Estruturas Eficientes para Buscas Locais Aplicadas a um Problema de Roteamento de Veículos

Igor M. Coelho[1,2] , Giancarlo F. de Roberto[1] , Raquel M. de Souza[1]
Paulo E. D. Pinto[1] , Luiz Satoru Ochi[2]

[1] Departamento de Computação – Universidade do Estado do Rio de Janeiro
Rua São Francisco Xavier, 524 – Bloco B, Sala 6019 – CEP 20550-900
Rio de Janeiro (RJ), Brasil
[2]Instituto de Computação – Universidade Federal Fluminense
Rua Passo da Pátria, 154 – Bloco E, 3^o andar – CEP 24.210-240
Niterói (RJ), Brasil

{igor.machado,pauloedp}@ime.uerj.br, giandroberto@yahoo.com.br
raquelmarcolino25@gmail.com, satoru@ic.uff.br

Abstract. *Vehicle Routing Problems are optimization problems with many practical applications, usually seeking to minimize the traveled distance in a route satisfying the demands of a given set of customers. We consider a single-vehicle routing problem with delivery and pickup customers, where only the deliveries are obligatory. However, for each pickup, a revenue is gained in return. Heuristic techniques usually adopted consist in using a set of operations that systematically change the route, by means of the so-called Local Search. By studying the nature of each operation, it is possible to reduce the space and time computational complexity. It is presented a technique inspired in the algorithm Range Minimum Query to reduce the computational time in the Local Search, when infeasible solutions are considered during the search. Finally, the behavior of local search methods is studied when three different infeasibility criteria are considered.*
KEYWORDS*: Vehicle Routing Problem. Range Minimum Query. Variable Neighborhood Search. Metaheuristics.*

Resumo. *Problemas de Roteamento de Veículos são problemas de otimização com diversas aplicações práticas, buscando-se geralmente otimizar a distância total percorrida em uma rota que visite um conjunto de clientes e satisfaça suas demandas. Consideramos um problema de roteamento no qual um único veículo é utilizado para fazer entregas e coletas, sendo que somente as entregas são obrigatórias. Para cada coleta, um valor em benefício é recebido em retorno. Técnicas heurísticas comumente adotadas consistem na utilização de um conjunto de operações que alteram sistematicamente a rota, através do que é definido por Buscas Locais. Estudando a natureza de cada operação envolvida nas buscas locais, é possível reduzir a complexidade de espaço e de tempo computacional. É apresentada uma técnica baseada no algoritmo de Range Minimum Query para reduzir o tempo computacional das buscas locais para o problema de roteamento em questão, com uma função de penalização de inviabilidade por valor máximo. Finalmente, é estudado o comportamento de métodos de busca local considerando três critérios de inviabilidade diferentes.*
PALAVRAS-CHAVE*: Problema de Roteamento de Veículos. Range Minimum Query. Busca em Vizinhanças Variáveis. Metaheurísticas.*

1 Introdução

O Problema de Roteamento de Veículos de Rota Única com Entregas Obrigatórias e Coletas Seletivas (SVRPDSP), ou *Single Vehicle Routing Problem with Deliveries and Selective Pickups* (Gribkovskaia et al., 2008), é uma variante do Problema de Roteamento de Veículos (PRV) clássico. Este problema também pode ser encontrado na literatura como Problema de Roteamento de Veículos de Rota Única com *Backhauls* Irrestrito, ou *Single Vehicle Routing Problem with Unrestricted Backhauls* (Süral and Bookbinder, 2003).

No SVRPDSP entregas devem ser efetuadas a uma série de clientes, mas também existem itens a serem coletados se possível, dependendo da receita gerada por estas coletas. O SVRPDSP pertence à classe $\mathcal{NP}$-Difícil, uma vez que ele pode ser reduzido ao Problema do Caixeiro Viajante quando nenhum cliente necessita de serviço de coleta. Formalmente, o problema consiste em um conjunto de clientes $\mathcal{C} = \{c_1, \ldots, c_n\}$ com demandas de entrega d_i e de coleta p_i, onde para cada coleta (não obrigatória) efetuada um benefício r_i é recebido. Um caminhão com capacidade limitada Q parte de um depósito c_0 e retorna de forma a satisfazer todas demandas de entrega e apenas algumas coletas (limitadas à capacidade do caminhão). O objetivo consiste em minimizar a distância total percorrida, dada por uma matriz de custos $c(i, j)$ entre os nós i e j, somando-se o valor recebido em benefícios pelas coletas. Dado S como o conjunto de todas possíveis rotas que satisfazem as restrições, o objetivo pode ser visto como na Equação (1).

$$\min_{R \in S} \sum_{(i,j) \in R} c(i,j) - \sum_{k \in R, \text{ tal que } k \text{ é coleta}} r_k \tag{1}$$

A Figura 1 ilustra uma rota com coletas e entregas de um SVRPDSP. Nela, um depósito é representado por um retângulo, enquanto os clientes são representados por círculos. Cada cliente tem demandas de entrega e de coleta, representados pelo número no lado esquerdo e direito de cada círculo, respectivamente. Os benefícios em fazer uma coleta são representados por estrelas ao lado de cada cliente.

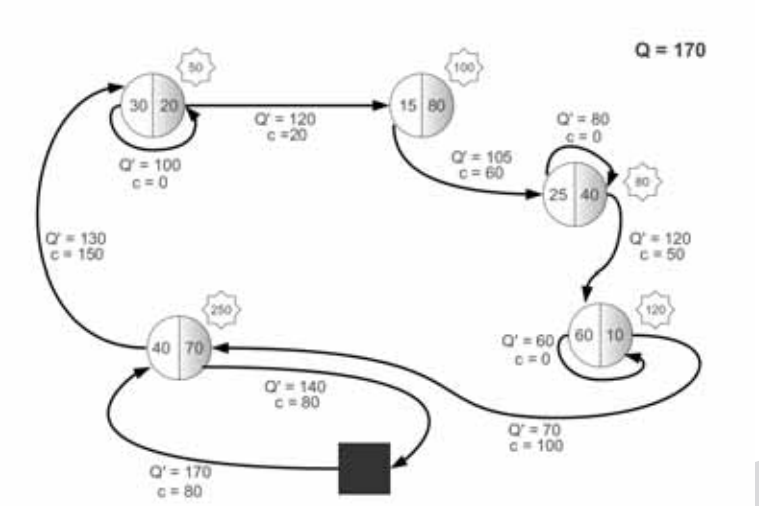

Figura 1. O Problema de Roteamento de Veículos de Rota Única com Entregas Obrigatórias e Coletas Seletivas

Na solução apresentada, o veículo parte do depósito com 170 unidades, percorre uma distância de valor 80 e visita o primeiro cliente entregando 40 unidades. O veículo agora carrega 130 unidades. O segundo cliente é visitado, com a entrega de 30 unidades seguido da coleta de 20 unidades, com um benefício de valor 50. No terceiro cliente pode-se perceber que é feita uma entrega e nenhuma coleta. No quarto e quinto clientes são feitas entregas e coletas conjuntas. Antes de retornar ao depósito o veículo passa no primeiro cliente novamente para fazer uma coleta de 70 unidades, com um benefício de valor 250. Desta forma, a distância total percorrida é de $80 + 150 + 0 + 20 + 60 + 0 + 50 + 0 + 100 + 80 = 540$, representando a primeira parcela da Equação (1). O total de benefícios é de $50 + 80 + 120 + 250 = 500$, representando a segunda parcela da Equação (1). Logo, o custo dessa solução para o problema é de $540 - 500 = 40$.

Neste trabalho, são propostas estruturas de dados eficientes para lidar com o SVRPDSP, considerando algoritmos recentes propostos para o problema na literatura. Este trabalho é organizado da seguinte maneira. A Seção 2 apresenta trabalhos relacionados da literatura, incluindo estratégias de busca recentes utilizando a tecnologia das *Graphics Processing Units* (GPUs). Na Seção 3 é apresentada a Busca Geral em Vizinhanças Variáveis, uma metaheurística com capacidade de escapar de ótimos locais por meio da troca sistemática de estruturas de vizinhança. Na Seção 4, são apresentadas melhorias nas estruturas de dados relacionadas a cada tipo de vizinhança de interesse no problema em foco. A Seção 5 apresenta resultados experimentais que exploram a capacidade das buscas locais escaparem de soluções inviáveis, ideia central das novas estruturas de dados propostas neste trabalho. Finalmente, a Seção 6 conclui o trabalho e apresenta extensões futuras das ideias discutidas neste trabalho.

2 Trabalhos Relacionados

Na literatura, as abordagens mais comuns para o Problema de Roteamento de Veículos de Rota Única com Entregas Obrigatórias e Coletas Seletivas (SVRPDSP) são por meio de métodos de programação matemática e metaheurísticas.

O SVRPDSP foi proposto por Süral and Bookbinder (2003), onde um modelo de programação matemática é utilizado para resolver problemas-teste criados pelos autores. Em Gribkovskaia et al. (2008) o problema é novamente abordado, porém com outra nomenclatura. Um novo modelo de programação matemática é proposto para resolver outros problemas-teste criados pelos autores. Visto que o modelo não é capaz de encontrar bons resultados, estratégias heurísticas são desenvolvidas. Basicamente, 12 heurísticas de construção e refinamento são desenvolvidas, e um *gap* médio de $30,62\%$ é obtido pela melhor heurística proposta **CI5+IMP**. A partir daí 12 estratégias inspiradas na metaheurística Busca Tabu são desenvolvidas, tendo bons resultados comparados a um limite inferior do problema.

Um problema similar ao SVRPDSP muito estudado na literatura é conhecido como Problema de Roteamento de Veículos com *Backhauls* (PRVB). O PRVB é uma extensão do PRV clássico, que considera clientes de entrega (*linehaul*) e de coleta (*backhaul*) e, por restrições operacionais, na rota de cada caminhão as entregas devem preceder as coletas. Cada veículo deve sair do depósito e retornar ao depósito, respeitando uma capacidade máxima estabelecida para toda a frota (frota homogênea), onde nem a soma das entregas e nem das coletas devem exceder essa capacidade. Todos os clientes (*linehaul* e *backhaul*) devem ser visitados, sendo que o problema se reduz ao PRV clássico, quando não exis-

tem clientes *backhaul*. Busca-se, então, minimizar a distância total percorrida. Existem problemas-teste para o problema criados por Goeschalekx and Jacobs (1989) e Toth and Vigo (1999). Abordagens heurísticas para o problema podem ser encontradas em Brandão (2006), Tavakkoli-Moghaddam et al. (2006) e Gajpal and Abad (2009).

Em Coelho et al. (2012b) é proposto um método de busca inspirado pela metaheurística Busca Geral em Vizinhança Variável (GVNS). Para efetuar a busca, são utilizadas quatro estruturas de vizinhança diferentes: *swap*, 2-Opt, OrOpt-1 e OrOpt-2. Também é apresentada uma heurística construtiva híbrida, que utiliza técnicas de programação matemática para resolver subproblemas de forma exata, gerando soluções iniciais com valores próximos ao limite inferior do problema.

Bruck et al. (2012) propõe uma metaheurística inspirada nos Algoritmos Meméticos para encontrar soluções de melhor qualidade. Algumas buscas locais propostas anteriormente são utizadas novamente, juntamente com uma nova técnica de diversificação. A comparação é feita através do conjunto de 68 problemas-teste de Gribkovskaia et al. (2008). Em Coelho et al. (2012a), é proposta a paralelização utilizando *Graphics Processing Units* (GPUs) de três das quatro buscas locais utilizadas neste trabalho. A ideia é diminuir o tempo computacional da etapa mais pesada do algoritmo, a busca local. Porém, o foco da paralelização em GPU não melhora a qualidade das soluções encontradas pelo algoritmo, sendo este paralelizado somente parcialmente.

Em Coelho et al. (2015) é apresentada uma paralelização GPU completa de uma metaheurística GVNS para o SVRPDSP. Uma nova estratégia de perturbação é desenvolvida, de forma a reduzir a comunicação com a GPU (reduzindo tempo computacional) e proporcionar maior diversificação durante a busca. Com isso, novas soluções de melhor qualidade são encontradas para os 68 problemas-teste clássicos, e mais 32 problemas-teste de maior porte foram gerados para aprofundar a análise do impacto da paralelização GPU.

3 Busca Geral em Vizinhanças Variáveis

Uma metaheurística é uma heurística de aplicação geral, com capacidade de escapar de ótimos locais (Mladenović and Hansen, 1997). A metaheurística Busca Geral em Vizinhanças Variáveis (GVNS) é inspirada na ideia de buscar boas soluções por meio de estruturas de vizinhança diferentes (Hansen et al., 2008b,a; Hansen and Mladenović, 2001). Uma das motivações desta abordagem é que um *ótimo global* é, consequentemente, um *ótimo local* para todas as possíveis vizinhanças de uma solução.

Assim, um método de busca local comumente empregado em um cenário com múltiplas estruturas de vizinhança é chamado *Variable Neighborhood Descent* (VND). Seja s a solução corrente e $\mathcal{N}$ a vizinhança de uma solução estruturada em r vizinhanças distintas, isto é, $\mathcal{N} = \mathcal{N}^{(1)} \cup \mathcal{N}^{(2)} \cup \cdots \cup \mathcal{N}^{(r)}$. O VND inicia-se analisando a primeira estrutura de vizinhança $\mathcal{N}^{(1)}$. A cada iteração, o método gera o melhor vizinho s' da solução corrente s na vizinhança $\mathcal{N}^{(k)}$, $k \in \{1, \ldots, r\}$. Caso s' seja melhor que s, então s' passa a ser a nova solução corrente e retorna-se à vizinhança $\mathcal{N}^{(1)}$. Caso contrário, passa-se para a próxima estrutura de vizinhança $\mathcal{N}^{(k+1)}$. O método termina quando não é possível encontrar uma solução $s' \in \mathcal{N}$ melhor que a solução corrente.

Na literatura, o algoritmo mais eficiente para o SVRPDSP foi proposto por Coelho et al. (2015), baseado na metaheurística VNS. O algoritmo conta com quatro estruturas de vizinhança distintas, sendo elas: Swap, 2-Opt, OrOpt-1 e OrOpt-2. Tais estruturas são

comumente utilizados em problemas de permutação em uma rota, como o Problema do Caixeiro Viajante (PCV). Para se assemelhar mais ao PCV, Coelho et al. (2015) separa os nós de entrega e coleta simultânea como dois nós isolados, desta maneira, nós com rótulo positivo efetuam coleta, enquanto nós com rótulo negativo efetuam entregas. Um nó especial de rótulo 0 é utilizado para simbolizar o depósito de onde sai e retorna o veículo. Embora isto aumente o número de elementos na rota, esta técnica facilita a implementação de diversas estratégias de busca local.

A estrutura de vizinhança Swap consiste na troca da ordem de visita entre dois clientes da rota. A Figura 2 exemplifica a troca entre os clientes +4 e -1. Já a estrutura de vizinhança 2-Opt consiste na remoção de duas arestas da rota, seguido da inclusão de duas novas arestas. A Figura 3 exemplifica uma operação de 2-Opt que remove os arcos entre $(-1, +4)$ e $(+3, 0)$. Finalmente, a operação OrOpt-c consiste na realocação de c clientes consecutivos na rota. A Figura 4 ilustra a aplicação de uma operação OrOpt-1 para a realocação do nó -6 para uma posição após o cliente $+5$.

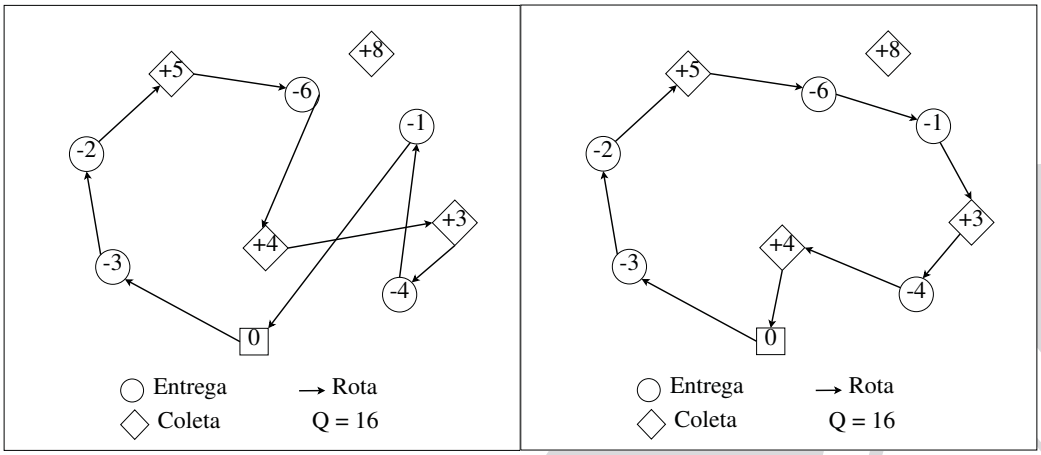

Figura 2. *Swap* **faz a troca dos clientes +4 e -1].**

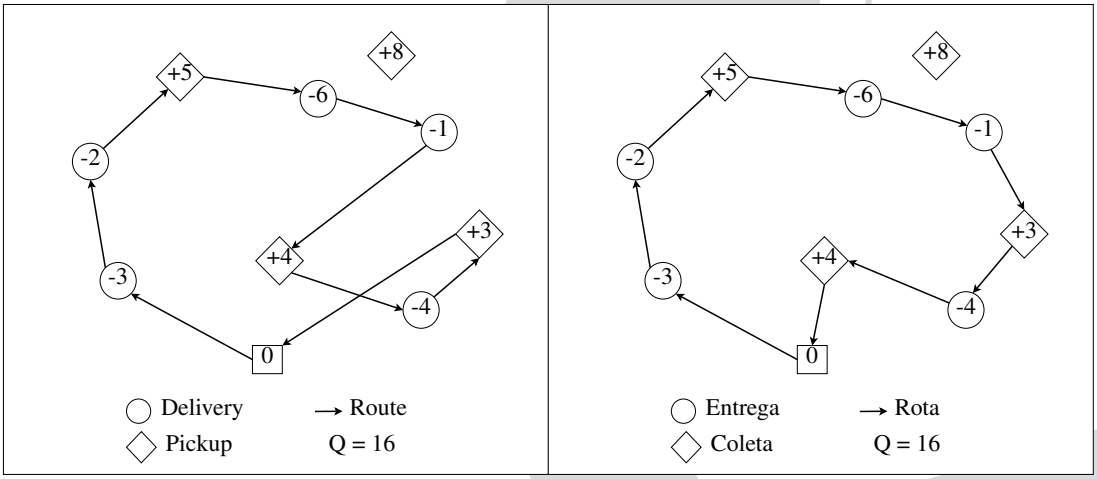

Figura 3. 2-Opt remove os arcos entre $(-1, +4)$ e $(+3, 0)$, refazendo a rota.

De forma a escapar de ótimos locais, uma estratégia de perturbação é incorpo-

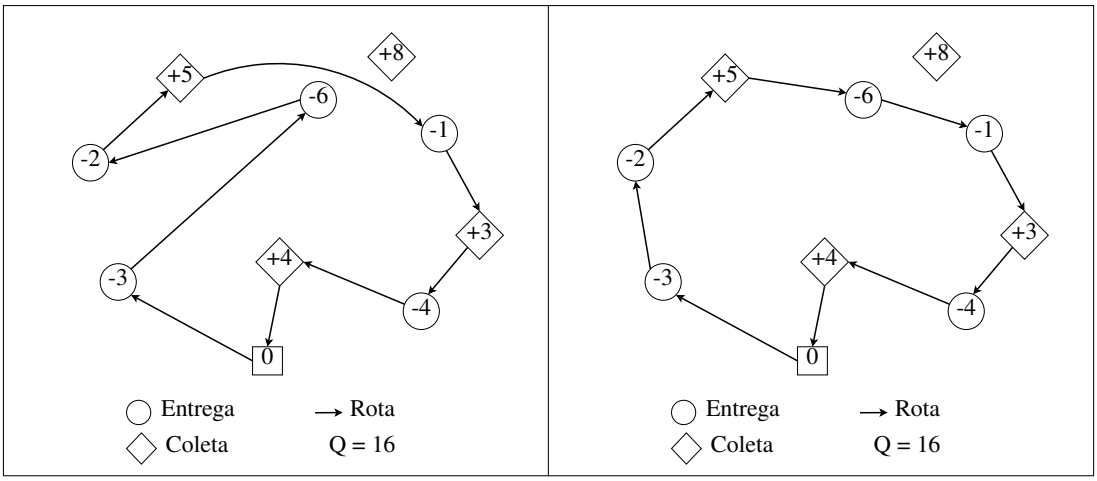

Figura 4. *OrOpt-1* **move o cliente -6 para uma posição após o cliente +5.**

rada ao algoritmo. Este algoritmo é geralmente chamado de Busca Geral em Vizinhanças Variáveis (GVNS). O Algoritmo 1 apresenta a ideia de um GVNS utilizando o método de busca VND descrito anteriormente.

Algoritmo 1: *GVNS*

Entrada: Solução s, Função $f(.)$, *kMax* Estruturas de Vizinhança, $IterMax$
Saída: Solução s^* de qualidade possivelmente superior a s de acordo com f

```
1  iter ← 1;
2  enquanto iter < IterMax faça
3  |  s' ← Perturbação(s);
4  |  s'' ← VND(s', f);
5  |  se f(s'') < f(s) então
6  |  |  s ← s'';
7  |  |  iter ← 0;
8  |  fim
9  |  iter ← iter + 1;
10 fim
11 retorna s;
```

4 Melhorias na Etapa das Buscas Locais

As quatro estruturas de vizinhança utilizadas (Swap, 2-Opt, OrOpt-1 e OrOpt-2) possuem um número quadrático de operações, comumente chamadas de *movimentos*, para uma dada rota de entrada. Porém, uma restrição do SVRPDSP é de que a capacidade do caminhão seja respeitada (não seja excedida) durante toda a rota (visita aos n clientes), o que pode acarretar a inclusão de um laço de tempo linear para a verificação da viabilidade da rota, após cada operação. Coelho et al. (2015) discute que é interessante relaxar esta restrição, permitindo que rotas inviáveis sejam aceitas pelo algoritmo de busca, que atribui um valor alto a tais soluções para que fiquem menos atrativas. Neste trabalho, exploramos três tipos de penalização: linear; por valor máximo e binária.

A penalização linear é utilizada por Coelho et al. (2015), considerando como função

de penalização o somatório das cargas excedidas no caminhão ao longo da rota. A vantagem desta técnica é que ela permite que o algoritmo diferencie rotas que estão *mais ou menos inviáveis*. O ponto negativo é que acarreta uma verificação linear na rota após cada movimento na busca. A penalização binária sinaliza se a rota é viável ou inviável, não fornecendo informações quantitativas da inviabilidade ocorrida. Este trabalho explora um novo tipo de penalização que somente considera o ponto *mais inviável* da rota.

4.1 Penalização por Valor Máximo

A penalização por valor máximo tem a vantagem de permitir a computação do valor de inviabilidade em tempo $\Theta(1)$, para cada uma das estruturas de vizinhança utilizadas. Esta estratégia exige que informações sejam pré-processadas, e em uma primeira abordagem consideraremos a criação de três novas matrizes de tamanho $\mathcal{O}(n^2)$: uma matriz de somas acumuladas, chamada $MSum$; uma de máximos acumulados, $MMax$, e outra para determinar se há clientes de entrega em determinado intervalo da rota, $MDel$. Seja z o marcador de fim de rota, para uma solução S (note que apenas clientes de coleta podem aparecer depois do z, pois não serão visitados).

Para encontrar o custo de inviabilidade em uma rota após uma operação 2-Opt(i,j), basta computar o valor máximo entre 3 casos, para a remoção das arestas $(i-1,i)$ e $(j,j+1)$, e a inclusão das arestas $(i-1,j)$ e $(i,j+1)$:

1. $MMax(0,i)$;
2. $MSum(0,i) + MMax(j,i+1)$;
3. $MSum(0,j) + MMax(j+1,z)$.

De acordo com Coelho et al. (2015), movimentos do tipo OrOpt-c podem ser vistos como uma operação de 3-Opt (quebra e religação de 3 arcos na rota). Para um 3-Opt(i,j,k) considere dois casos. Se $k < z$, então devemos determinar o máximo entre:

1. $MMax(0,i)$;
2. $MSum(0,i) + MMax(j+1,k)$;
3. $MSum(0,i) + MSum(j+1,k) + MMax(i+1,j)$;
4. $MSum(0,k) + MMax(k+1,z)$.

Se $j < z \leq k$, calculamos $max(MMax(0,i), MSum(0,1) + MMax(j+1,z))$. Se $i < z$ e $j > z$, determinamos o valor máximo entre:

1. $MMax(0,i)$;
2. $MSum(0,i) + MMax(j+1,k)$;
3. $MSum(0,i) + MSum(j+1,k) + MMax(i+1,z)$.

Por fim, a operação de Swap pode ser vista como uma operação de 4-Opt (quebra e religação de 4 arcos). Para encontrar a carga máxima na rota após uma operação de 4-Opt(i,j,k,l), considere os casos. Caso $k < z$:

1. $MMax(0,i)$;
2. $MSum(0,i) + MMax(k+1,l)$;
3. $MSum(0,i) + MSum(k+1,l) + MMax(j+1,k)$;
4. $MSum(0,i) + MSum(k+1,l) + MSum(j+1,k) + MMax(i+1,j)$;
5. $MSum(0,i) + MSum(k+1,l) + MSum(j+1,k) + MSum(i+1,j) + MMax(l+1,z)$.

Caso contrário, devemos determinar o máximo entre:

1. $MMax(0,i)$;
2. $MSum(0,i) + MMax(k+1,l)$;
3. $MSum(0,i) + MSum(k+1,l) + MMax(j+1,z)$.

4.2 Penalização por Valor Máximo de Forma Eficiente

A matriz $MSum$ pode ser substituída por um vetor $VSum$ de cargas acumuladas, em ordem da rota, utilizando a posição 0 como sentinela (valor 0), sendo que ao se atingir o ponto z a acumulação é reiniciada a partir do zero. Para se obter $MSum(i, j)$ é calculada a diferença entre $VSum(j) - VSum(i - 1)$. Perceba que intervalos contendo z não serão relevantes para o cálculo da solução.

De forma análoga, a matriz $MDel$ pode ser substituída por um vetor $VDel$ que acumula a quantidade de clientes (com demandas) de entrega, também em ordem da rota, igualmente usando um sentinela na posição 0. Para se obter $MDel(i, j)$, retorna-se verdadeiro quando a diferença $VDel(j) - VDel(i - 1)$ é positiva, e falso caso contrário.

4.2.1 Melhoria da matriz MMax com *Range Minimum Query*

A melhoria da matriz $MMax$ pode ser feita tanto em termos de espaço de armazenamento quanto em termos de tempo de geração, usando a técnica de RMQ (*Range Minimum Query*) com tabela esparsa. Utiliza-se RMQ para resolver o problema de busca de mínimo (ou máximo) em um intervalo $[i, j]$ qualquer de um vetor V dado, no qual é feito um número grande de consultas. A solução por força bruta para esse problema não requer nenhum pré-processamento e cada busca tem complexidade $\mathcal{O}(n)$.

Podemos obter a redução da complexidade através da utilização de estruturas auxiliares. Em casos como o atual, em que os dados são estáticos, é utilizada uma tabela esparsa T de dimensões $(\lfloor \log_2 n \rfloor + 1) \times n$ (reduzindo, assim, o espaço de armazenamento necessário) gerada em tempo $\mathcal{O}(n \log n)$ (reduzindo o tempo necessário). A consulta permanece $\mathcal{O}(1)$.

Cada linha $i (0 \leq i \leq \lfloor \log_2 n \rfloor)$ da tabela T contém, para cada posição j do vetor V, o índice do menor elemento compreendido no intervalo dado por $[j, j + 2^i - 1]$. A linha 0 contém o índice de cada elemento, e as linhas subsequentes são obtidas tomando-se $i \in \{1, \ldots, \lfloor log_2 n \rfloor\}$, $j \in \{0, \ldots, n - 2 \times (i - 1) - 1\}$:

$$T(i,j) = \begin{cases} T(i-1,j), & \text{se } V(T(i-1,j)) \leq V(T(i-1,j+2^{i-1})) \\ T(i-1,j+2^{i-1}), & \text{c.c.} \end{cases}$$

O preenchimento de cada célula $T(i, j)$ resume-se, portanto, a uma simples comparação entre dois intervalos que, se sobrepostos, são equivalentes ao intervalo de i a j desejado. O mesmo princípio é utilizado em uma consulta $RMQ(i, j)$, conforme mostraremos adiante.

Para ilustrar a criação de T consideremos o vetor V descrito abaixo:

1	6	4	7	3	5	10	2	8

Geramos, linha por linha, a tabela esparsa para V como foi apresentado, e temos a seguinte estrutura T:

0:	0	1	2	3	4	5	6	7	8
1:	0	2	2	4	4	5	7	7	-
2:	0	4	4	4	7	7	-	-	-
3:	0	7	-	-	-	-	-	-	-

Para responder a uma consulta do menor valor de qualquer intervalo de i a j de V, $i \leq j$ selecionamos a linha $k = \lfloor \log_2(j - i + 1) \rfloor$ da Tabela T, que contém as respostas precalculadas para o maior intervalo que seja potência de 2 e que esteja contido no intervalo $[i, j]$. Faz-se, então uma comparação análoga à do preenchimento da Tabela T:

$$RMQ(i,j) = \begin{cases} T(k,i), & \text{se } V(T(k,i)) \leq V(T(k, j - 2^k + 1)) \\ T(k, j - 2^k + 1), & \text{cc} \end{cases}$$

Vejamos como exemplo o cálculo de RMQ(3,8). Neste caso, $k = \lfloor \log_2(8 - 3 + 1) \rfloor = 2$. Então RMQ(3,8) = T(2,5) = 7, pois $V(T(2,3)) = V(4) = 7 > V(T(2,5)) = V(7) = 2$. Observe que tanto o cálculo de RMQ(i, j) quanto o preenchimento de cada célula da Tabela T é feito em $\mathcal{O}(1)$. Com isso, o preenchimento de T é feito em $\mathcal{O}(n \log n)$ e a consulta em $\mathcal{O}(1)$.

Assim, podemos perceber que os máximos acumulados antes armazenados em $MMax$ podem ser obtidos também em tempo constante por meio da aplicação dessa técnica sobre o vetor $VSum$, após gerada a tabela esparsa durante a etapa de pré-processamento. Para uma rota que se estende do ponto $i > 0$ ao ponto $j \geq i$, a carga máxima acumulada é dada por $VSum(RMQ_{VSum}(i,j)) - VSum(i-1)$.

Para que consultas a trechos de rota no sentido inverso sejam suportadas por essa adaptação, as informações em $VSum$ não são suficientes. Seja, então, o vetor $VRSum$, gerado de forma análoga a $VSum$ mas com acumulação de cargas no sentido invertido em relação ao da rota original. Após construção da tabela esparsa relativa a $VRSum$, torna-se possível determinar a acumulação máxima para trechos de j até i, com $j > i$ através de $VRSum(RMQ_{VRSum}(i,j)) - VRSum(j+1)$.

4.2.2 Exemplo de Geração das Estruturas Eficientes

Considere a rota descrita abaixo pelas variações de carga associadas ao depósito e a cada cliente de entrega ou coleta, com marcador $z = 7$. Suponha que foi realizada uma operação 2-Opt(2,6) e queremos determinar o custo:

Rota	0	1	2	3	4	5	6	7	8	9
	10	-2	3	1	-2	-1	-5	0	6	2

Perceba que a capacidade máxima de carga é 10, conforme foi carregada no depósito (ponto 0). Observe que as posições 8 e 9 correspondem a coletas não realizadas. Inicialmente, construimos as estruturas $VSum$, $VRSum$ e $VDel$ conforme explicitado.

VSum	0	1	2	3	4	5	6	7	8	9
	10	8	*11*	*12*	10	9	4	0	6	8

VRSum	0	1	2	3	4	5	6	7	8	9
	4	-6	-4	-7	-8	-6	-5	0	8	2

VDel	0	1	2	3	4	5	6	7	8	9
	0	1	1	1	2	3	4	4	4	4

É possível observar a **inviabilidade** a ser penalizada, que ocorre após as coletas dos clientes 2 e 3, que excedem a capacidade de carga do veículo em 1 e 2 unidades, respectivamente.

Geramos, a seguir, as tabelas esparsas T e T', com base em $VSum$ e $VRSum$ respectivamente, que serão auxiliares na determinação da acumulação máxima com RMQ:

T										
	0	1	2	3	4	5	6	7	x	x
	0	2	3	3	4	5	6	x	x	-
	3	3	3	3	4	x	x	-	-	-
	3	x	x	-	-	-	-	-	-	-

T'										
	0	1	2	3	4	5	6	7	x	x
	0	2	2	3	5	6	7	x	x	-
	0	2	2	6	7	x	x	-	-	-
	7	x	x	-	-	-	-	-	-	-

Por fim, basta converter a expressão para determinação do custo de inviabilidade do 2-Opt(2,6). Desse modo, selecionamos o máximo dentre os seguintes valores:

1. $VSum(RMQ_{VSum}(0,2)) = \mathbf{11}$
2. $VSum(2) + VRSum(RMQ_{VRSum}(3,6)) - VRSum(7) = 11 + (-5) - 0 = \mathbf{6}$
3. $VSum(6) + VSum(RMQ_{VSum}(7,7)) - VSum(6) = \mathbf{0}$

Concluímos, portanto, que a carga máxima na rota após a operação 2-Opt será **11**. Como o máximo atual é **12**, ocorrerá um decréscimo de uma unidade na inviabilidade da rota, valor multiplicado por um fator muito alto, tornando assim esta operação bastante interessante para o algoritmo de busca.

5 Impacto da Penalização na Busca Local

Nesta seção, são testadas as buscas locais apresentadas anteriormente considerando os três tipos diferentes de funções de penalização discutidos anteriormente. Denominaremos por P1, a penalização linear; por P2, a penalização por valor máximo; e, por P3, a penalização binária.

Como visto anteriormente para a P2, utilizando tabelas estendidas é possível diminuir a complexidade de espaço, ainda assim mantendo um tempo de acesso $\mathcal{O}(1)$ para acesso. Porém, é importante analisar o impacto da mudança da função de penalização em relação à capacidade das buscas locais para escaparem de soluções inviáveis. Neste experimento, foram utilizados os 68 problemas-teste propostos por Gribkovskaia et al. (2008) nos grupos *small*, *medium* e *large*. Partindo de 30 soluções aleatórias inviáveis para cada problema-teste, foi contabilizado quantos movimentos o algoritmo utiliza até que a solução corrente se torne viável.

A Tabela 1 compara a estratégias de penalização P1 e P2. Pela tabela é possível perceber que, na média, com P1 o algoritmo encontra uma solução viável com 2,06 operações, enquanto precisa de 2,12 operações utilizando P2. O uso de P2 acarreta um aumento de 3,08% no número de operações em relação a P1, demonstrando assim que o método se torna um pouco mais "cego" ao não considerar as informações de carga excedida em toda a rota, mas somente no ponto de inviabilidade máximo.

Na Tabela 2, é feita uma comparação entre as estratégias de penalização P1 e P3. Por esta tabela, a utilização da estratégia P3 acarreta em um aumento médio de 678% no número de operações para viabilizar uma solução de entrada inviável, em relação a P1. Este resultado era esperado, pois ao utilizar apenas por uma sinalização binária entre viável/inviável, o método considera soluções inviáveis como *igualmente ruins*, mesmo que a situação de sobrecarga seja radicalmente diferente nas rotas.

Tabela 1. Comparação das estratégias de penalização P1 e P2

Problemas-teste	P1	P2	Aumento(%)
small	1,80	1,87	3,64
medium	2,17	2,25	3,45
large	2,21	2,26	2,26
Média	**2,06**	**2,12**	**3,08**

Tabela 2. Comparação das estratégias de penalização P1 e P3

Problemas-teste	P1	P3	Aumento(%)
small	1,80	8,70	383,47
medium	2,17	19,47	796,48
large	2,21	19,95	803,58
Média	**2,06**	**16,04**	**678,74**

6 Conclusões e Trabalhos Futuros

Este trabalho teve foco em uma variante do Problema de Roteamento de Veículos (PRV), de classe $\mathcal{NP}$-Difícil. O PRV clássico consiste em visitar um conjunto de clientes com uma frota de veículos de capacidade limitada, percorrendo a rota com uma distância mínima. Na variante atacada, somente um veículo é utilizado e existem clientes com coleta opcional de produtos, que caso sejam satisfeitas, proporcionam um ganho extra, muitas vezes relacionado na literatura a processos de reciclagem em cadeias verdes de suprimentos. Tal problema é chamado SVRPDSP.

Neste trabalho, propomos novas estruturas de dados que permitem reduzir a complexidade computacional na etapa de buscas local. Porém, tais estruturas são dependentes da função de penalização aplicada a soluções inviáveis, nas quais a capacidade do veículo é excedida durante a rota. São estudados três tipos de penalização: linear, por valor máximo e binária. Experimentos computacionais demonstraram que a penalização linear proporciona mais informação ao método de busca, acelerando o processo de convergência a ótimos locais, enquanto a penalização binária tem um comportamente oposto. A penalização por valor máximo tem uma performance ligeiramente pior que a linear, mas proporciona a possibilidade de se otimizar as estruturas de dados auxiliares da busca.

Utilizando variantes do algoritmo *Range Minimum Query* foi possível reduzir a complexidade da busca, com tempo de acesso $\mathcal{O}(1)$ e utilizando espaço $\Theta(\log N \times N)$. Porém, a atualização de tais estruturas precisa ser refeita completamente a cada busca. Em trabalhos futuros, tais estruturas podem ser utilizadas para aperfeiçoar os melhores algoritmos de GPU da literatura, bem como novas estruturas de dados podem ser propostas para reduzir o tempo de atualização das estruturas após cada etapa de busca.

Agradecimentos

Os autores agradecem à agências CNPq, CAPES e FAPERJ, que apoiaram o desenvolvimento desta pesquisa.

Referências

Brandão, José. (2006). A new tabu search algorithm for the vehicle routing problem with backhauls. *European Journal of Operational Research*, v. 173, p. 540–555.

Bruck, Bruno P.; dosSantos, André G. and Arroyo, José E. C. (2012). Hybrid metaheuristic for the single vehicle routing problem with deliveries and selective pickups. *Proceedings of the WCCI 2012 IEEE World Congress on Computational Intelligence*, p. 10–15, Brisbane, Australia.

Coelho, Igor M.; Munhoz, Pablo Luiz A.; Haddad, Matheus N.; Souza, Marcone Jamilson F. and Ochi, Luiz S. (2012)a. A hybrid heuristic based on General Variable Neighborhood Search for the Single Vehicle Routing Problem with Deliveries and Selective Pickups. *Electronic Notes in Discrete Mathematics*, v. 39, n. 0, p. 99–106.

Coelho, Igor M.; Ochi, Luis S.; Munhoz, Pablo L.A.; Souza, Marcone J.F.; Farias, Ricardo and Bentes, Cristiana. June(2012)b. The Single Vehicle Routing Problem with Deliveries and Selective Pickups in a CPU-GPU Heterogeneous Environment. *2012 IEEE 14th International Conference on High Performance Computing and Communication & 2012 IEEE 9th International Conference on Embedded Software and Systems*, p. 1606–1611. IEEE, June(2012)b.

Coelho, I.M.; Munhoz, P.L.A.; Ochi, L.S.; Souza, M.J.F.; Bentes, C. and Farias, R. (2015). An integrated cpu–gpu heuristic inspired on variable neighbourhood search for the single vehicle routing problem with deliveries and selective pickups. *International Journal of Production Research*, v. 0, n. 0, p. 1–18. doi: 10.1080/00207543.2015.1035811. URL http://dx.doi.org/10.1080/00207543.2015.1035811.

Gajpal, Yuvraj and Abad, P.L. (2009). Multi-ant colony system (macs) for a vehicle routing problem with backhauls. *European Journal of Operational Research*, v. 196, p. 102 – 117. ISSN 0377-2217. doi: DOI:10.1016/j.ejor.2008.02.025. URL http://www.sciencedirect.com/science/article/B6VCT-4RY6WP4-2/2/56da7aec7d9797000d6b36125be3880c.

Goeschalekx, M. and Jacobs, B. (1989). The vehicle routing problem with backhauls. *European Journal of Operational Research*, v. 42, p. 39–51.

Gribkovskaia, Irina; Laporte, Gilbert and Shyshou, Aliaksandr. Setembro(2008). The single vehicle routing problem with deliveries and selective pickups. *Computers and Operations Research*, v. 35, p. 2908–2924. ISSN 0305-0548. doi: 10.1016/j.cor.2007.01.007. URL http://portal.acm.org/citation.cfm?id=1343112.1343206.

Hansen, P.; Mladenović, N. and Pérez, J. A. M. (2008)a. Variable neighborhood search. *European Journal of Operational Research*, v. 191, p. 593–595.

Hansen, P.; Mladenović, N. and Pérez, J. A. M. (2008)b. Variable neighborhood search: methods and applications. *4OR: A Quarterly Journal of Operations Research*, v. 6, p. 319–360.

Hansen, Pierre and Mladenović, Nenad. May(2001). Variable neighborhood search: Principles and applications. *European Journal of Operational Research*, v. 130, n. 3, p. 449–467.

Mladenović, N. and Hansen, P. (1997). Variable neighboorhood search. *Computers and Operations Research*, v. 24, p. 1097–1100.

Süral, Haldun and Bookbinder, James H. (2003). The single-vehicle routing problem with unrestricted backhauls. *Networks*, v. 41, p. 127–136.

Tavakkoli-Moghaddam, R.; Saremi, A.R. and Ziaee, M.S. (2006). A memetic algorithm for a vehicle routing problem with backhauls. *Applied Mathematics and Computation*, v. 181, p. 1049–1060.

Toth, P. and Vigo, D. (1999). A heuristic algorithm for the symmetric and asymmetric vehicle routing problem with backhauls. *European Journal of Operational Research*, v. 113, p. 528–543.