Universidade Federal Fluminense

SABIR RIBAS

Um algoritmo híbrido para o problema de roteamento de veículos com janelas de tempo

NITERÓI-RJ

2011

Universidade Federal Fluminense

SABIR RIBAS

Um algoritmo híbrido para o problema de roteamento de veículos com janelas de tempo

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

Orientador:
Prof. Luiz Satoru Ochi, D.Sc.

NITERÓI-RJ

2011

Um algoritmo híbrido para o problema de roteamento de veículos com janelas de tempo

Sabir Ribas

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

Aprovada por:

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

Prof. Marcone Jamilson Freitas Souza, D.Sc. / DECOM-UFOP

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

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

Niterói, 01 de Abril de 2011.

À minha noiva, família e amigos.

Resumo

O Problema de Roteamento de Veículos com Janelas de Tempo é uma variante do roteamento de veículos clássico, em que a demanda de cada consumidor deve ser atendida durante um intervalo temporal pré-estabelecido. Dado o caráter combinatório do problema, que pertence à classe NP-Difícil, sua resolução por abordagens puramente exatas é, em muitos casos, computacionalmente impraticável. Este fato motiva o desenvolvimento de algoritmos heurísticos para sua resolução, que são mais rápidos mas que, porém, não garantem a obtenção da melhor solução para o problema. Neste trabalho é proposto um algoritmo heurístico híbrido, que combina os métodos *Iterated Local Search*, *Variable Neighborhood Descent* e um procedimento exato de particionamento de conjunto. Esse procedimento de programação matemática é acionado periodicamente com vistas a combinar da melhor forma as rotas geradas ao longo do algoritmo. Adicionalmente, é desenvolvida uma versão paralela do método. Os experimentos computacionais mostraram que a abordagem híbrida proposta é competitiva, uma vez que dos 56 problemas considerados, o algoritmo foi capaz de melhorar a melhor solução da literatura em 16 casos e igualar ao melhor resultado em 31 casos.

Palavras-chave: Roteamento de Veículos com Janelas de Tempo, Algoritmo Híbrido, *Iterated Local Search*, *Variable Neighborhood Descent*, Particionamento de Conjunto.

Abstract

The Vehicle Routing Problem with Time Windows is a particular case of the classical Vehicle Routing Problem in which the demands of each customer should be met within an established time window. Due to the combinatorial complexity of the problem its resolution by pure exact methods is, in many cases, computationally impractical. This fact motivates the development of heuristic algorithms, which are usually faster but do not guarantee the best solution for the problem. This work proposes a hybrid algorithm that combines the metaheuristic Iterated Local Search, the method Variable Neighborhood Descent and an exact Set Partitioning model. The latter mathematical procedure is periodically activated in order to perform the best combination of the routes generated along the execution of the algorithm. In addition, we develop a parallel version of the method. The computational results demonstrate that the proposed hybrid approach is quite competitive, since out of the 56 test problems considered, the algorithm was found capable to improve the best known solution in 16 cases and to equal the result of another 31.

Keywords: Vehicle Routing Problem with Time Windows, Hybrid Algorithm, Iterated Local Search, Variable Neighborhood Descent, Set Partitioning.

Siglas e Abreviações

AG	:	Algoritmos Genéticos
BT	:	Busca Tabu
DPE	:	Distribuição de Probabilidade Empírica
GVNS	:	*General Variable Neighborhood Search*
IILS	:	*Intensified Iterated Local Search*
IILS-SP	:	*Intensified Iterated Local Search with Set Partitioning*
ILS	:	*Iterated Local Search*
IMB	:	Inserção Mais Barata
IMB-POP	:	Inserção Mais Barata com Princípio da Otimalidade Próxima
MPI	:	*Message Passing Interface*
MS	:	*Multi-Start*
PC	:	Particionamento de Conjunto
PIILS-SP	:	*Parallel Intensified Iterated Local Search with Set Partitioning*
POP	:	Princípio da Otimalidade Próxima
PPC	:	Problema de Particionamento de Conjunto
PRV	:	Problema de Roteamento de Veículos
PRVJT	:	Problema de Roteamento de Veículos com Janelas de Tempo
SA	:	*Simulated Annealing*
SP	:	*Set Partitioning*
TS	:	*Tabu Search*
VND	:	*Variable Neighborhood Descent*
VNS	:	*Variable Neighborhood Search*
VRP	:	*Vehicle Routing Problem*
VRPTW	:	*Vehicle Routing Problem with Time Windows*

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Algoritmos

Capítulo 1

Introdução

O Problema de Roteamento de Veículos (PRV) – referenciado na literatura inglesa como *Vehicle Routing Problem* (VRP) – é um dos problemas mais estudados na área de otimização combinatória. Neste problema, uma frota de veículos deve atender a um conjunto de clientes de forma que a soma das demandas dos clientes atendidos por um veículo k não ultrapasse a capacidade q_k deste veículo. O objetivo depende da aplicação. Os mais comuns são a minimização da distância total percorrida, do tempo de transporte, do número de veículos necessários e do custo total da operação.

Apesar do seu enunciado simples, este problema apresenta elevada complexidade computacional. Isto o torna bastante interessante como problema-teste de algoritmos de otimização. Diversos autores seguem esta ideia. Dantzig e Ramser [23], em 1959, foram os primeiros a formular o PRV na literatura científica, quando estudaram uma aplicação real sobre a distribuição de gasolina em estações de venda de combustíveis. Desde então, surgiram inúmeras variantes deste problema, cada uma especificando novas restrições com a finalidade de tratar diferentes cenários, cujas características centrais são:

Operação	coleta, entrega ou ambas
Tipo da Frota	homogênea/heterogênea
Tamanho da Frota	unitária ou vários veículos
Número de Depósitos	único ou vários depósitos
Natureza da Demanda	determinística/estocástica
Periodicidade	planejamento para um período temporal
Tempo de Serviço	tempo de serviço e janelas de tempo de clientes

Segundo Xu [76], os problemas de roteamento são basicamente focados no seguinte objetivo: "utilizar eficientemente uma frota de veículos para coletar e/ou entregar encomendas e mercadorias". Adicionalmente, Xu enfatiza as principais variantes como sendo: o Problema de Roteamento de Veículos Capacitado (PRVC), que considera uma frota uniforme de veículos de capacidade limitada, localizados inicialmente no mesmo depósito e sem limitação de tempo na entrega; o Problema de de Roteamento de Veículos com Coleta e Entrega (PRVCE), onde as encomendas dos clientes podem ser divididas em duas partes, a coleta em um local e a entrega em outro; e o Problema de Roteamento de Veículos com Janela de Tempo (PRVJT) – ou *VRP with Time Windows* –, que é uma generalização do PRVC, incluindo uma janela de tempo como intervalo obrigatório para começar o atendimento ao cliente.

No PRVJT, foco deste trabalho, uma frota de veículos deve partir do depósito, atender a demanda dos clientes e retornar ao depósito de forma que o custo total de viagem seja mínimo e o atendimento aconteça dentro do intervalo temporal especificado por cada cliente. Além disso, deve-se respeitar a capacidade dos veículos. Normalmente, considera-se permitido ao veículo chegar em um cliente antes do horário previsto. Porém, neste caso, é necessário esperar a "abertura" da janela de tempo para iniciar o serviço, isto é, carregar ou descarregar as mercadorias. Todo veículo tem um tempo de serviço, que somente depois de transcorrido poderá partir para outro cliente. Em alguns casos, costuma-se relaxar a janela de tempo, permitindo o início do serviço antes da abertura da janela ou após o fechamento da janela de tempo, somando-se um custo adicional por esta violação.

1.1 Motivação

Há duas motivações principais para o investimento em pesquisas que visam a proposição de novos algoritmos para a resolução do PRVJT: uma sob o ponto de vista operacional e outra relacionada a aspectos teóricos. No ponto de vista prático/operacional, os custos relacionados ao transporte de pessoas e/ou mercadorias geralmente são elevados, com tendência ao crescimento, motivado pela expansão atual das fronteiras comerciais de todo tipo de negócio [3]. Pesquisas sugerem que de 10% a 15% do valor final das mercadorias comercializadas no mundo correspondem ao custo de seu transporte [41]. Quanto aos aspectos teóricos, a versão de otimização do PRV e variantes, incluindo o PRVJT, pertence à classe NP-Difícil [45]. Desta forma, a resolução eficiente destes problemas corresponde a um desafio para os pesquisadores, que em sua maioria optam por adotar abordagens heurísticas para sua solução. Tal desafio é comprovado pela grande quantidade trabalhos

que tratam da resolução desses problemas.

Além disso, a resolução do PRVJT pela metaheurística *Iterated Local Search* (ILS) ainda é pouco explorada na literatura. Neste contexto, uma das metas ao propor mais uma heurística para resolver este problema foi desenvolver um ILS híbrido e mostrar que ele é competitivo frente às abordagens mais atuais. Em particular, frente as heurísticas baseadas em Busca Tabu, que tem até o momento vêm sendo as mais eficientes para o PRV e suas variantes.

1.2 Objetivos do trabalho

Este trabalho tem como objetivo geral desenvolver um algoritmo híbrido eficiente de otimização, baseado na metaheurística *Iterated Local Search*, para resolver o Problema de Roteamento de Veículos com Janelas de Tempo. Os objetivos específicos são:

1. Fazer uma revisão de literatura sobre o PRVJT e suas estratégias de resolução;
2. Estudar heurísticas construtivas e de refinamento, metaheurísticas e estratégias exatas para a resolução de problemas de otimização;
3. Desenvolver um algoritmo híbrido baseado na metaheurística *Iterated Local Search*, no método *Variable Neighborhood Descent* e um modelo exato de particionamento de conjunto para resolver o PRVJT;
4. Avaliar o desempenho do algoritmo desenvolvido com os resultados da literatura, tentando mostrar a competitividade de heurísticas híbridas como a proposta neste trabalho.

1.3 Organização do texto

O restante deste trabalho está organizado como segue. O Capítulo 2 apresenta o problema tratado neste trabalho, sua formulação e as ideias principais de alguns trabalhos presentes na literatura para sua resolução. No Capítulo 3, apresenta-se a metodologia híbrida proposta. Os resultados são reportados e discutidos no Capítulo 4. Por fim, no Capítulo 5 são apresentadas as conclusões e trabalhos futuros.

Capítulo 2

Problema de Roteamento de Veículos com Janelas de Tempo

2.1 Introdução

O Problema de Roteamento de Veículos com Janelas de Tempo (PRVJT) é um dos problemas mais importantes de otimização combinatória e mais estudados na literatura de pesquisa operacional. Neste problema, uma frota de veículos deve partir do depósito, atender a demanda dos consumidores e retornar ao depósito de forma que o custo total de viagem seja mínimo e o atendimento aconteça dentro do intervalo temporal especificado por cada consumidor. Além disso, deve-se respeitar a capacidade dos veículos.

São diversos os objetivos dos autores ao tratar o PRVJT. No presente trabalho, toma-se como objetivo a minimização da distância total percorrida que, segundo Oliveira e Vasconcelos [24], é o mais comum e mais encontrado na literatura. Backer e Furnon [6], Riise e Stølevik [63], Kilbi *et al.* [40], Ombuki *et al.* [53], Alvarenga *et al.* [4] e Oliveira e Vasconcelos [24], dentre outros, também adotam o mesmo objetivo. Outros autores consideram a geração de um conjunto mínimo de rotas como objetivo primário e a minimização da distância como um objetivo secundário. Os seguintes trabalhos seguem esta ideia: Bent e Hentenryck [9], Homberger e Gehring [33], Pisinger e Ropke [56], Lim e Zhang [46] e Pescott *et al.* [59].

2.2 Formulação matemática

O PRVJT pode ser definido num grafo completo orientado $G = (V, A)$ em que $V = \{0, \ldots, n+1\}$ é o conjunto de vértices e $A = \{(i, j) \mid i, j \in V\}$ é o conjunto de arcos. A cada arco (i, j) é associado um tempo t_{ij} e um custo de viagem c_{ij}.

Neste ponto faz-se necessária uma definição precisa do termo *custo de viagem*. Em casos práticos, o custo de viagem pode considerar diversos fatores tais como: distância, tempo, desgaste do veículo ao percorrer determinado caminho, entre outros fatores. Entretanto, quando se trata de problemas teóricos envolvendo janelas de tempo, é comum converter todas as medidas relevantes em unidades de tempo para fins de padronização e também para facilitar a comparação entre diferentes métodos. Isto posto, adota-se aqui a mesma definição de custo que a maioria dos trabalhos teóricos da literatura, isto é, o custo de viagem consiste na distância convertida em unidades de tempo.

Ao todo, um conjunto K de veículos idênticos com capacidade Q devem atender n clientes, representados pelos vértices $1, \cdots, n$. Considera-se que $N = V \setminus \{0, n+1\}$ representa o conjunto de clientes. Para atender tais clientes, os veículos devem partir do depósito e, após visitá-los, retornar ao mesmo local de onde partiram. Por questão de conveniência, o depósito é representado por dois vértices: o vértice 0, que representa a origem, e o vértice $n+1$, o destino. A cada cliente i, é associada uma demanda q_i que deve ser atendida por um único veículo. Além disso, todos os vértices possuem uma janela de tempo $[e_i, l_i]$, isto é, o serviço no vértice i deve ser iniciado dentro desse intervalo. Caso a chegada do veículo no cliente i ocorra antes do instante e_i, ele deve esperar a abertura da janela. O veículo não poderá chegar a i depois do instante l_i, pois isso faria violar a restrição de tempo do problema. Esse tipo de restrição é conhecido na literatura como janela de tempo rígida. A cada vértice é também associado um tempo de serviço, denotado por s_i. O objetivo é encontrar uma solução s de custo mínimo, isto é, minimizar a soma de todos os custos de viagem $\sum_{(i,j)\in s} c_{ij}$ que são associados aos arcos (i, j) presentes nas rotas que compõem essa solução.

A formulação matemática do PRVJT, adaptada de Cordeau *et al.* [21], é apresentada pelas expressões (2.1) a (2.9) a seguir. Nessa formulação, a variável binária x_{ijk} assume valor 1 se o veículo k passa pelo arco (i, j) e 0, caso contrário.

$$\text{Minimize} \sum_{k\in K} \sum_{i\in V} \sum_{j\in V} c_{ij} x_{ijk} \tag{2.1}$$

sujeito a:

$$\sum_{k\in K}\sum_{j\in V} x_{ijk} = 1, \forall i \in N \tag{2.2}$$

$$\sum_{j\in V} x_{0jk} = 1, \forall k \in K \tag{2.3}$$

$$\sum_{i\in V} x_{ijk} - \sum_{i\in V} x_{jik} = 0, \forall k \in K, \forall j \in N \tag{2.4}$$

$$\sum_{i\in V} x_{i(n+1)k} = 1, \forall k \in K \tag{2.5}$$

$$\sum_{i\in N} q_i \sum_{j\in V} x_{ijk} \leq Q, \forall k \in K \tag{2.6}$$

$$b_{ik} + s_i + t_{ij} - (1 - x_{ijk})M_{ij} \leq b_{jk}, \forall k \in K, \forall (i,j) \in A \tag{2.7}$$

$$e_i \leq b_{ik} \leq l_i, \forall k \in K, \forall i \in V \tag{2.8}$$

$$x_{ijk} \in \{0,1\}, \forall k \in K, \forall (i,j) \in A \tag{2.9}$$

A função objetivo (2.1) expressa o custo total a ser minimizado. As restrições (2.2) asseguram que somente um veículo k parte de cada cliente i. As restrições (2.3-2.5) garantem a continuidade do caminho a ser percorrido pelo veículo k, isto é, cada veículo parte do depósito, visita os clientes e, em seguida, retorna ao depósito. As restrições (2.6) fazem com que cada veículo k somente possa atender a um conjunto de clientes cuja demanda total não ultrapasse sua capacidade Q. As restrições (2.7)-(2.8) asseguram a viabilidade das rotas no que diz respeito às restrições de janelas de tempo, em que b_{ik} representa o tempo em que o veículo k começa a atender o cliente i e M_{ij} são constantes de valor suficientemente grande. Segundo os autores, essas constantes podem ser substituídas por $\max\{b_i + s_i + t_{ij} - e_j, 0\}$ $\forall (i,j) \in A$. As restrições (2.9) definem o domínio das variáveis de decisão.

2.3 Complexidade do problema

Encontrar a solução ótima do PRVJT implica em obter simultaneamente a solução de vários problemas NP-difíceis, dentre os quais citam-se: o Problema do Caixeiro Viajante (PCV) e o Problema da Mochila. Assim, tal tarefa é também NP-difícil. Encontrar uma simples solução viável para o PRVJT dispondo de um conjunto limitado de veículos é NP-difícil no sentido forte [42]. Porém, uma solução inicial viável é trivial caso o número de veículos seja ilimitado, bastando atender cada consumidor com um veículo.

Os resultados presentes na literatura referentes ao PRVJT comprovam que os algo-

ritmos exatos restringem-se à resolução problemas-teste de tamanho reduzido e janelas de tempo apertadas. Embora o tamanho possível hoje seja ligeiramente maior que o de alguns anos atrás – atualmente muitas instâncias de 100 consumidores já foram resolvidas na otimalidade – o crescimento da capacidade dos computadores e da eficiência dos algoritmos está muito distante da curva exponencial representada por este problema. Consequentemente, pode-se dizer que os métodos exatos não são uma alternativa viável para situações onde há um número maior de consumidores, como ocorre na maioria dos casos reais.

Abordagens heurísticas e algoritmos aproximativos também têm sido utilizadas na resolução do PRVJT. A resolução de problemas da classe NP-difícil também representa um desafio para tais abordagens. Heurísticas buscam obter uma boa solução em tempo hábil. Esta fato torna as estratégias heurísticas muito poderosas se comparadas com abordagens exatas, que focam exclusivamente na obtenção da solução ótima. Uma boa heurística deve ser capaz de encontrar soluções próximas da ótima, em tempo bem inferior ao necessário pelos métodos exatos; afinal esta é sua principal justificativa. Além disto, a qualidade da solução não deve variar demasiadamente ao aplicá-la em diferentes problemas-teste e também ao aplicá-la várias vezes ao mesmo problema-teste.

Os métodos aproximativos vêm ao encontro destas características almejadas. Um método aproximativo nada mais é que uma heurística com garantia de qualidade no resultado. A melhor solução encontrada por um algoritmo de aproximação está sempre a uma distância percentual previamente definida do ótimo desconhecido. A "distância do ótimo" é particular de cada algoritmo, podendo até mesmo não ser muito relevante em termos práticos. Um exemplo bem conhecido é o algoritmo PRIM, para árvore geradora mínima, que é capaz de oferecer um limite superior (solução viável) para o PCV, que é no máximo duas vezes o ótimo em distância total percorrida [2].

2.4 Abordagens exatas

O estado da arte dos algoritmos exatos é apresentado por Chabrier [16], Irnich e Villeneuve [36], Jepsen *et al.* [37, 38] e Kallehauge *et al.* [39]. Segundo [37], até 2006, 45 dos 56 problemas-teste de Solomon foram resolvidos na otimalidade [37]. Os autores, destacam que, em alguns casos, são gastos mais que cinco horas de processamento na resolução de algumas instâncias, enquanto outras podem ser resolvidas em menos de um minuto.

Dada a complexidade do problema, sua resolução por abordagens puramente exatas é,

em muitos casos, uma tarefa extremamente árdua, pois demanda um tempo computacional muito elevado. Este fato motiva o desenvolvimento de novos algoritmos heurísticos para a resolução do PRVJT. Vale observar que tais algoritmos não garantem a obtenção da solução ótima, porém são mais rápidos e geralmente fornecem soluções próximas da ótima.

2.5 Abordagens heurísticas

No contexto de otimização, heurísticas são procedimentos de busca que visam a obtenção de soluções de qualidade satisfatória em um tempo computacional aceitável. Tais procedimentos, no entanto, não garantem encontrar a solução ótima nem são capazes de mensurar o quão próxima a solução obtida está da ótima. Entretanto, a utilização de algoritmos heurísticos para a resolução do PRVJT tem se tornado cada vez mais atraente. As próximas seções apresentam as ideias centrais de algumas heurísticas construtivas e de refinamento disponíveis na literatura.

2.5.1 Heurísticas de construção de rotas

O primeiro trabalho sobre construção heurística de rotas para o PRVJT foi o de Baker e Schaffer [8], em 1989. Trata-se de uma adaptação da heurística das economias de Clarke e Wright [20], proposta para construir soluções para o PRV. Inicialmente, o algoritmo cria uma rota partindo do depósito para cada cliente i. Em seguida, são efetuadas uma série de iterações. Em cada iteração o algoritmo calcula quais as duas rotas que podem ser combinadas de forma a gerar a maior economia possível.

Uma heurística semelhante baseada no algoritmo das economias foi desenvolvida por Solomon [69] para resolver o PRVJT. Dada a existência de janelas de tempo, deve-se neste caso considerar a orientação da rota. Além disto, deve-se checar a violação das janelas de tempo quando duas rotas são combinadas. Tanto a heurística das economias quanto sua adaptação para o PRVJT possuem complexidade $O(n^2 \log n^2)$.

No trabalho de Landeghem [43], também é proposta uma heurística baseada na das economias. Trata-se de uma heurística de dois critérios, onde as janelas de tempo são utilizadas para mensurar o quanto uma ligação entre dois clientes é boa em termos de tempo.

Solomon [69] também descreve uma heurística de melhor vizinho orientada por tempo. Nesta heurística, toda rota é inicializada encontrando-se o cliente mais próximo ao depó-

sito que ainda não pertença a nenhuma rota. A relação de proximidade considera aspectos tanto geográficos quanto temporais dos clientes em questão. A cada iteração subsequente o cliente mais próximo ao último adicionado à rota é considerado para inserção ao final da rota que está sendo gerada. Quando a busca falha, uma nova rota é inicializada.

De forma geral, as heurísticas de Solomon [69] e Landeghem [43] retornam soluções rapidamente. Entretanto, estas soluções são geralmente de baixa qualidade. Na maioria dos casos estão a mais de 10% da ótima [27].

O problema de se criar uma rota por vez é que usualmente as rotas geradas por último são de baixa qualidade uma vez que os clientes sem rota tendem estar distantes geograficamente [27]. No trabalho de Potvin e Russeau [57], pode-se encontrar uma tentativa de sanar este problema de inserção por meio da construção simultânea de várias rotas. A inicialização das rotas é feita usando a heurística de inserção de Solomon [69]. Em cada rota o cliente mais distante do depósito é selecionado como semente. A partir de então, computa-se a melhor inserção viável para cada cliente ainda não visitado. Este método é melhor que as heurísticas de Solomon, porém as soluções geradas continuam longe das ótimas. Em Russell [65], são apresentadas melhorias nas abordagens de inserção.

Antes e Derigs [5] vão além das ideias clássicas de inserção. Nesse trabalho, todo cliente sem rota designada recebe um custo de inserção de cada uma das rotas. A definição desse custo é semelhante ao adotado nas heurísticas de Solomon. Cada cliente sem rota envia uma proposta à rota com a melhor oferta e cada rota aceita a melhor proposta dos clientes com menor número de alternativas. Vale observar que mais clientes podem ser inseridos em cada iteração. Se houver alguma violação nas rotas, um certo número de veículos é removido e o processo é inicializado novamente. Os resultados dos autores em questão são comparados àqueles apresentados por Potvin e Robillard [57]. De forma geral, segundo os autores, construir rotas paralelamente produz soluções de maior qualidade que construir rotas uma a uma.

Para maiores informações sobre algoritmos de construção de rotas, sugere-se ao leitor o trabalho de Braysy e Gendreau [14], onde os autores descrevem as características básicas de vários métodos disponíveis na literatura e os resultados obtidos são apresentados e analisados. No trabalho em questão, eles discutem tanto heurísticas de construção de rotas quanto algoritmos de busca local.

2.5.2 Heurísticas de aprimoramento de rotas

A base de quase todas as heurísticas de melhoria de rotas é a noção de vizinhança. A vizinhança de uma solução s é um conjunto de soluções $N(s)$ que pode ser geradas pela aplicação de uma única alteração – denominada *movimento* – na solução s.

Checar algumas ou todas as soluções de uma vizinhança pode revelar soluções melhores em relação a uma determinada função objetivo. Esta ideia pode ser repetida partindo-se da melhor solução obtida até o momento. Se em algum momento nenhuma solução melhor for encontrada em uma vizinhança, um *ótimo local* foi obtido. Trata-se definitivamente de um ótimo local, porém este pode eventualmente ser um ótimo global. A este algoritmo dá-se o nome de *busca local*. Para maiores informações sobre algoritmos de aprimoramento de rotas, sugere-se ao leitor também o trabalho de Braysy e Gendreau [14].

Na próxima seção serão introduzidas várias estruturas de vizinhança empregadas na literatura para melhorar soluções do PRVJT. Em seguida serão descritos alguns dos algoritmos que a utilizam.

2.5.3 Estruturas de vizinhança

Uma das vizinhanças mais utilizadas em roteamento é a *k-opt*, onde k arcos são removidos e substituídos por outros k arcos. Um ótimo local obtido utilizando-se a vizinhança *k-opt* é dita solução *k-optimal*. Normalmente, k é no máximo 3.

Para todas as possíveis trocas *2-opt* e algumas das permutações da vizinhança *3-opt*, partes da rota é invertida. Isto comumente acarreta violações nas janelas de tempo. No trabalho de Potvin e Robillard [58], são apresentadas duas variantes, a *2-opt** e a *Or-opt*, que mantêm a direção da rota.

Na vizinhança *Or-opt*, um conjunto contíguo de até 3 clientes é realocado para outra posição na mesma rota. Uma vez que nessa vizinhança três arcos são trocados por outros três, é fácil observar que ela é um subconjunto da vizinhança *3-opt*. Desta forma, o tamanho da vizinhança é reduzido de $O(n^3)$ para $O(n^2)$. De forma geral, o tamanho da vizinhança *k-opt* é da ordem de $O(n^k)$. A vizinhança *2-opt** consiste na troca de um segmento de uma rota por um segmento de outra rota. Esses operadores de vizinhança são muitas vezes denotados na literatura por *crossover* ou simplesmente *cross*.

Movimentos da vizinhança *exchange* alteram diferentes rotas através da troca simultânea de dois clientes. A vizinhança *k-node*, proposta no trabalho de Christofides

e Beasley [19], tem sido adaptada por alguns autores para que esta leve em consideração aspectos referentes às janelas de tempo. Nesta estrutura, cada cliente i é considerado e os conjuntos M_1 e M_2 são identificados. Em M_1 são alocados o cliente i e seu sucessor j. O conjunto M_2 é formado pelos clientes mais próximos aos clientes i e j que não estejam na mesma rota que i e j (encontrados pela minimização do custo de inserção considerando distância euclidiana). A vizinhança é então definida pela remoção dos elementos desses conjuntos e posterior inserção em qualquer outra possível localização. Como trata-se de uma vizinhança de dimensões muito elevadas, apenas os k candidatos mais promissores são considerados.

Outra vizinhança explorada na literatura é a λ-*interchange*, desenvolvida em Osman [55] originalmente para o PRV. Trata-se de uma generalização do operador *relocate*. Nesta estrutura, um subconjunto de clientes de uma mesma rota é trocado por outro conjunto de outra rota. O mecanismo de geração λ-*exchange* pode ser descrito como segue. Dada uma solução para o problema, representada pelo conjunto de rotas $S = \{r_1, ..., r_p, ..., r_q, ..., r_k\}$, um λ-*interchange* entre um par de rotas (r_p, r_q) consiste na troca dos clientes $S_1 \cup r_p$ de tamanho $|S_1| \leq \lambda$ por outro subconjunto $S_2 \cup r_q$ de tamanho $|S_2| \leq \lambda$ para gerar novas rotas $r_p^* = (r_p - S_1) \cup S_2$, $r_q^* = (r_q - S_2) \cup S_1$ e uma nova solução $S' = \{r_1, ..., r_p^*, ..., r_q^*, ..., r_k\}$. A vizinhança $N_\lambda(S)$ de uma dada solução S é o conjunto de todos os vizinhos S' gerados para um dado valor de λ.

Finalmente, a vizinhança denotada por *shift-sequence* é proposta em Schulze e Fahle [67]. Nesta, um cliente é movido de uma rota para outra checando-se todas as possibilidades de inserção. Caso uma inserção possa se tornar viável pela remoção de outro cliente j, este é removido e inserido em outra rota. Este procedimento é repetido até que a viabilidade seja restabelecida.

A Tabela 2.1 apresenta uma breve descrição das estruturas de vizinhança comumente utilizadas na literatura por algoritmos de busca local para a resolução do PRVJT. Observa-se que algumas dessas são também utilizadas neste trabalho.

Tabela 2.1: Estruturas de vizinhança para o PRVJT

Vizinhança	Descrição
Relocate	Move um cliente de uma rota para outra.
Exchange	Troca dois clientes entre duas rotas.
*2-opt**	Troca um segmento de uma rota por um segmento de outra rota.
Or-opt	Um segmento contínuo de clientes é movido de uma posição em uma rota para outra posição da mesma rota.
k-node	Os clientes i, seu sucessor j e os dois clientes mais próximos que não estão na mesma rota são removidos. Tenta-se então inserir os quatro vértices em todas as possíveis localizações. Como trata-se de uma vizinhança de dimensões muito elevadas, apenas os k candidatos mais promissores são considerados.
λ-*interchange*	Um subconjunto S_1 de clientes de tamanho $\|S_1\| \leq \lambda$ de uma rota é trocado por um subconjunto S_2 de tamanho $\|S_2\| \leq \lambda$ de outra rota.
Shift-sequence	Um cliente é movido de uma rota para outra checando-se todas as possibilidades de inserção. Caso uma inserção se torne viável pela remoção de um consumidor j, este é removido e inserido em alguma outra rota. Este procedimento é repetido até que a viabilidade seja reestabelecida.

2.6 Metaheurísticas

Metaheurísticas são procedimentos destinados a encontrar uma boa solução, eventualmente a ótima, consistindo na aplicação, em cada passo, de uma heurística subordinada, a qual tem que ser modelada para cada problema específico [62]. Contrariamente às heurísticas convencionais, as metaheurísticas são de caráter geral e providas de mecanismos para tentar escapar de ótimos locais ainda distantes dos ótimos globais. As metaheurísticas diferenciam-se entre si pelo mecanismo usado para escapar dos ótimos locais [70].

O estado da arte das abordagens baseadas em metaheurísticas para resolver o PRVJT consiste em Estratégias Evolutivas [50, 33], Busca em Grandes Vizinhanças [9, 56, 59], *Simulated Annealing* [17, 24], Busca Tabu [13], *Iterated Local Search* [35, 34] e *Multi-start Local Search* [35, 46].

Thamgiah [74] desenvolveu um algoritmo *Simulated Annealing* com uma função de probabilidade não-monótona, utilizando a vizinhança λ-*interchange* com $k = 2$ para explorar o espaço de soluções. No método SA original, a temperatura é decrescida após cada iteração. Porém, neste trabalho, caso toda a vizinhança tenha sido explorada e nenhum movimento tenha sido aceito, a temperatura é incrementada – esta estratégia recebe o nome de *reset*. A temperatura é acrescida até o máximo entre a temperatura onde se

obteve a melhor solução e metade da temperatura do último *reset*. Após R *resets* sem melhora, o algoritmo termina. A qualidade das soluções obtidas em [74] foi semelhante à obtida em [17].

Chiang e Russell [17] desenvolveram três métodos baseados em *Simulated Annealing*: (i) com uma versão modificada da vizinhança *k-node*; (ii) usando a vizinhança λ-*interchange* como proposta em [55], com $k = 1$; e (iii) adotando o conceito de lista tabu da metaheurística Busca Tabu. Nesta última, o método utiliza uma lista tabu de movimentos proibidos da vizinhança λ-*interchange*. Destes três métodos, o primeiro é o que apresenta convergência mais demorada; entretanto, seus resultados são um pouco melhores que o dos demais. A distância total percorrida obtida pelos métodos propostos apresentaram-se entre 7% e $11,5\%$ acima da ótima.

Rouchat e Taillard [64] propuseram uma abordagem Busca Tabu baseada em uma então chamada "memória adaptativa" e uma busca local usando a vizinhança *2-opt*. Nesta abordagem a memória adaptativa consiste em um conjunto elite de rotas tomadas a partir das melhores soluções visitadas durante a busca. O objetivo disto é prover novas soluções iniciais para a Busca Tabu a partir das rotas extraídas da memória. Na primeira fase, a Busca Tabu é usada para criar uma certa quantidade de soluções diversificadas, armazenando-as na memória adaptativa. A seleção de rotas da memória é feita probabilisticamente e a probabilidade de se selecionar uma rota particular depende do valor de avaliação da solução a qual esta pertence. As rotas selecionadas são então aprimoradas pela Busca Tabu e inseridas novamente na memória adaptativa. Ao final, resolve-se de forma exata o problema de particionamento de conjunto usando as rotas elite agrupando-as da melhor forma na tentativa de gerar a melhor solução possível.

Chiang e Russell [18] propuseram uma Busca Tabu reativa que dinamicamente varia o tamanho da lista de movimentos proibidos na tentativa de evitar ciclagem. Mais precisamente, isto é feito pelo aumento no tamanho da lista tabu caso soluções idênticas ocorram frequentemente e reduzindo o tamanho da lista caso nenhuma solução viável puder ser encontrada. A busca local é baseada no mecanismo λ-*interchange* proposto em [55].

Taillard *et al.* [73] propuseram uma heurística Busca Tabu para o problema de roteamento de veículos com janelas de tempo flexíveis. Neste problema, o atraso no cliente é apenas uma inconveniência (a qual é penalizada na função objetivo) e não uma inviabilidade como no problema original. Tal problema pode ser resolvido de maneira satisfatória ao se estabelecer penalidades elevadas. Os autores propuseram uma nova estrutura de vizinhança chamada *Cross-exchange*, que pode ser usada tanto para trocar clientes de

rotas distintas quanto da mesma rota.

Cordeau *et al.* [22] introduziram uma Busca Tabu simples para duas variantes do PRVJT, o periódico e o com múltiplos depósitos. Uma característica importante desta abordagem foi a possibilidade de explorar soluções inviáveis durante a busca. As violações das restrições são penalizadas na função de custo e os parâmetros que penalizam cada violação (restrições de carga, duração e janelas de tempo) são ajustadas dinamicamente. O único operador de busca local utilizado foi o de inserção, o qual insere um cliente em um tempo em outra localidade. A melhor solução identificada após n iterações entra em processo de pós-otimização por meio da aplicação individual em cada rota de uma heurística especializada para o Problema do Caixeiro Viajante com Janelas de Tempo [30].

Gehring e Hoberger [28] estudaram uma abordagem de duas fases, onde a Busca Tabu é combinada com o algoritmo evolutivo ES_1 de [32]. Na parte evolutiva do algoritmo, a busca é principalmente guiada pela mutação baseada nas vizinhanças *Or-opt* [54], *2-opt** [58] e λ-*interchange* com $\lambda = 1$. Adicionalmente, um operador baseado em Or-opt é usado para minimizar o número de rotas. Os indivíduos da população inicial são gerados por meio de uma abordagem estocástica baseada no algoritmo das economias de Clarke e Wright [20]. O algoritmo evolucionário é usado na primeira fase com o intuito de minimizar o número de rotas. Na segunda fase, a distância total é minimizada utilizando um algoritmo de Busca Tabu com os mesmos operadores de vizinhança que na fase anterior.

Braysy [12] apresenta uma revisão sobre a resolução de problemas de roteamento de veículos com janelas de tempo por metodologias baseadas em Busca Tabu. Nesse trabalho, são descritas as características básicas de vários métodos disponíveis na literatura e resultados para os problemas de Solomon são discutidos e analisados. Há uma grande quantidade de autores que atuam neste sentido. Porém, segundo Braysy os algoritmos mais eficientes até aquela data foram os desenvolvidos em Chiang e Russell [18], Taillard *et al.* [73], Cordeau *et al.* [22] e Gehring e Homberger [28].

De Backer e Furnon [6] investigaram técnicas de melhora iterativa por meio de programação por restrições acopladas às metaheurísticas Busca Tabu e *Guided Local Search*. A programação por restrições foi usada para acelerar a verificação da validade das soluções encontradas durante a busca.

Riise e Stlevik [63] fizeram um estudo do *Guided Local Search* e do *Fast Local Search* combinados com movimentos simples de realocação de tarefas.

Kilby *et al.* [40] introduziram um *Guided Local Search* determinístico que usa operadores de busca local *2-opt*, *relocate*, *exchange* e *2-opt**, com uma estratégia de melhor aceitação.

Ombuki *et al.* [53] descreveram uma outra abordagem para o problema em questão. Segundo eles, o PRVJT pode ser tratado como um problema multiobjetivo, ponderando a importância de se reduzir o número total de veículos e a distância total percorrida.

Alvarenga *et al.* [4] desenvolveram um Algoritmo Genético (AG) seguido de uma estratégia exata formulada como um problema de particionamento de conjunto (PPC). Nessa abordagem, cada coluna representa uma rota viável, candidata a pertencer à solução, enquanto que as linhas representam os consumidores a serem atendidos por uma dada rota. As rotas das soluções geradas pelo AG foram combinadas pela resolução do PPC correspondente, encontrando, assim, a melhor combinação das rotas sem violar as restrições do PRVJT.

Oliveira e Vasconcelos [24] desenvolveram um algoritmo que associa a metaheurística *Simulated Annealing* com controle não-monótono de temperatura à heurística *Hill-Climbing* e um reinício aleatório. Os resultados foram comparados aos melhores resultados publicados na literatura para os 56 problemas de Solomon [69]. Segundo os autores, esse trabalho mostrou como estratégias estocásticas podem ser usadas para melhorar o desempenho dos métodos.

Capítulo 3

Metodologia proposta

Neste trabalho é proposto um algoritmo híbrido combinando conceitos da metaheurística *Iterated Local Search*, do método *Variable Neighborhood Descent* e de um modelo exato de particionamento de conjunto que, periodicamente, combina da melhor forma as rotas geradas ao longo do algoritmo.

3.1 Representação de uma solução

Uma solução do PRVJT é representada como uma permutação de clientes, numerados de 1 a n e separadas em k partições, sendo k o número de rotas ou veículos utilizados. Por exemplo, se existem 19 clientes a serem atendidos e 3 veículos disponíveis, então uma possível solução é [0 7 14 15 4 13 12 0], [0 1 6 16 11 5 18 17 3 0] e [0 10 8 19 9 2 0]. A rota [0 10 8 19 9 2 0] indica que o veículo sai do depósito, visita os clientes 10, 8, 19, 9 e 2 nesta ordem e retorna ao depósito.

Para facilitar a visualização e o entendimento do trabalho, as soluções neste trabalho são representadas graficamente, conforme exemplificada na Figura 3.1.

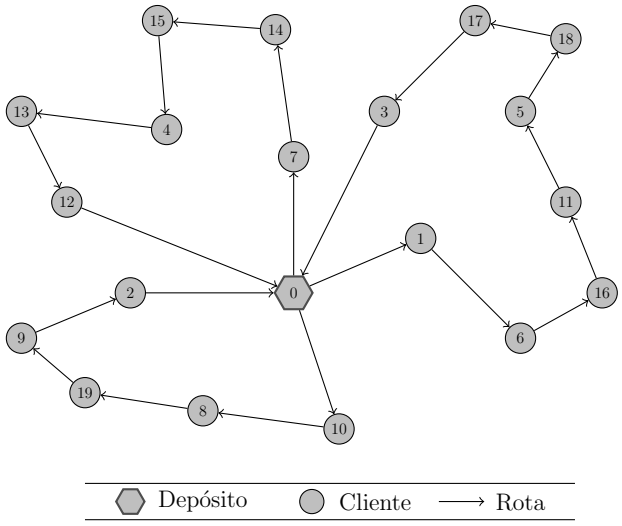


Figura 3.1: Exemplo de uma solução do PRVJT

3.2 Função de avaliação

Em alguns momentos no algoritmo desenvolvido, torna-se difícil gerar soluções viáveis. Assim, faz sentido permitir que o método caminhe por regiões inviáveis do espaço de busca. De fato, esta é uma estratégia muito adotada em metaheurísticas e, assim, penalidades são adicionadas para guiar a busca às soluções viáveis, como em: Gendreau *et al.* [29] e Toth e Vigo [75] para o problema de roteamento de veículos capacitado e Badeau *et al.* [7], Berger e Barkoui [10], Bouthillier *et al.* [11] e Ibaraki *et al.* [35, 34], entre outros, para resolver o PRVJT.

Uma solução s é avaliada pela função f, dada pela equação (3.1), a qual deve ser minimizada:

$$f(s) = \sum_{r \in s} g(r) = \sum_{r \in s} (c(r) + w_l.l(r) + w_e.e(r)) \tag{3.1}$$

em que: g é a função de avaliação da rota r; $c(r)$ é o custo de deslocamento da rota r; $l(r)$ corresponde ao tempo de atraso em r; $e(r)$ é o excesso de carga na solução r; w_l e w_e são penalidades por unidade de atraso e excesso de carga, respectivamente (os quais foram fixados de forma empírica em $w_l = 200$ e $w_e = 300$). Observa-se que caso s seja viável, o valor reportado por f corresponde apenas ao custo total de deslocamento, pois, neste caso, $l(s) = e(s) = 0$.

3.3 Construção de uma solução inicial

Para obter uma solução inicial para o PRVJT foi desenvolvido um método de inserção mais barata que explora o Princípio da Otimalidade Próxima [60]. Segundo este princípio, em uma sequência ótima de escolhas, cada subsequência deve também ser ótima. Vale observar que, apesar de este princípio tratar de casos ótimos, no algoritmo desenvolvido não há nenhuma garantia da obtenção de soluções ótimas ou mesmo de partes ótimas de soluções. Neste caso, tal princípio é utilizado apenas com o intuito de obter soluções iniciais de maior qualidade. O pseudocódigo do método construtivo é apresentado pelo Algoritmo 1.

Algoritmo 1 Inserção Mais Barata com Princípio da Otimalidade – IMB-POP

```
1: Seja s_0 uma solução com |K| rotas vazias
2: para cada cliente c ∈ {1,...,n} faça
3:    melhor_custo ← ∞
4:    para cada rota r ∈ s_0 faça
5:       para cada posição p da rota r em que se possa inserir o cliente c faça
6:          custo ← custo da inserção do cliente c na posição p da rota r
7:          se custo < melhor_custo então
8:             melhor_r ← r ;
9:             melhor_p ← p ;
10:            melhor_custo ← custo
11:         fim se
12:      fim para
13:   fim para
14:   Adicione o cliente c na posição melhor_p da rota melhor_r da solução s_0
15:   se (c mod ⌈n/5⌉) = 0 então
16:      Refine a solução parcial incompleta s_0 por uma heurística h
17:   fim se
18: fim para
19: retorne s_0
```

Seja $|K|$ o número de veículos disponíveis para utilização. Inicialmente, o algoritmo construtivo cria $|K|$ rotas vazias (linha 1) e uma lista de candidatos a serem inseridos no conjunto de rotas (linha 2). Observe que, inicialmente, a lista de candidatos deve conter todos os clientes que devem ser atendidos pela frota. A ideia do algoritmo é inserir, passo a passo, cada cliente da lista de candidatos à sua melhor localização no momento (linha 14), efetuando periodicamente uma busca local na solução que está sendo construída (linha 16). O algoritmo de construção termina quando a lista de candidatos se torna vazia. Mais especificamente, os parâmetros da linha 15 foram dimensionados de forma que ocorram cinco buscas locais ao longo da construção. Por exemplo, caso haja um total de 100

clientes, a busca local é efetuada a cada vinte clientes adicionados à solução. Neste caso, a busca local é feita pelo algoritmo RVND (*vide* descrição na Seção 3.5.2).

3.4 Estruturas de vizinhança

Para explorar o espaço de soluções do problema, aplicam-se, neste trabalho, dez diferentes estruturas de vizinhança, as quais são apresentadas a seguir. É importante destacar que são permitidos movimentos que conduzam a soluções inviáveis.

3.4.1 *Shift*

A estrutura de vizinhança *Shift* gera movimentos de realocação. Um movimento desta estrutura consiste na transferência um cliente i de uma rota para outra. A Figura 3.2 ilustra um exemplo em que o cliente 6 é transferido da Rota 2 à Rota 3.

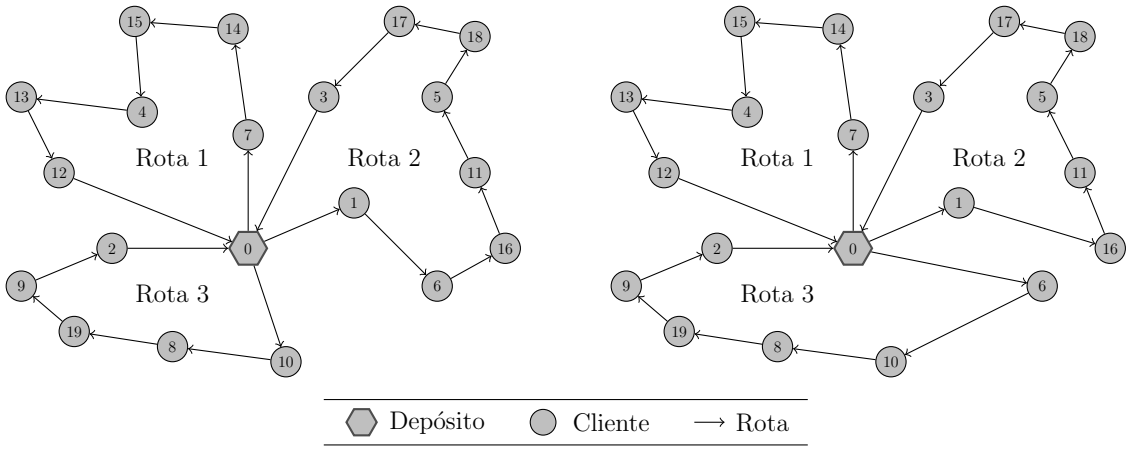


Figura 3.2: Exemplo do movimento *Shift*

3.4.2 *Shift*(2,0)

A estrutura *Shift*(2,0) gera movimentos semelhantes aos da estrutura *Shift*, porém realocando-se dois clientes consecutivos de uma rota para outra. A Figura 3.3 exemplifica a realocação dos clientes 1 e 6 da Rota 2 para a Rota 3.

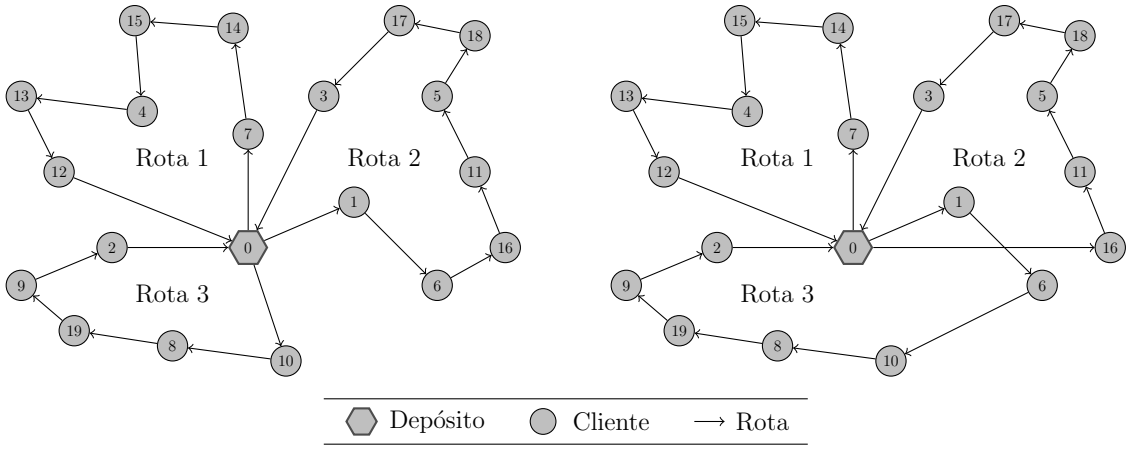


Figura 3.3: Exemplo do movimento *Shift*(2,0)

3.4.3 *Shift*(3,0)

Shift(3,0) também é semelhante à estrutura de vizinhança *Shift*. Nesta estrutura, três clientes consecutivos são realocados de uma rota para outra. A Figura 3.4 exemplifica a realocação dos clientes 1, 6 e 16 da Rota 2 para a Rota 3.

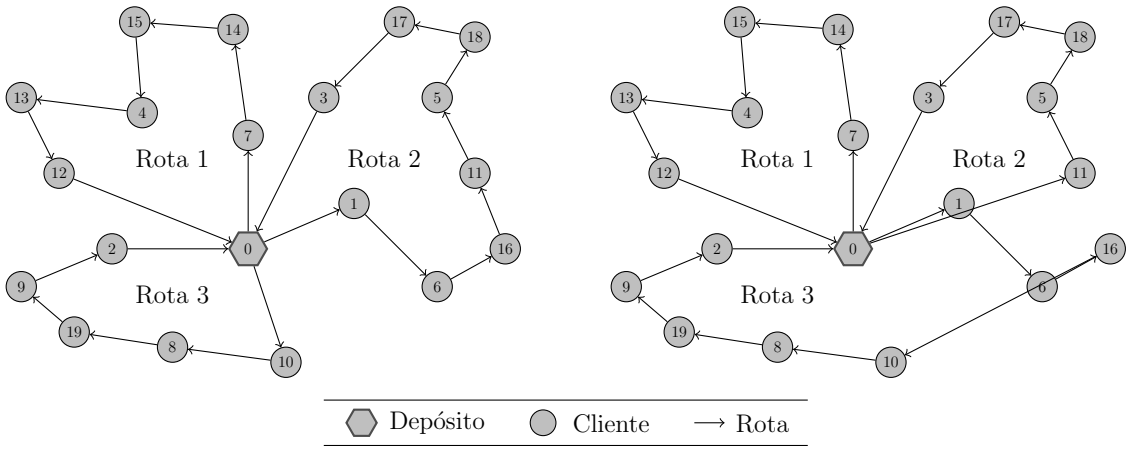


Figura 3.4: Exemplo do movimento *Shift*(3,0)

3.4.4 *Swap*

Um movimento da estrutura de vizinhança *Swap* consiste em trocar um cliente i de uma rota r_p com um outro cliente j de uma rota r_q. A Figura 3.5 mostra a aplicação do movimento *Swap* para os clientes 1 e 10 das rotas 2 e 3, respectivamente.

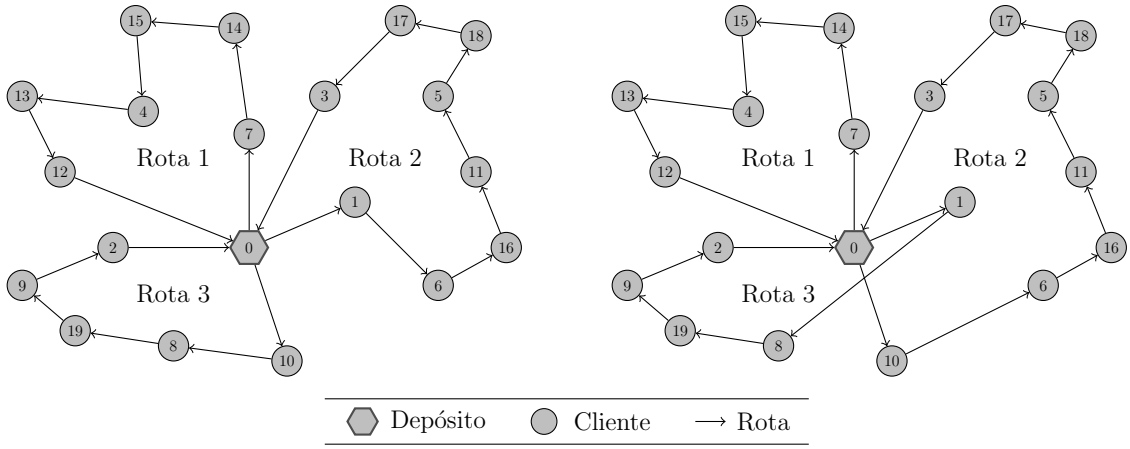


Figura 3.5: Exemplo do movimento *Swap*

3.4.5 *Swap*(2,1)

A estrutura de vizinhança *Swap*(2,1) é análoga à *Swap*. Porém, neste caso, trocam-se dois clientes consecutivos de uma rota por um cliente de outra rota. A Figura 3.6 mostra a aplicação do movimento *Swap*(2,1) considerando os clientes 9 e 2 da Rota 3 e o cliente 12 da Rota 1.

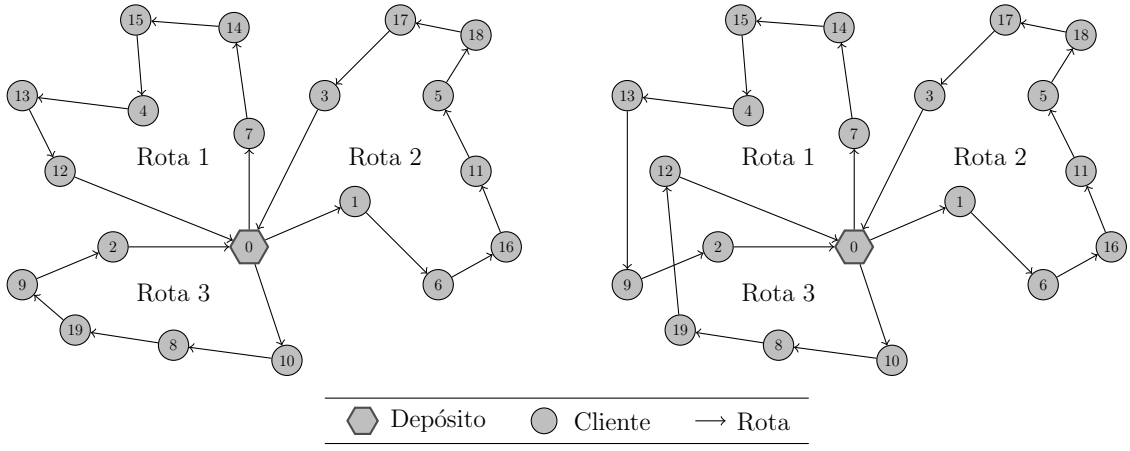


Figura 3.6: Exemplo do movimento *Swap*(2,1)

3.4.6 *Swap*(2,2)

Um movimento da estrutura *Swap*(2,2) também é análoga à *Swap*. Porém, trocando-se dois clientes consecutivos de uma rota com dois clientes de outra rota. A aplicação do movimento *Swap*(2,2) para os clientes 13 e 12 da Rota 1 e 9 e 2 da Rota 3 é ilustrado na Figura 3.7.

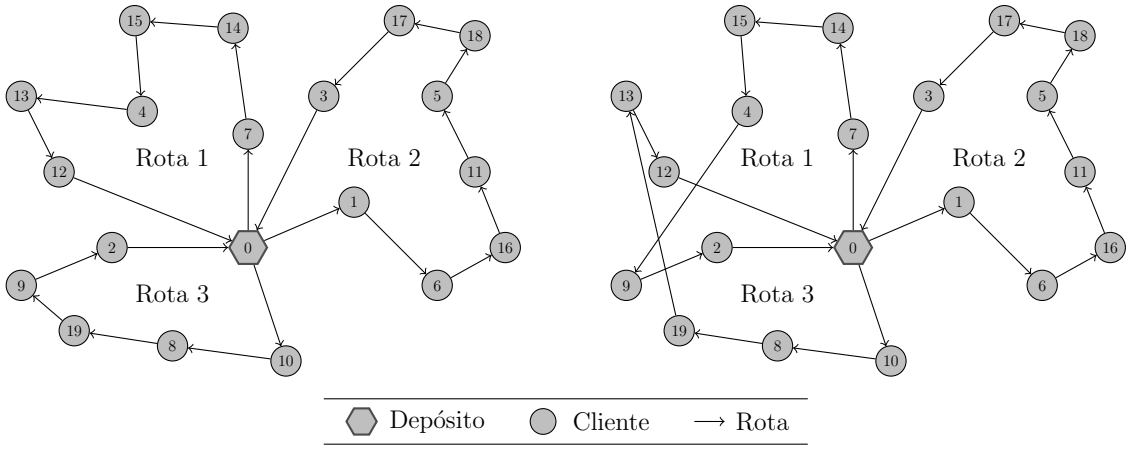


Figura 3.7: Exemplo do movimento *Swap*(2,2)

3.4.7 *Swap-intra*

Um movimento da estrutura de vizinhança *Swap-intra* consiste na troca da ordem de atentimento de dois clientes presentes em uma mesma rota. A Figura 3.8 mostra a troca dos clientes 1 e 3.

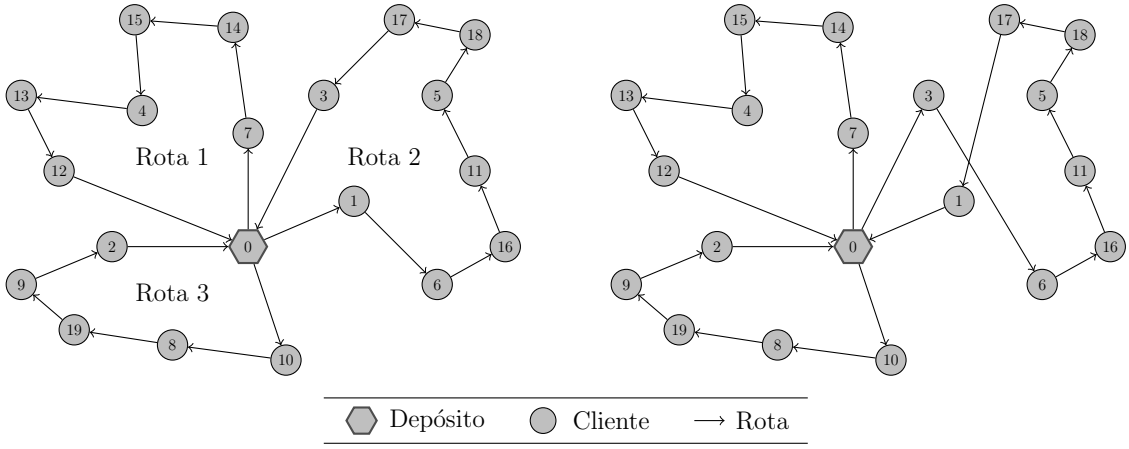


Figura 3.8: Exemplo do movimento *Swap-intra*

3.4.8 *1Or-opt, 2Or-opt e 3Or-opt*

Um movimento da estrutura *kOr-opt* consiste em remover k clientes consecutivos de uma rota r e, em seguida, reinseri-los em uma outra posição nessa mesma rota. Neste trabalho, consideram-se três estruturas de vizinhança baseadas na *kOr-opt*: a *1Or-opt*, a *2Or-opt* e a *3Or-opt*. A Figura 3.9 ilustra a aplicação do movimento *kOr-opt* ($k = 3$), considerando a reinserção dos clientes 18, 17 e 3 da Rota 1 na posição do arco $(11, 5)$.

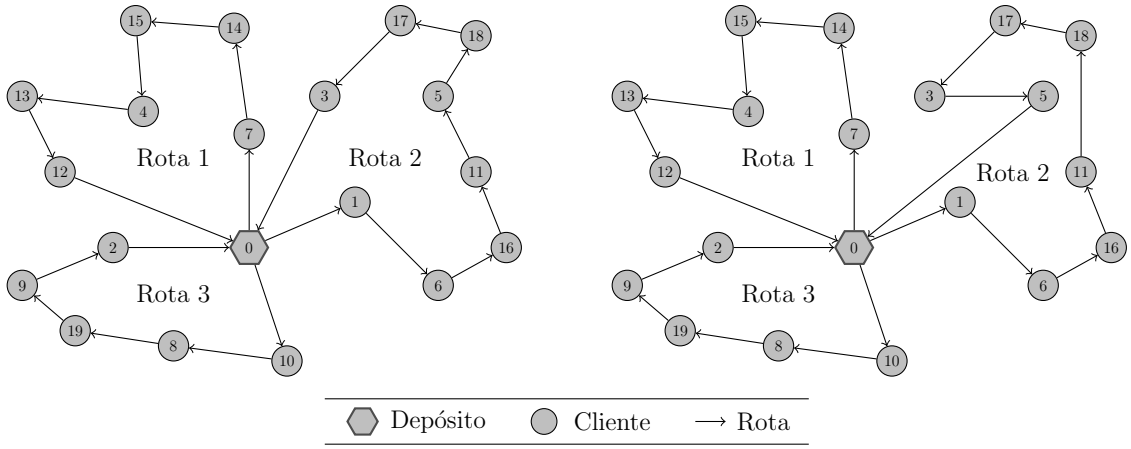


Figura 3.9: Exemplo do movimento *kOrOpt*

3.5 O algoritmo proposto

O algoritmo proposto, denotado por IILS-SP, consiste na construção de uma solução inicial pelo procedimento apresentado na Seção 3.3, seguida de um refinamento que combina versões adaptadas dos métodos *Iterated Local Search* (ILS) e *Variable Neighborhood Descent* (VND) com uma abordagem exata baseada na formulação matemática do problema de particionamento de conjunto (*Set Partition* - SP). O pseudocódigo do algoritmo proposto é apresentado no Algoritmo 2, onde: s_0 é uma solução inicial; s^* é a melhor solução obtida durante a execução do procedimento; s' é a solução perturbada e s'' é um ótimo local obtido pela aplicação da busca local RVND à solução perturbada.

Algoritmo 2 *Intensified Iterated Local Search* – IILS-SP

```
1: s_0 ← IMB-POP()
2: s* ← RVND(s_0)
3: enquanto critério de parada satisfeito faça
4:    s' ← Perturbação(s*, histórico)
5:    s'' ← RVND(s')
6:    se MomentoApropriado(histórico) então
7:       s'' ← Intensificação (s'')
8:    fim se
9:    s* ← CritérioAceitação(s'', s*, histórico)
10: fim enquanto
11: retorne s*
```

As seções seguintes detalham cada parte desse algoritmo.

3.5.1 *Iterated Local Search*

Intensified Iterated Local Search é uma extensão da metaheurística *Iterated Local Search* – ILS [47]. ILS explora o espaço de soluções por meio da aplicação de perturbações à solução ótima local corrente. Em linhas gerais, tal metaheurística parte de uma solução inicial s_0 e a esta aplica uma busca local, obtendo s^*. Posteriormente, o método efetua iterativamente os seguintes passos: (i) perturbação na solução corrente s^*, obtendo uma solução s' e (ii) busca local em s', obtendo uma solução s'', que é um ótimo local. Caso s'' seja melhor que a solução corrente s^* (isto é, melhora a melhor solução obtida até o momento), o método torna s'' a nova solução corrente. Caso contrário, o método efetua uma nova iteração. Este procedimento é repetido até que um critério de parada seja satisfeito.

É importante salientar que o sucesso do ILS depende fortemente das perturbações realizadas. Portanto, a perturbação aplicada a uma dada solução deve ser dosada de maneira tal que as alterações decorrentes de sua aplicação sejam suficientes para se escapar de ótimos locais e explorar diferentes regiões sem, contudo, causar um caos absoluto que implicaria na perda de características do ótimo local corrente.

Na adaptação feita, as perturbações realizadas (linha 4 do Algoritmo 2) consistem na aplicação de *nível*+2 movimentos aleatoriamente escolhidos na vizinhança *Shift*, apresentada na Seção 3.4, onde *nível* $\in \{0, 1, 2, \dots\}$ representa o nível de perturbação. Logo, quanto maior for este valor, maior será o número de modificações feitas na solução. Neste trabalho, em um mesmo nível de perturbação são aplicadas $iter_{max}$ iterações sem melhora. Tão logo este valor seja alcançado, o nível de perturbação é aumentado em uma unidade.

Por outro lado, a busca local do ILS (linhas 2 e 5 do Algoritmo 2) é feita pelo método *Variable Neighborhood Descent* com exploração randômica de vizinhanças, denotado por RVND e descrito na Seção 3.5.2.

Finalmente, o algoritmo proposto contém um módulo de intensificação (linha 7 do Algoritmo 2). Este módulo é acionado em momentos "apropriados" da busca e consiste na chamada de um procedimento de programação matemática, baseado em Particionamento de Conjuntos, para encontrar o melhor conjunto de rotas dentre aquelas geradas ao longo da busca. Considera-se, neste trabalho, que o momento apropriado para aplicação da intensificação é a última iteração de cada nível de perturbação. Mais especificamente, o modelo de particionamento é aplicado ao conjunto formado por todas as rotas pertencentes às soluções geradas após a fase de busca local do algoritmo IILS. Isto é, a cada

iteração do IILS, as rotas de s'' (linha 5 do Algoritmo 2) são adicionadas ao conjunto a ser particionado. Observa-se que isto é feito evitando-se a existência de rotas repetidas neste conjunto, sendo que não há limite em seu tamanho. A descrição desse módulo é feita na Seção 3.6.

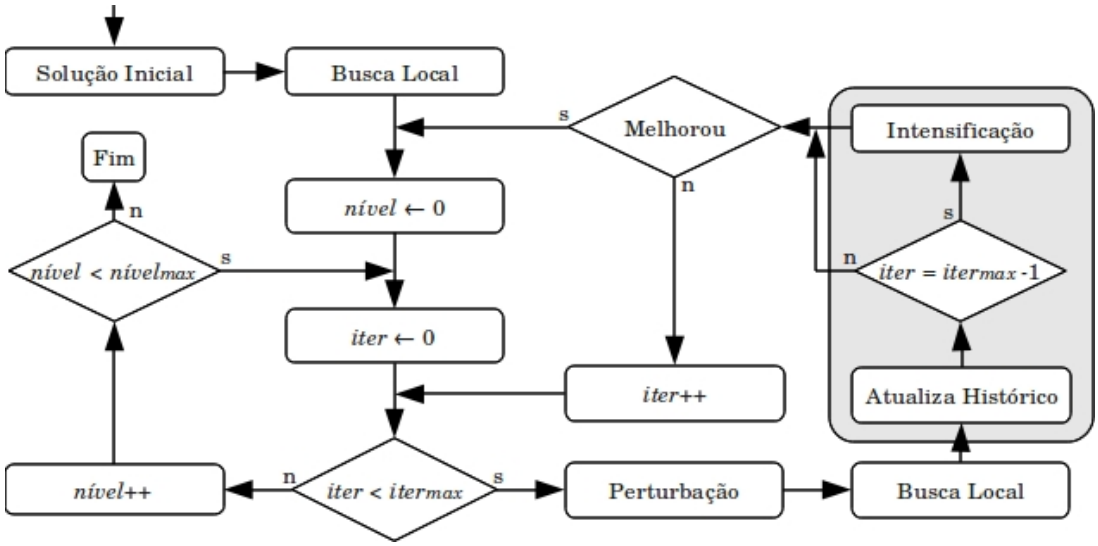


Figura 3.10: Diagrama do algoritmo IILS

A Figura 3.10 apresenta um diagrama mais detalhado do algoritmo desenvolvido. A parte destacada em cinza consiste na etapa de intensificação adicionada ao método original. Vale destacar que, ao utilizar o conceito de níveis de perturbação e refinamento pela versão clássica do *Variable Neighborhood Descent* (VND) na metaheurística *Iterated Local Search*, esta se torna bastante semelhante ao método *General Variable Neighborhood Search* (GVNS) proposto por Hansen e Mladenovic [31]. Assim, o algoritmo desenvolvido poderia sim ser chamado de *Intensified* GVNS. Porém, embora a ideia de níveis seja utilizada neste trabalho, optou-se manter o nome *Iterated Local Search*. Esta escolha se deve principalmente ao fato de que o momento apropriado de intensificação da versão proposta não necessariamente precisa ser definida com base em informações de nível. Por exemplo, caso seja do interesse da aplicação, pode-se definir que um momento apropriado é obtido periodicamente contando um certo número de iterações. Outra ideia é considerar que este momento ocorre quando um ótimo local melhor que os anteriores é alcançado. Outro ponto é que qualquer método de refinamento pode ser usado na fase de busca local.

3.5.2 *Variable Neighborhood Descent*

O procedimento *Variable Neighborhood Descent* (VND), desenvolvido por Mladenović e Hansen [51], consiste em explorar exaustivamente o espaço de soluções por meio de

trocas sistemáticas de estruturas de vizinhança. No decorrer da busca somente são aceitas as soluções de melhora em relação à solução corrente. Quando uma solução melhor é encontrada, o método retoma sua busca partindo da primeira estrutura de vizinhança.

O método VND baseia-se em três princípios: (i) um ótimo local com relação a uma dada estrutura de vizinhança não corresponde necessariamente a um ótimo local com relação a uma outra estrutura de vizinhança; (ii) um ótimo global corresponde a um ótimo local para todas as estruturas de vizinhança; e (iii) para muitos problemas, ótimos locais com relação a uma ou mais estruturas de vizinhança são relativamente próximas.

O último princípio, de natureza empírica, indica que um ótimo local frequentemente fornece algum tipo de informação sobre o ótimo global. Esse é o caso em que os ótimos local e global compartilham muitas variáveis com o mesmo valor, o que sugere uma investigação sistemática da vizinhança de um ótimo local até a obtenção de uma nova solução de melhor valor.

Na versão clássica, o VND efetua buscas por ótimos locais seguindo uma ordem pré-estabelecida de estruturas de vizinhança. Esta estratégia é vastamente aplicada na literatura e sua eficácia é comprovada em diversos trabalhos. No presente trabalho, porém, opta-se por não fixar a ordem de exploração da vizinhança. Dessa forma, ocorre maior diversidade durante a busca. Tal estratégia, adotada com sucesso em [72] e [71], é denominada *Variable Neighborhood Descent with random neighborhood ordering.*

3.5.3 Modelo de Particionamento de Conjunto

A etapa de intensificação do algoritmo proposto consiste na resolução exata de um Problema de Particionamento de Conjunto (PPC). Seja $\mathcal{R}$ o conjunto de rotas e y_j, $\forall j \in \mathcal{R}$, as variáveis binárias que indicam se a rota $j \in \mathcal{R}$ faz parte da solução ($y_j = 1$) ou não ($y_j = 0$). A cada rota $j \in \mathcal{R}$ tem-se um custo g_j associado. O parâmetro m_{ij} é igual a 1 se o cliente $i \in N$ for atendido pela rota $j \in \mathcal{R}$ e 0 caso contrário.

O modelo matemático do PRVJT na forma de um PPC utilizado neste trabalho é apresentado pelas expressões (3.2) a (3.5).

$$Minimize \sum_{j \in \mathcal{R}} g_j y_j \tag{3.2}$$

sujeito a:

$$\sum_{j \in \mathcal{R}} m_{ij} y_j = 1, \forall i \in N \tag{3.3}$$

$$\sum_{j \in \mathcal{R}} y_j \leq |K| \tag{3.4}$$

$$y_j \in \{0, 1\}, \forall j \in \mathcal{R} \tag{3.5}$$

O objetivo dessa formulação é encontrar o conjunto de rotas de custo mínimo (expressão 3.2) atendendo às restrições do problema. As restrições (3.3) garantem que exatamente uma rota passa por cada cliente e as restrições (3.4) asseguram que a solução contém no máximo $|K|$ rotas.

São inúmeros os problemas em otimização combinatória que podem ser descritos como um PPC. Especificamente para roteamento de veículos, o modelo tem sido adotado com sucesso por diversos autores, entre eles: [1], [26], [42] e [44]. Os dois últimos abordaram especificamente o PRVJT, obtendo novos resultados ótimos para as instâncias de [69] até então desconhecidos.

Neste trabalho, o modelo em questão foi implementado utilizando a API Concert para C++ e resolvido, ao longo do algoritmo, pelo otimizador CPLEX, versão 12.

3.6 Versão paralela

Como intuito de aproveitar melhor os recursos de multiprocessamento atualmente disponíveis, propõe-se também uma versão paralela do algoritmo IILS-SP. Em geral, ao se considerar a paralelização de algoritmos de otimização, dois objetivos podem ser explorados: (i) a redução do tempo de execução ou (ii) o aprimoramento da qualidade das soluções finais geradas caso o tempo seja fixado o tempo de processamento. A versão paralela desenvolvida, denominada PIILS-SP, explora o segundo objetivo, a melhoria da qualidade das soluções finais geradas fixado um tempo de processamento.

Na versão sequencial, o algoritmo acumula uma certa quantidade de rotas a cada iteração. Periodicamente durante o método estas rotas são agrupadas da melhor forma pelo modelo de particionamento de conjunto discutido na Seção . Nesse sentido, quanto maior for o número de rotas boas e diferentes, maior é a chance de que a solução gerada pelo modelo exato ainda não tenha sido explorada durante a fase heurística. Caso isto ocorra, pode-se garantir que a solução gerada é melhor que as geradas até então.

A base da versão paralela é a ideia de que é possível gerar uma quantidade maior de rotas boas e diferentes ao se executar o algoritmo IILS-SP de forma independente. Obviamente, para que esta estratégia tenha efeito, cada busca deve percorrer um caminho diferente. Caso contrário, bastaria uma única busca.

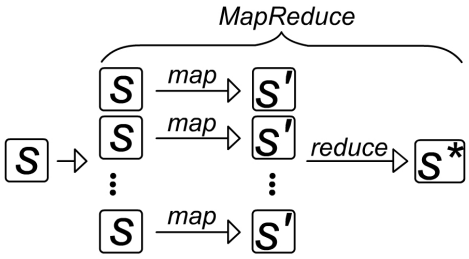
Nesse sentido, a versão paralela é baseada no conceito de *Multi-Start* [49] em forma de *MapReduce*, uma abstração simples e poderosa para o processamento ou geração de grandes massas de dados. Segundo Lämmel [48], esse modelo foi feito para processar de maneira massivamente paralela e é baseado nos seguintes fatores: iteração sobre a entrada, computação sobre cada um dos pares (*chave,valor*) da entrada, agrupamento de todos os valores intermediários por chaves, iteração sobre os grupos resultantes e redução de cada grupo. Dean e Ghemawat [25] apresentam uma visão geral sobre a implementação do *MapReduce* da Google, a qual facilita muito o trabalho de seus programadores.

O usuário da biblioteca *MapReduce* expressa a computação como duas funções: *map* e *reduce*. A função *map* recebe um par como entrada e produz um conjunto de pares intermediários também na forma (*chave,valor*). A biblioteca *MapReduce* agrupa todos os valores intermediários associados à mesma chave intermediária e os passa à função *reduce*. A função *reduce* aceita uma chave intermediária e o conjunto de valores relacionados aquela chave. Essa função junta esses valores para formar um conjunto possivelmente menor de valores. Tipicamente, zero ou apenas um valor é produzido pela função *reduce*.

Especificamente para problemas de otimização, onde os itens a serem mapeados não necessitam ser ordenados ou agrupados por chaves, pois o que interessa é a solução e uma forma de se medir sua qualidade, o modelo pode ser simplificado. Usuários especificam as funções *map* e *reduce*. A primeira é responsável por processar um item de um tipo α resultando em um item do tipo β e a segunda mescla todos os dados intermediários, representados por um conjunto de elementos do tipo β, gerado a partir da aplicação da função *map* a um conjunto de dados do tipo α.

A Figura 3.11 apresenta uma forma de se aplicar a abstração *MapReduce* a problemas de otimização, onde os tipos α e β representam soluções (s e s') e a função *map* pode ser qualquer método de refinamento, o resultado da função *reduce*, s^*, é geralmente a melhor dentre as soluções intermediárias (doravante usaremos o termo *bestReduce* para referir à função de redução que seleciona o melhor elemento de uma sequência de entrada).

Supondo que na Figura 3.11 sejam feitas n aplicações da função de mapeamento (*map*), tal situação é reproduzida por meio da criação de $n + 1$ processos MPI pelo



Fonte: [61]

Figura 3.11: Abstração *MapReduce* em problemas de otimização

framework MaPI[1] [61], utilizado neste trabalho. O processo adicional é responsável por fazer a chamada do *MapReduce*.

No PIILS-SP, em vez de receber uma solução s e retornar uma solução s' (como apresentado na Figura 3.11), cada mapeamento recebe uma semente única para números aleatórios e retorna o conjunto de todas as rotas acumuladas durante a busca. Assim, o PIILS-SP pode ser visto como um programa *MapReduce* distribuído onde o mapeamento consiste na execução do IILS-SP e, a redução, na otimização pelo modelo exato de particionamento de conjunto considerando todas as rotas acumuladas em cada mapeamento.

[1]Disponível em: *http://sourceforge.net/projects/mapreducepp/*

Capítulo 4

Resultados Computacionais

O algoritmo proposto (IILS-SP) foi desenvolvido na linguagem de programação C++ e os testes foram efetuados em um computador com microprocessador Intel Quad Core 2.4 GHz com 8 GB de memória RAM utilizando o sistema operacional Ubuntu Linux 9.10 Kernel 2.6.31.

4.1 Comparações com a literatura

O IILS-SP foi aplicado à resolução do conjunto de problemas-teste proposto por [69], que é bastante conhecido e amplamente explorado na literatura. Este conjunto consiste em 56 problemas-testes de 100 clientes e é dividido em seis grupos: C1, C2, R1, R2, RC1 e RC2. Os problemas contidos nos grupos C1 e C2 possuem clientes alocados de forma geograficamente agrupada, enquanto que em R1 e R2 os clientes são alocados em posições aleatórias. Os problemas de RC1 e de RC2 mesclam agrupamento e aleatoriedade de localização de clientes. Além disso, os grupos C2, R2 e RC2 possuem horizontes de planejamento mais longos e os veículos são de maior capacidade que em C1, R1 e RC1. Isto significa que cada veículo de C2, R2 e RC2 é capaz de atender um número maior de clientes.

Foram feitas 5 execuções do algoritmo para cada um dos 56 problemas-teste utilizando como critério de parada 10 minutos de processamento para cada execução. O algoritmo foi calibrado empiricamente e os parâmetros foram fixados como segue. Na construção de uma solução inicial, ao longo das inserções dos clientes são efetuadas 5 buscas locais conforme descreve a Seção 3.3. O número de iterações sem melhora em um dado nível de perturbação do IILS foi fixado em 20. O procedimento é executado iterativamente,

seguindo a mesma ideia do método *Multi-Start* [49]. Isto é, a cada iteração, a solução inicial é construída pelo método não determinístico descrito na Seção 3.3 e a busca local é feita pelo IILS-SP. Além disto, o tempo máximo de processamento de cada chamada do resolvedor matemático na etapa de intensificação foi limitado a 5 segundos.

As Tabelas 4.1, 4.2, 4.3, 4.4, 4.5 e 4.6 apresentam os melhores resultados disponíveis na literatura para cada um dos problemas-teste analisados e os resultados obtidos pela aplicação do algoritmo proposto. A primeira coluna indica o nome do problema. As três seguintes fornecem respectivamente o número de veículos ("$|K|$"), a distância total percorrida ("Distância") da melhor solução obtida para o problema e o primeiro trabalho que a obteve ("Trabalho"). O próximo grupo de colunas apresenta os resultados obtidos pela aplicação da versão sequencial do método proposto, apresenta-se o número de veículos e a distância da melhor solução obtida nas 5 execuções do algoritmo, a média das distâncias obtidas ("Média") e o desvio da média em relação à melhor solução ("Desvio"), conforme a seguinte definição: $Desvio = (Média\ -\ Melhor)\ / Melhor$, sendo $Melhor$ o melhor valor encontrado para o problema-teste em questão. A última coluna deste grupo apresenta a melhora do valor das soluções obtidas pelo IILS-SP em relação às melhores soluções da literatura, de acordo com a seguinte expressão: $Melhora = (MelhorLit\ -\ Melhor)\ / Melhor$. As últimas cinco colunas apresentam os resultados da versão paralela do método. Os resultados de melhora e os empates estão destacados na tabela com sublinhado e negrito, respectivamente.

Tabela 4.1: Resultados para o grupo de problemas C1

Problema	Melhores da literatura			IILS-SP					PIILS-SP				
	$\|K\|$	Distância	Trabalho	$\|K\|$	Distância	Média	Desvio	Melhora	$\|K\|$	Distância	Média	Desvio	Melhora
C101	10	**828,94**	RT95 [64]	10	**828,94**	**828,94**	0,00%	0,00%	10	**828,94**	**828,94**	0,00%	0,00%
C102	10	**828,94**	RT95 [64]	10	**828,94**	**828,94**	0,00%	0,00%	10	**828,94**	**828,94**	0,00%	0,00%
C103	10	**828,06**	RT95 [64]	10	**828,06**	**828,06**	0,00%	0,00%	10	**828,06**	**828,06**	0,00%	0,00%
C104	10	**824,78**	RT95 [64]	10	**824,78**	**824,78**	0,00%	0,00%	10	**824,78**	**824,78**	0,00%	0,00%
C105	10	**828,94**	RT95 [64]	10	**828,94**	**828,94**	0,00%	0,00%	10	**828,94**	**828,94**	0,00%	0,00%
C106	10	**828,94**	RT95 [64]	10	**828,94**	**828,94**	0,00%	0,00%	10	**828,94**	**828,94**	0,00%	0,00%
C107	10	**828,94**	RT95 [64]	10	**828,94**	**828,94**	0,00%	0,00%	10	**828,94**	**828,94**	0,00%	0,00%
C108	10	**828,94**	RT95 [64]	10	**828,94**	**828,94**	0,00%	0,00%	10	**828,94**	**828,94**	0,00%	0,00%
C109	10	**828,94**	RT95 [64]	10	**828,94**	**828,94**	0,00%	0,00%	10	**828,94**	**828,94**	0,00%	0,00%

Tabela 4.2: Resultados para o grupo de problemas C2

Problema	Melhores da literatura			IILS-SP					PIILS-SP				
	$\|K\|$	Distância	Trabalho	$\|K\|$	Distância	Média	Desvio	Melhora	$\|K\|$	Distância	Média	Desvio	Melhora
C201	3	**591,56**	RT95 [64]	3	**591,56**	**591,56**	0,00%	0,00%	3	**591,56**	**591,56**	0,00%	0,00%
C202	3	**591,56**	RT95 [64]	3	**591,56**	**591,56**	0,00%	0,00%	3	**591,56**	**591,56**	0,00%	0,00%
C203	3	**591,17**	RT95 [64]	3	**591,17**	**591,17**	0,00%	0,00%	3	**591,17**	**591,17**	0,00%	0,00%
C204	3	**590,60**	RT95 [64]	3	**590,60**	596,42	0,99%	0,00%	3	**590,60**	591,27	0,11%	0,00%
C205	3	**588,88**	RT95 [64]	3	**588,88**	**588,88**	0,00%	0,00%	3	**588,88**	**588,88**	0,00%	0,00%
C206	3	**588,49**	RT95 [64]	3	**588,49**	**588,49**	0,00%	0,00%	3	**588,49**	**588,49**	0,00%	0,00%
C207	3	**588,29**	RT95 [64]	3	**588,29**	**588,29**	0,00%	0,00%	3	**588,29**	**588,29**	0,00%	0,00%
C208	3	**588,32**	RT95 [64]	3	**588,32**	**588,32**	0,00%	0,00%	3	**588,32**	**588,32**	0,00%	0,00%

Tabela 4.3: Resultados para o grupo de problemas R1

Problema	Melhores da literatura			IILS-SP					PIILS-SP				
	$\|K\|$	Distância	Trabalho	$\|K\|$	Distância	Média	Desvio	Melhora	$\|K\|$	Distância	Média	Desvio	Melhora
R101	20	**1642,88**	AM04 [3]	20	**1642,88**	**1642,88**	0,00%	0,00%	20	**1642,88**	**1642,88**	0,00%	0,00%
R102	18	**1472,62**	AM04 [3]	18	1472,81	1472,81	0,01%	-0,01%	18	1472,81	1472,81	0,00%	-0,01%
R103	14	**1213,62**	RT95 [64]	14	**1213,62**	1214,40	0,06%	0,00%	14	**1213,62**	**1213,62**	0,00%	0,00%
R104	11	**986,10**	AL07 [4]	11	**982,30**	988,65	0,26%	0,39%	11	**976,61**	**983,25**	0,68%	0,96%
R105	15	**1360,78**	AL07 [4]	15	**1360,78**	**1360,78**	0,00%	0,00%	15	**1360,78**	**1360,78**	0,00%	0,00%
R106	13	**1241,52**	AL07 [4]	13	**1239,37**	1242,48	0,08%	0,17%	13	**1239,37**	**1239,37**	0,00%	0,17%
R107	11	**1076,13**	AL07 [4]	11	**1075,21**	1076,23	0,01%	0,09%	11	**1072,12**	**1073,61**	0,14%	0,37%
R108	10	**948,57**	AL07 [4]	10	951,22	955,39	0,72%	-0,28%	10	**941,89**	**945,53**	0,39%	0,70%
R109	13	**1151,84**	AL07 [4]	13	**1151,84**	1152,06	0,02%	0,00%	13	**1151,84**	1152,06	0,02%	0,00%
R110	11	**1080,36**	RT95 [64]	12	**1072,41**	**1072,41**	0,00%	0,74%	12	**1072,41**	**1075,24**	0,26%	0,74%
R111	12	**1053,50**	AL07 [4]	12	**1053,50**	1054,43	0,09%	0,00%	12	**1053,50**	1055,88	0,23%	0,00%
R112	10	**953,63**	RT95 [64]	10	956,36	961,66	0,84%	-0,29%	10	**953,63**	963,31	1,01%	0,00%

Tabela 4.4: Resultados para o grupo de problemas R2

Problema	Melhores da literatura			IILS-SP					PIILS-SP				
	$\|K\|$	Distância	Trabalho	$\|K\|$	Distância	Média	Desvio	Melhora	$\|K\|$	Distância	Média	Desvio	Melhora
R201	8	**1147,80**	OV08 [24]	8	**1147,80**	1149,94	0,19%	0,00%	8	**1147,80**	1149,11	0,11%	0,00%
R202	8	**1039,32**	OV08 [24]	8	**1034,35**	**1039,19**	0,47%	0,48%	8	**1034,35**	**1035,24**	0,09%	0,48%
R203	6	**874,87**	OV08 [24]	6	881,12	892,38	2,00%	-0,71%	6	**874,87**	877,92	0,35%	0,00%
R204	5	**735,80**	OV08 [24]	4	745,12	759,48	3,22%	-1,25%	5	737,24	744,32	0,96%	-0,20%
R205	5	**954,16**	OV08 [24]	5	955,96	967,51	1,40%	-0,19%	6	957,85	962,66	0,50%	-0,39%
R206	5	**884,25**	OV08 [24]	5	**879,89**	891,15	0,78%	0,50%	5	**879,89**	890,64	1,22%	0,49%
R207	4	**797,99**	OV08 [24]	4	808,23	820,73	2,85%	-1,27%	4	804,44	814,33	1,23%	-0,81%
R208	4	**705,62**	OV08 [24]	3	724,98	734,94	4,16%	-2,67%	3	711,14	721,65	1,48%	-0,78%
R209	5	**860,11**	OV08 [24]	5	**859,39**	866,29	0,72%	0,08%	5	**859,39**	863,35	0,46%	0,08%
R210	5	**910,98**	OV08 [24]	7	914,84	919,89	0,98%	-0,42%	6	**906,97**	912,62	0,62%	0,44%
R211	4	**755,82**	OV08 [24]	4	762,38	772,97	2,27%	-0,86%	4	760,77	769,44	1,14%	-0,66%

Tabela 4.5: Resultados para o grupo de problemas RC1

Problema	Melhores da literatura			IILS-SP					PIILS-SP				
	$\|K\|$	Distância	Trabalho	$\|K\|$	Distância	Média	Desvio	Melhora	$\|K\|$	Distância	Média	Desvio	Melhora
RC101	15	**1623,58**	RT95 [64]	15	**1623,58**	1626,82	0,20%	0,00%	15	**1623,58**	**1623,58**	0,00%	0,00%
RC102	14	**1466,84**	AL07 [4]	14	**1461,23**	1472,15	0,36%	0,38%	14	**1461,23**	**1465,08**	0,26%	0,38%
RC103	11	**1261,67**	S98 [68]	11	1262,02	1273,51	0,94%	-0,03%	11	**1261,67**	1267,01	0,42%	0,00%
RC104	10	**1135,48**	C01 [21]	10	1135,52	1135,79	0,03%	0,00%	10	**1135,48**	1136,48	0,09%	0,00%
RC105	16	**1518,60**	AM04 [3]	16	**1518,58**	**1518,58**	0,00%	0,00%	16	**1518,58**	**1518,58**	0,00%	0,00%
RC106	13	**1377,35**	AM04 [3]	13	**1376,99**	1378,25	0,07%	0,03%	12	**1376,26**	**1377,21**	0,07%	0,08%
RC107	12	**1212,83**	AM04 [3]	12	**1212,83**	**1212,83**	0,00%	0,00%	12	**1211,11**	**1212,78**	0,14%	0,14%
RC108	11	**1117,53**	AM04 [3]	11	**1117,53**	1120,99	0,31%	0,00%	11	**1117,53**	1125,73	0,73%	0,00%

Tabela 4.6: Resultados para o grupo de problemas RC2

Problema	Melhores da literatura			IILS-SP					PIILS-SP				
	$\|K\|$	Distância	Trabalho	$\|K\|$	Distância	Média	Desvio	Melhora	$\|K\|$	Distância	Média	Desvio	Melhora
RC201	9	**1266,11**	OV08 [24]	9	**1265,90**	1268,59	0,20%	0,02%	9	**1265,90**	1268,29	0,19%	0,02%
RC202	8	**1096,75**	OV08 [24]	8	**1095,64**	1099,17	0,22%	0,10%	8	**1095,64**	1099,36	0,34%	0,10%
RC203	5	**926,89**	OV08 [24]	5	937,45	949,94	2,49%	-1,13%	6	938,19	940,99	0,30%	-1,22%
RC204	4	**786,38**	OV08 [24]	4	796,55	810,88	3,12%	-1,28%	4	794,38	799,84	0,69%	-1,02%
RC205	7	**1157,55**	OV08 [24]	7	1157,66	1161,82	0,37%	-0,01%	7	**1157,55**	1158,55	0,09%	0,00%
RC206	6	**1056,21**	OV08 [24]	6	1069,00	1076,84	1,95%	-1,20%	7	**1054,61**	1060,79	0,59%	0,15%
RC207	6	**966,08**	OV08 [24]	6	**966,08**	982,07	1,66%	0,00%	6	**966,08**	972,57	0,67%	0,00%
RC208	4	**779,84**	OV08 [24]	5	785,07	788,78	1,15%	-0,67%	5	782,70	788,22	0,71%	-0,37%

4.1.1 Análise do algoritmo sequencial

Em suma, quanto às melhores soluções encontradas ao longo das execuções, a versão sequencial do algoritmo proposto (IILS-SP) obteve, comparativamente aos melhores resultados encontrados na literatura: 100% (9/9) de empates para C1; 100% (8/8) de empates para C2; 33,3% (4/12) de melhorias e 41,6% (5/12) de empates para R1; 27,3% (3/11) de melhorias e 9,1% (1/11) de empates para R2; 37,5% (3/8) de melhorias e 37,5% (3/8) de empates para RC1; e 25% (2/8) de melhorias e 12,5% (1/8) de empates para RC2. Ao todo, o IILS-SP superou os resultados da literatura em 21,4% (12/56) dos casos, empatou em 48,2% (27/56) e perdeu em 30,4% (17/56) ao se considerar todos os grupos.

O IILS-SP se mostrou robusto, visto que ele apresentou desvios relativamente baixos. Em 80,4% (45/56) dos problemas-teste analisados, o desvio foi inferior a 1,0%. Quando extrapolou esse valor, o desvio não passou de 4,16% (como é o caso em R208). Dessa forma, pode-se concluir que o algoritmo produz soluções finais com baixa variabilidade em relação à qualidade.

4.1.2 Análise da versão paralela

Embora a versão paralela tenha sido baseada em MPI e estivesse pronta para execução em um *cluster*, os testes desta versão contaram apenas com os oito núcleos de processamento de uma mesma máquina. Quanto à esta versão, denominada PIILS-SP, os seguintes resultados foram alcançados: 100% (9/9) de empates para C1; 100% (8/8) de empates para C2; 41,7% (5/12) de melhorias e 50% (6/12) de empates para R1; 36,4% (4/11) de melhorias e 18,2% (2/11) de empates para R2; 50% (4/8) de melhorias e 50% (4/8) de empates para RC1; e 37,5% (3/8) de melhorias e 25% (2/8) de empates para RC2. Ao todo, o PIILS-SP superou os resultados da literatura em 28,6% (16/56) dos casos, empatou em 55,4% (31/56) e perdeu em 16,1% (9/56) ao se considerar todos os grupos.

O PIILS-SP se mostrou robusto, visto que ele apresentou desvios relativamente baixos. Em 91,1% (51/56) dos problemas-teste analisados, o desvio foi inferior a 1,0%. O maior desvio da versão paralela também foi obtido em R208, porém o valor não passou de 1,48%. Assim, o algoritmo paralelo produz soluções finais com variabilidade em relação à qualidade ainda mais baixa que a produzida pela versão sequencial. Além disso, em alguns casos (R104, R106, R107, R108, R110, R202, RC102, RC105, RC106 e RC107) o algoritmo obteve soluções em média melhores que as melhores soluções da literatura.

4.1.3 Visão geral e comparativo por grupos

A Tabela 4.7 apresenta os resultados de diferentes trabalhos que têm como objetivo primário a minimização da distância total percorrida. Os resultados de cada trabalho são apresentados em colunas onde cada célula possui duas linhas. A linha superior ("NV") indica o número de veículos e a inferior ("DT") indica a distância total percorrida. Nas linhas de "C1" a "RC2" são apresentados os resultados para cada um dos grupos de problemas-teste analisados. Cada célula apresenta a média do número de veículos e a média das distâncias das melhores soluções de cada grupo. As últimas duas linhas apresentam o número total de veículos e a distância total percorrida considerando-se todos os problemas, isto é, os valores acumulados destas linhas são calculados somando-se o número de veículos e a distância percorrida das melhores soluções obtidas por cada algoritmo para os 56 problemas-teste de [69].

Tabela 4.7: Comparação entre trabalhos que otimizam a distância total percorrida

Grupo		Trabalho					IILS-SP	PIILS-SP
		RT95 [64]	CA99 [15]	SC00 [66]	AL07 [4]	OV08 [24]		
C1	NV	10,00	10,00	10,00	10,00	10,00	10,00	10,00
	DT	**828,38**	**828,38**	**828,38**	**828,38**	**828,38**	**828,38**	**828,38**
C2	NV	3,00	3,00	3,00	3,00	3,00	3,00	3,00
	DT	**589,86**	596,63	**589,86**	**589,86**	**589,86**	**589,86**	**589,86**
R1	NV	12,16	12,42	12,08	13,25	13,33	13,17	13,25
	DT	1208,50	1233,34	1211,53	1183,38	1186,94	**1181,03**	**1179,29**
R2	NV	2,91	3,09	2,82	5,55	5,36	5,36	5,45
	DT	961,71	990,99	949,27	899,90	**878,79**	883,10	879,52
RC1	NV	11,87	12,00	11,88	12,88	13,25	12,75	12,63
	DT	1377,39	1403,74	1361,76	1341,67	1362,44	**1338,54**	**1338,18**
RC2	NV	3,37	3,38	3,38	6,50	6,13	6,13	6,50
	DT	1119,59	1220,99	1097,63	1015,90	**1004,59**	1009,17	1006,88
Todos	NVA	414	420	412	489	488	482	486
	DTA	57231	58927	56830	55134	55021	**54842**	**54761**

RT95 $92,5 \times 5$; CA99 5×1; SC00 30×10; AL07 60×3; OV08 $35,1 \times 15$; IILS-SP e PIILS-SP 10×5.
Notação: (tempo médio do algoritmo em minutos) $\times$ (número de execuções por problema).

Ao se observar os resultados separadamente para cada grupo, conclui-se que as duas versões do algoritmo desenvolvido empataram com os melhores resultados nos grupos clusterizados (C1 e C2) e superaram todos os trabalhos nos grupos R1 e RC1. Porém, apesar de apresentarem resultados próximos aos melhores nos grupos R2 e RC2, eles não foram capazes de superá-los nesses dois grupos. Por outro lado, ao se considerar todos os grupos, tanto o PIILS-SP quanto o IILS-SP superaram os demais algoritmos.

4.2 Eficiência do método

A versão clássica da metaheurística *Iterated Local Search* é, por si só, bastante poderosa, tendo sido aplicada por diversos autores na resolução de inúmeros problemas de otimização. No presente trabalho foi introduzida uma estratégia de intensificação, que consiste em aplicar, em momentos apropriados, o modelo de particionamento de conjuntos a um conjunto de rotas promissoras geradas ao longo do algoritmo. Para validar sua contribuição na exploração do espaço de soluções, são apresentados nesta seção os resultados comparativos entre a metodologia proposta para a resolução do PRVJT utilizando o algoritmo IILS-SP desenvolvido neste trabalho e a versão clássica da metaheurística *Iterated Local Search*, onde não há etapa de intensificação. Para analisar os dois algoritmos, este trabalho utiliza a Distribuição de Probabilidade Empírica (doravante DPE), que consiste em fixar um valor alvo e fazer n execuções do algoritmo, no caso $n = 30$, guardando o tempo gasto para se alcançar o objetivo em cada execução. O ideal é que a escolha do valor alvo seja de forma que o objetivo seja alcançado em um tempo razoável. Em posse dos tempos de execução, um gráfico é feito associando-se o i-ésimo tempo à probabilidade $p_i = \left(i - \frac{1}{2}\right)/n$, gerando pontos $z_i = (t_i, p_i)$, para $i = 1, ..., n$.

A Figura 4.1 apresenta a DPE para o problema-teste R208 de Solomon com alvo de 5% da melhor solução disponível na literatura. Considerando a melhor solução no valor de $705,62$, o alvo pode ser calculado como segue: $alvo_{R208} = 705,62 + 705,62 \times 0,05 = 705,62 + 35,28 = 740,90$. A figura apresenta duas curvas, as quais correspondem à probabilidade de se atingir o alvo especificado. Assim, quanto mais à esquerda estiver a curva, menor é o tempo necessário para que o método correspondente atinja seu objetivo.

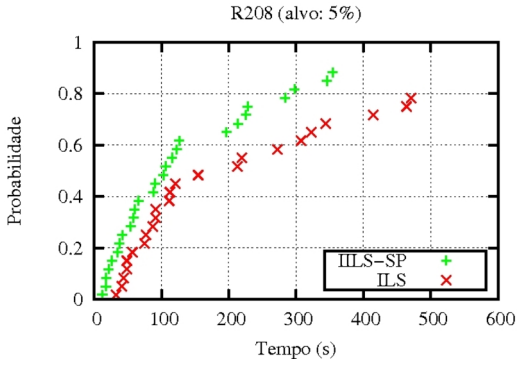


Figura 4.1: DPE para o problema-teste R208 com alvo de 5%

Pela Figura 4.1, pode-se observar que a curva referente ao algoritmo IILS-SP apresenta-se mais à esquerda da curva do ILS. Isto mostra que, no problema R208, a versão intensificada tem maior probabilidade de alcançar soluções boas, gastando menos tempo que a

versão não intensificada.

Uma particularidade deste tipo de teste é que os algoritmos não produzem os mesmos resultados ao se alterar os valores alvo. Desta forma, definimos alvos de acordo com dois níveis de dificuldade para analisar o comportamento dos algoritmos. Consideramos, neste trabalho, que gerar uma solução 5% distante da melhor solução disponível na literatura seja uma tarefa fácil e que gerar uma solução distante em 1% da melhor seja uma tarefa difícil. A Figura 4.2 apresenta a distribuição de probabilidade empírica para os problemas-teste C103, R107 e RC208 considerando diferentes valores alvo.

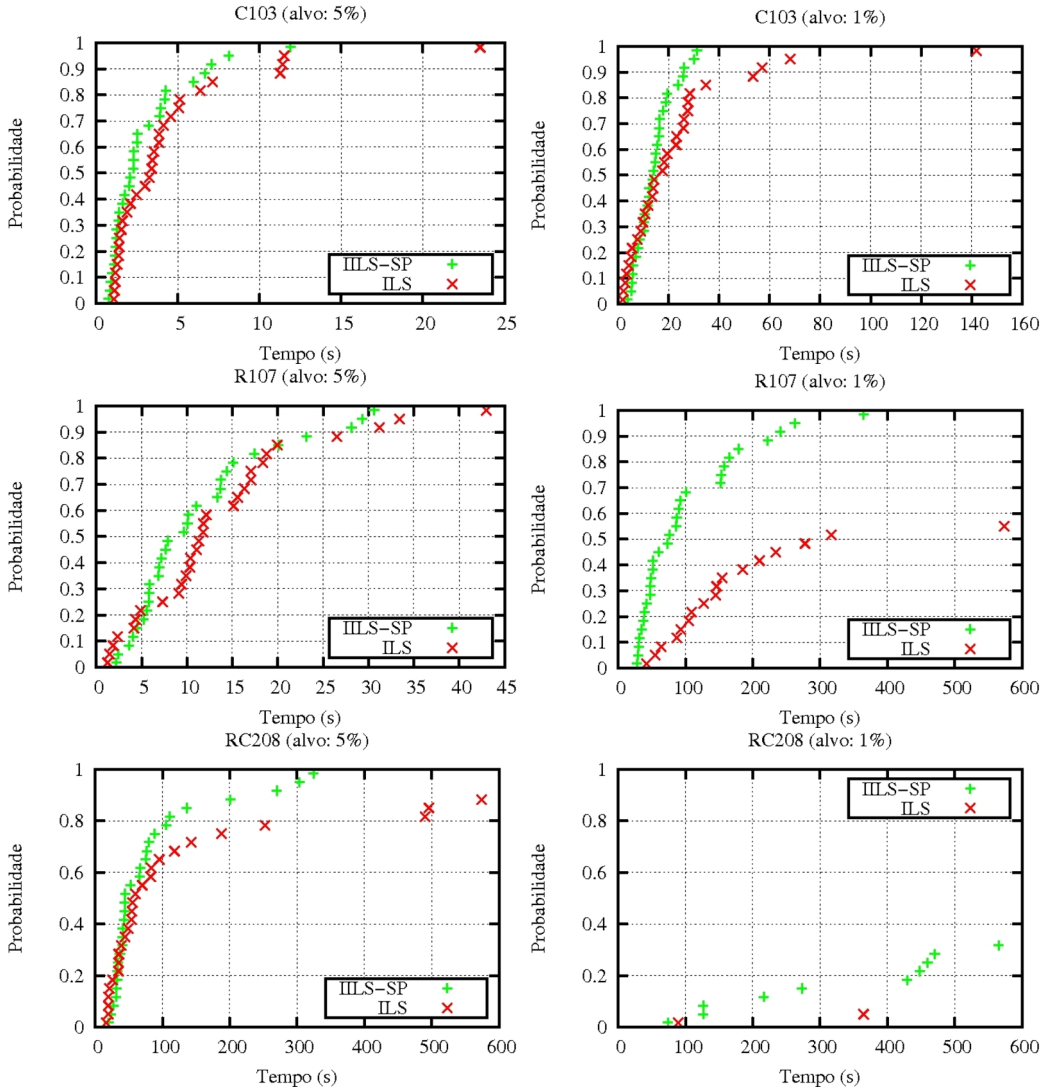


Figura 4.2: DPE para C103, R107 e RC208 considerando alvos de 1% e 5%

No problema-teste C103 com alvo fácil, o IILS-SP e o ILS tem probabilidade de quase 100% de alcançar o alvo em menos que 13 segundos e em menos que 25 segundos,

respectivamente. Para alcançar alvos difíceis, o IILS-SP não gasta mais que 30 segundos e, embora o ILS tenha gasto mais que 140 segundos no pior caso, este algoritmo tem probabilidade de aproximadamente 85% de alcançar o alvo em 30 segundos. Isto mostra que, embora nos testes realizados na Seção 4.1 o critério de parada tenha sido 10 minutos de processamento, em alguns casos o algoritmo não leva todo esse tempo para convergir.

Considerando o problema-teste R107, pode-se observar que as duas curvas apresentam comportamentos semelhantes para os alvos fáceis, chegando a se cruzar em alguns instantes. Neste caso, os dois algoritmos são capazes de alcançar o alvo em menos de 45 segundos. Porém, apenas a versão intensificada do algoritmo é capaz alcançar os alvos difíceis com probabilidade de quase 100%. Neste caso, a versão não intensificada tem apenas 55% de chance de convergir às soluções alvo no tempo limite estabelecido. Com probabilidade de 95%, o IILS-SP atinge o alvo em pouco mais que 250 segundos. Adicionalmente, com probabilidade de 50%, o IILS-SP atinge o alvo em pouco menos de 90 segundos e o ILS leva mais que 300 segundos. Nesse sentido, pode-se concluir que a fase de intensificação realmente é efetiva durante a busca.

A eficiência da proposta de intensificação também pode ser notada no problema-teste RC208, onde o alvo fácil foi alcançado em pouco mais que 310 segundos em todos os casos. Observando o gráfico referente ao alvo difícil para o mesmo problema, podemos verificar que nenhum dos dois algoritmos foi capaz de alcançar o alvo satisfatoriamente, isto é, com probabilidade elevada. Isto ocorre pois a distância percentual do alvo difícil, que é de 1% da melhor solução da literatura, é inferior ao valor do desvio, que é de 1,15% para este problema-teste (conforme Tabela 4.6). Entretanto, vale lembrar que esses testes foram feitos para a versão sequencial do algoritmo e que a versão paralela é capaz de gerar resultados ainda melhores.

Capítulo 5

Conclusões e Trabalhos Futuros

Este trabalho apresenta um algoritmo híbrido para o Problema de Roteamento de Veículos com Janelas de Tempo. O algoritmo proposto, denominado IILS-SP, combina a metaheurística *Iterated Local Search*, o método *Variable Neighborhood Descent* e um modelo exato de Particionamento de Conjunto que, periodicamente, agrega da melhor forma as rotas geradas ao longo do algoritmo.

Dado o caráter combinatório do problema, que pertence à classe NP-Difícil, sua resolução por abordagens puramente exatas é, em muitos casos, computacionalmente impraticável. Este fato motiva o desenvolvimento de algoritmos heurísticos para a resolução deste problema. Heurísticas são desenvolvidas para serem rápidas, porém não garantem encontrar a melhor solução para os problemas tratados. O algoritmo proposto combina a flexibilidade dos métodos heurísticos com o poder da programação matemática.

Adicionalmente, foi desenvolvida uma versão paralela do algoritmo, o PIILS-SP, nos moldes da abstração *MapReduce*. Neste modelo, a computação é expressa por meio de duas funções, a de mapeamento e a de redução. A versão paralela desenvolvida pode ser vista como um programa *MapReduce* onde o mapeamento consiste na execução da versão sequencial híbrida proposta e a redução consiste na resolução exata do modelo de particionamento de conjunto considerando as rotas acumuladas em cada nó de processamento.

Os dois algoritmos foram aplicado aos 56 problemas-teste de Solomon e os resultados foram comparados aos melhores encontrados na literatura. Os experimentos computacionais mostraram que a abordagem híbrida proposta é competitiva, uma vez que dos 56 problemas considerados, a versão paralela foi capaz de melhorar a melhor solução da literatura em 16 casos e igualar ao melhor resultado em 31 casos. Além disso, embora o critério de parada do algoritmo tenha sido dez minutos de processamento, os testes de

análise de probabilidade empírica mostraram que, para alguns problemas-teste, o versão sequencial gasta (com probabilidade de aproximadamente 100%) apenas alguns segundos para atingir os valores alvo.

Ao longo do desenvolvimento deste trabalho, algumas decisões no projeto dos algoritmos foram tomadas com base em testes rápidos e informais. Além disso, a calibração do método foi feita de forma empírica. Assim, apesar de que o algoritmo apresente resultados em geral melhores que os da literatura, acredita-se que ainda seja possível aprimorá-lo. Para isto, propõe-se, como trabalho futuro, a calibração sistemática dos parâmetros e a realização de experimentos para verificar a real contribuição de cada parte do algoritmo.

Como trabalho futuro propõe-se também a adaptação da reavaliação proposta em Nagata [52] aos operadores de busca local utilizados neste trabalho. Para os movimentos *swap*, *shift* e *2-opt**, que também são utilizados no presente trabalho, os autores reduziram o custo computacional da reavaliação de uma solução para $O(1)$. Espera-se, desta forma, uma redução no tempo demandado pelo algoritmo proposto. Além disto, com tal estratégia de reavaliação o algoritmo poderia ser aplicado a problemas-teste maiores.

Referências

[1] Agarwal, Y., Mathur, K., Salkin, H. M. A set-partitioning-based exact algorithm for the vehicle routing problem. *Networks 19* (1989), 731–740.

[2] Alvarenga, G. B. *Um algoritmo híbrido para o problema de roteamento de veículos estático e dinâmico com janela de tempo.* Tese de doutorado, Departamento de Ciência da Computação, Universidade Federal de Minas Gerais, Belo Horizonte, MG, 2005.

[3] Alvarenga, G. B., Mateus, G. R. A two-phase genetic and set partitioning approach for the vehicle routing problem with time windows. In *HIS '04: Proceedings of the Fourth International Conference on Hybrid Intelligent Systems* (Washington, DC, USA, 2004), IEEE Computer Society, p. 428–433.

[4] Alvarenga, G. B., Mateus, G. R., de Tomi, G. A genetic and set partitioning two-phase approach for the vehicle routing problem with time windows. *Computers and Operations Research 34* (2007), 1561–1584.

[5] Antes, J., Derigs, U. A new parallel tour construction algorithm for the vehicle routing problem with time windows. Tech report, Technical Report, Lehrstuhl fur Wirtschaftsinformatik und Operations Research, Universitat zu Koln, 1995.

[6] Backer, B. D., Furnon, V. Meta-heuristics in constraint programming experiments with tabu search on the vehicle routing problem. In *MIC'97: Proceedings of the Second International Conference on Metaheuristics* (Sophia Antipolis, France, 1997).

[7] Badeau, P., Guertin, F., Gendreau, M., Potvin, J., Taillard, E. A parallel tabu search heuristic for the vehicle routing problem with time windows. *Transportation Research 5*, 2 (1997), 109–22.

[8] Baker, E., Schaffer, J. Solution improvement heuristics for the vehicle routing and scheduling problem with time windows constraints. *American Journal of Mathematics and Management Sciences 6*, 3,4 (1989), 261–300.

[9] Bent, R., Hentenryck, P. V. A two-stage hybrid local search for the vehicle routing problem with time windows. *Transportation Science 38*, 4 (2004), 515–530.

[10] Berger, J., Barkoui, M. A parallel hybrid genetic algorithm for the vehicle routing problem with time windows. *Computers and Operations Research 31*, 12 (2004), 2037–53.

[11] Bouthillier, A., Crainic, T. A cooperative parallel meta-heuristic for the vehicle routing problem with time windows. *Computers and Operations Research 32*, 7 (2005), 1685–708.

[12] Braysy, O. Fast local searches for the vehicle routing problem with time windows. *Inform. Systems Oper. Res. 40* (2002), 319–330.

[13] Braysy, O., Gendreau, M. Tabu search heuristics for the vehicle routing problem with time windows. *TOP: An Official Journal of the Spanish Society of Statistics and Operations Research 10*, 2 (2002), 211–237.

[14] Braysy, O., Gendreau, M. Vehicle routing problem with time windows, part i: Route construction and local search algorithms. *Transportation Science 39*, 1 (2005), 104–118.

[15] Caseau, Y., Laburthe, F. Heuristics for large constrained vehicle routing problems. *Journal of Heuristics 5* (1999), 281–303.

[16] Chabrier, A. Vehicle routing problem with elementary shortest path based column generation. *Computers and Operations Research 33*, 10 (2006), 2972–90.

[17] Chiang, W., Russell, R. Simulated annealing metaheuristics for the vehicle routing problem with time windows. *Annals of Operations Research 63* (1996), 3–27.

[18] Chiang, W. C., Russell, R. A. A reactive tabu search metaheuristic for the vehicle routing problem with time windows. *INFORMS J. Comput. 9* (1997), 417–430.

[19] Christofides, N., Beasley, N. The period routing problem. *Networks 14* (1984), 237–241.

[20] Clarke, G., Wright, W. Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research 12* (1964), 568–581.

[21] Cordeau, J. F., Desaulniers, G., Desrosiers, J., Solomon, M. M., Soumis, F. *The Vehicle Routing Problem*. Paolo Toth and Daniele Vigo, SIAM Monographs on Discrete Mathematics and Applications, 2001, ch. The VRP with Time Windows, p. 157–193.

[22] Cordeau, J. F., Laporte, G., Mercier, A. A unified tabu search heuristic for vehicle routing problems with time windows. *J. Oper. Res. Soc. 52* (2001), 928–936.

[23] Dantzig, G., Ramser, J. H. The truck dispatching problem. *Management Science 6* (1959), 80–91.

[24] de OLIVEIRA, H., Vasconcelos, G. A hybrid search method for the vehicle routing problem with time windows. *Annals of Operations Research* (2008).

[25] Dean, J., Ghemawat, S. Mapreduce: Simplified data processing on large clusters. *In OSDI'04, 6th Symposium on Operating Systems Design and Implementation, Sponsored by USENIX, in cooperation with ACM SIGOPS* (2004), 137–150.

[26] Desrosiers, J., Soumis, F., Desrochers, M. Routing with time windows by column generation. *Networks 14*, 4 (1984), 545–565.

[27] El-Sherbeny, N. A. Vehicle routing with time windows: An overview of exact, heuristic and metaheuristic methods. *Journal of King Saud University 22* (2010), 123–131.

[28] Gehring, H., Homberger, J. Parallelization of a two-phase metaheuristic for routing problems with time windows. *Asia-Pacific J. Oper. Res. 18* (2001), 35–47.

[29] Gendreau, M., Hertz, A., Laporte, G. A tabu search heuristic for the vehicle routing problem. *Management Science 40*, 10 (1994), 1276–90.

[30] Gendreau, M., Hertz, A., Laporte, G., Stan, M. A generalized insertion heuristic for the traveling salesman problem with time windows. *Operations Research 46* (1998), 330–335.

[31] Hansen, P., Mladenovic, N. A tutorial on variable neighborhood search. Tech report, Les Cahiers du GERAD, HEC Montreal and GERAD, 2003.

[32] Homberger, J., Gehring, H. Two evolutionary metaheuristics for the vehicle routing problem with time windows. *Inform. Systems Oper. Res. 37* (1999), 297–318.

[33] Homberger, J., Gehring, H. A two-phase hybrid metaheuristic for the vehicle routing problem with time windows. *European Journal of Operational Research 162*, 1 (2005), 220–238.

[34] Ibaraki, T., Imahori, S., nad K. Sobue, K. N., Uno, T., Yagiura, M. An iterated local search algorithm for the vehicle routing problem with convex time penalty functions. *Discrete Applied Mathematics 156*, 11 (2008), 2050–69.

[35] Ibaraki, T., Kubo, M., Masuda, T., Uno, T., Yagiura, M. Effective local search algorithms for the vehicle routing problem with general time window constraints. *Transportation Science 39*, 2 (2005), 206–32.

[36] Irnich, S., Villeneuve, D. The shortest path problem with resource constraints and k-cycle elimination. *INFORMS Journal on Computing 18*, 3 (2006), 391–406.

[37] Jepsen, M., Spoorendonk, S., Petersen, B., Pisinger, D. A non-robust branch-and-cut-and-price algorithm for the vehicle routing problem with time windows. Tech report, Department of Computer Science, University of Copenhagen, 2006.

[38] Jepsen, M., Spoorendonk, S., Petersen, B., Pisinger, D. Subset-row inequalities applied to the vehicle-routing problem with time windows. *INFORMS Journal on Computing 56*, 2 (2008), 479–511.

[39] Kallehaugea, B., Larsenb, J., Madsena, O. Lagrangian duality applied to the vehicle routing problem with time windows. *Computers and Operations Research 33*, 5 (2006), 1464–87.

[40] Kilby, P., Prosser, P., Shaw, P. Guided local search for the vehicle routing problem with time windows. In *META-HEURISTICS advances and trends in local search paradigms for optimization* (Boston: Kluwer Academic, 1999), S. Voss, S. Martello, I. H. Osman, & C. Roucairol (Eds.), p. 473–486.

[41] King, G. F., Mast, C. F. Excess travel: causes, extent and consequences. *Transportation Research Record*, 1111 (1997), 126–134.

[42] Kohl, N. *Exact methods for Time Constained Routing and Related Scheduling Problems*. PhD thesis, Department of Mathematical Modelling, Technical University of Denmark, Lyngby, Denmark, 1995.

[43] Landeghem, A. A bi-criteria heuristic for the vehicle routing problem with time windows. *European Journal of Operational Research 36* (1988), 217–26.

[44] Larsen, J. *Parallelization of the vehicle routing problem with time windows*. PhD thesis, Department of Mathematical Modelling, Technical University of Denmark, Lyngby, Denmark, 1995.

[45] Lenstra, J. K., Kan, A. H. G. R. Complexity of vehicle routing and scheduling problems. *Networks 11*, 2 (2006), 221–227.

[46] Lim, A., Zhang, X. A two-stage heuristic with ejection pools and generalized ejection chains for the vehicle routing problem with time windows. *INFORMS Journal on Computing 19*, 3 (2007), 443–457.

[47] Lourenco, H. R., Martin, O. C., Stutzle, T. Iterated local search. In *Handbook of Metaheuristics*, F. Glover and G. Kochenberger, Eds. Kluwer Academic Publishers, Boston, 2003, ch. 11.

[48] Lämmel, R. Google's mapreduce programming model, microsoft corp.

[49] Martí, R. Multi-start methods. In *Handbook of Metaheuristics*, F. Glover and G. Kochenberger, Eds. Kluwer Academic Publishers, Boston, 2003, ch. 12.

[50] Mester, D., Braysy, O. Active guided evolution strategies for large scale vehicle routing problems with time windows. *Computers and Operations Research 32*, 6 (2005), 1593–614.

[51] Mladenovic, N., Hansen, P. A variable neighborhood search. *Computers and Operations Research 24* (1997), 1097–1100.

[52] Nagata, Y., Braysy, O., Dullaert, W. A penalty-based edge assembly memetic algorithm for the vehicle routing problem with time windows. *Computers and Operations Research 37* (2010), 724–737.

[53] Ombuki, B., Ross, B. J., Hanshar, F. Multi-objective genetic algorithms for vehicle routing problem with time windows. *Applied Intelligence 24* (2006), 17–30.

[54] Or, I. *Traveling salesman-type combinatorial problems and their relation to the logistics of regional blood banking*. PhD thesis, Northwestern University, IL, 1976.

[55] Osman, I. Metastrategy simulated annealing and tabu search heuristic algorithms for the vehicle routing problem. *Annals of Operations Research 41* (1993), 421–434.

[56] Pisinger, D., Ropke, S. A general heuristic for vehicle routing problems. *Comput. Oper. Res. 34*, 8 (2007), 2403–2435.

[57] Potvin, J., Rousseau, J. A parallel route building algorithm for the vehicle routing and scheduling problem with time windows. *European Journal of Operational Research 66* (1993), 331–40.

[58] Potvin, J. Y., Robillard, C. Clustering for vehicle routing with a competitive neural network. *Neurocomputing 8* (1995), 125–139.

[59] Prescott-Gagnon, E., Desaulniers, G., Rousseau, L. M. A branch-and-price-based large neighborhood search algorithm for the vehicle routing problem with time windows. *Networks 54*, 4 (2009), 190–204.

[60] Resende, M. G. C., Ribeiro, C. C. Grasp. In *Search Methodologies*, E. K. Burke and G. Kendall, Eds., 2 ed. Springer (to appear), 2010. Available at: http://www.ic.uff.br/∼celso/artigos/grasp.pdf.

[61] Ribas, S., Perché, M., Coelho, I., Munhoz, P., Souza, M., Aquino, A. A framework for developing parallel optimization algorithms, 2010. [Online; acessado em 16 de Março de 2011].

[62] Ribeiro, C. C. Metaheuristics and applications. In *Advanced School on Artificial Intelligence* (Estoril, Portugal, 1996).

[63] Riise, A., Stølevik, M. Implementation of guided local search for the vehicle routing problem. Tech report, Department of Computer Science, Michigan State University, SINTEF Applied Mathematics, Norway, 1999.

[64] Rochat, Y., Taillard, E. Probabilistic diversification and intensification in local search for vehicle routing. *Journal of Heuristics 1* (1995), 147–167.

[65] Russell, R. Hybrid heuristics for the vehicle routing problem with time windows. *Transportation Science 29*, 2 (1995), 156–66.

[66] Schrimpf, G., Schneider, J., Stamm-Wilbrandt, H., Dueck, G. Record breaking optimization results using the ruin and recreate principle. *Journal of Computational Physics 159* (2000), 139–171.

[67] Schulze, J., Fahle, T. A parallel algorithm for the vehicle routing problem with time windows constraints. *Annals of Operations Research 86* (1999), 585–607.

[68] Shaw, P. Using constraint programming and local search methods to solve vehicle routing problems. *Lecture notes in computer science* (1998), 417–431.

[69] Solomon, M. M. Algorithms for the vehicle routing and scheduling problem with time window contraints. *Operational Research 35* (1987), 254–265.

[70] Souza, M. J. F. Inteligência computacional para otimização, 2011. [Online; acessado em 31 de Janeiro de 2011].

[71] Souza, M. J. F., Coelho, I. M., Ribas, S., Santos, H. G., Merschmann, L. H. C. A hybrid heuristic algorithm for the open-pit-mining operational planning problem. *European Journal of Operational Research 207*, 2 (2010), 1041–1051.

[72] Subramanian, A., Drummond, L., Bentes, C., Ochi, L., Farias, R. A parallel heuristic for the vehicle routing problem with simultaneous pickup and delivery. *Computers and Operations Research 37* (2010), 1899–1911.

[73] Taillard, E., Badeau, P., Gendreau, M., Guertin, F., Potvin, J. Y. A tabu search heuristic for the vehicle routing problem with soft time windows. *Transportation Science 31* (1997), 170–186.

[74] Thamgiah, S., Osman, I., Sun, T. Metaheuristics for the vehicle routing problem with time windows. Tech report, Artificial Intelligence and Robotics Laboratory, Computer Science Department, Slippery Rock University, PA, 1995.

[75] Toth, P., Vigo, D. The granular tabu search and its application to the vehicle routing problem. *INFORMS Journal on Computing 15*, 4 (2003), 333–4.

[76] Xu, H., Chen, Z. L., Rajagopal, S., Arunapuram, S. Solving a practicalpickup and delivery problem. Tech report, Technical Report, Department of Systems Engineering, University of Pennsylvania, 2001.