Alexandre Romanelli

Meta-heurística Guided Local Search aplicada à resolução do Problema de Roteamento de Veículos Capacitados

Brasil

Novembro de 2016

Alexandre Romanelli

Meta-heurística Guided Local Search aplicada à resolução do Problema de Roteamento de Veículos Capacitados

Trabalho apresentado como requisito parcial para obtenção de aprovação na disciplina Metaheurísticas.

Universidade Federal do Espírito Santo – UFES

Departamento de Informática

Programa de Pós-Graduação

Professora: Maria Cristina Rangel

Brasil

Novembro de 2016

Sumário

1 Introdução

Este trabalho visa estudar a aplicação da meta-heurística *Guided Local Search* como abordagem à resolução do Problema de Roteamento de Veículos Capacitados. Esta meta-heurística é um método usado para resolver problemas de otimização combinatória baseado em busca local. Sua ideia central é mapear características das soluções, de modo que seja possível evitar as características presentes em soluções conhecidas e, dessa forma, escapar de ótimos locais.

Para avaliar a aplicação desta meta-heurística ao problema indicado, a metodologia utilizada foi a implementação de um programa de computador, a posterior realização de experimentos com instâncias do problema de roteamento de veículos capacitados usadas para comparação de abordagens, e a análise dos resultados obtidos.

O trabalho está estruturado da seguinte forma. O capítulo 2 apresenta a meta-heurística *Guided Local Search*, com a apresentação dos conceitos básicos, do algoritmo e uma análise do comportamento deste, seguindo a exposição de trabalhos recentes que fazem uso desta meta-heurística para solucionar problemas diversos. O capítulo 3 traz a definição do problema de roteamento de veículos capacitados. O capítulo 4 é dedicado à aplicação de *Guided Local Search* ao problema definido no capítulo anterior, envolvendo as estruturas de dados, os métodos implementados e os resultados dos testes realizados. O capítulo 5 traz a conclusão deste trabalho.

2 A meta-heurística Guided Local Search

Publicada originalmente em (VOUDOURIS; TSANG, 1995), a meta-heurística *Guided Local Search* (GLS) foi estabelecida para a resolução de problemas de otimização combinatória. O método faz uso de informações do domínio do problema e relativas à busca para guiar o procedimento de busca local no espaço de soluções. Resumidamente, isto é feito com a inclusão de um conjunto de termos de penalização à função que avalia o custo de uma solução. Neste capítulo serão apresentados os conceitos envolvidos nesta meta-heurística, seu algoritmo, uma análise do funcionamento do método e algumas aplicações atuais deste na resolução de problemas de otimização combinatória.

2.1 Conceitos básicos

Para as definições a seguir, apresentadas conforme (VOUDOURIS; TSANG, 1995), será considerado o contexto de um problema de otimização combinatória. Seja o par (S, g), onde S é o conjunto de todas as soluções possíveis e g é a função objetivo que mapeia cada elemento s em S para um número real. Considera-se que o objetivo é minimizar a função $g(s)$, conforme equação 2.1.

$$\min g(s), s \in S \tag{2.1}$$

Define-se como "vizinhança" N para uma instância de problema (S, g) o mapeamento de S para o seu conjunto das partes, como mostra a equação 2.2.

$$N : S \to 2^S \tag{2.2}$$

A partir de uma solução $s \in S$ qualquer, as soluções que podem ser alcançadas a partir de s por um único movimento são consideradas "vizinhas" de s. O conjunto de vizinhas $N(s)$ é chamado de "vizinhança" de s. Um movimento é uma operação que transforma uma solução em outra com algumas pequenas modificações. A definição dessas modificações forma o que se chama de "estrutura de vizinhança".

Dentre todas as soluções de uma vizinhança N, a que possui o menor custo definido pela função g é chamada de mínimo local de g em relação a N. Isto é apresentado pela equação 2.3.

$$g(x) \leq g(y), \forall y \in N(x) \tag{2.3}$$

Assim como outras meta-heurísticas, GLS também age sobre um procedimento de minimização da função g chamado de "busca local". Este procedimento consiste em fazer

uma sucessão de substituições da solução atual x por uma solução y que seja "melhor" em relação à função de custo g, conforme a equação 2.4.

$$g(y) \leq g(x), y \in N(x) \tag{2.4}$$

De modo geral, a busca local pode ser definida como um procedimento que tem como entradas uma solução inicial s_1 e uma função de custo g, e produz como saída uma solução final, s_2, considerada o mínimo local. Para partir de s_1 e chegar a s_2, o procedimento efetua as trocas sucessivas de soluções por vizinhos melhores, em uma vizinhança N. A equação 2.5 mostra de maneira simbólica a definição da busca local.

$$s_2 \leftarrow \textit{procedimento } \textbf{BuscaLocal}(s_1, g) \tag{2.5}$$

Uma **característica de solução** f_i é alguma propriedade que descreve parcialmente a solução e pode ser relacionada com o custo desta. Ou seja, o custo c_i é aplicado a uma solução se esta tem a característica f_i. A definição das características de solução é uma tarefa dependente de conhecimento sobre o domínio do problema. É importante ressaltar que essa definição deve ser feita de tal forma que algumas soluções tenham determinada característica, outras não. A presença, ou não, de uma característica em uma solução é indicada pela função I, que é definida como na equação 2.6.

$$I_i(s) = \begin{cases} 1, \text{ se a solução } s \text{ tem a propriedade } i \\ 0, \text{ caso contrário} \end{cases}, s \in S \tag{2.6}$$

Um conceito chave do método GLS é a penalização de características, cuja definição é apresentada a seguir. Sejam os elementos abaixo:

p_i: um parâmetro de penalidade aplicado a uma característica f_i de uma solução s;

M: o número de características definidas nas soluções;

λ: o parâmetro de regularização.

Uma função de custo aumentada h é definida pela equação 2.7.

$$h(s) = g(s) + \lambda \cdot \sum_{i=0}^{M} p_i \cdot I_i(s) \tag{2.7}$$

Observa-se que as penalidades aplicadas incidem apenas sobre as soluções que possuem as características que foram penalizadas. Isto é garantido pela função indicadora I. Os parâmetros de penalidades são incrementados um por um para todas as características que maximizam a função utilidade mostrada na equação 2.8. Esta incrementação ocorre dinamicamente durante o procedimento GLS.

$$\textit{util}(s_*, f_i) = I_i(s_*) \cdot \frac{c_i}{1 + p_i} \tag{2.8}$$

Como a função de utilidade faz uma relação entre o custo associado a uma característica e seu respectivo parâmetro de penalidade, as características de uma solução selecionadas para serem penalizadas não serão sempre as que possuem o maior custo associado. Isso garante que, embora as características mais "caras" serão penalizadas prioritariamente, também haverá uma distribuição de penalizações entre as outras características da solução, ao longo das iterações do procedimento GLS, que é apresentado a seguir.

2.2 Algoritmo GLS

O algoritmo 2.1 é uma adaptação dos que foram apresentados em (VOUDOURIS; TSANG, 1995) e (VOUDOURIS; TSANG; ALSHEDDY, 2010). Este algoritmo recebe como entradas os elementos abaixo:

- O conjunto de soluções viáveis S;
- A função de custo g, a ser minimizada;
- O parâmetro de regularização λ;
- Um vetor de funções indicadoras I_i para cada característica f_i observável;
- Um vetor de custos c_i associados às características f_i;
- O número de características observáveis.

E produz como saída uma solução s^*, que é a solução com o menor valor da função g dentre as que foram avaliadas.

Algoritmo 2.1 Guided Local Search.

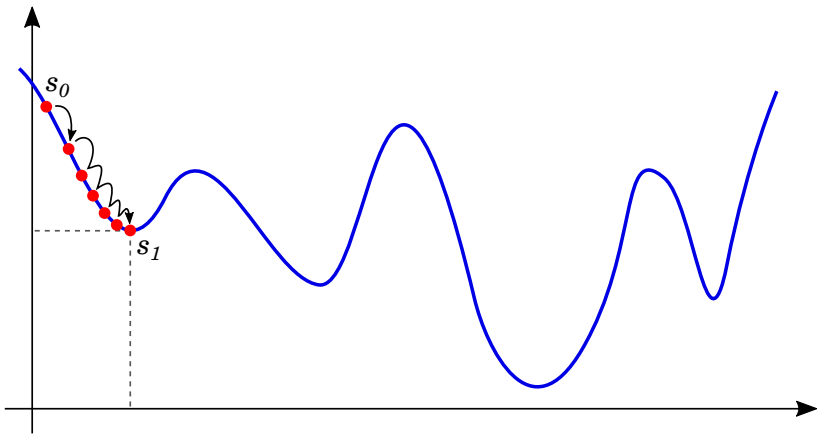
```
1:  function GuidedLocalSearch(S, g, λ, [I_1, ..., I_M], [c_1, ..., c_M], M)
2:      k ← 0
3:      s_0 ← solução aleatória em S
4:      for i ← 1 to M do
5:          p_i ← 0
6:      h ← g + λ * Σ p_i * I_i
7:      while Critério de parada não satisfeito do
8:          s_{k+1} ← BuscaLocal(s_k, h)
9:          for i ← 1 to M do
10:             util_i ← I_i(s_{k+1}) * c_i/(1 + p_i)
11:         for all i tal que util_i é máximo do
12:             p_i ← p_i + 1
13:         k ← k + 1
14:     s* ← melhor solução encontrada com relação à função custo g
15: return s*
```

O procedimento inicia com a definição de um indexador de solução k e uma solução inicial s_0 escolhida aleatoriamente. Isto pode ser feito com o uso de algum método construtivo. Nas linhas 4 e 5, os parâmetros de penalização são iniciados com zero. Ou seja, nenhuma característica é penalizada *a priori*. Na linha 6 está a definição da função de custo aumentada h. A estrutura repetitiva entre as linhas 7 e 13 será executada enquanto o critério de parada não for satisfeito. Este critério pode ser definido de acordo com a necessidade específica do contexto da aplicação. Alguns de muitos critérios de parada aplicáveis são: tempo decorrido, número total de iterações, número de iterações sem melhoria na solução e distância para um limitador de valor mínimo para a função de custo. O processo executado repetidamente inicia com uma busca local que parte da solução s_k, com objetivo de minimizar a função de custo aumentada h. Isto gera a solução seguinte, s_{k+1}, que é o mínimo local. As características presentes nessa solução s_{k+1} serão submetidas a análise pela função de utilidade, nas linhas 9 e 10. Nas linhas 11 e 12, as características que maximizam a função de utilidade serão penalizadas, com o incremento dos respectivos parâmetros de penalização. A repetição é encerrada com o incremento do indexador de soluções, para que na próxima iteração haja o sequenciamento da busca local no último ótimo local encontrado. Durante todo o processo, as soluções percorridas são avaliadas também em relação à função de custo original g. A solução com o menor valor de g, armazenada em s^*, é retornada como o resultado do método GLS.

2.3 Análise do método GLS

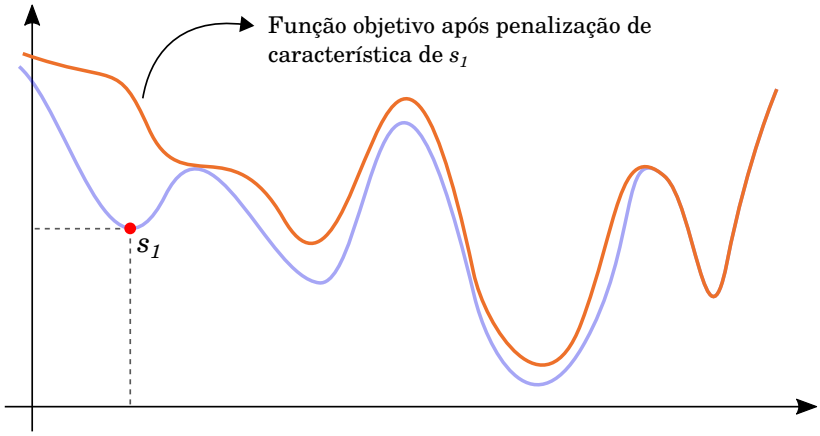
Assim como outras meta-heurísticas, o GLS inicia com uma busca local que conduz da solução inicial s_0 até uma solução ótima local s_1, como se pode observar na figura 1, com uma função cujo intuito é meramente ilustrativo. Inicialmente, a função de custo aumentada h é igual à função de custo g, pois os parâmetros de penalização são iguais a zero.

Figura 1 – Intensificação inicial com busca local



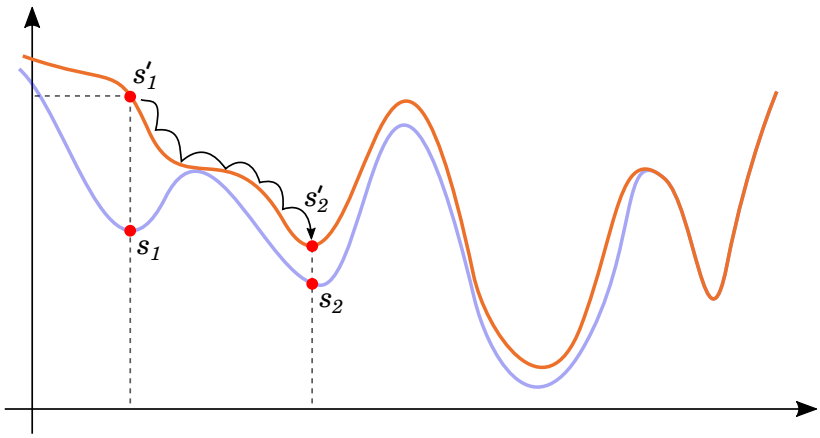
O passo seguinte do algoritmo é aplicar penalidades às características de maior impacto na função objetivo, como indicado em (SILVA; SILVA, 2015). Dentre as características presentes no ótimo local, as que tenham maior relação entre custo e penalizações prévias são penalizadas. Isto garante a diversificação das soluções na próxima etapa de busca, em que a função de custo aumentada h é diferente da que foi usada como guia para a última busca local realizada. A figura 2 ilustra a função objetivo após penalização aplicada.

Figura 2 – Função objetivo modificada por penalização de característica de s_1



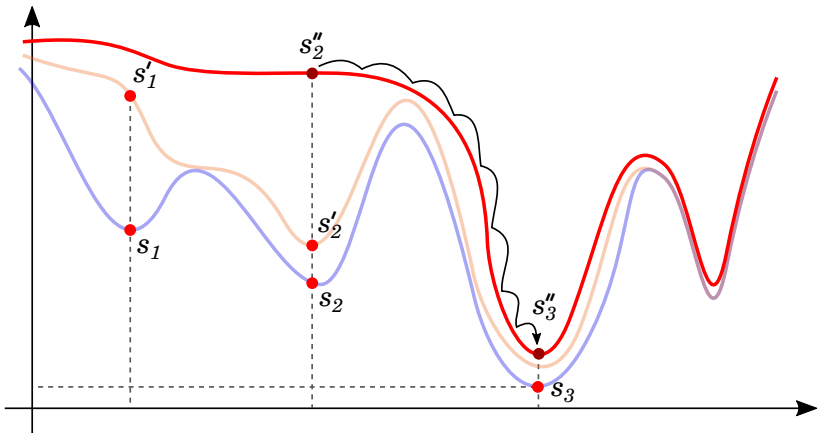
O ótimo local s_1 passa a possuir um novo valor na função de custo aumentada h, o que é ilustrado pela figura 3 com a projeção s'_1. Isto significa que a diversificação, em GLS, é feita com a imposição de restrições às soluções que serão avaliadas a seguir. Assume-se que o ótimo global não está em torno do ótimo local encontrado, e as penalidades excluem esta área das buscas feitas nas iterações seguintes, conforme (VOUDOURIS, 1998). A busca local seguinte pode conduzir a um outro ótimo local s'_2, que possui sua projeção s_2 na função g.

Figura 3 – Diversificação de solução após penalização



A cada iteração, as penalizações aplicadas fazem com que as buscas locais levem a soluções que tenham cada vez menos características em comum com os ótimos locais encontrados previamente, o que aumenta a chance de escapar dos vales em que estão os mínimos locais. A figura 4 exemplifica como as soluções afastadas dos ótimos locais já observados, ou seja, com poucas características em comum, possuem valores da função h muito próximos dos valores da função g. Com penalizações suficientes aplicadas, é possível que a busca local seja conduzida a soluções bem diferentes, ampliando assim o espaço de busca observado.

Figura 4 – Diversificação de solução após penalização



Como observado em (VOUDOURIS, 1998), as penalizações no GLS são feitas suavemente. Isto se deve à ausência de certeza de que, quando uma solução é visitada, como um ótimo local, por exemplo, todas as soluções na área de busca ao seu redor foram observadas pela busca local.

É possível perceber que o diferencial da meta-heurística GLS em relação a outras baseadas em busca local, como *Variable Neighborhood Search*, *Iterated Local Search*, GRASP, entre outras, reside na diversificação pela penalização imposta a características, o que modifica a função objetivo, e não por perturbações feitas a soluções ou geração de novas soluções aleatórias. Alguns pesquisadores obtiveram êxito na aplicação de GLS na solução de vários problemas de otimização combinatória. Alguns desses trabalhos são expostos a seguir.

2.4 Aplicações atuais do método GLS

Nesta seção, serão apresentados alguns trabalhos recentes que aplicaram a meta-heurística GLS na resolução de problemas de otimização combinatória. A exposição dos trabalhos será focada na definição do problema e na identificação das características das soluções que impactam no custo.

2.4.1 Problema de programação de tripulações de ônibus urbano

Segundo (SILVA; SILVA, 2015), o problema de programação de tripulações (PPT) consiste em gerar as jornadas de trabalho para os funcionários que compõem as tripulações das viagens dos veículos em operação em um dado local. Os dados de entrada incluem a programação de viagens dos veículos em operação, as regras operacionais e elementos das leis trabalhistas. As viagens são agrupadas em tarefas. As substituições de tripulações, durante a operação de um veículo, ocorrem em intervalos entre tarefas que são maiores do que determinada quantidade de tempo, e em certos locais apropriados. Esses eventos são chamados de oportunidades de troca. O objetivo é alocar as tarefas às tripulações para minimizar o custo. O conjunto de tarefas atribuído a cada tripulação constitui sua jornada de trabalho. Jornadas com tempo acima do período normal de trabalho contém horas extras, enquanto jornadas com tempo abaixo contém ociosidade. Jornadas com intervalos maiores do que duas horas entre duas tarefas são chamadas de duplas pegadas. Em duplas pegadas, os intervalos maiores do que duas horas não são remunerados.

Para aplicar o GLS, o PPT foi modelado como um problema de particionamento de tarefas. O critério de parada do algoritmo foi definido como tempo de processamento de trinta minutos. O método construtivo guloso adotado faz a adição sucessiva, à jornada corrente, da tarefa ainda não alocada que levar ao menor acréscimo de custo à solução. A estrutura de vizinhança definida consiste em movimentos de realocação de tarefas a jornadas ou trocas de tarefas entre jornadas, sempre respeitando as restrições impostas.

As características identificadas para receberem possíveis penalizações foram:

- Quantidade de horas extras;
- Tempo ocioso;
- Quantidade de duplas pegadas.

Assim, a penalização dessas características leva à substituição das tarefas que estão gerando custo excessivo. Em testes computacionais, foi observado que a aplicação de GLS levou a algumas soluções melhores do que as encontradas com técnicas amplamente usadas para a resolução do PPT, como VNS.

2.4.2 Problema de posicionamento de formas irregulares

O problema de posicionamento de formas irregulares (PFI), como tratado em (EGEBLAD; NIELSEN; ODGAARD, 2007), consiste em determinar os locais em que algumas formas irregulares serão posicionados em um dado objeto, que pode ser uma chapa retangular, na versão do problema em 2D, ou um contêiner, na versão 3D, sem que haja sobreposição entre as formas. O objetivo é fazer o posicionamento de todas as formas

de modo que a área ou o espaço ocupado seja mínimo. Como a abordagem para a versão 3D do problema foi uma generalização da usada para a versão 2D, a exposição a seguir será concentrada nesta última.

O PFI foi modelado, para aplicar o GLS, definindo metas de área máxima a ocupar e, progressivamente, tentando obter soluções válidas para áreas cada vez menores. As formas eram inicialmente distribuídas sobre a chapa as sobreposições observadas eram minimizadas com movimentos de deslocamentos horizontais, deslocamentos verticais, rotação e espelhamento. As características penalizadas pelo método foram as sobreposições entre pares de itens. O critério de parada estabelecido foi o tempo total de dez minutos para cada execução.

Os resultados computacionais observados foram satisfatórios, levando às melhores soluções conhecidas até então para algumas instâncias do PFI usadas para comparações entre abordagens.

2.4.3 Posicionamento de formas irregulares para corte em chapas irregulares

O problema tratado em (BALDACCI et al., 2014) difere do citado anteriormente apenas pela forma da chapa. Neste, a chapa pode assumir a forma de um polígono irregular. No caso, a aplicação em estudo era o corte de couro, possivelmente com áreas defeituosas de menor valor. Na abordagem utilizada, as formas foram representadas como matrizes de *pixels*, assim como a área da chapa, e o posicionamento das formas seguia os pixels definidos na chapa, de modo que todos as áreas defeituosas fossem evitadas e que não houvesse sobreposição entre as formas posicionadas. O objetivo é reduzir o desperdício de material e maximizar o valor das formas posicionadas.

A meta-heurística GLS foi aplicada com a penalização de *pixels* da chapa que foram cobertos pelas formas posicionadas no ótimo local. Após a penalização, novas soluções eram definidas de forma diversificada, pois consideravam os pixels previamente ocupados como áreas de menor valor.

Segundo os autores do trabalho, embora não tenham encontrado meios de comparálo a outras abordagens para resolução do mesmo problema, os resultados foram satisfatórios, pois encontraram boas soluções para as instâncias testadas e em tempo suficiente para sua aplicação na indústria.

2.5 Considerações sobre o capítulo

Neste capítulo foi feita uma introdução à meta-heurística *Guided Local Search*, que usa uma técnica peculiar de busca e diversificação que combina a penalização de características dos ótimos locais com a modificação da função objetivo, acrescentando a esta o

custo extra relativo às penalidades aplicadas. Foi feito um breve estudo do comportamento deste método e apresentadas algumas aplicações atuais deste na resolução de problemas de otimização. No próximo capítulo, é apresentado o Problema de Roteamento de Veículos Capacitados (PRVC), cuja resolução pela aplicação de GLS é tratada na sequência.

3 O problema de roteamento de veículos capacitados

O PRVC, apresentado em (DANTZIG; RAMSER, 1959), consiste em se obter a melhor distribuição de rotas a uma frota de veículos para fazer entregas a determinados locais, que têm demandas específicas[1] quantificáveis. Cada veículo da frota deve partir de um ponto inicial que é comum a todos, com a carga suficiente para atender completamente aos locais que visitará em sua rota, passar por cada local e, por fim, retornar ao ponto de partida. Os veículos da frota possuem uma capacidade de carga máxima e não podem transportar excesso de carga. Este problema é formulado como segue.

Sejam os seguintes valores (baseados em (TOTH; VIGO, 2002) e (DANTZIG; RAMSER, 1959)):

1. Um conjunto V de n locais V_i, $i = 0, 1, \ldots, n$, sendo que as entregas são feitas a partir do local V_0 aos demais locais.

2. Um número K de veículos disponíveis[2]. K será usado para denotar o número de circuitos da solução, correspondentes às rotas dos veículos.

3. Um conjunto de arestas E com uma aresta para cada par de locais V_i e V_j, em que $i \neq j$, e $i, j \in \{0, 1, \ldots, n\}$. Cada aresta $e \in E$ é um caminho que pode ser percorrido por um veículo entre os locais V_i e V_j. Os custos destes caminhos são dados pela matriz de custos.

4. Uma matriz de custos $[c_{ij}]$, em que cada elemento c_{ij} especifica a distância entre o par de locais V_i e V_j. Assim, pela definição, $c_{ii} = 0$. Denomina-se "PRVC simétrico" qualquer instância do PRVC em que $c_{ij} = c_{ji}$, $\forall i, j \in \{0, 1, \ldots, n\}$. Um custo c_{ij} pode ser referido como c_e, onde e é a aresta entre V_i e V_j.

5. Um vetor de demandas (d_i), $i = 1, 2, \ldots, n$, em que cada elemento d_i representa numericamente a quantidade que deve ser entregue ao local V_i.

6. Um número C que indica a capacidade de carga de todos os veículos da frota. Considera-se que $C > d_i$ $\forall i \in \{1, 2, \ldots, n\}$.

[1] Neste trabalho, não será feita qualquer consideração sobre a natureza do que é entregue, mesmo que se possa deduzir que, dependendo desta natureza, várias informações adicionais poderiam ser relevantes e causariam impacto à forma de modelagem e resolução do problema.

[2] Como defendido em (UCHOA et al., 2014), há casos em que o número de veículos poderia ser considerado apenas como um limite mínimo

7. Um conjunto de arestas $\delta(i)$, que determina as arestas que possuem uma de suas extremidades incidindo sobre o local V_i.

8. Um conjunto de valores $X = \{x_e | e \in E\}$, em que cada elemento x_e indica a quantidade de vezes que a aresta e foi percorrida por um veículo na solução.

9. Dado um conjunto de locais $S \subseteq V$, $\gamma(S)$ indica o número mínimo de veículos necessários para atender a todos os locais em S, e $d(S) = \sum_{j \in S} d_j$ é a demanda total do conjunto.

O problema, portanto, possui um conjunto de variáveis X e a solução consiste em apresentar o conjunto de valores dessas variáveis que resolve o seguinte modelo de programação inteira (TOTH; VIGO, 2002).

$$\min \sum_{e \in E} c_e x_e \tag{3.1}$$

$$\text{s.a} \sum_{e \in \delta(i)} x_e = 2 \qquad \forall i \in V \setminus \{0\} \tag{3.2}$$

$$\sum_{e \in \delta(0)} x_e = 2K \tag{3.3}$$

$$\sum_{e \in \delta(S)} x_e \geq 2\gamma(S) \qquad \forall S \subseteq V \setminus \{0\}, S \neq \emptyset \tag{3.4}$$

$$x_e \in \{0, 1, 2\} \qquad \forall e \in \delta(0) \tag{3.5}$$

$$x_e \in \{0, 1\} \qquad \forall e \notin \delta(0) \tag{3.6}$$

As restrições 3.2 e 3.3 determinam que haverá duas arestas incidentes em cada local que não é o depósito, e neste o número de arestas incidentes será igual a $2K$, pois cada rota deverá iniciar e terminar no depósito. Sendo $\gamma(S)$ substituído pelo limite mínimo obtido com o problema da mochila associado ao carregamento de veículos para atender às demandas dos locais em S, a restrição 3.4 assegura que o número suficiente de arestas entrará em cada subconjunto de vértices, para que a conectividade dos circuitos seja atingida, e de modo que a capacidade dos veículos seja respeitada (TOTH; VIGO, 2002). As restrições finais indicam os valores possíveis para as variáveis x_e, sendo que o valor 2 será usado em casos de rotas nas quais o veículo sai do depósito, visita apenas um local e retorna.

Como está definido, uma instância do PRVC é um grafo conexo $G = (V, E)$, não sendo incomum encontrar estudos que usam como instâncias grafos completos. A solução para o PRVC é um subgrafo conexo $G' = (V', E')$ de G, sendo que $V' \subseteq V$ possui todos os vértices de V cujas demandas são maiores do que zero, e $E' \subset E$. Graficamente, as soluções assemelham-se à figura 5, na qual cada rota é apresentada com uma cor diferente. Todas as

rotas partem do mesmo local, o depósito, percorrem locais (cada local será visitado apenas uma vez por um veículo) e retornam ao depósito[3]. A figura 6 apresenta a representação textual para a mesma solução, com as rotas descritas em linhas como a sequência de locais visitados, com o depósito omitido por ser implícito.

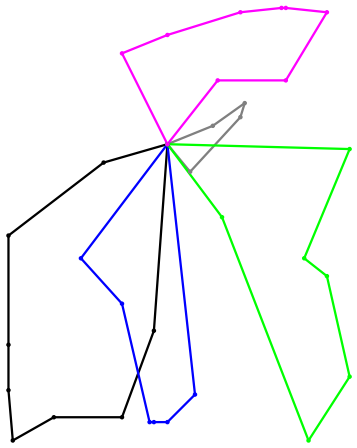


Figura 5 – Exemplo de solução para o PRVC.

Rota 1:	15	17	9	3	16	29		
Rota 2:	12	5	26	7	8	13	32	2
Rota 3:	20	4	27	25	30	10		
Rota 4:	23	28	18	22				
Rota 5:	24	6	19	14	21	1	31	11

Figura 6 – Representação textual de solução.

Este problema é classificado como NP-difícil, como informado em (TOTH; VIGO, 2002) e (FUKASAWA et al., 2006). Sendo assim, embora haja esforços significativos para determinar soluções ótimas com métodos exatos, como em (LAPORTE, 1992), (TOTH; VIGO, 2002), (FUKASAWA et al., 2006) e (RALPHS et al., 2003), instâncias com elevado número de vértices podem levar a esperas longas pela execução de tais métodos. Consequentemente, em determinadas circunstâncias, pode ser viável aceitar soluções não ótimas, desde que sejam suficientemente boas e obtidas em tempo hábil. Este trabalho visa apresentar uma técnica para resolução do PRVC com uso da meta-heurística GLS, com objetivo inicial de determinar soluções satisfatórias em tempo relativamente pequeno. Este é o assunto do próximo capítulo.

[3] No PRVC simétrico, o sentido do percurso de cada veículo, que pode ser horário ou anti-horário, não é relevante para a solução.

4 Aplicação de GLS ao PRVC

Neste capítulo será apresentada a aplicação da meta-heurística GLS como abordagem à resolução do PRVC. Inicialmente, são expostos os dados do problema organizados para esta aplicação. Em seguida, é feita uma explicação das estruturas de dados usadas no algoritmo implementado. Logo após, são apresentados os métodos construtivos usados para gerar soluções iniciais. Na sequência, são apresentadas as estruturas de vizinhança empregadas nesta implementação. A seção seguinte apresenta os algoritmos de busca local que foram empregados. A implementação do algoritmo GLS é discutida na seção seguinte. Por fim, são mostrados os resultados obtidos e é feita uma breve análise destes.

4.1 Organização de dados para aplicação do GLS

Para resolver o PRVC com a aplicação da meta-heurística GLS, foi necessário estabelecer os seguintes termos.

1. Seja chamado de τ_i cada um dos ciclos que juntos formam a solução. E seja $|\tau_i|$ o número de locais percorridos no ciclo τ_i.

2. O conjunto de variáveis do problema passou a ser $X = \{x_{ij} = m\}$. Cada variável e seu respectivo valor indicam que o j-ésimo local visitado pelo veículo i será V_m. Os limites dos índices são $i \in \{1, 2, \ldots, K\}$, $j \in \{1, 2, \ldots, |\tau_i|\}$ e $m \in \{1, 2, \ldots, n\}$.

3. Se $x_{ij} = m$, então $x_{pq} \neq m$, $\forall p, q$ tal que $p \neq i$ ou $q \neq j$. Assim, cada local será visitado por apenas um veículo.

4. A demanda total de cada ciclo i, $d(\tau_i)$, é dada pela soma das demandas dos locais de τ_i. Como restrição de capacidade, têm-se que $d(\tau_i) \leq C$, $\forall i \in \{1, 2, \ldots, K\}$.

5. Seja o conjunto $E^{(i)}$ de arestas percorridas por cada veículo i, sendo que o número de arestas $|E^{(i)}| = |\tau_i| + 1$, composto de acordo com as seguintes regras aplicadas a cada local de τ_i:

   - Para x_{i1}, a aresta entre o depósito V_0 e o local x_{i1}.
   - Para x_{ij}, $j > 1$, a aresta entre x_{il} e x_{ij}, em que $l = j - 1$.
   - A aresta entre x_{il} e o depósito V_0, sendo que $l = |\tau_i|$.

6. O custo de um ciclo i, $c(\tau_i)$, é a soma das distâncias das arestas percorridas neste. Ou seja, $c(\tau_i) = \sum_{e \in E^{(i)}} c_e$.

7. O custo de uma solução s é dado pela função g apresentada na equação 4.1.

$$g(s) = \sum_{i=1}^{K} c(\tau_i) \tag{4.1}$$

8. Dado o parâmetro λ, a função de custo aumentada h foi definida pela equação 4.2.

$$h(s) = g(s) + \lambda \sum_{i=1}^{K} \sum_{e \in \tau_i} I_e p_e \tag{4.2}$$

As características enumeráveis do problema, requisito do GLS, foram definidas como as arestas entre os locais. Como o cálculo do custo de uma solução está diretamente associado aos custos individuais de cada aresta que foi usada na solução, isto torna esta escolha de características apropriada para a aplicação desta meta-heurística. Foi adicionada uma função indicadora $I_{ij}(s) \in \{0, 1\}$, que indica a utilização da aresta e_{ij} na solução s (0 se não foi usada, 1 se foi).

Uma vez organizados os dados que são necessários para a resolução do PRVC, passou-se à definição das estruturas de dados que suportassem a implementação deste método. A explicação dessas estruturas de dados é o assunto da próxima seção.

4.2 Estruturas de dados

Para implementar o GLS aplicado à resolução do PRVC, foi necessário tomar decisões que envolviam o pleno suporte às operações realizadas durante a busca por soluções, o consumo de tempo de processamento dessas operações e o intervalo de tempo disponível para concluir a implementação. Diante deste último aspecto, não foi possível aproveitar as técnicas de redução da complexidade computacional conhecidas para os problemas da família do PRVC (como o problema do caixeiro viajante), como as que se encontram em (FREDMAN et al., 1995).

Para acomodar os dados de uma solução, foi usada uma matriz $X = [x_{ij}]$, com n linhas e $n + 3$ colunas. Cada linha corresponde a uma rota τ_i, sendo que a quantidade máxima de linhas foi dimensionada para atender ao caso extremo de uma instância em que seja necessário alocar um veículo para visitar cada local. As colunas representam os locais visitados em cada rota. O dimensionamento visa atender às possíveis instâncias em que apenas um veículo é suficiente para realizar as entregas a todos os locais. As três colunas adicionais correspondem aos valores de número de locais visitados em cada rota, $|\tau_i|$, da carga atribuída ao veículo, $d(\tau_i)$, e da distância percorrida pelo veículo em sua rota, $c(\tau_i)$.

A representação da função indicadora foi feita com uma matriz $I = [i_{qr}]$ com $q, r \in \{1, 2, \ldots, n\}$. Cada elemento i_{qr} possui o valor 0 se a aresta e_{qr} não for percorrida

por algum veículo na solução, e possui o valor 1 se a aresta correspondente for parte de alguma rota. A princípio, $i_{qr} = 0, \forall q, r$. Uma matriz $P = [p_{ij}]$ foi usada para armazenar as penalidades às características do problema. Como estas são as arestas do grafo associado ao problema, cada elemento p_{ij} indica o valor da penalidade aplicada ao uso da aresta e_{ij}. Inicialmente, $p_{ij} = 0, \forall i, j$.

É fácil perceber o desperdício dos elementos i_{qq} e p_{ii}, pois correspondem a arestas inexistentes pela definição do problema. Evidentemente, há um vasto universo de possibilidades de melhorias que podem ser efetuadas nas estruturas de dados usadas na implementação feita neste trabalho. Como a realização dessas melhorias pode elevar a qualidade dos resultados obtidos, a continuidade da pesquisa pode explorar esse caminho. Na próxima seção, a discussão passa aos métodos construtivos propostos.

4.3 Métodos construtivos

No algoritmo GLS apresentado na seção 2.2, uma solução aleatória s_0 é obtida para iniciar a busca local da primeira iteração. Na implementação feita, com objetivo de realizar os testes que são apresentados na seção 4.7, dois métodos foram propostos. Um constrói aleatoriamente uma solução e o outro usa uma heurística gulosa.

4.3.1 Método construtivo aleatório

A construção de uma solução aleatória foi implementada com a preocupação simples de respeitar a restrição de capacidade dos veículos. O método foi elaborado com os seguintes passos.

1. Alocar inicialmente $K = \gamma(V)$ rotas.
2. Escolher aleatoriamente um local V_i ainda não atendido.
3. Escolher aleatoriamente uma rota τ_j, $1 \leq j \leq K$, tal que $d(\tau_j) + d_i \leq C$, ou seja, que ainda comporte atender à demanda do local V_i. Se não houver rota alocada que possa atender a esta demanda, deve-se alocar outro veículo, acrescentar uma unidade ao valor de K, e selecionar τ_j com $j = K$, que é a rota recém-alocada.
4. Incluir V_i ao final da rota τ_j.
5. Se ainda houver local não atendido, retornar ao passo 2.

O passo 3 é responsável por fazer a solução respeitar a restrição de capacidade imposta. Este método não faz qualquer consideração em respeito ao número de veículos da frota.

4.3.2 Método construtivo guloso

Para a realização de testes com variação do parâmetro λ, foi elaborado um método construtivo guloso para gerar soluções de modo determinístico. Isto teve como objetivo permitir observar as implicações de variações do parâmetro a partir de uma solução única para cada instância testada, ao eliminar prováveis variações da qualidade da solução inicial que poderiam influenciar nos resultados, obscurecendo os efeitos do valor de λ. O método proposto consiste nos seguintes passos.

1. Alocar inicialmente $K = \gamma(V)$ rotas.
2. Para cada rota τ_i, $1 \leq i \leq K$, fazer:
   a) Atribuir dois locais V_q e V_r à rota τ_i, escolhidos pela menor distância a partir do depósito, e desde que não excedam a capacidade do veículo.
   b) Enquanto $d(\tau_i) < C$, ou seja, a soma das demandas dos locais atendidos na rota τ_i for menor do que a capacidade, incluir um novo local com a menor soma das distâncias para o último e para o penúltimo locais visitados nesta rota, e que mantenha $d(\tau_i) \leq C$.
3. Se houver local não atendido, deve-se alocar outro veículo, acrescentar uma unidade ao valor de K, e fazer, para esta rota, o mesmo processo descrito nos subitens do passo 2.

∗ *Durante qualquer passo, se não houver mais local não atendido, o método deve ser encerrado.*

A qualidade das soluções geradas por este método, assim como no aleatório, não é satisfatória. A busca realizada a partir dessas soluções será feita com pequenas transformações aplicadas sucessivamente. Isto será descrito a seguir.

4.4 Estruturas de vizinhanças

Para realizar a busca local como definida na seção 2.1, primeiro foram definidos os métodos usados para fazer as pequenas modificações nas soluções, que são chamados de "estruturas de vizinhança". As soluções geradas a partir das modificações são chamadas "soluções vizinhas", e o conjunto delas é chamado de "vizinhança" da solução modificada. Nesta implementação, foram empregadas três estruturas de vizinhanças, chamadas de 2-OPT, TROCA e INSERÇÃO, que são expostas a seguir.

Nas três estruturas, foi usado um operador de indexação sobre as rotas, definido pela equação 4.3.

$$\tau_i[j] = V_l, \text{ se há } j-1 \text{ locais visitados na rota } \tau_i \text{ antes de } V_l. \tag{4.3}$$

4.4.1 Vizinhança 2-OPT

A vizinhança 2-OPT usada neste trabalho foi inspirada no método proposto em (LIN, 1965) para melhoria de rotas, originalmente aplicado no problema do caixeiro viajante. O nome "2-OPT" refere-se ao resultado da busca local, que é uma solução que não pode ser melhorada com a troca de duas arestas, ou seja, é ótima nesta vizinhança.

Neste trabalho, a vizinhança 2-OPT foi restringida aos ciclos. Isto significa que as arestas trocadas faziam parte de uma mesma rota. A obtenção de vizinhos nesta vizinhança segue os passos descritos a seguir, que são executados para cada rota τ_i da solução original s.

1. Para cada local $\tau_i[j]$ atendido pela rota, fazer o que segue:
   a) Para cada local $\tau_i[l]$ também atendido pela mesma rota, $j < l \leq |\tau_i|$, gerar uma solução s' da seguinte forma:
      i. Todas as rotas de s devem ser replicadas em s'.
      ii. A ordem da sequência de locais τ_i de s' deve ser invertida entre os locais $\tau_i[j]$ e $\tau_i[l]$, inclusive estes.

A figura 7 apresenta um exemplo de obtenção de vizinho com esta estrutura. Acima, nesta figura, está a solução original. Abaixo, o resultado da troca das arestas $e_{6,19}$ e $e_{1,31}$ pelas arestas $e_{6,1}$ e $e_{19,31}$.

Como as arestas não são diretamente referenciadas na estrutura de dados que representa uma solução, a troca de arestas é realizada com a inversão da sequência dos locais visitados. Não havendo transferências entre diferentes rotas, não é preciso recalcular os valores das demandas atendidas por cada rota. Contudo, a mudança afeta o valor da função de custo g, desde que os custos das arestas trocadas sejam diferentes. O recálculo desta função pode ser feito apenas considerando essas diferenças, sem que seja necessário realizar todo o somatório novamente.

4.4.2 Vizinhança TROCA

Definida como uma variação da estrutura 2-OPT, a vizinhança TROCA é constituída das soluções obtidas a partir da solução s pela troca de posições de dois locais. O processo de obtenção de vizinhos segue os seguintes passos.

τ_1:	15	17	9	3	16	29		
τ_2:	12	5	26	7	8	13	32	2
τ_3:	20	4	27	25	30	10		
τ_4:	23	28	18	22				
τ_5:	24	6	19	14	21	1	31	11
τ_1:	15	17	9	3	16	29		
τ_2:	12	5	26	7	8	13	32	2
τ_3:	20	4	27	25	30	10		
τ_4:	23	28	18	22				
τ_5:	24	6	**1**	**21**	**14**	**19**	31	11

Figura 7 – Obtenção de vizinho na estrutura 2-OPT.

- Para cada rota τ_i, fazer os passos a seguir.
  1. Para cada local $\tau_i[l]$, fazer os passos a seguir.
     - Para cada rota τ_j, $i \leq j \leq K$, fazer os passos a seguir.
       * Para cada local $\tau_j[m]$, com $1 \leq m \leq |\tau_j|$ se $i \neq j$, ou com $l < m \leq |\tau_i|$ se $i = j$, gerar uma solução s' com os passos a seguir.
         a) Todas as rotas de s devem ser replicadas em s'.
         b) Em τ_i de s', o local $\tau_i[l]$ deve ser substituído por $\tau_j[m]$.
         c) Em τ_j de s', o local $\tau_j[m]$ deve ser substituído por $\tau_i[l]$ de s.

A figura 8 apresenta um exemplo de obtenção de vizinho com esta estrutura. Acima, nesta figura, está a solução original. Abaixo, o resultado da troca do local da quarta posição de τ_1 com o local da terceira posição de τ_5. Esta troca modificou o total de quatro arestas. As arestas removidas da rota τ_1 foram $e_{9,3}$ e $e_{3,16}$. As arestas removidas da rota τ_5 foram $e_{6,19}$ e $e_{19,14}$. As arestas inseridas na rota τ_1 foram $e_{9,19}$ e $e_{19,16}$. As arestas inseridas na rota τ_5 foram $e_{6,3}$ e $e_{3,14}$.

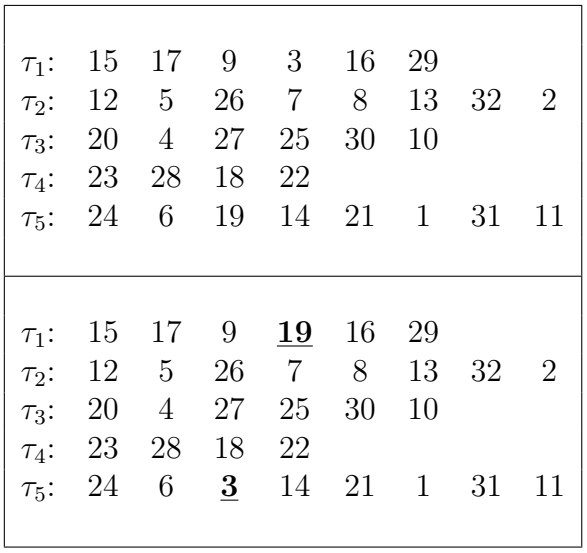


Figura 8 – Obtenção de vizinho na estrutura TROCA.

A operação da vizinhança de TROCA requer a observação da restrição de capacidade dos veículos, pois as demandas dos locais trocados podem ser diferentes. Assim, para alguns casos, é possível que a troca não seja viável e deixe de ser realizada. As atualizações das demandas atendidas pelas rotas envolvidas na troca e suas respectivas distâncias totais podem ser atualizadas sem a necessidade de recálculo completo, observando-se apenas as modificações pontuais que foram realizadas.

4.4.3 Vizinhança INSERÇÃO

A terceira vizinhança utilizada, chamada de INSERÇÃO, foi adotada para possibilitar a busca por soluções com variação nas quantidades de locais visitados pelas rotas. Pode-se observar que as estruturas 2-OPT e TROCA mantêm as rotas com os mesmos números de locais visitados da solução original. Se mantida, esta condição poderia restringir excessivamente o espaço de soluções avaliado pela busca.

A inserção é feita com a remoção de um local de uma rota e sua posterior inclusão em alguma posição de outra rota, ou em uma posição diferente na mesma rota. A obtenção de vizinhos de uma solução s pela vizinhança INSERÇÃO é feita com os passos abaixo.

- Para cada rota τ_i, fazer os passos a seguir.
  1. Para cada local $\tau_i[l]$, fazer os passos a seguir.
     - Para cada rota τ_j, $1 \leq j \leq K$, fazer os passos a seguir.
       * Para cada posição m, $1 \leq m \leq |\tau_j|$, gerar uma solução s' com os passos a seguir.
         a) Todas as rotas de s devem ser replicadas em s'.

b) Em τ_i de s', os locais das posições p, $l < p \leq |\tau_i|$, devem ser deslocados em uma posição à esquerda. Com isto, a rota passa a ter um elemento a menos, que é o elemento $\tau_i[l]$ de s.

c) Em τ_j de s', os locais das posições p', $m \leq p' \leq |\tau_j|$, devem ser deslocados em uma posição à direita. O espaço aberto na posição m deve ser ocupado com o elemento $\tau_i[l]$ de s. Assim, a rota τ_j passa a ter um elemento a mais.

Como a inserção modifica a demanda total atendida por uma ou duas rotas, é necessário fazer a verificação de atendimento à restrição de capacidade dos veículos, para garantir que as soluções geradas sejam viáveis. Inserções que geram soluções inviáveis devem ser descartadas. Algumas inserções também são descartadas por redundância. São os casos de $j = i$ e $l = m$, que não modificariam a solução s, e de $j = i + 1$ ou $j = i - 1$, que são posições adjacentes e a inserção geraria soluções idênticas a outras que já fazem parte da vizinhança TROCA.

A figura 9 apresenta a operação de inserção para obtenção de vizinho com a estrutura INSERÇÃO. Nesta figura, o local da terceira posição da rota τ_5 é inserido na quarta posição da rota τ_1. Duas arestas foram removidas da rota τ_5, que foram $e_{6,19}$ e $e_{19,14}$. Uma aresta foi incluída na rota τ_5, que foi $e_{6,14}$. Uma aresta foi removida da rota τ_1, que foi $e_{9,3}$. E duas arestas foram incluídas na rota τ_1, que foram $e_{9,19}$ e $e_{19,3}$.

τ_1:	15	17	9	3	16	29		
τ_2:	12	5	26	7	8	13	32	2
τ_3:	20	4	27	25	30	10		
τ_4:	23	28	18	22				
τ_5:	24	6	<u>19</u>	14	21	1	31	11

τ_1:	15	17	9	**<u>19</u>**	3	16	29	
τ_2:	12	5	26	7	8	13	32	2
τ_3:	20	4	27	25	30	10		
τ_4:	23	28	18	22				
τ_5:	24	6	14	21	1	31	11	

Figura 9 – Obtenção de vizinho na estrutura INSERÇÃO.

Assim como nas outras estruturas de vizinhança apresentadas, nesta também é possível calcular as variações na função de custo g e nos valores de demandas totais das rotas modificadas sem a necessidade de recalcular completamente esses dados. Para as demandas, basta reduzir a demanda do local removido da rota de origem e adicioná-la

à demanda total da rota de destino. Para o custo, também basta remover os custos das arestas removidas da rota de origem e da rota de destino e adicionar os custos das arestas incluídas nas rotas de origem e de destino.

Em todas as estruturas de vizinhanças, também é possível calcular as modificações feitas em relação à função de custo aumentada h sem refazer os somatórios, já que a alteração do valor do termo de penalização é proporcional aos valores de custos das arestas removidas e adicionadas às rotas. A seguir, são apresentados os métodos de busca local que foram implementados neste trabalho.

4.5 Busca Local

Na implementação feita, foram incorporados dois métodos de busca local, denominados Busca Local Completa (BLC) e Busca Local Rápida (BLR). Os dois métodos aplicam as três estruturas de vizinhanças definidas anteriormente para obter soluções vizinhas e caminhar com trocas sucessivas de soluções até encontrar o ótimo local. Os métodos BLC e BLR são apresentados nesta seção.

4.5.1 Busca Local Completa

O método BLC recebe uma solução inicial s como parâmetro e faz melhorias sucessivas nesta com a seguinte sequência de passos.

1. Usar uma variável s_A, que inicialmente é uma cópia de s, para armazenar a solução atualmente considerada o ótimo local.
2. Repetir os passos abaixo enquanto for encontrada solução vizinha com menor valor da função de custo h.
    a) Usar uma variável s_V para armazenar o melhor vizinho encontrado. Inicialmente, $s_V = s_A$.
    b) Gerar todas as soluções vizinhas s' pela estrutura 2-OPT. Se a solução s', com relação à função h, for melhor do que s_V, deve-se armazenar s' em s_V.
    c) Gerar todas as soluções vizinhas s' pela estrutura TROCA. Se a solução s', com relação à função h, for melhor do que s_V, deve-se armazenar s' em s_V.
    d) Gerar todas as soluções vizinhas s' pela estrutura INSERÇÃO. Se a solução s', com relação à função h, for melhor do que s_V, deve-se armazenar s' em s_V.
    e) Se s_V for melhor do que s_A em relação à função h, deve-se armazenar s_V em s_A e realizar uma nova iteração.
3. O resultado do método é a solução armazenada em s_A.

Como se pode observar, este método de busca local usa o melhor aprimorante como opção para caminhar no espaço de soluções. Isto amplia o número de soluções avaliadas, pois explora exaustivamente as vizinhanças. O método seguinte faz uma redução tanto por usar outra opção para caminhar pelas soluções vizinhas, quanto por reduzir as vizinhanças avaliadas.

4.5.2 Busca Local Rápida

Proposto em (VOUDOURIS; TSANG, 1995), o método BLR acrescenta o conceito de sub-vizinhanças, definidas nesta abordagem à resolução do PRVC como as modificações feitas a partir de um determinado local. Assim, cada local possui um valor correspondente de ativação ou desativação de sua respectiva sub-vizinhança.

Em termos práticos, as mudanças feitas em relação ao método BLC são referentes à execução dos procedimentos de obtenção de vizinhos e à opção para caminhar no espaço de soluções. Quanto a este último aspecto, a BLR faz uso do primeiro aprimorante, ou seja, a primeira solução vizinha encontrada que melhorar o valor da função de custo h em comparação à solução s_A faz esta ser atualizada e o algoritmo passar imediatamente a uma nova iteração. Quanto aos procedimentos de obtenção de vizinhos, as alterações afetam o passo 1 do procedimento da estrutura 2-OPT, o passo 1 do procedimento da estrutura TROCA, e o passo 1 do procedimento da estrutura INSERÇÃO. Nesses passos, deve-se ignorar os referidos locais se suas respectivas sub-vizinhanças estiverem desativadas.

Inicialmente, todas as sub-vizinhanças são ativadas. Se uma sub-vizinhança for explorada e não for encontrada vizinha que melhore a função de custo h em comparação à solução s_A, esta sub-vizinhança será desativada. Em contrapartida, quando uma solução vizinha promover uma melhoria, as sub-vizinhanças dos locais afetados pela modificação feita pela estrutura de vizinhança serão ativadas. São considerados afetados todos os locais com arestas incidentes que foram envolvidas na operação efetuada para obter o vizinho.

O método BLR tende a reduzir o número de soluções avaliadas, por priorizar as que são consideradas mais promissoras. Isto é feito com a ativação das sub-vizinhanças que foram envolvidas em melhorias recentes no ótimo local conhecido. Na sequência, é descrita a implementação do GLS.

4.6 Algoritmo GLS

Após a preparação de todas as operações de suporte, o núcleo da meta-heurística GLS pode ser implementado seguindo o algoritmo 2.1, adaptando apenas as indexações de características para obter os valores correspondentes aos custos das arestas, à função indicadora e às penalidades. Durante as execuções de busca local, todas as soluções obtidas foram avaliadas também quanto à função de custo g, pois a solução com o menor valor

desta função é a que deve ser retornada como resultado do algoritmo GLS. A condição de parada usada foi a quantidade de iterações, e este valor foi parametrizado.

O parâmetro λ foi calculado, inspirado no que foi proposto para resolução do problema do caixeiro viajante em (VOUDOURIS; TSANG, 1995), usando a fórmula da equação 4.4. Nesta, s' refere-se à solução obtida com uma iteração da busca local a partir de uma solução inicial gerada com um dos métodos apresentados na seção 4.3. Este cálculo de λ visa obter valores que tornem significativo o termo de penalização da função de custo aumentada h em relação ao custo g. Na prática, substitui-se o parâmetro λ por a, que possui valores situados em um intervalo mais restrito e, logo, isto simplifica o processo de calibração do método.

$$\lambda = a \times \frac{g(s')}{n}, a \leq 0 \leq 1 \tag{4.4}$$

Um parâmetro adicional permitiu definir o método de busca local que seria utilizado na execução do GLS. Com isso, foi concluída a implementação do GLS aplicado à resolução do PRVC. Na próxima seção são apresentados os testes que foram realizados e os resultados obtidos.

4.7 Testes e resultados

Para avaliar a efetividade da aplicação da meta-heurística GLS como abordagem à resolução do PRVC, de modo que fosse possível compará-la a outras técnicas, foram selecionadas instâncias para a realização de testes que foram usadas por vários outros estudos. Um total de 53 instâncias foram escolhidas, sendo 50 do conjunto gerado pelo método de criação de instâncias proposto em (AUGERAT, 1995) e três obtidas de (FISHER, 1994). Cada instância é identificada por um nome formado por uma cadeia de caracteres com a estrutura "X-nY-kZ", onde "X" assume um dos valores "A", "B" ou "F", sendo os dois primeiros associados a instâncias de AUGERAT e o último, de FISHER, referindo-se aos autores dos trabalhos de onde foram obtidas, "Y" é o número de locais envolvidos no problema (inclusive o depósito), e "Z" é o número de veículos. A tabela 1 apresenta os nomes das instâncias usadas nos testes.

O programa de computador foi implementado em linguagem C++ e compilado com gcc 5.4.0 em sistema Linux versão 4.4.0-43-generic. O hardware utilizado foi um computador com processador Intel(R) Core(TM) i3-2328M, a 2.20GHz, com 3072 KB de cache e 3.7GiB de memória primária.

Foram realizadas duas etapas de testes. Na primeira, os testes foram considerados exploratórios, visando obter "*insights*" sobre o comportamento do método e identificar o número de iterações necessário para obter soluções relativamente boas para cada iteração.

Tabela 1 – Instâncias usadas nos testes.

A-n32-k5	A-n45-k6	A-n63-k9	B-n43-k6	B-n63-k10
A-n33-k5	A-n45-k7	A-n64-k9	B-n44-k7	B-n64-k9
A-n33-k6	A-n46-k7	A-n65-k9	B-n45-k5	B-n66-k9
A-n34-k5	A-n48-k7	A-n69-k9	B-n45-k6	B-n67-k10
A-n36-k5	A-n53-k7	A-n80-k10	B-n50-k7	B-n68-k9
A-n37-k5	A-n54-k7	B-n31-k5	B-n50-k8	B-n78-k10
A-n37-k6	A-n55-k9	B-n34-k5	B-n51-k7	F-n135-k7
A-n38-k5	A-n60-k9	B-n35-k5	B-n52-k7	F-n45-k4
A-n39-k5	A-n61-k9	B-n38-k6	B-n56-k7	F-n72-k4
A-n39-k6	A-n62-k8	B-n39-k5	B-n57-k7	
A-n44-k7	A-n63-k10	B-n41-k6	B-n57-k9	

Na segunda, os testes visavam avaliar como o parâmetro λ interfere na qualidade das soluções. Esses testes são detalhados a seguir.

4.7.1 Testes exploratórios

A primeira etapa de testes foi executada com a seguinte configuração:

- Método construtivo: aleatório.
- Método de busca local: BLC.
- Número de iterações: 2000, 5000 e 2000000.
- Parâmetro a: valores $0.1, 0, 2, 0.3, \ldots, 1.0$ para 2000 iterações, 0.1 e 0.2 para 5000 iterações, 0.4 para 2000000 de iterações.
- Quantidade de execuções: 10 por combinação de parâmetros.

Os resultados obtidos estão resumidos na tabela 2. Nesta, há uma linha para cada instância testada e as colunas representam, na ordem em que aparecem da esquerda para a direita, o que é descrito abaixo:

1ª) Nome da instância.

2ª) Valor do custo da solução ótima conhecida, obtido de (UCHOA et al., 2014).

3ª) Valor da média dos custos das soluções obtidas nos testes.

4ª) Melhor valor de custo dentre as soluções obtidas.

5ª) Pior valor de custo de solução obtida.

6ª) Desvio padrão dos valores de custos das soluções obtidas.

7ª) Distância do ótimo D_O, calculada pela equação 4.5, onde $c(s_O)$ é o custo da solução ótima e $c(s_*)$ é o melhor valor de custo dentre as soluções obtidas.

$$D_O = \left(1 - \frac{c(s_O)}{c(s_*)}\right) \times 100 \tag{4.5}$$

8ª) Média do tempo de processamento das execuções do método, em milissegundos.

É possível observar que o método de aplicação do GLS ao PRVC conseguiu atingir a solução ótima, nos testes exploratórios, para 37 das 53 instâncias testadas. Para 14 das demais, o melhor resultado foi distante do ótimo em menos de 0.4%. Para uma instância, "A-n69-k9", o melhor resultado obtido foi 0.77% distante do ótimo. Por não haver uma condição, na implementação feita, que impusesse ao método de busca restringir o espaço de soluções viáveis apenas às que tivessem o número de rotas igual a k, para a instância "B-n57-k7" a busca foi encaminhada para soluções próximas ou até abaixo do ótimo conhecido, mas com 8 rotas. Embora haja aplicações práticas do PRVC em que seja aplicável usar mais rotas do que o valor estabelecido em k, neste trabalho esses valores foram descartados.

Os valores das médias dos custos das soluções obtidas ficaram menos de 6% distantes do valor ótimo para 51 das 53 instâncias usadas nos testes. Para as instâncias "F-n72-k4" e "F-n135-k7", esses valores ficaram 10.2% e 15% distantes do ótimo, respectivamente. Algum aspecto dessas instâncias pode ter tornado mais difícil para que a busca convergisse corretamente para o valor ótimo. Já os piores resultados obtidos ficaram até 30% distantes dos valores ótimos, mas para 46 instâncias essa distância foi menor que 15%.

Os tempos de processamento dependem das quantidades de soluções avaliadas nas buscas, de locais e de veículos das instâncias. A quantidade média de soluções avaliadas variou de 4.4×10^6 até 4.6×10^7, enquanto as médias do tempo de processamento variaram de 1.2s, com as instâncias com menos locais, até 3min32s com a instância "F-n135-k7".

Os testes exploratórios indicaram fortemente que a meta-heurística GLS pode ser aplicada satisfatoriamente ao PRVC em casos de restrição de tempo disponível para a determinação de soluções, com possibilidade de aceitar soluções aproximadas. A seguir, são descritos os testes feitos para avaliar a influência do parâmetro λ nos resultados.

4.7.2 Testes de parâmetro da meta-heurística GLS

Na segunda etapa de testes, como a meta foi o estudo do impacto de variações do parâmetro λ na qualidade dos resultados obtidos, optou-se por usar soluções iniciais

Tabela 2 – Resultados dos testes exploratórios.

Instância	v.ótimo	v.médio	melhor	pior	desv.pad	dist.ótimo	t.médio
A-n32-k5	784	803.03	784	829	20.22	0.00	1198.7
A-n33-k5	661	661.42	661	671	1.67	0.00	1243.4
A-n33-k6	742	742.67	742	774	2.94	0.00	1260.3
A-n34-k5	778	782.53	778	812	5.03	0.00	1339.7
A-n36-k5	799	810.36	799	840	7.63	0.00	1303.6
A-n37-k5	669	673.93	669	711	8.62	0.00	1746.9
A-n37-k6	949	959.08	949	982	8.79	0.00	1263.9
A-n38-k5	730	737.97	730	787	11.12	0.00	1182.0
A-n39-k5	822	827.09	822	839	3.07	0.00	1416.8
A-n39-k6	831	837.00	831	923	8.99	0.00	4683.6
A-n44-k7	937	945.05	937	981	8.68	0.00	1667.3
A-n45-k6	944	994.25	944	1096	34.30	0.00	1617.8
A-n45-k7	1146	1160.43	1146	1217	13.43	0.00	1967.6
A-n46-k7	914	930.59	914	991	19.72	0.00	1859.6
A-n48-k7	1073	1105.03	1073	1156	15.52	0.00	2007.8
A-n53-k7	1010	1029.84	1010	1153	15.53	0.00	7925.8
A-n54-k7	1167	1194.06	1167	1272	18.90	0.00	3575.3
A-n55-k9	1073	1085.16	1073	1126	12.59	0.00	2869.4
A-n60-k9	1354	1381.58	1354	1505	23.60	0.00	13737.9
A-n61-k9	1034	1067.24	1035	1143	29.13	0.10	2791.6
A-n62-k8	1288	1318.47	1289	1454	16.69	0.08	34051.3
A-n63-k10	1314	1333.71	1315	1420	13.31	0.08	10321.5
A-n63-k9	1616	1645.48	1618	1722	18.09	0.12	25960.7
A-n64-k9	1401	1440.83	1402	1574	20.65	0.07	90639.1
A-n65-k9	1174	1210.20	1177	1314	25.11	0.25	3884.4
A-n69-k9	1159	1198.30	1168	1292	23.42	0.77	4972.3
A-n80-k10	1763	1820.91	1764	1933	26.05	0.06	22352.6
B-n31-k5	672	677.61	672	685	3.21	0.00	1218.8
B-n34-k5	788	789.15	788	795	1.51	0.00	1533.0
B-n35-k5	955	965.63	955	988	10.85	0.00	1457.5
B-n38-k6	805	809.10	805	843	4.81	0.00	1501.3
B-n39-k5	549	561.38	549	705	27.45	0.00	1753.4
B-n41-k6	829	852.30	829	929	16.14	0.00	2317.1
B-n43-k6	742	749.51	742	765	4.62	0.00	1881.1
B-n44-k7	909	931.55	909	990	15.03	0.00	1603.1
B-n45-k5	751	769.18	751	892	17.68	0.00	2235.4
B-n45-k6	678	714.75	678	777	21.66	0.00	1436.4
B-n50-k7	741	761.63	741	972	36.80	0.00	3016.2
B-n50-k8	1312	1327.11	1313	1345	6.24	0.08	2698.3
B-n51-k7	1032	1058.05	1032	1215	48.00	0.00	2526.0
B-n52-k7	747	769.38	747	870	31.11	0.00	3018.1
B-n56-k7	707	730.27	707	827	23.61	0.00	3416.4
B-n57-k7	1153	1163.29	1169	1604	66.20	1.37	3801.0
B-n57-k9	1598	1637.51	1598	1674	18.12	0.00	49293.9
B-n63-k10	1496	1546.88	1497	1620	21.25	0.07	20624.2
B-n64-k9	861	893.33	861	999	28.49	0.00	12256.2
B-n66-k9	1316	1346.61	1320	1445	23.60	0.30	26922.6
B-n67-k10	1032	1058.07	1033	1185	16.37	0.10	23000.1
B-n68-k9	1272	1311.23	1276	1431	26.29	0.31	28708.2
B-n78-k10	1221	1264.96	1222	1360	24.53	0.08	43510.5
F-n135-k7	1162	1367.27	1165	1678	106.21	0.26	212008.8
F-n45-k4	724	742.95	724	810	20.60	0.00	3412.7
F-n72-k4	237	263.92	237	305	14.30	0.00	11497.2

únicas para cada instância. Essas soluções foram geradas pelo método descrito na seção 4.3.2. O número de iterações também foi fixado a somente um valor por instância. O método de busca local foi estabelecido como BLR, visando reduzir o esforço computacional despendido. A configuração dos testes foi a seguinte:

- Método construtivo: guloso.
- Método de busca local: BLR.
- Número de iterações: disposto na tabela 3, calculado de acordo com os valores obtidos na primeira etapa de testes, aproximando[1] para o próximo número múltiplo de 100 o resultado da equação 4.6, onde:
  - $N_{iter}[instância]$ é o número de iterações calculado para ser usado com uma dada instância,
  - $\min(N^*_{\text{iter}}[\text{instância}])$ corresponde ao número mínimo de iterações até atingir a melhor solução encontrada para uma dada instância, na etapa de testes anterior, e
  - $\text{avg}(N^*_{\text{iter}}[\text{instância}])$ é a média do número de iterações até atingir a melhor solução encontrada para uma dada instância, também na etapa de testes anterior.

$$N_{\text{iter}} = \lfloor \frac{\text{avg}(N^*_{\text{iter}}[\text{instância}]) + 4 \times \min(N^*_{\text{iter}}[\text{instância}])}{5} \rfloor \tag{4.6}$$

- Parâmetro a: valores $0.01, 0,02, 0.03, \ldots, 1.0$.
- Quantidade de execuções: uma por combinação de parâmetros.

Os resultados obtidos na segunda etapa de testes estão expostos na tabela 4, que tem a mesma estrutura da tabela de resultados da primeira etapa de testes. Pode-se observar que a configuração desta etapa de testes, que envolveu avaliar mais valores de λ dentro dos mesmos limites mínimo e máximo, possibilitou atingir as soluções ótimas globais para 44 das 53 instâncias usadas nos testes. Para 5 das 9 demais instâncias, foram obtidas soluções com custos menos de 0.1% distantes do valor ótimo. Para 3 das instâncias restantes, a distância do custo da melhor solução obtida para o ótimo foi de menos de 1%.

Apenas a melhor solução obtida para a instância "B-n57-k7" ficou aquém da qualidade geral do método, distante do valor ótimo em 13.76%. Para esta instância, somente com $\lambda = 27\frac{2}{3} \times 10^{-2}$ ($a = 0.01$) foi obtida solução com $k = 7$. O motivo que afetou

[1] Esta adaptação do número de iterações para um valor múltiplo de 100 teve por objetivo a simplificação do agrupamento de dados para o estudo iniciado sobre a convergência do método.

Tabela 3 – Número de iterações por instância na segunda etapa de testes.

Instância	Iterações
A-n32-k5	12600
A-n33-k5	7600
A-n33-k6	6100
A-n34-k5	8100
A-n36-k5	12100
A-n37-k5	8400
A-n37-k6	19400
A-n38-k5	10500
A-n39-k5	25300
A-n39-k6	10300
A-n44-k7	12400
A-n45-k6	130800
A-n45-k7	57100
A-n46-k7	13300
A-n48-k7	38600
A-n53-k7	88800
A-n54-k7	88700
A-n55-k9	33600

Instância	Iterações
A-n60-k9	40100
A-n61-k9	138100
A-n62-k8	650200
A-n63-k10	170200
A-n63-k9	94400
A-n64-k9	1301000
A-n65-k9	1519700
A-n69-k9	913000
A-n80-k10	801300
B-n31-k5	20400
B-n34-k5	13100
B-n35-k5	59900
B-n38-k6	73400
B-n39-k5	20500
B-n41-k6	124700
B-n43-k6	25800
B-n44-k7	25100
B-n45-k5	73500

Instância	Iterações
B-n45-k6	37200
B-n50-k7	9000
B-n50-k8	76600
B-n51-k7	46200
B-n52-k7	72800
B-n56-k7	201800
B-n57-k7	28500
B-n57-k9	753200
B-n63-k10	40900
B-n64-k9	302700
B-n66-k9	1184400
B-n67-k10	1480400
B-n68-k9	453700
B-n78-k10	836000
F-n135-k7	2101300
F-n45-k4	14000
F-n72-k4	31400

este resultado foi o mesmo relatado na apresentação dos resultados da primeira etapa de testes.

As piores soluções encontradas ficaram menos de 6% distantes do ótimo para 51 das instâncias usadas para os testes. Para a instância "F-n72-k4", a pior solução ficou 10.23% distante do ótimo. Quanto às médias dos custos das soluções encontradas, para 51 instâncias esses valores ficaram menos de 2% distantes do ótimo, enquanto para uma, "B-n63-k10", esta distância ficou em 3%. Os valores do desvio padrão dos custos das soluções encontradas ficou abaixo de 11% para 52 das instâncias.

As médias de quantidades de soluções avaliadas nos testes variou de 6.9×10^5 a 1.4×10^7, este último para a instância "F-n135-k7". Observou-se que o uso do método BLR possibilitou avaliar menos soluções durante a busca, permitindo que mais iterações do GLS pudessem ser executadas, por intervalo de tempo, do que com o uso de BLC. As médias de tempo de processamento variaram de 0.3s, para instâncias com poucos locais, a 9min56s, para a instância "F-n135-k7". Esses valores variaram menos entre os testes para uma mesma instância em relação à primeira etapa, pois a quantidade de iterações era constante.

Para investigar o impacto do parâmetro λ sobre a busca, os testes foram repetidos com o registro das iterações em que havia mudança na melhor solução encontrada. O objetivo foi estabelecer possíveis relações entre λ e os resultados, para que fosse possível calibrar este parâmetro para reduzir o esforço computacional até chegar às melhores so-

Tabela 4 – Resultados dos testes com variação do parâmetro λ.

Instância	v.ótimo	v.médio	melhor	pior	desv.pad	dist.ótimo	t.médio
A-n32-k5	784	786.01	784	828	7.10	0.00	562.4
A-n33-k5	661	661.51	661	669	1.55	0.00	362.6
A-n33-k6	742	743.18	742	766	3.37	0.00	306.1
A-n34-k5	778	781.37	778	792	3.80	0.00	382.7
A-n36-k5	799	807.85	799	820	5.84	0.00	525.9
A-n37-k5	669	672.39	669	697	5.08	0.00	426.1
A-n37-k6	949	951.64	949	960	2.90	0.00	798.8
A-n38-k5	730	735.00	730	747	4.63	0.00	437.6
A-n39-k5	822	824.06	822	833	1.93	0.00	1163.2
A-n39-k6	831	834.02	831	840	1.47	0.00	564.6
A-n44-k7	937	942.65	937	963	5.04	0.00	587.3
A-n45-k6	944	956.68	944	979	5.26	0.00	5428.5
A-n45-k7	1146	1147.71	1146	1152	1.78	0.00	3273.7
A-n46-k7	914	919.12	914	967	8.28	0.00	773.3
A-n48-k7	1073	1088.02	1073	1111	10.76	0.00	2149.1
A-n53-k7	1010	1016.47	1010	1024	3.45	0.00	5525.1
A-n54-k7	1167	1171.04	1167	1182	3.39	0.00	5541.5
A-n55-k9	1073	1076.17	1073	1088	3.23	0.00	2471.2
A-n60-k9	1354	1365.72	1354	1399	6.95	0.00	3314.6
A-n61-k9	1034	1038.05	1034	1051	2.96	0.00	8439.1
A-n62-k8	1288	1297.58	1288	1309	6.06	0.00	46011.3
A-n63-k10	1314	1319.79	1314	1326	2.40	0.00	14362.9
A-n63-k9	1616	1632.09	1617	1643	4.12	0.06	6098.3
A-n64-k9	1401	1412.11	1402	1420	4.08	0.07	94843.7
A-n65-k9	1174	1179.00	1174	1184	2.72	0.00	87095.6
A-n69-k9	1159	1167.12	1159	1173	2.90	0.00	77945.6
A-n80-k10	1763	1778.57	1764	1791	5.98	0.06	79216.8
B-n31-k5	672	672.49	672	675	1.01	0.00	1145.9
B-n34-k5	788	788.72	788	794	1.23	0.00	735.6
B-n35-k5	955	956.18	955	961	1.60	0.00	3039.8
B-n38-k6	805	805.45	805	806	0.50	0.00	4098.7
B-n39-k5	549	550.35	549	565	2.03	0.00	1371.1
B-n41-k6	829	831.84	829	865	3.95	0.00	5169.7
B-n43-k6	742	745.08	742	749	1.76	0.00	1601.6
B-n44-k7	909	914.18	909	935	5.04	0.00	1376.5
B-n45-k5	751	752.39	751	757	1.35	0.00	3493.1
B-n45-k6	678	691.32	678	716	6.59	0.00	1471.2
B-n50-k7	741	743.27	741	750	2.26	0.00	818.2
B-n50-k8	1312	1319.49	1313	1326	3.19	0.08	5070.0
B-n51-k7	1032	1044.78	1039	1053	5.07	0.67	2191.0
B-n52-k7	747	748.75	747	753	1.42	0.00	5737.7
B-n56-k7	707	710.12	707	714	1.76	0.00	16169.4
B-n57-k7	1153	1337.00	1337	1337	0.00	13.76	1326.0
B-n57-k9	1598	1599.18	1598	1601	0.79	0.00	58515.5
B-n63-k10	1496	1542.31	1502	1561	13.68	0.40	3085.9
B-n64-k9	861	864.95	861	878	2.62	0.00	21947.5
B-n66-k9	1316	1319.37	1316	1322	1.31	0.00	92044.3
B-n67-k10	1032	1038.74	1032	1046	2.74	0.00	138721.3
B-n68-k9	1272	1282.03	1273	1295	4.07	0.08	41388.7
B-n78-k10	1221	1230.62	1221	1243	5.92	0.00	86592.3
F-n135-k7	1162	1176.97	1164	1195	8.07	0.17	595925.2
F-n45-k4	724	725.73	724	733	2.18	0.00	1238.7
F-n72-k4	237	241.77	237	264	5.76	0.00	5321.0

luções. Os gráficos apresentados na figura 4.7.2 mostram a convergência das buscas para a instância "F-n135-k7" com variação de a entre 0.01 e 0.10. O eixo horizontal de cada gráfico corresponde aos números das iterações, e o eixo vertical ao custo da melhor solução encontrada. Dada a oscilação da convergência observada nos diferentes gráficos, que também ocorreu com as outras instâncias, e a obscuridade dos fatores que influenciam isto, esta análise mostrou-se insuficiente para identificar relações entre λ e a qualidade dos resultados ou à convergência da busca. Diante deste cenário, sugere-se que um estudo mais detalhado, possivelmente auxiliado por técnicas de ajuste de parâmetros com aplicação de redes neurais artificiais, como foi proposto em (DOBSLAW, 2010), seja aplicado para a obtenção do melhor valor de λ de acordo com algum conjunto de características específicas de cada instância.

4.8 Considerações sobre o capítulo

Este capítulo apresentou o estudo da aplicação da meta-heurística GLS como abordagem à resolução do PRVC. Foram expostas as estruturas de dados usadas na implementação feita, os métodos empregados na construção de soluções iniciais, na busca em vizinhança e na busca local, além das funções usadas no núcleo do algoritmo GLS. Experimentos foram realizados com uma coleção de instâncias, obtidas a partir da literatura citada, usadas para comparar diferentes métodos de resolução. Com os resultados dos testes realizados, foi possível observar que houve um bom desempenho geral, obtendo soluções ótimas ou próximas ao ótimo para a maior parte das instâncias avaliadas. No entanto, a calibragem do parâmetro λ permaneceu obscura, sendo necessário intensificar o estudo da relação entre este, os dados das instâncias e a convergência da busca para chegar a informações conclusivas sobre este parâmetro.

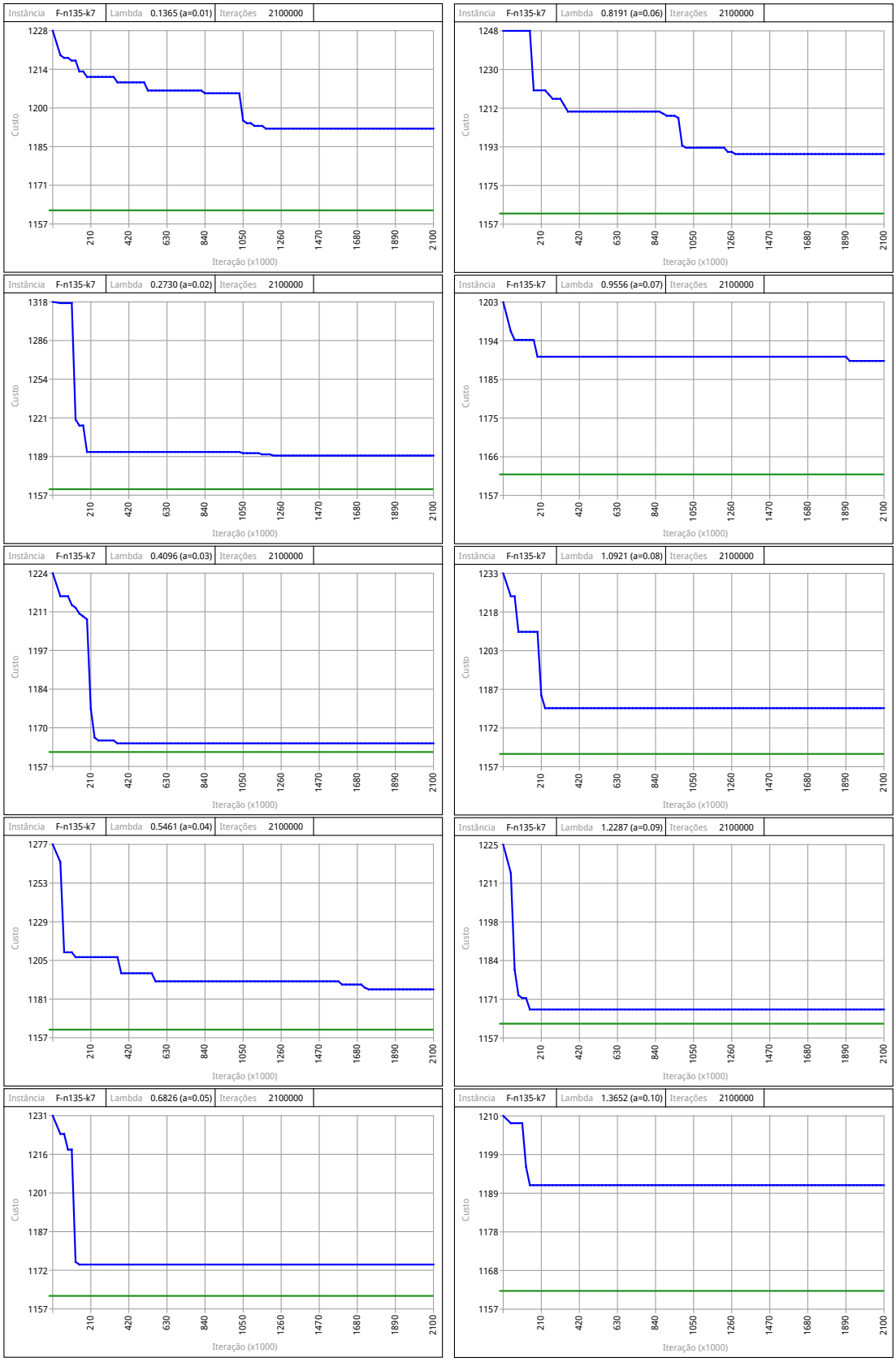


Figura 10 – Convergência com variação do parâmetro λ.

5 Conclusão

A meta-heurística GLS constitui uma variação dos métodos de otimização combinatória baseados em busca local, pois trata da diversificação através da aplicação de penalidades a características dos ótimos locais encontrados. Esta diversificação, feita suavemente, altera a função objetivo e conduz as buscas locais subsequentes a outros mínimos locais, possibilitando expandir o espaço de soluções avaliado e, consequentemente, levar ao conhecimento de boas soluções.

Neste trabalho, foi feita a aplicação de GLS à resolução do PRVC. Os resultados obtidos indicam que esta meta-heurística pode ser usada para resolver este problema em casos de necessidade de obtenção de soluções aproximadas em tempo relativamente pequeno. A continuidade desta pesquisa pode ser feita aplicando a implementação feita a instâncias maiores do problema, o que poderia levar a análises mais conclusivas sobre a real aplicabilidade do método empregado.

Algumas melhorias podem ser feitas na implementação proposta neste trabalho, como o uso de estruturas de dados mais sofisticadas que reduzam a complexidade das operações de obtenção de vizinhos e, consequentemente, da busca local. O uso de outras estruturas de vizinhança também pode levar a melhor convergência para soluções ótimas. Além disso, o ajuste do algoritmo de busca em vizinhança para impor o número fixo de rotas possibilitaria atender plenamente aos casos em que há esta restrição. Estas melhorias, bem como o estudo da calibragem do parâmetro λ e a continuidade dos experimentos, ficam como sugestões para trabalhos futuros.

Referências

AUGERAT, P. *Approche polyèdrale du problème de tournées de véhicules*. Tese (Doutorado) — Institut National Polytechnique de Grenoble-INPG, 1995. Citado na página 26.

BALDACCI, R. et al. Algorithms for nesting with defects. *Discrete Applied Mathematics*, Elsevier, v. 163, p. 17–33, 2014. Citado na página 11.

DANTZIG, G. B.; RAMSER, J. H. The truck dispatching problem. *Management science*, Informs, v. 6, n. 1, p. 80–91, 1959. Citado na página 13.

DOBSLAW, F. A parameter tuning framework for metaheuristics based on design of experiments and artificial neural networks. In: WASET. *International Conference on Computer Mathematics and Natural Computing*. [S.l.], 2010. Citado na página 33.

EGEBLAD, J.; NIELSEN, B. K.; ODGAARD, A. Fast neighborhood search for two- and three-dimensional nesting problems. *European Journal of Operational Research*, v. 183, n. 3, p. 1249–1266, 2007. ISSN 0377-2217. Citado na página 10.

FISHER, M. L. Optimal solution of vehicle routing problems using minimum k-trees. *Operations research*, INFORMS, v. 42, n. 4, p. 626–642, 1994. Citado na página 26.

FREDMAN, M. et al. Data structures for traveling salesmen. *Journal of Algorithms*, v. 18, n. 3, p. 432 – 479, 1995. ISSN 0196-6774. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0196677485710188>. Citado na página 17.

FUKASAWA, R. et al. Robust branch-and-cut-and-price for the capacitated vehicle routing problem. *Mathematical programming*, Springer, v. 106, n. 3, p. 491–511, 2006. Citado na página 15.

LAPORTE, G. The vehicle routing problem: An overview of exact and approximate algorithms. *European Journal of Operational Research*, Elsevier, v. 59, n. 3, p. 345–358, 1992. Citado na página 15.

LIN, S. Computer solutions of the traveling salesman problem. *The Bell System Technical Journal*, Alcatel-Lucent, v. 44, n. 10, p. 2245–2269, 1965. Citado na página 20.

RALPHS, T. K. et al. On the capacitated vehicle routing problem. *Mathematical programming*, Springer, v. 94, n. 2-3, p. 343–359, 2003. Citado na página 15.

SILVA, T. A.; SILVA, G. P. O uso da metaheurística guided local search para resolver o problema de escala de motoristas de ônibus urbano. 2015. Citado 2 vezes nas páginas 8 e 10.

TOTH, P.; VIGO, D. Models, relaxations and exact approaches for the capacitated vehicle routing problem. *Discrete Applied Mathematics*, Elsevier, v. 123, n. 1, p. 487–512, 2002. Citado 3 vezes nas páginas 13, 14 e 15.

UCHOA, E. et al. *New Benchmark Instances for the Capacitated Vehicle Routing Problem*. [S.l.], 2014. Citado 2 vezes nas páginas 13 e 27.

VOUDOURIS, C. Guided local search — an illustrative example in function optimisation. *BT Technology Journal*, v. 16, n. 3, p. 46–50, 1998. ISSN 1573-1995. Citado 2 vezes nas páginas 8 e 9.

VOUDOURIS, C.; TSANG, E. P. K. *Guided Local Search*. Colchester, UK, 1995. Citado 4 vezes nas páginas 4, 6, 25 e 26.

VOUDOURIS, C.; TSANG, E. P. K.; ALSHEDDY, A. Guided Local Search. *Handbook of Metaheuristics*, v. 146, n. August, p. 321–361, 2010. ISSN 978-1-4419-1665-5. Citado na página 6.