não pode haver nenhum item pertencente a outro cliente

na frente ou em cima de uma caixa designada ao cliente corrente. Essa política é denotada por *carregamento sequencial* (IORI; SALAZAR-GONZALEZ; VIGO, 2007).

A Figura 2.2 demonstra um carregamento factível, bem como o sistema de coordenadas utilizado ao se referenciar um contêiner.

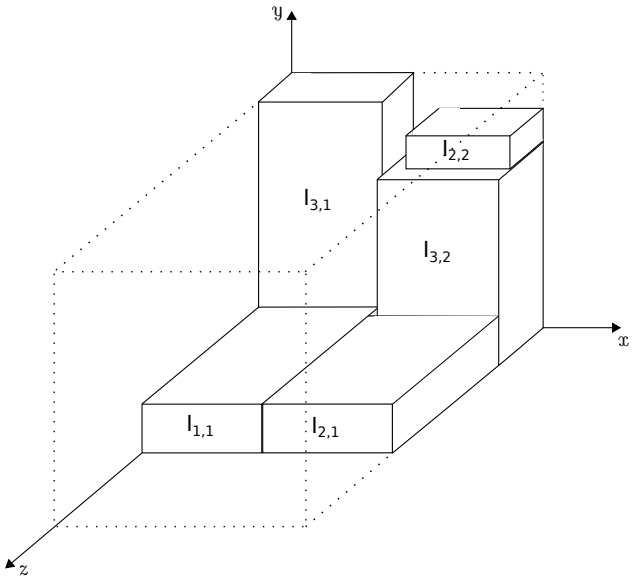


Figura 2.2: Carregamento factível

2.2 A Meta-heurística Busca Tabu

Os problemas de roteamento e empacotamento, em seus diversos formatos, tendem a ser problemas de alta complexidade e de difícil resolução. No problema estudado, ambos o roteamento e o empacotamento são problemas fortemente *NP*-difíceis (TOTH; VIGO, 2002; MARTELLO; PISINGER; VIGO, 2000), e consequentemente o 3L-CVRP como um todo é também. Na prática, isso implica na dificuldade em obter soluções exatas para esses tipos de problemas que requeiram tempo e poder de processamento praticáveis. O que se pode fazer é criar métodos aproximativos que tentam encontrar soluções de boa qualidade em um tempo razoável. Os métodos de resolução usados exploram o espaço de todas as soluções possíveis do problema, buscando soluções de boa qualidade. No final, a melhor solução encontrada é retornada. As diferentes maneiras de explorar o espaço de soluções constituem as diferentes meta-heurísticas propostas (TOTH; VIGO, 2002; LODI; MARTELLO; VIGO, 1999).

Uma meta-heurística que aparece com grande frequência nos problemas de roteamento e é usada também neste trabalho é a busca tabu (GLOVER, 1989). A ideia geral consiste em iterativamente avaliar a vizinhança da solução atual, sendo que a solução inicial deve ser construída previamente por algum outro método, e escolher a melhor solução vizinha para ser a nova atual. Para evitar o retorno à soluções prévias, e permitir assim o método a não ficar preso em mínimos locais, cria-se uma *lista tabu*, que é usada para proibir certos movimentos enquanto estamos explorando o espaço de busca.

O Algoritmo 2.1 descreve uma busca tabu, onde queremos encontrar uma solução s que minimize a função-objetivo f. O procedimento básico executa até que a condição de

parada seja atingida, e nesse momento retorna a melhor solução encontrada previamente. Os critérios de parada são normalmente um número de iterações, tempo ou iterações que não produzam melhora. A cada iteração, analisa-se a vizinhança da solução atual de acordo com alguma definição de vizinhança dependente do problema. Seleciona-se então o melhor vizinho que não seja tabu, ou seja, que não está presente na lista tabu e que passará então a ser a nova solução atual.

Algoritmo 2.1 Meta-heurística Busca Tabu

Entrada: Uma solução s_0
Saída: Uma solução s', tal que $f(s') \leq f(s_0)$
1: $s \leftarrow s_0$
2: $s' \leftarrow s_0$
3: $T \leftarrow \emptyset$
4: **repita**
5: $\quad \tilde{V} \leftarrow V(s) \setminus T$
6: $\quad s \leftarrow$seleciona $s \in \tilde{V}$ com $f(s)$ mínimo
7: $\quad T \leftarrow T \cup movimento\_aplicado$
8: $\quad$ **se** $f(s) < f(s')$ **então**
9: $\quad\quad s' \leftarrow s$
10: $\quad$ **fim se**
11: $\quad \mathrm{atualiza}(T)$ {envelhecimento}
12: **até** condição de parada
13: **retorna** s'

A lista tabu implementa o conceito de memória de curta duração, geralmente representada por uma lista de movimentos ditos tabu. O objetivo é impedir a busca de ficar presa a mínimos locais, bem como direcioná-la para soluções de maior qualidade. Os movimentos são acrescentados à lista segundo regras de ativação, em geral sempre que são aplicados, e permanecem tabu por um período de iterações determinado *tabu tenure*. Tanto o tamanho da lista quanto a duração de permanência dos seus elementos constituem importantes parâmetros.

Existem diversas variações deste algoritmo básico e vários procedimentos e estruturas para auxiliar a busca, como os conceitos de diversificação e intensificação. Para uma exposição e análise destes aspectos, bem como uma explicação detalhada sobre a metaheurística ver Glover e Laguna (1998).

2.3 Resultados Prévios

Gendreau et al. (2006) introduziu o 3L-CVRP estendendo um trabalho prévio que tratava o problema similar considerando o empacotamento em duas dimensões. A estratégia utilizada foi a de duas buscas tabu: a primeira responsável pelo roteamento (que é uma das maneiras de resolver o CVRP), e outra que trata apenas do carregamento. A cada iteração, um cliente é movido entre rotas e os carregamentos então são calculados gerando uma vizinhança que consiste de sequências de carregamentos a serem seguidas por duas heurísticas construtivas. Uma penalização é acrescida ao valor retornado do empacotamento se não foi possível, depois de uma série de tentativas, carregar os itens utilizando um comprimento menor ou igual ao comprimento do veículo.

De Araújo (2006), aproveitando o seu algoritmo proposto para problemas de carrega-

mento de contêineres, foi o primeiro a melhorar os resultados do problema. A abordagem é semelhante a de Gendreau et al. (2006), mas difere quanto à vizinhança considerada na busca tabu, que é gerada a partir de movimentos de inserção e troca, e quanto ao algoritmo de empacotamento, baseado no seu carregamento de contêineres com cubóides com múltiplos inícios. A melhora média em relação a distância total percorrida nas instâncias reportada é de 4.52%.

O método adotado por Tarantilis et al. (2009) difere dos anteriores em dois pontos. Primeiro, a meta-heurística utilizada para o roteamento é a *Guided Tabu Search* e segundo, no lugar de uma meta-heurística ou de um algoritmo sofisticado para o empacotamento, é usado uma série de heurísticas construtivas mais simples que provêm uma boa qualidade de empacotamento a um custo computacional reduzido. A melhora publicada em relação às soluções de Gendreau et al. (2006) é de 3.54%.

Fuellerer et al. (2010) publicou um método diferente de abordar o problema. Para o roteamento, foi utilizada a meta-heurística de otimização de colônia de formigas que resumidamente funciona através da melhora iterativa do processo de geração de uma solução completa através da identificação de características que levam às melhores soluções até então encontradas. Assume-se portanto que mantendo essas características as soluções tendem a melhorar. O método parte também da solução inicial produzida pelo algoritmo de Clarke e Wright (1964). O carregamento é feito novamente com heurísticas construtivas simples e uma exploração parcial de sua vizinhança. Uma melhora significativa de 6.43% foi alcançada em relação a Gendreau et al. (2006).

Tabela 2.1: Comparativo de métodos da literatura

Método	Roteamento	Empacotamento	Melhora
Gendreau et al. (2006)	busca tabu	busca tabu	0.0%
de Araújo (2006)	busca tabu	método próprio (cubóides)	-4.52%
Tarantilis et al. (2009)	GTS (busca tabu)	heurísticas rápidas	-3.54%
Fuellerer et al. (2010)	ACO	heurísticas rápidas	-6.43%
Tao e Wang (2010)	busca tabu	heurística *least waste*	-8.56%
Wang et al. (2010)	busca tabu	heurísticas rápidas	-8.51%
Bortfeldt (2010)	busca tabu	busca em árvore	-8.00%

Com relação à qualidade das soluções encontradas, o melhor método atualmente é o de Tao e Wang (2010). O roteamento é feito por uma busca tabu, na qual a vizinhança é definida por três tipos de movimentos: inserção de clientes, troca de clientes e troca de pedaços de rotas inteiras. Um vizinho é escolhido aleatoriamente com uma probabilidade fixa para cada tipo de movimento. Novamente o carregamento é feito por rápidas heurísticas construtivas (com consideração também do espaço desperdiçado). O método de Wang et al. (2010) obtém resultados bem próximos com uma abordagem semelhante, mas uma vizinhança diferente, gerada a partir de movimentos 2-*opt*, troca de clientes intra-rota, inserção de cliente, *crossover* e divisão de uma rota em duas. A melhoria média destes dois métodos em relação a Fuellerer et al. (2010) é respectivamente de 1.4% e 1.31%.

O tempo computacional necessário para o método de Bortfeldt (2010) é bastante reduzido em relação aos métodos precedentes. O algoritmo ainda é uma busca tabu, mas uma importante mudança foi feita: ao avaliar uma vizinhança, os vizinhos são ordenados em relação a sua qualidade e o procedimento de empacotamento é então seletivamente chamado para as rotas mais promissoras. Além disso, o método de carregamento difere

dos outros por se tratar de uma busca em árvore, com características especialmente feitas para tratar a restrição de carregamento sequencial. Em relação à qualidade das soluções, uma melhora em relação a Fuellerer et al. (2006) de 0.8% foi obtida. O tempo total usado para encontrar as soluções foi de 5.2% e 12.2% dos tempos respectivos tomados pelos outros métodos.

Uma observação importante é que a maneira de calcular o *gap* usada em cada publicação não é igual, para uma análise consistente ver Capítulo 5. O capítulo apresenta também detalhes dos resultados obtidos na literatura.

Vale notar que de uma certa forma todos os métodos publicados utilizam heurísticas construtivas *Bottom-Left* e *Maximum Touching Area*, com a exceção de de Araújo (2006), que baseia-se na construção de cubóides. Todos também resolvem o problema em dois passos: uma meta-heurística para o roteamento que iterativamente chama uma sub-rotina responsável por determinar a factibilidade dos empacotamentos para as rotas geradas. Outro ponto em comum é o uso do método de economias de Clarke e Wright (1964) para a geração da solução inicial.

As pesquisas de Iori e Martello (2010) e Wang et al. (2009) detalham os resultados obtidos por algumas dessas abordagens, bem como descrevem esse e diversos outros problemas relacionados de roteamento e carregamento de veículos. Em relação a soluções exatas, Iori et al. (2007) e Azevedo et al. (2009) apresentaram algoritmos *branch-and-cut* para o 2L-CVRP, mas até agora nenhum algoritmo exato para o 3L-CVRP de que temos conhecimento foi publicado.

3 CARREGAMENTO DE UM CONTÊINER

A busca tabu responsável por resolver o roteamento chama diversas vezes, a cada iteração, uma sub-rotina responsável por resolver o subproblema do empacotamento. Especificamente, dada uma lista ordenada de clientes $\{i_1, \ldots, i_n\}$ que serão visitados em uma determinada rota e a correspondente lista de itens de cada cliente $\{I_{j,k} : j = 1, \ldots, n; k = 1, \ldots, k_{i_j}\}$, o procedimento deve criar um empacotamento de todos os itens em um veículo de largura L, altura H e comprimento infinito e retornar o comprimento total utilizado, que deve ser minimizado. Caso esse comprimento ainda seja maior do que C, a busca tabu é responsável por aplicar uma penalidade à solução, ou descartá-la completamente, conforme descrito no Capítulo 4. O carregamento deve sempre respeitar todas as restrições da seção 2.1. O problema é fortemente *NP*-difícil, já que o problema mais simples, sem as restrições de fragilidade, superfície de apoio e carregamento sequencial, já o é (*bin packing* tridimensional, ver por exemplo Lodi (2002)).

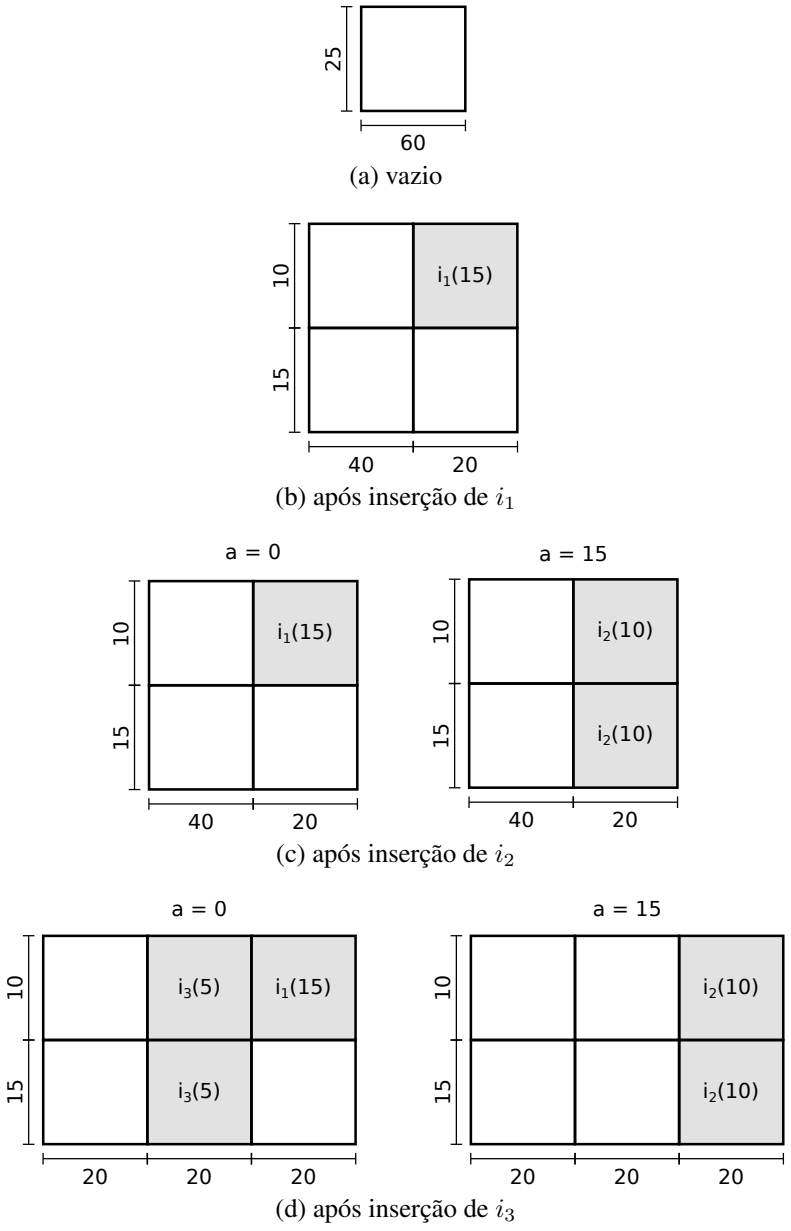
3.1 Representação de um Empacotamento

O fato de que o procedimento de empacotamento é chamado um número elevado de vezes ao longo de uma execução completa do método torna a estrutura de dados usada para representação de um contêiner crítica do ponto de vista da eficiência. Além disso, o objetivo é usar uma representação tão poderosa quanto possível para que seja possível identificar empacotamentos factíveis sempre que o algoritmo gere uma configuração válida, caso contrário corremos o risco de descartar empacotamentos que deveriam ser aceitos.

A estrutura proposta é uma extensão de uma matriz dinâmica (NGOI; TAY; CHUA, 1994) para múltiplas vistas. Uma matriz dinâmica representa a princípio um veículo em duas dimensões dividindo-o em um número mínimo de *células*. Cada caixa tem então sua posição completamente definida em termos das células que ela ocupa dentro do veículo. Inicialmente, o contêiner possui uma única célula representando todo o espaço interno. Quando mais itens são empacotados, as células subsequentes são divididas conforme o necessário.

Um problema ao utilizar a matriz dinâmica para o 3L-CVRP é que ela não possui informação total referente à terceira dimensão (altura). Isso implica no desperdício de espaço já que sempre temos apenas uma vista de cima que não é capaz de representar vários níveis diferentes de caixas. Consequentemente, é comum não conseguir representar carregamentos factíveis pois o algoritmo não é capaz de usar eficientemente os espaços livres que ficam escondidos embaixo de caixas já posicionadas. Esses espaços surgem sempre que uma caixa é carregada em uma posição com área de apoio inferior a cem porcento. Este problema, porém, pode ser eficientemente resolvido com o uso de múltiplas

vistas, ou seja, uma vista é criada na altura da base de cada caixa inserida no contêiner quando a mesma já não exista. Isso cria um custo computacional pois temos que propagar as subdivisões de células através das vistas, mas, graças ao baixo número de células e vistas devido às dimensões envolvidas nas instâncias, esse custo não acarreta uma perda de desempenho significativa.



(a) vazio

(b) após inserção de i_1

(c) após inserção de i_2

(d) após inserção de i_3

Figura 3.1: Representação de uma matriz dinâmica de múltiplas vistas

Uma simulação de uso de uma matriz dinâmica de múltiplas vistas pode ser vista na Figura 3.1. Os dois primeiros estados são como os de uma matriz dinâmica de uma única vista. Inicialmente, temos uma única célula representando todo o espaço livre do veículo de dimensões $60 \times 25 \times 30$ (comprimento, largura e altura). A inserção de um item i_1 de dimensões $20 \times 10 \times 15$ no canto superior direito, no nível da base do contêiner, resulta em 3.1b. O número dentro da célula indica a sua ocupação por uma caixa daquela altura. Assim que inserimos um item acima de i_1 uma nova vista é criada em 3.1c. A primeira vista, agora indexada por sua altura igual a zero, continua igual devido as dimensões da

nova caixa inserida. A nova vista, de altura quinze (ou seja, a altura onde o item i_2 foi inserido), representa o item de dimensões $20 \times 25 \times 10$ apoiado sobre a face superior de i_1. A Figura 3.1d mostra a propagação de subdivisões. Um novo item i_3 ($20 \times 25 \times 5$) é inserido no mesmo nível de i_1. Como devem ser criadas novas células, elas são criadas também na vista de altura quinze, mesmo que o novo item não seja representado nessa vista, visto que não é posicionado nessa altura. Como o nível em que i_3 é colocado já é representado por uma vista, não é necessária a criação de uma nova.

Uma clara vantagem sobre representações mais simples é em casos onde a inserção de novas caixas estende uma área de suporte provida pela parte superior de uma caixa adjacente. Por exemplo, na Figura 2.2, $I_{1,1}$ e $I_{2,1}$ geram uma superfície de apoio conjunta onde uma caixa grande pode ser posicionada. Esse ponto continua valendo mesmo se existem espaços vazios ou outras caixas entre as que estão sendo consideradas.

3.2 Construção de uma Solução

Devido à complexidade e ao grande número de vezes que uma solução é executada, os procedimentos de empacotamento devem ser relativamente simples. Os carregamentos são sempre gerados com uma heurística construtiva, que empacota os itens de forma gulosa na mesma ordem em que os recebe. Essa ordem inicialmente é inversa a qual os clientes serão visitados, utilizando fragilidade e depois volume como critérios de desempate: itens não-frágeis e itens maiores são carregados primeiro. Usa-se uma adaptação proposta por Gendreau et al. (2006) do algoritmo inferior-esquerda (Baker, Coffman e Rivest (1980), inicialmente para itens bidimensionais). Empacota-se o item corrente na posição normal com menor coordenada z disponível, usando o menor valor nas coordenadas x e depois y em caso de empate. Os itens são rotacionados em 90 ° no plano x-z caso não se encontre nenhuma inserção possível.

3.3 Geração de Permutações de Empacotamentos

Explorar o espaço de busca do problema do empacotamento como um todo é inviável pelo seu tamanho, i.e. tentar colocar cada caixa em todas as posições dentro do veículo. O que pode-se fazer é trabalhar com a ordem utilizada pela heurística construtiva. Ao colocar itens em ordens diferentes no veículo seguindo a heurística inferior-esquerda obtemos diferentes resultados finais. Precisa-se então algum método para escolher quais permutações do conjunto de itens I serão avaliadas, já que tentar todas ainda é inviável (temos $n!$ permutações).

Ao isolar algumas instâncias do empacotamento e executar todas as $n!$ permutações observamos que os resultados ótimos eram significativamente melhores que os gerados por uma heurística gulosa. Se escolhemos aleatoriamente um número pequeno de permutações dentre todas possíveis, porém, obtemos resultados muito piores, o que indica que apesar de existirem várias ordens com resultados bons, uma estratégia completamente randomizada não é adequada para encontrá-las.

O algoritmo utilizado para gerar as permutações consiste em atribuir a cada item a ser empacotado um número e, na hora de carregar o veículo, utiliza-se esse número para determinar a ordem em que eles serão considerados pela heurística construtiva. Os índices gerados são randômicos, mas é importante criar um viés para ordens que tenham uma maior probabilidade de respeitar o carregamento sequencial: a permutação $(1, 2, 3, 4, 5, 6, 7)$, por exemplo, tem, em média, mais chances de gerar um empacota-

mento factível (em especial, vai respeitar a restrição de carregamento sequencial) do que a permutação $(7, 6, 5, 4, 3, 2, 1)$, assumindo que o índice refere-se a ordem inicial considerada (i.e. a ordem inversa a qual os clientes de cada item serão visitados). Deve-se lembrar, porém, que é possível em algumas instâncias que ordens mais próximas à segunda sejam ótimas, e, portanto, apesar de tendermos para ordens próximas a inicial, eventualmente consideramos ordens mais distantes.

Algoritmo 3.1 Gerando uma permutação de itens

Entrada: Uma lista indexada l de tamanho n
Saída: A lista l permutada
1: cria vetor auxiliar *idx* de tamanho n
2: **para** $i = 1$ **até** n **faça**
3: $r_0 \leftarrow random()$
4: $r_1 \leftarrow random()$
5: $r_2 \leftarrow random()$
6: $t \leftarrow r_0/(r_1 \bmod (n - i) + 1)$
7: $idx[i] \leftarrow t/(r_2 \bmod (n - i) + 1)$
8: **fim para**
9: ordena l em ordem não-decrescente, utilizando $idx[i]$ para as comparações de $l[i]$
10: **retorna** l

Na linha 9, ordenamos os itens a serem carregados usando idx para as comparações, i.e. i será empacotado antes de j somente se $idx[i] \leq idx[j]$. No final l contém os índices dos itens a serem carregados (item $l[1]$ é o primeiro, $l[2]$ o segundo, etc.). *random()* é uma rotina que retorna um número positivo aleatório qualquer. O algoritmo gera permutações usando uma distribuição não-uniforme que torna permutações similares à entrada mais prováveis.

O método age apenas sobre os índices, ignorando por completo características e representação dos itens, e o número de itens empacotados nas instâncias utilizadas não passa de 30, devido às dimensões envolvidas. O procedimento tem complexidade de tempo $O(n \log n)$ para gerar uma permutação já que a ordenação domina a complexidade do laço. Para gerar $O(n^2)$ permutações, o tempo tomado é pequeno, não impactando o algoritmo de forma significativa, considerando o n encontrado ao executá-lo nas instâncias tratadas.

O procedimento para gerar o empacotamento de uma rota é então bastante simples. Considera-se um número θn^2 de permutações, onde n é o número de clientes pertencentes à rota. As permutações são todas geradas através do Algoritmo 3.1. Chama-se então a heurística construtiva inferior-esquerda que gera um empacotamento seguindo a ordem passada. Assim que se encontra um empacotamento que ocupe um comprimento igual ou inferior a C ele é retornado (já que não nos interessa um empacotamento ainda melhor, basta que ele não tenha penalidades). No final, caso tal carregamento não tenha sido encontrado, retorna-se aquele cujo comprimento ocupado seja mínimo. É responsabilidade da busca tabu o tratamento de empacotamentos infactíveis.

3.4 Memória de Empacotamentos Realizados

A busca tabu gera uma vizinhança baseada em trocas simples de clientes entre rotas: avalia-se o estado atual tentando empacotar todos os veículos com as suas caixas corres-

pondentes, e gera-se um novo estado baseado no antigo com algum cliente movido de uma rota para outra (ver Capítulo 4). Devido à grande proximidade dos estados, é de se esperar que alguns empacotamentos sejam feitos mais de uma vez. Assim, devido à quantidade de empacotamentos em uma execução, é provável que um mesmo carregamento seja feito diversas vezes. O método de empacotamento, apesar de relativamente eficiente, ainda acarreta em um custo computacional considerável, e, se for possível eliminar estas repetições, espera-se um aumento no número de iterações realizadas e, consequentemente, melhores soluções.

Utiliza-se então, com base nessa ideia, uma *cache* implementada através de uma árvore de busca binária, onde cada elemento consiste de uma rota (uma lista ordenada de clientes) e um empacotamento associado. Primeiramente, guardando todos resultados recentes e eliminando algum resultado aleatório quando o tamanho máximo da *cache* era excedido, obtivemos, em média, um total de 65% de *hits*, o que ajuda significativamente no número total de iterações realizadas.

Como geramos permutações randômicas, para uma mesma entrada podemos esperar resultados diferentes em execuções seguidas. Assim quando salvamos um empacotamento factível garantimos que em chamadas futuras ele será considerado factível. Da mesma forma, se marcamos um empacotamento com infactível impedimos que em futuras chamadas um empacotamento melhor, eventualmente factível, seja encontrado, e portanto não o fazemos. Após um número τ de tentativas, porém, desistimos de procurar por empacotamentos para a mesma rota, isto é, a salvamos como infactível a fim de não desperdiçar tempo com rotas impossíveis ou muito difíceis de serem empacotadas. As duas estratégias em conjunto impactam significativamente a qualidade da solução encontrada.

Algoritmo 3.2 Gerenciamento da *Cache*

Entrada: uma rota R
Saída: um empacotamento para itens de R

seja C_p a cache de empacotamentos factíveis
$C_p : Rota \rightarrow Empacotamento$
seja C_n a cache de empacotamentos não factíveis
$C_n : Rota \rightarrow \mathbb{N}$
se $R \in C_p$ **então**
    retorna $C_p[R]$
senão se $R \in C_n$ **e** $C_n[R] = \tau$ **então**
    retorna *impossível*
fim se
$e \leftarrow$ empacota R
se e é factível **então**
    $C_p \leftarrow C_p \cup (R, e)$
    remove R de C_n se necessário
senão
    $C_n \leftarrow C_n \cup (R, 0)$
    $C_n[R] \leftarrow C_n[R] + 1$
fim se
remove elemento aleatório de C_p ou C_n se tamanho máximo excedido
retorna e

4 UMA BUSCA TABU PARA O 3L-CVRP

A parte de roteamento do 3L-CVRP é feita através de uma busca tabu. A solução inicial é construída através do algoritmo de economias (CLARKE; WRIGHT, 1964) e então, a partir desta, o espaço de busca é explorado através de sua vizinhança. No escopo do roteamento, o problema de empacotamento é abstraído na forma de uma rotina usada simplesmente para determinar a factibilidade de uma solução, em particular uma rota.

4.1 Solução Inicial

Algoritmo 4.1 Algoritmo de Economias adaptado ao 3L-CVRP

Entrada: uma instância do 3L-CVRP
Saída: uma solução não necessariamente factível
1: cria uma rota R_i para cada cliente i
2: cria uma lista de economias eco
3: **para cada** par de clientes (i, j) **faça**
4: insere em eco o par i, j com valor $c_{i0} + c_{0j} - c_{ij}$
5: **fim para**
6: ordena eco em ordem não-crescente de valor (economia)
7: **para cada** par (i, j) em eco **faça**
8: **se** é possível unir as rotas R_i e R_j respectivas nos pontos i e j **então**
9: $R_r \leftarrow R_i R_j$
10: $e \leftarrow$ empacota R_r
11: **se** e não é factível **então**
12: $e \leftarrow$ empacota R_r invertida
13: **fim se**
14: **se** e é factível **então**
15: une definitivamente as rotas
16: **fim se**
17: **fim se**
18: **fim para**
19: **se** número de rotas existentes $>$ veículos disponíveis **então**
20: repete o laço para todos pares (i, j) aceitando empacotamentos infactíveis
21: **fim se**
22: **retorna** o conjunto de rotas com os respectivos empacotamentos

O método utilizado para gerar a solução inicial é o de Clarke e Wright (1964), e funciona da seguinte maneira. Primeiro criamos uma rota para cada um dos clientes.

Construímos então uma lista de pares de clientes (i, j) , e atribuímos a cada um valor de economia, calculado por $c_{i0} + c_{0j} - c_{ij}$ (considerando os custos simétricos). Este valor corresponde à redução da distância total percorrida por todas as rotas quando unimos duas rotas com pontos finais i e j. Ordenamos a lista de forma que os pares de maior economia sejam processados primeiro. Por fim, basta percorrer a lista em ordem (o que acarreta em uma política gulosa) e a cada par realizar a união das rotas correspondentes se possível, i.e. os pontos devem ser pontos finais nas suas respectivas rotas e devemos prover um empacotamento factível para a rota resultante. Caso não seja encontrado um empacotamento factível, tenta-se reverter a rota unida e empacotar novamente. Este procedimento baseia-se na variação paralela do algoritmo de economias.

É possível que após este processamento, a solução resultante possua mais do que v rotas. Neste caso, a lista de pares de economias é mais uma vez percorrida, mas desta vez aceitamos uniões de rotas mesmo quando o empacotamento encontrado não é factível, e escolhe-se agora o par de menor função objetivo. Esse procedimento continua até que a solução possua um número de rotas igual a v. Note que neste caso a solução inicial gerada não é factível, o que será tratado mais tarde.

4.2 Algoritmo Busca Tabu

O procedimento geral utilizado para a otimização é apresentado no Algoritmo 4.2. Primeiro, gera-se uma solução inicial conforme a Seção 4.1 e inicializa-se os parâmetros, com a lista tabu inicialmente vazia. Depois, enquanto há tempo restante, faz-se o seguinte. Os movimentos são gerados em um determinado número e ordem, que serão especificados depois e um deles, que não esteja presente na lista tabu, é escolhido, que determinará a próxima solução. Esse movimento é então adicionado à lista tabu, que é atualizada conforme processo de envelhecimento. Quando o tempo de execução limite é atingido, retorna-se a melhor solução encontrada.

Algoritmo 4.2 Procedimento de otimização

Entrada: Instância, parâmetros
Saída: Uma solução s'

```
gera solução inicial s_0 {método de economias}
s' ← s_0
s ← s_0
lista_tabu ← ∅
enquanto tempo restante faça
    determina próximo vizinho não-tabu aceito s
    adiciona movimento à lista_tabu
    atualiza lista_tabu
    atualiza s' quando necessário
    se estava no modo infactível e s é factível, entra em modo factível
fim enquanto
retorna s'
```

Com respeito à factibilidade das soluções, uma execução pode estar em dois modos. O primeiro só ocorre quando a solução inicial gerada não é factível, denominado *modo infactível*. Os movimentos são gerados da mesma maneira independentemente do modo que estamos, mas no caso do modo infactível, aceita-se qualquer movimento que gere

uma solução factível, independentemente do valor da função objetivo, que é $f(x) = d + \alpha \cdot p_e + \beta \cdot c_e$, onde d é a distância total de todas rotas, p_e é o peso total excedido e c_e o comprimento total excedido. Uma vez que uma solução factível é encontrada, nenhuma solução infactível é aceita posteriormente, e entramos no *modo factível*. Uma solução só é dita factível se junto com ela são providos empacotamentos factíveis para todas as rotas que a formam. Isso tem a importante consequência de que precisamos executar o procedimento de empacotamento para todas as rotas quando quisermos determinar a factibilidade de uma solução, o que implica um custo computacional significativo.

4.2.1 Vizinhança

A cada iteração da busca tabu, uma série de movimentos (i.e. troca de clientes intra ou inter-rotas) é avaliada, a fim de determinar a próxima solução a ser gerada. É desta forma que o algoritmo explora o espaço de busca. Um movimento sempre tem uma ou mais rotas originais, às quais ele será aplicado, e as transforma em novas rotas. Os tipos de movimentos que serão considerados no momento da avaliação da vizinhança no algoritmo são os seguintes:

- *Shift*: insere-se um cliente $i \in R_0$ em uma posição definida em R_1 e se remove i da rota R_0, com $R_1 \neq R_0$.
- *Swap*: dados clientes $i \in R_0$ e $j \in R_1$ ($R_1 \neq R_0$), troca-se i e j de posição.
- *Crossover*: define-se dois pontos de corte, um em R_0 e outro em R_1 ($R_1 \neq R_0$). Constrói-se então uma nova R_0, que constitui a primeira metade da R_0 original (do início ao ponto de corte) e a segunda metade de R_1 (do ponto de corte ao fim da rota). O procedimento análogo é realizado para construir uma nova rota R_1.
- *Intra-swap*: troca-se dois clientes $i, j \in R_0$ de posição.

A vizinhança de uma solução é então definida da seguinte forma, sendo a ordem importante (ver próxima subseção). Primeiro, leva-se em conta todos os movimentos *shift*, i.e. todos os pares (cliente, posição) possíveis. Em seguida todos os movimentos do tipo *swap* e depois *crossover* são avaliados, considerando todos os pontos de inserção existentes. Os movimentos do tipo *intra-swap* são feitos de forma seletiva, sendo que considera-se aleatoriamente n_{iswaps} movimentos deste tipo.

Nenhum dos movimentos resulta em uma solução com um número maior de rotas do que a original. Assim, observa-se que a solução utiliza ao longo da execução um menor número de veículos, o que faz com que a solução final tenha *v* ou menos rotas, em outras palavras, que saiamos eventualmente do modo infactível.

4.2.2 Determinando a próxima solução

Uma maneira de se escolher o movimento a ser aplicada é avaliar toda a vizinhança e então escolher o melhor vizinho não-tabu como próxima solução. Isso acarreta, porém, em um grande custo computacional no 3L-CVRP, já que a avaliação da vizinhança é custosa: para avaliar cada vizinho precisamos calcular um empacotamento. A maneira adotada pelo nosso algoritmo é então imediatamente aceitar qualquer movimento que resulte em alguma melhora, conforme detalhado no algoritmo abaixo.

Os movimentos são avaliados na ordem que é descrita na subseção precedente, sendo que os movimentos de uma mesma categoria são gerados em ordem randômica. Por

Algoritmo 4.3 Política de aceitação de movimentos

Entrada: Solução corrente s
Saída: Movimento a ser aplicado
1: $m' \leftarrow \emptyset$
2: **repita**
3: gera próximo movimento m
4: **se** modo factível **e** distância total resultante maior que de m' **então**
5: descarta(m)
6: **fim se**
7: empacota nova solução
8: **se** modo factível **e** não encontrou empacotamento factível **então**
9: descarta(m)
10: **senão se** aplicar m em s resulta em melhor solução sem penalidades que s **então**
11: **retorna** m
12: **senão se** aplicar m resulta em melhor solução levando em conta penalidades do que s **e** modo infactível **então**
13: **retorna** m
14: **senão se** aplicar m resulta em solução com função-objetivo menor que aplicar m' **então**
15: $m' \leftarrow m$
16: **fim se**
17: **até** aceitar m ou acabar movimentos
18: **retorna** m'

exemplo, os clientes i que serão trocados de rotas nos movimentos *shift* são escolhidos em uma ordem aleatória, mas garante-se que todos movimentos *shift* serão avaliados antes dos movimentos *swap*.

Qualquer movimento que gera uma solução sem penalidades melhor que a atual é imediatamente aceito. Aceita-se no modo infactível o primeiro que melhora a função objetivo. Escolhe-se o melhor movimento encontrado se nenhum resulta em uma solução melhor, o que resulta em uma piora da solução atual. O movimento escolhido é aplicado e adicionado então a lista tabu, que sempre contém os últimos movimentos realizados.

Uma observação interessante é que se estamos no modo factível pode-se descartar todas as próximas soluções que não provoquem uma melhora em relação à distância total percorrida em relação ao melhor vizinho encontrado, ou seja, não precisamos chamar o algoritmo de empacotamento para estas soluções.

Após todos os movimentos serem avaliados, se o melhor vizinho é ainda infactível mas melhor do que a melhor solução global em relação a distância total percorrida, intensifica-se a busca por empacotamentos, chamando novamente o algoritmo com $\theta = 30$ para todas as rotas que atualmente possuem penalidades.

Assim que um movimento é aceito, todos os movimentos individuais de clientes contidos no movimento aplicado são declarados tabu pelas próximas T iterações (*tabu tenure*). Por exemplo, depois de um *swap*, o cliente i não pode retornar à rota R_0 e o cliente j não pode retornar à rota R_1 durante o período de *tabu tenure*. A mesma ideia se aplica a todos os outros tipos de movimentos, ou seja, proíbe-se a reversão de movimentos individuais.

5 RESULTADOS EXPERIMENTAIS

Os testes foram realizados em duas etapas. A primeira consiste em testes feitos para avaliar o desempenho do algoritmo de empacotamento, com base em instâncias geradas ao longo de uma execução da busca tabu. A segunda etapa é feita diretamente sobre o 3L-CVRP, e produz resultados que são comparados à literatura. Os algoritmos foram testados com base no mesmo conjunto de instâncias utilizados na literatura, descrito por Gendreau et al. (2006), disponível em Toth et al. (2011). O volume disponível para carga nos veículos tem dimensões $L = 25$, $A = 30$ e $C = 60$. Cada cliente tem uma demanda entre um e três itens, e as dimensões dos itens ficam no intervalo relativo de 20% a 60% de cada coordenada em relação a cada dimensão do veículo. Para detalhes de como as instâncias foram geradas ver Toth e Vigo (2002) e Gendreau et al. (2006). A Tabela 5.1 mostra as principais características das instâncias, que serão a partir de agora referenciadas pelos seus índices.

Os experimentos foram realizados em um Intel Core i7 930 com 2.8 GHz, usando apenas um núcleo. Os algoritmos foram implementados em ANSI C++ e compilados com o GCC com otimização -O3.

5.1 Comparativo de Métodos de Empacotamento

Para corretamente avaliar a eficiência do algoritmo de empacotamento com respeito ao 3L-CVRP, as instâncias testadas foram retiradas de uma execução do método de roteamento nas mesmas condições da próxima seção. Para fins de comparação, implementou-se os seis seguintes algoritmos:

- PK1 corresponde ao algoritmo detalhado no Capítulo 3.
- PK2 difere de PK1 somente quanto ao método usado para gerar permutações de itens: no lugar do Algoritmo 3.1 uma permutação randômica é usada.
- PK3 é a heurística construtiva da Seção 3.2 executada uma única vez sobre a ordem inicial dos itens.
- PK4 utiliza uma busca local *first improvement* sobre as permutações. A partir da ordem inicial, realiza-se trocas (*swaps*) de itens i, j para todos $i = 0, \ldots, n$ e $j = i + 1, \ldots, n$ até que alguma resulte em uma melhora com relação à melhor ordem atual. O mesmo procedimento se repete até que um empacotamento de comprimento menor ou igual ao do contêiner ($C = 60$) seja encontrado ou até que uma iteração não resulte em melhora.

Tabela 5.1: Instâncias do 3L-CVRP usadas para os testes

Índice	Nome	Clientes	Caixas	Veículos
1	E016-03m	15	32	5
2	E016-05m	15	26	5
3	E021-04m	20	37	5
4	E021-06m	20	36	6
5	E022-04g	21	45	7
6	E022-06m	21	40	6
7	E023-03g	22	46	6
8	E023-05s	22	43	8
9	E026-08m	25	50	8
10	E030-03g	29	62	10
11	E030-04s	29	58	9
12	E031-09h	30	63	9
13	E033-03n	32	61	9
14	E033-04g	32	72	11
15	E033-05s	32	68	10
16	E036-11h	35	63	11
17	E041-14h	40	79	14
18	E045-04f	44	94	14
19	E051-05e	50	99	13
20	E072-04f	71	147	20
21	E076-07s	75	155	18
22	E076-08s	75	146	19
23	E076-10e	75	150	18
24	E076-14s	75	143	18
25	E101-08e	100	193	24
26	E101-10c	100	199	28
27	E101-14s	100	198	25

- PK5 possui a mesma ideia de viés do algoritmo PK1, mas é implementado de uma forma diferente. O Algoritmo 5.1 descreve como a próxima permutação é gerada. Os itens são escolhidos com base no seu índice: itens com índice menor tem maior probabilidade de ficarem no início da lista. Fora o método de permutação, o algoritmo é igual a PK1.

- PK6 é a nossa implementação do método de busca em árvore introduzido por Bortfeldt (2010). Esse é o único algoritmo que não utiliza uma matriz dinâmica de múltiplas vistas para a representação. Os parâmetros utilizados foram, conforme notação do artigo: 3000 maxApCalls, 5 maxRefPoints e 2 maxBoxRankDiff.

Todos algoritmos PK1-PK5 retornam a primeira solução de comprimento não superior a 60 ou a melhor solução encontrada. PK6 não retorna soluções com penalidades: a solução retornada é sempre de comprimento igual ou inferior a 60. Caso nenhuma seja encontrada, nenhum empacotamento é retornado, situação representada nas tabelas pelo símbolo ∞.

Algoritmo 5.1 Detalhamento de PK5

Entrada: Uma lista indexada l de tamanho n, onde $l[1]$ referencia a primeira caixa, $l[2]$ a segunda, etc.
Saída: A lista l permutada

```
1: cria nova lista nl
2: C ← 1, ..., n
3: enquanto C ≠ ∅ faça
4:     a ← 0
5:     para cada i ∈ C faça
6:         a ← a + (n − i)^3
7:     fim para
8:     r ← (random() mod a) + 1
9:     a, i ← 0
10:    enquanto a < r faça
11:        i ← i + 1
12:        a ← a + (n − i)^3
13:    fim enquanto
14:    insere l[i] no final de nl
15:    C ← C − i
16: fim enquanto
17: retorna nl
```

A Tabela 5.2 relata os resultados obtidos sobre 4869581 instâncias. Durante uma execução da solução proposta para o 3L-CVRP, conforme a Seção 5.2, cada chamada para a sub-rotina de empacotamento foi substituída por seis, uma para cada método. Todos os métodos executaram para todas as rotas geradas em cada uma das 27 instâncias. O tempo levado por cada chamada foi somado. Analisou-se apenas se o algoritmo retornava uma solução factível. O mecanismo de memória (*cache*) foi desabilitado para estes testes. O número de iterações para cada execução dos métodos PK1, PK2 e PK5 foi de n^2, onde n é o número de itens a serem carregados. Os outros métodos são executados conforme suas descrições.

Tabela 5.2: Comparativo de algoritmos de empacotamento

	PK1	PK2	PK3	PK4	PK5	PK6
soluções factibilizadas	74.9%	49.1%	41.6%	57.4%	59.6%	64.3%
tempo [s]	3861	3487	267	10269	8025	12892

Pode-se notar que o algoritmo PK1 é claramente superior às outras implementações, conseguindo factibilizar 74.9% de todas as rotas. Com relação ao tempo tomado o algoritmo é o segundo mais rápido, perdendo apenas obviamente para PK2 e PK3. O baixo custo computacional do método pode ser constatado se o comparamos com PK2: um aumento de apenas 10.7% no tempo computacional em relação a geração completamente randômica de permutações.

Mesmo o método mais simples, PK3, foi capaz de prover um carregamento para uma fração significativa de 41.6%. Pode-se concluir, portanto, que para um grande número de chamadas, os itens podem ser empacotados de forma trivial. Da mesma forma, uma parte

das rotas geradas são impossíveis de serem factibilizadas. O intervalo entre esses dois tipos de rotas é então o mais crítico, composto de rotas difíceis de empacotar. Além disso, é de se esperar que quanto melhor a qualidade da solução para o 3L-CVRP, mais difíceis serão as rotas de serem empacotadas, já que o espaço dentro dos contêineres será melhor aproveitado, com mais itens. Isso tem uma grande implicação na solução do problema. Mesmo que o algoritmo de roteamento explore o espaço de busca de maneira eficiente, se o método de empacotamento falha em prover um carregamento factível para essas rotas, o algoritmo não reconhece as melhores soluções como factíveis, e consequentemente a solução final será pior.

Tabela 5.3: Desempenho dos algoritmos de empacotamento para rotas exemplo

Índice	Rota	n_{itens}	PK1	PK2	PK3	PK4	PK5	PK6
1	11,2,9,10,5	8	59	59	70	66	70	60
2	8,7,14,6	9	61	61	89	66	61	56
3	13,14,15,4	6	48	48	48	48	48	56
4	17,6,12,16,13,14	10	59	65	76	63	59	∞
5	6,2,1,3,4,5	11	57	69	69	62	69	58
6	21,15,9,7,13,10	11	55	76	76	60	60	57
7	14,22,20,19,18	9	59	63	1	59	59	60
8	7,8,9,11	10	60	70	89	72	60	∞
9	21,5,7,2	5	63	63	69	63	63	∞
10	19,25,24,6,3	11	60	72	75	72	60	∞
11	3,19,29,28,26	10	63	63	82	65	63	∞
12	6,2,30	5	58	58	75	58	58	58
13	12,10,11	12	58	75	75	71	60	55
14	13,11,4	9	57	57	62	57	57	53
15	29,16,27,28	12	60	81	81	63	60	∞
16	27,13,15,20	10	56	63	91	63	60	∞
17	25,18,24,23	9	51	56	72	55	60	59
18	40,34,31,28	9	61	61	67	61	61	∞
19	2,20,36,35,3	9	58	80	97	69	58	60
20	24,42,44,46,43,27,21	13	60	82	82	72	60	∞
21	28,69,71,60,70,20,57	12	60	77	88	74	60	∞
22	8,19,14,59,53,7	9	60	88	103	76	60	∞
23	2,73,33	7	47	66	66	55	55	55
24	74,47,37,36,69,61	11	57	57	57	57	57	57
25	85,61,16,91,93,99,89	12	59	90	90	72	72	∞
26	90,84,82,83,91	11	63	91	91	63	63	∞
27	78,34,35,65,71,66,27	12	59	83	83	73	63	59
média			58,07	69,41	78,63	64,26	59,96	∞
número de soluções factibilizadas			22	6	2	8	19	14

Para exemplificar empacotamentos difíceis, a Tabela 5.3 mostra o desempenho dos seis algoritmos para 27 rotas, retiradas cada uma da respectiva instância de mesmo índice na Tabela 5.1. Durante uma execução da solução do 3L-CVRP sobre cada instância, foi escolhida aleatoriamente uma rota em que algum método retorna uma solução com $55 \leq C \leq 65$ e na qual pelo menos uma das soluções retornadas fosse diferente das outras. A primeira coluna indica de qual instância a rota foi retirada. A coluna *rota* é

uma lista de clientes determinando a ordem em que eles devem ser visitados. Os itens a serem empacotados, cuja quantidade aparece em n_{items}, são definidos através da rota. Os resultados são dados como comprimento total ocupado pelo plano de empacotamento retornado. São apresentados também a média e o número de empacotamentos com comprimento menor ou igual a 60.

As conclusões sobre a Tabela 5.2 são reafirmadas. PK1 factibiliza o maior número de rotas e o comprimento médio ocupado pelos seus empacotamentos é o menor. Nota-se também que a combinação de uma matriz dinâmica de múltiplas vistas com a heurística construtiva inferior-esquerda e um método de geração de permutações é eficiente mesmo com outros métodos, como exemplificado por PK5. De fato, foram testadas diversas outras distribuições de probabilidade para as permutações, mas a definida no Capítulo 3 foi a que obteve melhores resultados.

5.2 Resultados do 3L-CVRP

Os experimentos sobre as instâncias da Tabela 5.1 foram realizados de forma similar à literatura. Cada teste roda até que seu correspondente *time_limit* seja atingido. O tempo dado para cada instância é a metade do utilizado por Gendreau et al. (2006): 900 segundos para as instâncias 1-9, 1800 segundos para as instâncias 10-18 e 3600 segundos para as instâncias 19-27. O tempo é o único critério de parada utilizado. A Tabela 5.4 descreve os valores usados para os parâmetros, que foram definidos através de experimentação.

Tabela 5.4: Valores para os parâmetros usados nos experimentos

Parâmetro	Descrição	Valor
τ	tentativas de empacotamento	10
θ	const. permutações de empacotamento	1
n_{iswaps}	número de intra-swaps randomizados	$\min(n^2/4, 250)$
T	tabu tenure	$\min(15, n/2)$
α	penalidade para excesso de peso	$20\delta/P$, onde δ é o c_{ij} médio
β	penalidade para excesso de comprimento	$20\delta/C$
tl	tempo limite dado para a execução	$900s$ se $n < 35$, $1800s$ se $35 \leq n < 50$, $3600s$ caso contrário

A Tabela 5.5 apresenta os resultados médios obtidos por nosso algoritmo e os compara aos da literatura. Os algoritmos se referem às seguintes publicações: TS (GENDREAU et al., 2006), AAcr (DE ARAÚJO, 2006), GTS (TARANTILIS; ZACHARIADIS; KIRANOUDIS, 2009), ACO (FUELLERER et al., 2010), DMTS (WANG et al., 2010), TSLW (TAO; WANG, 2010), VRLH1 (BORTFELDT, 2010) e TSFI (este trabalho).

Os resultados correspondem à média de dez execuções das soluções factíveis encontradas de melhor qualidade em cada execução para todos métodos menos TS, AAcr e GTS, que são executados apenas uma vez (GENDREAU et al., 2006; DE ARAÚJO, 2006; TARANTILIS; ZACHARIADIS; KIRANOUDIS, 2009). As médias de todas as instâncias são apresentadas embaixo dos resultados para cada método, bem como a diferença percentual média entre o valor de cada instância e àquele inicialmente publicado por Gendreau et al. (2006), sendo que um valor negativo indica que os resultados do método em

questão são melhores, ou seja:

$$gap_{gen}(X) = \text{average}\left(\frac{ttd_{avg}(X,i)}{ttd_{avg}(TS,i)}\right)$$

para toda instância *i*, onde $ttd_{avg}(X,i)$ corresponde ao valor da solução média dada pelo método X na instância *i*. Da mesma forma, a média das diferenças relativas entre cada método e a melhor solução conhecida é fornecida.

$$gap_{bkv}(X) = \text{average}\left(\frac{ttd_{avg}(X,i)}{\min(ttd_{min}(M,i)\forall M)}\right)$$

Nota-se que as publicações mais recentes apresentam resultados próximos, e que o TSFI produziu soluções de melhor qualidade em média, uma melhora de 0.4% a 1.87% quanto aos métodos mais recentes (a partir de Fuellerer et al. (2009)). Dos resultados da literatura, destacam-se atualmente Tao e Wang (2010), Wang et al. (2010) e Bortfeldt (2010). Para cada instância, o melhor valor encontrado está em negrito.

Tabela 5.5: Resultados médios do nosso algoritmo e da literatura

Inst.	TS[a] (2006)	AAcr[a] (2006)	GTS[a] (2009)	ACO (2009)	DMTS (2010)	TSLW (2010)	VRLH1 (2010)	TSFI
1	316.32	304.13	321.47	305.35	**301.77**	302.02	302.02	304.84
2	350.58	**334.96**	**334.96**	**334.96**	**334.96**	**334.96**	**334.96**	**334.96**
3	447.73	391.65	430.95	409.79	387.91	**381.37**	404.34	384.90
4	448.48	447.98	458.04	440.68	438.59	440.68	437.94	**437.19**
5	464.24	454.14	465.79	453.19	440.23	**438.43**	447.49	449.66
6	504.46	499.07	507.96	501.47	499.48	**498.32**	501.47	501.77
7	831.66	865.77	796.61	797.47	771.09	773.01	791.03	**770.22**
8	871.77	823.21	880.93	820.67	**805.95**	808.59	824.00	808.14
9	666.10	645.14	642.22	635.50	630.90	634.00	663.39	**630.13**
10	911.16	849.33	884.74	841.12	832.46	839.76	829.31	**825.86**
11	819.36	822.17	873.43	821.04	**781.85**	794.03	815.35	788.77
12	651.58	625.92	624.24	629.07	614.78	616.30	636.10	**610.43**
13	2928.34	2807.56	2799.74	2739.80	2715.82	**2698.25**	2701.10	2713.05
14	1559.64	1494.67	1504.44	1472.26	1456.13	1432.44	**1419.84**	1425.18
15	1452.34	1467.70	1415.42	1405.48	1371.26	1377.06	**1357.01**	1370.41
16	707.85	702.70	**698.61**	698.92	699.54	698.92	704.24	699.49
17	920.87	879.57	872.79	870.33	875.19	**867.28**	969.59	871.71
18	1400.52	1316.12	1296.59	1261.07	1248.28	1236.77	**1233.99**	1248.09
19	871.29	823.48	818.68	781.29	776.35	768.58	**755.05**	763.44
20	732.12	623.35	641.57	611.26	593.17	603.49	598.23	**586.00**
21	1275.20	1168.79	1159.72	1124.55	1121.60	1123.37	1108.40	**1099.24**
22	1277.94	1252.62	1245.35	1197.43	1176.76	1179.88	1175.82	**1159.85**
23	1258.16	1216.21	1231.92	1171.77	1148.02	1160.05	1138.19	**1128.71**
24	1307.09	1193.12	1201.96	1148.70	1144.56	1141.02	1127.76	**1126.55**
25	1570.72	1499.02	1457.46	1436.32	1457.09	1421.36	1436.63	**1405.00**
26	1847.95	1782.83	1711.93	1616.99	1616.61	**1608.16**	1630.63	1612.31
27	1747.52	1675.28	1646.44	1573.50	1574.23	1555.34	**1541.99**	1560.06
avg	1042.26	998.76	997.18	966.67	956.10	953.09	958.74	948.74
gap_{gen}[b]	0.0%	-4.17%	-3.54%	-6.43%	-7.66%	-7.80%	-6.90%	-8.16%
gap_{bkv}[c]	10.50%	5.72%	6.41%	3.18%	1.80%	1.64%	2.60%	1.22%

[a] Valor referente a uma única execução
[b] Melhora em relação a Gendreau (2006)
[c] Diferença média em relação aos melhores valores conhecidos

Em relação ao tempo computacional tomado por cada método uma análise se torna difícil devido a quantidade de fatores envolvidos. A Tabela 5.6 mostra o tempo médio tomado por cada algoritmo para encontrar a solução retornada. Acreditamos que a maioria dos métodos estejam em um nível semelhante, com a exceção de Bortfeldt (2010), que é claramente mais rápido. Com o mesmo intuito, a Tabela 5.7 mostra as máquinas utilizadas por cada publicação para a realização de seus testes.

Por completude, a Tabela 5.8 apresenta os resultados mínimos publicados, isto é, a distância total percorrida mínima retornada pelo método nas execuções realizadas para os testes. Deve-se notar, porém, que para os métodos TS, AAcr e GTS, os mínimos são os mesmos que os resultados médios, já que os resultados publicados se referem a uma execução apenas. Já para o método TSLW, apenas o resultado médio das execuções foi

Tabela 5.6: Tempo em segundos tomado por cada método para encontrar a solução retornada

Inst.	TS (2006)	AAcr (2006)	GTS (2009)	ACO (2009)	DMTS (2010)	TSLW (2010)	VRLH1 (2010)	TSFI
1	129.5	73.52	7.8	12.0	193.0	58.3	41.6	7.8
2	5.3	224.9	7.2	0.6	9.3	12.7	0.3	0.5
3	461.1	180.8	352.6	121.8	87.0	117.3	159.1	126.4
4	181.1	111.9	204.0	5.4	91.3	14.5	12.4	12.1
5	75.8	12.8	61.3	30.9	444.7	163.1	170.5	138.0
6	1167.9	707.1	768.8	18.4	125.9	37.9	15.3	43.0
7	181.1	27.4	241.5	67.4	394.9	89.2	62.8	96.7
8	156.1	13.1	140.0	78.6	331.0	113.6	98.9	337.4
9	1468.5	331.9	604.7	16.3	197.9	19.5	11.2	82.1
10	714.0	275.4	803.1	246.7	707.0	293.1	139.8	593.1
11	396.4	353.2	308.5	199.8	820.9	273.6	118.8	643.1
12	268.1	474.6	180.8	48.2	194.7	129.3	12.8	227.6
13	1639.1	154.9	1309.5	308.8	859.4	382.1	232.9	971.9
14	3451.6	589.4	2678.1	642.8	1638.7	608.6	312.2	922.1
15	2327.4	1914.7	1466.3	656.8	1537.3	512.8	299.9	755.5
16	2550.3	88.2	2803.2	14.8	46.5	6.5	2.4	225.6
17	2142.5	2114.2	1208.6	14.9	731.7	10.3	1.7	87.7
18	1452.9	2227.6	1300.9	2209.8	1748.8	1123.7	315.1	1045.2
19	1822.3	2921.0	1438.4	623.6	1376.9	572.5	419.2	2147.8
20	790.0	7153.5	1284.8	3901.0	1647.8	2564.1	432.1	2482.9
21	2370.3	2408.4	1704.8	5180.6	1594.5	2043.6	452.3	2895.0
22	1611.3	1218.9	1663.5	2290.3	1287.7	1371.5	428.6	2056.6
23	6725.6	5566.5	3048.2	3727.6	1091.0	1778.1	430.5	1998.5
24	6619.3	7188.7	2876.8	1791.5	469.8	2023.5	413.3	1232.4
25	5630.9	6976.2	3432.0	8817.1	1582.8	3795.7	463.5	2922.7
26	4123.7	7157.8	3974.8	6904.3	1488.7	2550.6	436.7	1586.5
27	7127.2	2304.7	5864.2	10483.9	1440.1	4848.3	441.6	1481.4
avg	2058.9	1954.5	1471.6	1793.1	820.0	945.0	219.5	930.4

Tabela 5.7: Máquinas utilizadas por cada método para testes

Publicação	Método	Máquina
Gendreau et al. (2006)	TS	Pentium IV 3.0 GHz
de Araújo (2006)	AAcr	Pentium IV 2.8 GHz
Tarantilis et al. (2009)	GTS	Pentium IV 2.8 GHz
Fuellerer et al. (2010)	ACO	Pentium IV 3.2 GHz
Wang et al. (2010)	DMTS	Intel Xeon E5430 2.66 GHz
Tao e Wang (2010)	TSLW	Pentium IV 2.39 GHz
Bortfeldt (2010)	VRLH1	Core 2 Duo E8500 3.17 GHz
Este trabalho	TSFI	Core i7 930 2.8 GHz

publicado, e então é esse que aparece. Para os outros métodos, o resultado é referente a melhor solução encontrada dentre as mesmas 10 execuções usadas para alcançar os

resultados da Tabela 5.5. Apesar de uma análise dos resultados ser mais difícil, devido à maior instabilidade dos mesmos quando extraímos o mínimo do conjunto (e não a média), o mesmo padrão pode ser observado.

O nosso método se mostra particularmente bom em instâncias maiores. Dentre as nove instâncias que possuem 50 clientes ou mais, nós encontramos seis novas melhores soluções. O mesmo comportamento aparece quando analisamos a média, onde a nossa é a melhor publicada também em seis dessas instâncias.

Tabela 5.8: Melhores soluções publicadas

Inst.	TS[a] (2006)	AAcr[a] (2006)	GTS[a] (2009)	ACO (2009)	DMTS (2010)	TSLW[b] (2010)	VRLH1 (2010)	TSFI
1	316.32	304.13	321.47	304.13	**301.74**	302.02	302.02	304.13
2	350.58	**334.96**	**334.96**	**334.96**	**334.96**	**334.96**	**334.96**	**334.96**
3	447.73	391.65	430.95	399.68	387.34	**381.37**	388.10	**381.37**
4	448.48	447.98	458.04	440.68	**437.19**	440.68	**437.19**	**437.19**
5	464.24	454.14	465.79	450.93	**436.48**	438.43	443.61	442.21
6	504.46	499.07	507.96	498.32	498.32	498.32	**498.16**	499.02
7	831.66	865.77	796.61	792.13	**767.46**	773.01	769.68	769.68
8	871.77	823.21	880.93	820.67	**803.98**	808.59	810.89	806.25
9	666.10	645.14	642.22	635.50	**630.13**	634.00	**630.13**	**630.13**
10	911.16	849.33	884.74	840.75	826.39	839.76	**820.35**	822.46
11	819.36	822.17	873.43	818.87	**768.25**	794.03	800.52	781.17
12	651.58	625.92	624.24	626.37	**610.23**	616.30	**610.23**	**610.23**
13	2928.34	2807.56	2799.74	2739.80	2697.70	2698.25	**2679.86**	2693.24
14	1559.64	1494.67	1504.44	1466.84	1428.99	1432.44	**1368.42**	1390.18
15	1452.34	1467.70	1415.42	1367.58	1352.94	1377.06	**1331.27**	1350.85
16	707.85	702.70	**698.61**	698.92	**698.61**	698.92	**698.61**	**698.61**
17	920.87	879.57	872.79	868.59	871.63	**867.28**	876.22	871.24
18	1400.52	1316.12	1296.59	1255.64	1227.07	1236.77	**1206.67**	1227.08
19	871.29	823.48	818.68	777.18	762.47	768.58	757.04	**755.56**
20	732.12	623.35	641.57	604.28	583.45	603.49	585.65	**579.18**
21	1275.20	1168.79	1159.72	1110.09	1094.78	1123.37	1091.33	**1085.53**
22	1277.94	1252.62	1245.35	1194.18	1170.89	1179.88	1162.11	**1151.48**
23	1258.16	1216.21	1231.92	1158.51	1137.90	1160.05	1144.46	**1119.08**
24	1307.09	1193.12	1201.96	1136.80	1132.05	1141.02	**1114.54**	1116.57
25	1570.72	1499.02	1457.46	1429.64	1434.00	1421.36	1398.42	**1389.95**
26	1847.95	1782.83	1711.93	1611.78	1606.85	1608.16	**1590.31**	1594.45
27	1747.52	1675.28	1646.44	1560.70	1551.68	1555.34	**1528.43**	1537.10
avg	1042.26	998.76	997.18	960.87	943.43	953.09	939.97	939.96

[a] Valor referente a uma única execução
[b] Valor médio, mínimo não publicado

6 CONSIDERAÇÕES FINAIS

A modelagem de problemas reais de logística está se tornando altamente prática com o maior poder de processamento e melhor entendimento das características relevantes. Neste trabalho analisou-se o problema de roteamento de veículos capacitados com restrições de empacotamento tridimensionais, com diversas restrições tiradas diretamente de aplicações de mundo real. Ainda há espaço, porém, tanto para melhorar resultados nesse problema quanto para explorar problemas adjacentes, com mais restrições e que modelem realisticamente outras situações encontradas.

O resultado deste trabalho advém de três fatores principais. Primeiro, no subproblema do empacotamento, o qual se constatou de essencial importância em relação ao resultado da solução final obtida: pequenas mudanças no método de carregamento influenciam bastante a solução encontrada. Confirmou-se que estratégias menos pesadas computacionalmente são preferíveis, visto que os resultados decaem drasticamente quando começamos a usar algoritmos mais custosos. Há, porém, tempo suficiente para explorar o espaço de busca relativo a ordem do empacotamento de uma forma relevante, e fica claro que o principal desafio é explorá-lo de forma eficiente: manter-se em estados que probabilisticamente tenham uma maior chance de respeitar a restrição de empacotamento sequencial mostrou-se a melhor estratégia. O armazenamento de resultados em $cache$ também teve um impacto positivo, principalmente devido ao fator randômico do algoritmo.

Segundo, para que seja possível o uso de técnicas rápidas de empacotamento, é essencial o uso de uma estrutura de representação eficiente e capaz de representar os espaços dentro dos contêineres com uma certa fidedignidade. Com estruturas simples e incompletas, constatou-se uma grande dificuldade do algoritmo em produzir empacotamentos factíveis, devido ao grande desperdício de espaços livres. A estrutura proposta de matriz dinâmica com múltiplas vistas se mostrou adequada às necessidades do problema.

Por fim, a busca tabu, cujas características se mostraram sensíveis e de difícil determinação. A proposta *first improvement* mostrou-se capaz de produzir resultados de estado-da-arte, e a possível perda em qualidade de busca em não escolher o melhor vizinho é facilmente compensada pela diversificação e o maior número de iterações que essa estratégia permite.

Diversos trabalhos significativos ainda podem ser realizados nessa área. Uma série de problemas surgem ao retirarmos ou inserirmos restrições, como *pickup-and-delivery*, frota heterogênea de veículos, janelas de tempo, restrições de estabilidade de carga e distribuição entre os eixos do veículo, etc. Além disso, devido à complexidade do problema, estima-se que ainda há possibilidade de uma melhora significativa quanto à qualidade das soluções do estado-da-arte e um melhor entendimento do problema. Dois trabalhos importantes a serem realizados sobre o 3L-CVRP são a formulação do problema e o desenvolvimento de uma solução exata para o mesmo.

REFERÊNCIAS

AZEVEDO, B. et al. A Branch-and-Cut Approach for the Vehicle Routing Problem with Two-Dimensional Loading Constraints. **Anais do XLI Simpósio Brasileiro de Pesquisa Operacional**, Porto Seguro, 2009.

BAKER, B. S.; COFFMAN, E. G.; RIVEST, R. L. Orthogonal Packings in Two Dimensions. **SIAM Journal on Computing**, [S.l.], v.9, n.4, p.846–855, 1980.

BORTFELDT, A. **A Hybrid Algorithm for the Capacitated Vehicle Routing Problem with Three-Dimensional Loading Constraints**. Hagen: Fakultät für Wirtschaftswissenschaft der FernUniversität in Hagen, 2010. (460).

CLARKE, G.; WRIGHT, J. W. Scheduling of Vehicles from a Central Depot to a Number of Delivery Points. **Operations Research**, [S.l.], v.12, p.568–581, 1964.

DE ARAÚJO, O. C. B. **Problemas de Corte e Empacotamento Tridimensional e Integração com Roteamento de Veículos**. 2006. Tese (Doutorado em Ciência da Computação) — Faculdade de Engenharia Elétrica e de Computação - UNICAMP, Campinas.

FUELLERER, G. et al. Metaheuristics for Vehicle Routing Problems with Three-Dimensional Loading Constraints. **European Journal of Operational Research**, [S.l.], v.201, n.3, p.751–759, 2010.

GENDREAU, M. et al. A Tabu Search Algorithm for a Routing and Container Loading Problem. **Transportation Science**, Institute for Operations Research and the Management Sciences (INFORMS), Linthicum, Maryland, USA, v.40, p.342–350, Agosto 2006.

GLOVER, F. Tabu Search–Part I. **INFORMS Journal on Computing**, [S.l.], v.1, n.3, p.190–206, 1989.

GLOVER, F. Tabu Search–Part II. **INFORMS Journal on Computing**, [S.l.], v.2, n.1, p.4–32, 1990.

GLOVER, F.; LAGUNA, M. **Tabu search**. [S.l.]: Kluwer Academic Publishers, 1998.

IORI, M.; MARTELLO, S. Routing Problems with Loading Constraints. **TOP: An Official Journal of the Spanish Society of Statistics and Operations Research**, [S.l.], v.18, n.1, p.4–27, Julho 2010.

IORI, M.; SALAZAR-GONZALEZ, J.-J.; VIGO, D. An Exact Approach for the Vehicle Routing Problem with Two-Dimensional Loading Constraints. **Transportation Science**, [S.l.], v.41, n.2, p.253–264, 2007.

LODI, A. Multi-Dimensional Packing by Tabu Search. **Studia Informatica Universalis**, Bologna, v.2, p.111–126, 2002.

LODI, A.; MARTELLO, S.; VIGO, D. Heuristic and Metaheuristic Approaches for a Class of Two-Dimensional Bin Packing Problems. **INFORMS Journal on Computing**, Institute for Operations Research and the Management Sciences (INFORMS), Linthicum, Maryland, USA, v.11, p.345–357, Abril 1999.

MARTELLO, S.; PISINGER, D.; VIGO, D. The Three-Dimensional Bin Packing Problem. **OPERATIONS RESEARCH**, [S.l.], v.48, n.2, p.256–267, 2000.

NGOI, B. K. A.; TAY, M.; CHUA, E. Applying Spatial Representation Techniques to the Container Packing Problem. **International Journal of Production Research**, [S.l.], v.32, p.111–123, 1994.

PORTAL, G. et al. Uma Busca Tabu Aplicada ao Problema de Roteamento com Restrições de Empacotamento Tridimensionais. **Anais do XLI Simpósio Brasileiro de Pesquisa Operacional**, Porto Seguro, 2009.

TAO, Y.; WANG, F. A New Packing Heuristic Based Algorithm for Vehicle Routing Problem with Three-dimensional Loading Constraints. In: CONFERENCE ON AUTOMATION SCIENCE AND ENGINEERING. **Anais...** [S.l.: s.n.], 2010. p.972–977.

TARANTILIS, C. D.; ZACHARIADIS, E. E.; KIRANOUDIS, C. T. A Hybrid Metaheuristic Algorithm for the Integrated Vehicle Routing and Three-Dimensional Container-Loading Problem. **Transactions on Intelligent Transportation Systems**, [S.l.], v.10, p.255–271, Junho 2009.

TOTH, P. et al. **DEIS University of Bologna - Operations Research Group, Library of Instances**. Disponível em: <http://www.or.deis.unibo.it/research.html>. Acesso em: junho 2011.

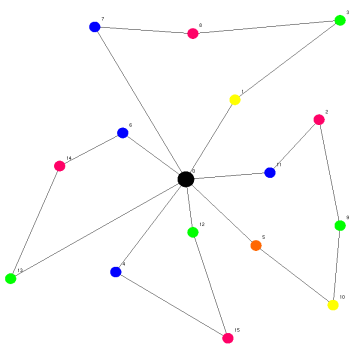
TOTH, P.; VIGO, D. The Vehicle Routing Problem. **SIAM Monographs on Discrete Mathematics and Applications**, Philadelphia, 2002.

WANG, F.; TAO, Y.; SHI, N. A Survey on Vehicle Routing Problem with Loading Constraints. In: INTERNATIONAL JOINT CONFERENCE ON COMPUTATIONAL SCIENCES AND OPTIMIZATION. **Anais...** [S.l.: s.n.], 2009. v.2, p.602–606.

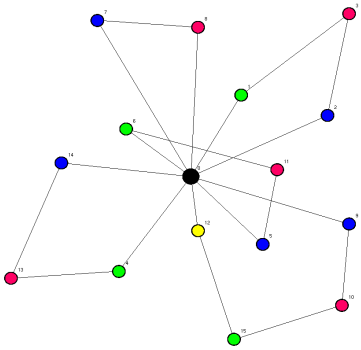
WANG, L. et al. Two Natural Heuristics for 3D Packing with Practical Loading Constraints. In: PACIFIC RIM INTERNATIONAL CONFERENCE ON TRENDS IN ARTIFICIAL INTELLIGENCE, 11., Daegu, Korea. **Proceedings...** Springer-Verlag, 2010. p.256–267.

APÊNDICE SOLUÇÕES PARA AS INSTÂNCIAS DA LITERATURA

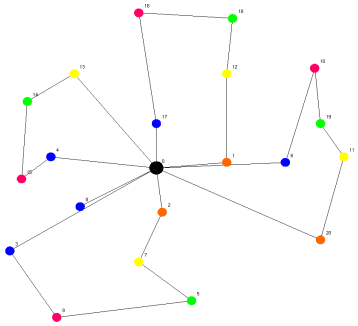
Roteamento das melhores soluções encontradas nas 10 execuções realizadas para obter os resultados do Capítulo 5 para todas as 27 instâncias testadas:



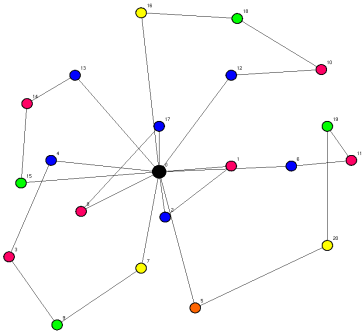
(1)



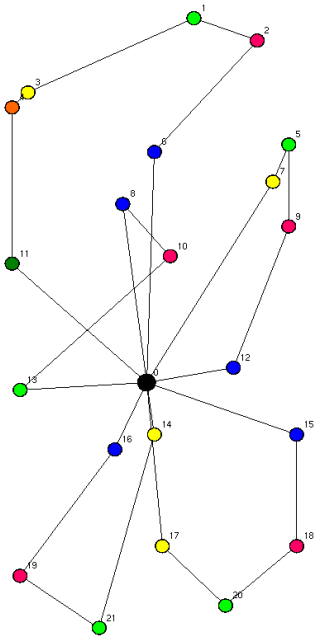
(2)



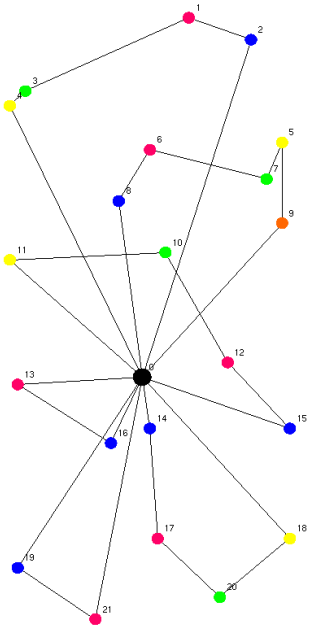
(3)



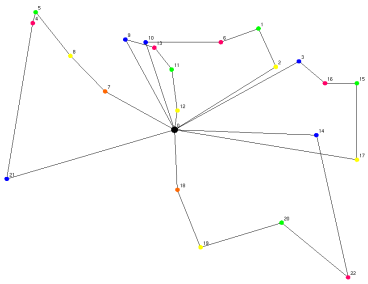
(4)



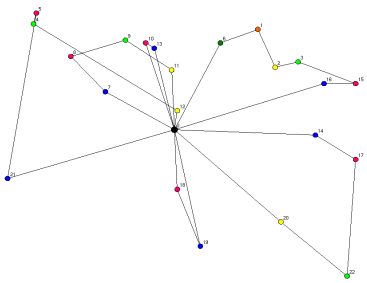
(5)



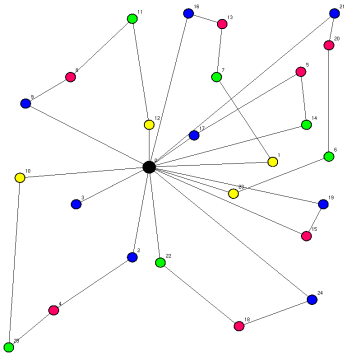
(6)



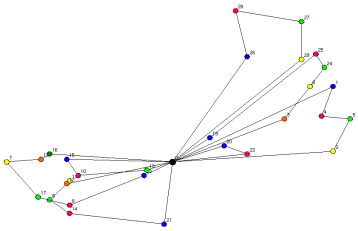
(7)



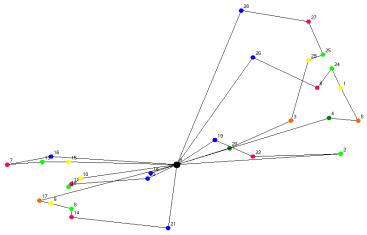
(8)



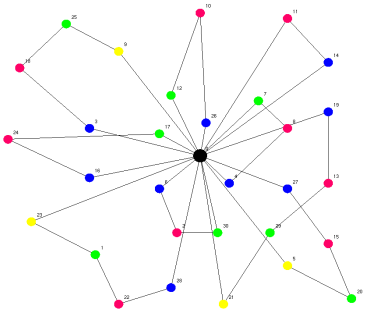
(9)



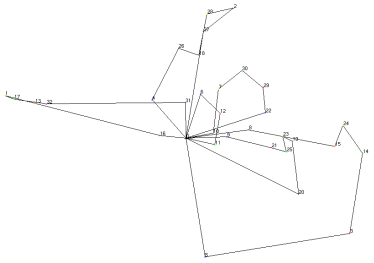
(10)



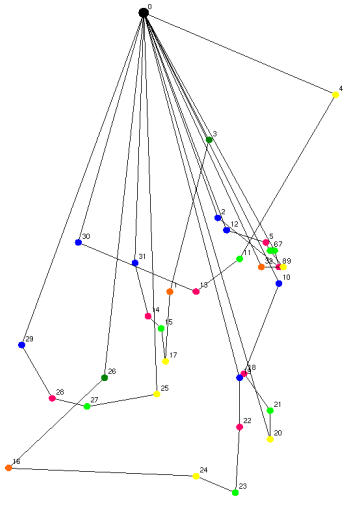
(11)



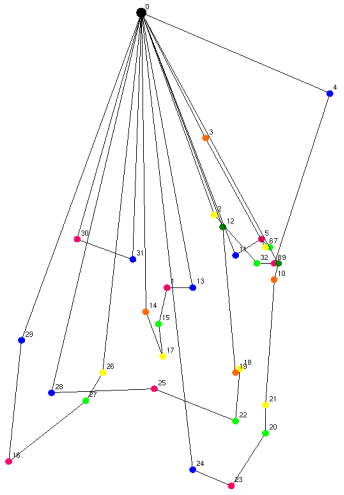
(12)



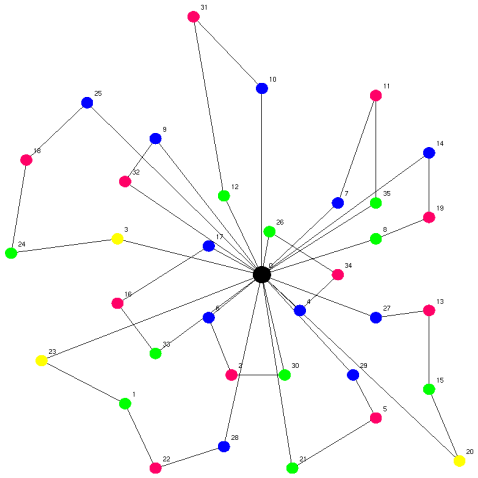
(13)



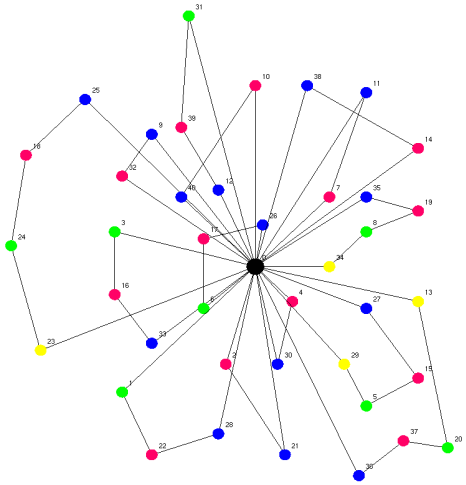
(14)



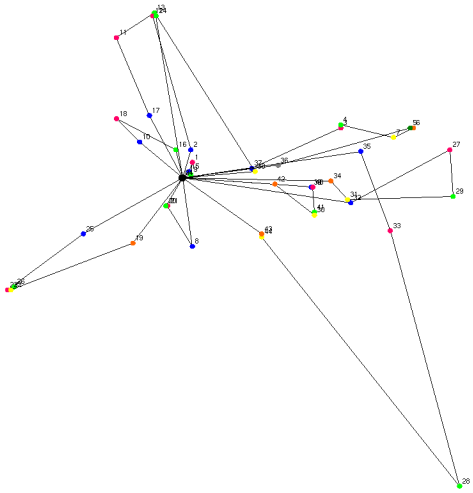
(15)



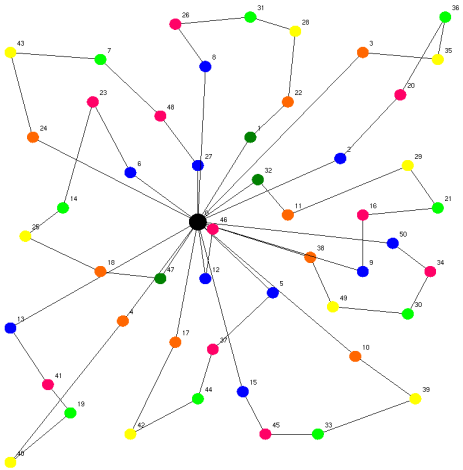
(16)



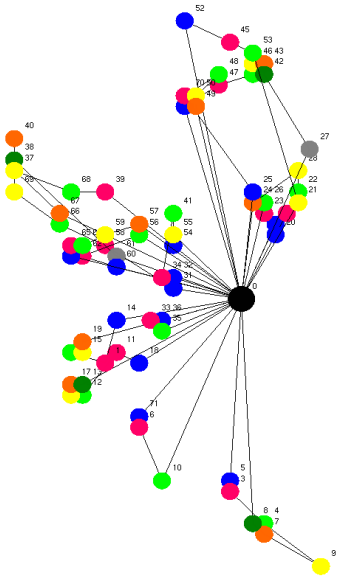
(17)



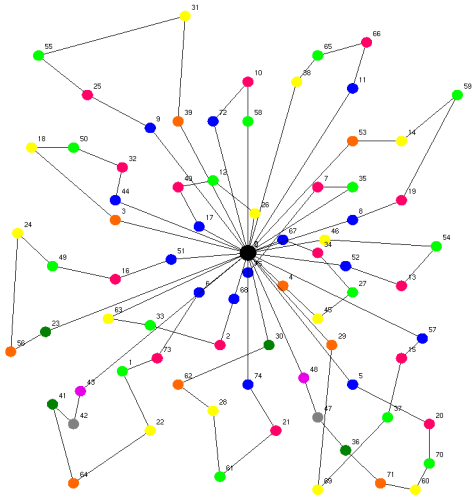
(18)



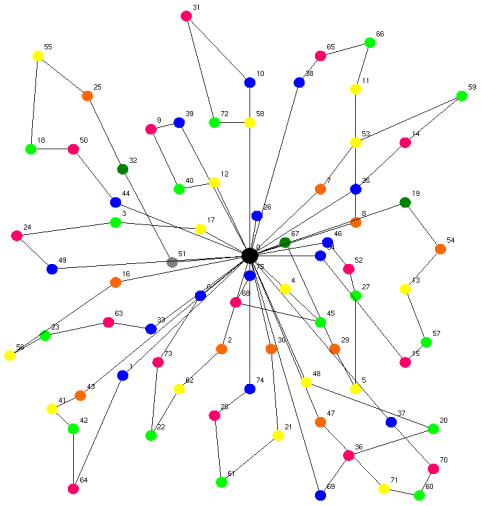
(19)



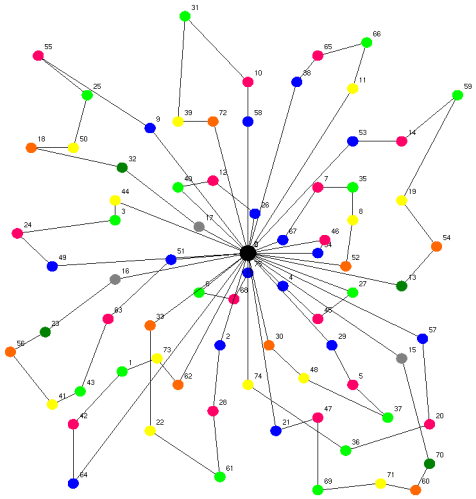
(20)



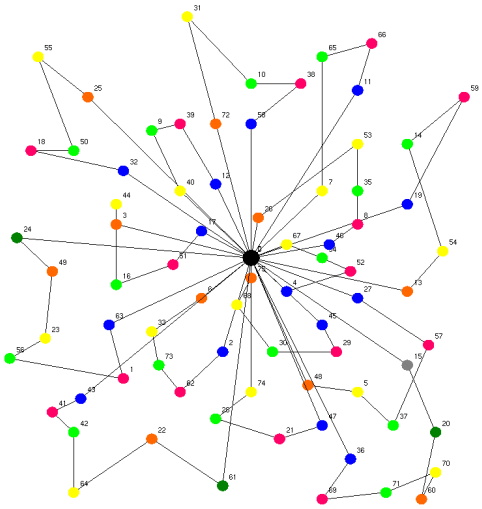
(21)



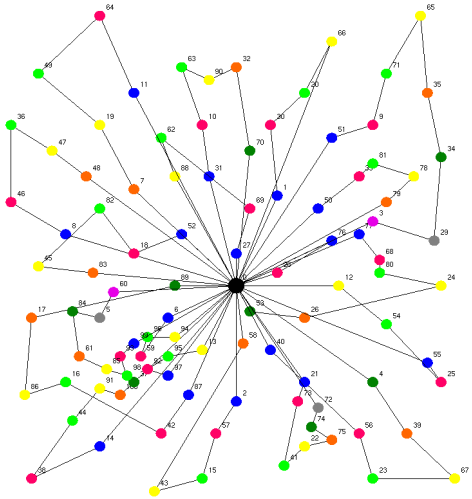
(22)



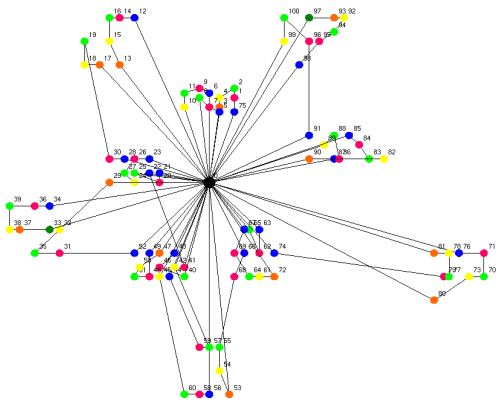
(23)



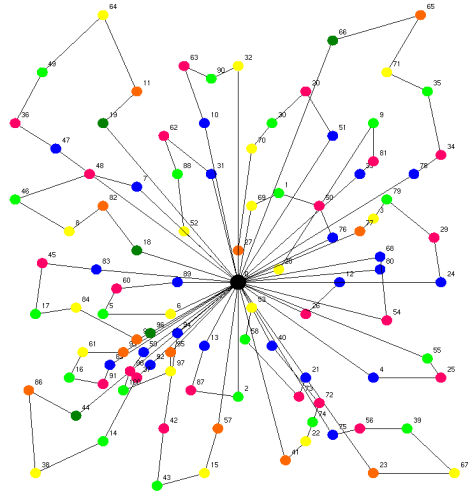
(24)



(25)



(26)



(27)