Pontifícia Universidade Católica
do Rio de Janeiro

Toni Tiago da Silva Pacheco

Buscas Eficientes em Vizinhanças Largas para o Problema do Caixeiro Viajante com Coleta e Entrega

Dissertação de Mestrado

Dissertação apresentada como requisito parcial para obtenção do grau de Mestre pelo Programa de Pós–graduação em Informática do Departamento de Informática do Centro Técnico Científico da PUC-Rio.

Orientador: Prof. Thibaut Victor Gaston Vidal

Rio de Janeiro
Abril de 2018

Pontifícia Universidade Católica
do Rio de Janeiro

Toni Tiago da Silva Pacheco

Buscas Eficientes em Vizinhanças Largas para o Problema do Caixeiro Viajante com Coleta e Entrega

Dissertação apresentada como requisito parcial para obtenção do grau de Mestre pelo Programa de Pós–graduação em Informática do Departamento de Informática do Centro Técnico Científico da PUC-Rio. Aprovada pela Comissão Examinadora abaixo assinada.

Prof. Thibaut Victor Gaston Vidal
Orientador
Departamento de Informática – PUC-Rio

Prof. Rafael Martinelli Pinto
Departamento de Engenharia Industrial – PUC-Rio

Prof. Túlio Angelo Machado Toffolo
Departamento de Computação – UFOP

Prof. Márcio da Silveira Carvalho
Coordenador Setorial do Centro Técnico Científico – PUC-Rio

Rio de Janeiro, 12 de Abril de 2018

Toni Tiago da Silva Pacheco

Graduou-se em Ciência da Computação pela Universidade Federal Fluminense em 2010. Concluiu a Pós-Graduação em Engenharia de Petróleo pela Universidade Federal Fluminense em 2015. Atua na Indústria de Óleo e Gás há mais de 10 anos como Consultor de Tecnologia. Em 2018, recebeu o certificado profissional de *Data Science* e *Big Data Analytics* pelo *Massachusetts Institute of Technology*, MIT.

Ficha Catalográfica

Pacheco, Toni Tiago da Silva

Buscas Eficientes em Vizinhanças Largas para o Problema do Caixeiro Viajante com Coleta e Entrega / Toni Tiago da Silva Pacheco ; orientador: Thibaut Victor Gaston Vidal. – 2018.

67 f. : il. color. ; 30 cm

Dissertação (Mestrado)–Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Informática, 2018.

Inclui bibliografia

1. Informática – Teses. 2. Caxeiro viajante; 3. Coleta-entrega; 4. Busca local; 5. Vizinhança; 6. Programação dinâmica. I. Vidal, Thibaut Victor Gaston II. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Informática. III. Título.

CDD: 004

Agradecimentos

A Deus por tudo.

Aos meus amados pais Rosa e Carlos agradeço por todo amor, carinho, educação e princípios que me foram dado. Sem vocês não chegaria até aqui.

Ao meu irmão pela amizade leal e companheirismo que sei que posso contar a todo momento.

Ao meu orientador, Professor Thibaut Vidal, expresso minha profunda gratidão pela orientação, paciência, amizade, brilhantismo e por todo conhecimento transmitido.

Ao Professor Rafael Martinelli e ao Professor Túlio Toffolo pela ativa colaboração e ajuda neste trabalho.

À Halliburton, principalmente ao meu gerente Carlos Henrique, por todo o período concedido e incentivo para realização do mestrado.

Agradeço à CAPES pelo apoio financeiro e à PUC-Rio pela bolsa de isenção de mensalidades do mestrado.

A todos os amigos que sempre estiveram presentes dando conselhos e disponíveis para ouvir quando precisei. Dentre estes não posso deixar de citar Marcelo Gomes, pela companhia em algumas disciplinas nessa jornada e Martha Salles pela companhia, força e risadas em momentos árduos.

Resumo

Pacheco, Toni Tiago da Silva ; Vidal, Thibaut Victor Gaston. **Buscas Eficientes em Vizinhanças Largas para o Problema do Caixeiro Viajante com Coleta e Entrega**. Rio de Janeiro, 2018. 67p. Dissertação de Mestrado – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

Em vários problemas de distribuição e logística, os produtos devem ser coletados em uma origem e entregues em um destino. Exemplos incluem o transporte de pessoas com deficiência, serviços de correio expresso, logística de suprimentos médicos, etc. O problema de roteamento abordado neste trabalho, conhecido como *Traveling Salesman Problem with Pickup and Delivery* (TSPPD), é da classe de problemas do caixeiro viajante com restrições de precedência. Neste problema, existe um mapeamento um-para-um entre coleta-entrega no qual cada cliente do tipo coleta possui um cliente do tipo entrega associado. Os clientes do tipo entrega somente podem ser visitados posteriormente à coleta associada. O TSPPD é um problema NP-difícil uma vez que generaliza o *Traveling Salesman Problem* (TSP). O TSP pode ser visto como um caso particular do TSPPD onde cada coleta coincide espacialmente com a respectiva entrega. As variantes com restrições de capacidade, janelas de tempo e diferentes políticas de carregamento têm recebido maior atenção na última década, embora ainda existam significantes avanços a serem realizados em termos de qualidades de soluções na versão básica do problema. Para resolver este problema, propomos um algoritmo meta-heurístico híbrido com vizinhanças largas exploradas eficientemente em $\mathcal{O}(n^2)$. Nossos experimentos demonstram uma redução significativa no tempo computacional e também melhoria na qualidade de soluções previamente conhecidas na literatura.

Palavras-chave

Caxeiro viajante; Coleta-entrega; Busca local; Vizinhança; Programação dinâmica.

Abstract

Pacheco, Toni Tiago da Silva ; Vidal, Thibaut Victor Gaston (Advisor). **Efficient Large Neighborhood Searches for the Traveling Salesman Problem with Pickup and Delivery**. Rio de Janeiro, 2018. 67p. Dissertação de Mestrado – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

In various distribution and logistics issues, products must be collected at one source and delivered to a destination. Examples include disabled people transportation, express mail services, medical supplies logistics, etc. The routing problem addressed by this work, known as Traveling Salesman Problem with Pickup and Delivery (TSPPD), belongs to the class of traveling salesman problems with precedence constraints. In this problem, there is a one-to-one pickup-delivery mapping in which, for each pickup-type client, there is exactly one associated delivery-type client. Delivery clients can only be visited after the associated pickup. Since the TSPPD generalizes the TSP it is also a NP-hard problem, as the TSP is a particular casa of TSPPD where each pickup matches spatially with it's respective delivery. Variants with capacity constraints, time windows and different loading policies have received more attention in the last decade, although there are still significant advances to be made in terms of solution quality for the basic version of the problem. To solve this problem, we propose a hybrid metaheuristic algorithm with large neighborhoods efficiently explored in $\mathcal{O}(n^2)$. Our experiments demonstrate a significant computational time reduction and also solutions quality improvement compared to the previous works.

Keywords

Traveling salesman problem; Pickup-and-delivery; Local search; Neighborhood; Dynamic programming.

Sumário

Lista de figuras

Lista de tabelas

1
Introdução

Provedores de serviços são comumente contratados para transportar mercadorias de um local para outro. Assim, o problema de transporte com coleta e entrega é tratado em diversas aplicações como coleta de produtos em fornecedores e entregas em supermercados, coleta de medicamentos em hospitais e entrega em casa de pacientes e, ainda, serviços de transporte de pessoas idosas ou portadoras de deficiência. Cenários em que há busca pela qualidade dos serviços são problemas típicos de otimização que envolvem minimizar custos de transporte. Em virtude disto, alguns estudos envolvendo esta temática de otimização de custo de transportes foram propostos ao longo dos anos.

O problema do caixeiro viajante com coleta e entrega (*Traveling Salesman Problem with Pickup and Delivery, TSPPD*) surge nesse contexto como uma variante do bem conhecido problema do caixeiro viajante (*Traveling Salesman Problem, TSP*) com a adição de restrições de precedência. No TSPPD, partindo do depósito, uma quantidade n de requisições para coletar um objeto em um ponto e o entregar em um outro ponto específico devem ser realizadas, visitando cada local apenas uma vez. Estes locais são mapeados um-para-um e as entregas só podem ser visitadas após as respectivas coletas serem efetuadas.

Diversos levantamentos foram feitos na literatura sobre os problemas de roteamento com coleta e entrega [1, 2, 3, 4, 5, 6]. Estes levantamentos mostram que esta classe de problemas tem sido abordada de forma exata e heurística por anos. Outras variantes do TSPPD com números de veículos maiores que um, janela de tempo de serviço e diferentes políticas de coleta e entrega foram o foco da atenção.

Em trabalhos que tratam diretamente o TSPPD [7, 8, 9, 10] foi enfatizado que pouca atenção foi dada a esse problema recentemente. Os algoritmos exatos foram capazes de solucionar problemas com até 35 requisições, um número pequeno comparado à quantidade de requisições abordadas pelos métodos heurísticos. Embora heurísticas mostrem melhor eficiência ao lidar com quantidades maiores de requisições, existe uma maior dificuldade de alcançar os melhores valores conhecidos em problemas com número de requisições maiores. Tal fato sugere a possibilidade de avanços neste campo de estudo.

O presente trabalho propõe um algoritmo genético híbrido para o TSPPD com buscas eficientes em vizinhanças largas (*Hybrid Genetic Search with Large Neighborhoods*, HGS-LN). A busca local realiza uma pré busca em três vizinhanças e quando não é mais possível melhorar a solução, o mesmo processo é executado em três vizinhanças largas. Além do potencial de escapar de mínimos locais do algoritmo genético com controle avançado de diversidade, a grande quantidade de movimentos cobertos pelas vizinhanças largas é capaz de evitar mínimos locais atingidos pela busca anterior e continuar a melhoria da solução.

Os experimentos computacionais mostraram que o HGS-LN melhorou resultados obtidos anteriormente. Em um total de 163 instâncias divididas em três classes, os melhores valores conhecidos para duas classes de instâncias foram encontrados em todas as rodadas, enquanto na terceira classe, com 108 instâncias, as soluções foram melhoradas em média de 0.30%, encontrando novas melhores soluções para mais da metade das instâncias. O tempo computacional também foi inferior quando comparado aos trabalhos anteriores.

Os capítulos seguintes foram estruturados de forma a facilitar a compreensão do algoritmo proposto. No Capítulo 2 é realizada uma descrição breve do problema proposto e uma revisão dos principais trabalhos da literatura relacionados ao mesmo. No Capítulo 3, as vizinhanças utilizadas nas buscas locais e suas estruturações são detalhadas. O Capítulo 4 apresenta o algoritmo genético híbrido desenvolvido para o TSPPD e o funcionamento dos operadores de seleção, reprodução e mutação. No Capítulo 5 estão expostos os resultados computacionais, e no Capítulo 6, as conclusões obtidas e as etapas futuras de pesquisa.

2
O Problema do Caixeiro Viajante com Coleta e Entrega

O problema de caixeiro viajante implica em fazer coletas ou entregas, sem repetir qualquer local, minimizando a distância total da viagem. Em 1979, uma discussão sobre adicionar restrições de precedência ao TSP foi iniciada por Lokin [11]. Desde então, muitos estudos relacionados a este problema propondo diferentes considerações de precedências foram realizados.

Após a introdução do primeiro método exato formulado por Kalantari et al. [12], em 1985, considerando um ou mais veículos com capacidades finitas ou infinitas, o problema com precedência de coleta-entrega ganhou mais atenção de pesquisadores. Savelsbergh e Sol [13], em 1995, realizaram um levantamento dos métodos exatos estudados e propuseram uma notação geral para diversas variantes do problema de coleta e entrega. Na década passada, outras revisões sobre variantes do roteamento de veículos com coleta e entrega foram também realizadas [1, 2, 3, 4, 5, 6]. Tendo em vista estas revisões, foi possível notar que, apesar do problema TSP com restrições de precedência ter sido bastante discutido, poucos trabalhos foram dedicados ao problema base do TSPPD. Este fato também foi percebido por Venstra et al. [7], Dumitrescu et al. [8] e Renaud et al. [9, 10] cujos estudos abordaram diretamente o TSPPD.

2.1
Formulação Matemática

O problema do caixeiro viajante com coleta e entrega pode ser descrito por um grafo não orientado, nos quais os vértices representam os locais ou cidades e as arestas, os deslocamentos percorridos entre os mesmos. É considerada a premissa de que a partir de um vértice inicial, o depósito, todos os demais vértices devem ser visitados uma única vez e então, regressar ao depósito. Além disto, as coletas e entregas associadas são uma relação de um-para-um, isto é, um produto coletado em um vértice deverá ser entregue em outro vértice específico. Assim, o objetivo é minimizar a distância total da viagem de modo que cada vértice de coleta seja visitado antes do vértice de entrega correspondente.

Da mesma forma, podemos definir matematicamente o problema, sendo $G = (V, E)$ o grafo não orientado em que V é o conjunto de vértices e E o

conjunto de arestas. O conjunto V consiste de vértices de coletas, entregas e dois vértices referentes ao depósito, representando o mesmo local em contextos de início e fim da rota. Seja n o número de requisições, $P = \{1, ..., n\}$ o conjunto de vértices de coletas e $D = \{n+1, ...2n\}$ o conjunto de entregas, o conjunto de vértices é descrito como $V = P \cup D \cup \{0, 2n+1\}$, 0 e $2n+1$ representando o depósito. Com estas definições, o par coleta-entrega (p–d) é mapeado onde um vértice de coleta $p = i$ é associado à entrega $d = n+i$, para $p \in P$ e $d \in D$. As arestas de G são tais que $E = \{(i,j) : i \neq j;\ i,j \in V\}$. A distância entre dois vértices i e j é definido pelo custo c_{ij} da aresta $(i,j) \in E$.

O objetivo deste problema é minimizar a distância total da rota, Equação (2-1), para atender todas as requisições iniciando e terminando o percurso no depósito.

Utilizando a notação introduzida por Dumitrescu et al. [8], tem-se que $\delta(S) = \{(i,j) \in E : i \in S, j \notin S \ ou \ i \notin S, j \in S\}$ para qualquer subconjunto de vértices, $S \subseteq V$ e, se $S = \{i\}$, assume-se $\delta(i)$ ao invés de $\delta(\{i\})$. Também, a variável binária de decisão x_{ij} é definida para toda aresta $(i,j) \in E$. Com $x(E')$ para $\sum_{(i,j)\in E'} x_{ij}$ no qual $E' \subseteq E$, pode-se:

$$\text{minimizar} \sum_{(i,j)\in E} c_{ij}x_{ij} \tag{2-1}$$

sujeito a:

$$x_{0,2n+1} = 1 \tag{2-2}$$

$$x(\delta(i)) = 2 \quad \forall i \in V \tag{2-3}$$

$$x(\delta(S)) \geq 2 \quad \forall S \subseteq V,\ 3 \leq |S| \leq |V|/2 \tag{2-4}$$

$$x(\delta(S)) \geq 4 \quad \forall S \in \mathcal{U} \tag{2-5}$$

$$x_{ij} \in 0, 1 \quad \forall (i,j) \in E, \tag{2-6}$$

onde $\mathcal{U}$ é o conjunto de subconjuntos $S \subset V$ que satisfaz $3 \leq |S| \leq |V| - 2$ com $0 \in S$, $2n+1 \notin S$ para os quais existe $i \in P$ onde $i \notin S$ e $n+i \in S$. Sendo a restrição (2-2) indicando que a variável binária relacionada ao depósito será sempre igual a 1, (2-3) é relativa ao grau dos vértices, (2-4) elimina sub-rotas e (2-5) são restrições de precedência que garantem que o vértice i seja visitado antes do vértice $n+i$ para todo o $i \in P$.

Nessa formulação o número total de vértices no conjunto V é igual a $2n + 2$, contendo $2n$ vértices de requisições e outros dois para definir o

mesmo depósito como ponto inicial e final. Por fim de simplificação e de uma melhor adequação com outros trabalhos, a quantidade prática de vértices n_v é introduzida se referindo ao número de vértices total sem repetir a localidade do depósito, sendo $n_v = 2n + 1$.

2.2 Métodos exatos

Os métodos exatos visam encontrar a melhor solução para um dado problema, sendo esta o ótimo global. Nas últimas décadas, estes métodos evoluíram consideravelmente e lograram êxito em instâncias com grande número de requisições para o problema do TSP, sendo possível, por exemplo, alcançar a solução ótima para todas instâncias simétricas do TSPLIB [14, 15] com até 85900 vértices, apesar de tempos muito elevados [16].

Em variantes do TSP com restrições de precedência, como o caso do TSPPD, o tamanho máximo das instâncias resolvidas otimamente é bem menor. Kalantari et al. [12] apresentou o primeiro método exato para o TSPPD. Este método baseado no algoritmo de *branch-and-bound*, introduzido por Little et al. [17], conseguiu solucionar instâncias com até 31 vértices que se referem as 15 requisições de coleta-entrega e o vértice de depósito. Posteriormente, Ruland e Rodin [18] propuseram um algoritmo de *branch-and-cut* incluindo a geração de plano de cortes. Neste trabalho, obtiveram ganhos consideráveis em tempo computacional comparados a Kalantari et al. [12], mas sem êxito em instâncias maiores. Na última década, Dumitrescu et al. [8] apresentaram avanços em um algoritmo *branch-and-cut* e resultados poliédricos, com novas desigualdades válidas e procedimentos de separação para o TSPPD. Este algoritmo de *branch-and-cut* foi capaz de resolver novas instâncias propostas e instâncias de Renaud et al. [9], envolvendo até 35 requisições. O resultado foi expressivo mostrando sucesso em mais do que o dobro de vértices comparado aos trabalhos anteriores.

2.3 Métodos meta-heurísticos

Os métodos meta-heurísticos são procedimentos não-exatos e não necessariamente encontram soluções ótimas globais. Porém, estes métodos permitem encontrar boas soluções em tempos computacionais aceitáveis tornando possível trabalhar com instâncias maiores que nos métodos exatos.

Glover e Kochenberger [19] definem como meta-heurísticos os métodos que orquestram a interação entre procedimentos de melhoria local com estratégias capazes de escapar de ótimos locais, realizando assim, uma busca robusta em um espaço de soluções.

Procedimentos de buscas locais utilizam conjuntos de movimentos admissíveis para transição entre soluções. Tal conjunto, chamado de estruturas de vizinhanças, define como a busca local deve guiar a solução inicial para soluções de melhor qualidade. Por conta disto, diversas estratégias para escapar de ótimos locais ganharam atenção nestes espaços complexos de busca, especialmente os que utilizam vizinhanças largas.

No contexto do TSPPD, Renaud et al. [9] apresentaram um algoritmo baseado na heurística de construção-melhoria. A etapa construtiva produz uma solução válida utilizando um algoritmo guloso de dupla inserção em que, repetidamente, todos os pares coleta-entrega são avaliados buscando o que pode ser inserido à rota com o menor custo. O processo termina quando todos os pares são inseridos garantindo uma solução inicial que mantém as restrições de precedência. No processo de melhoria, uma busca local utilizando a vizinhança 4-Opt** [9] precede o procedimento de deleções e reinserções de pares coleta-entrega as quais são aplicadas até que não seja mais possível a melhoria da solução. Além disto, os mesmos autores mostraram outro estudo [10] adicionando sete heurísticas de perturbação diferentes para gerar soluções ótimas ou quase-ótimas, aumentando a capacidade do algoritmo de escapar de ótimos locais. A vizinhança 4-Opt** é um subconjunto da vizinhança 4-Opt explorado em $\mathcal{O}(n^3)$ respeitando as restrição de precedência, que mostrou eficiência nos testes realizados com tempo computacional aproximadamente 2% da vizinhança 3-Opt e apenas 0.61% de aumento no custo das soluções. Veenstra et al. [7] recentemente introduziram um problema que generaliza o TSPPD e desenvolveram um algoritmo de *Large Neighborhood Search*. Em seus experimentos utilizando as instâncias RBO00, também testadas em outros trabalhos [8, 9, 10], trouxeram 53 novas melhores soluções conhecidas e, ainda, com uma redução de 0.75% em média para uma classe importante de instâncias.

2.3.1 Tipos de vizinhanças

Como definido por Papadimitriou e Steiglitz [20], um problema de otimização combinatória é definido para o par (F, c), onde F é o conjunto de todas as soluções válidas ou viáveis, e c é uma métrica de custo que atribui um valor real para cada solução $\sigma' \in F$. Dentro deste conjunto, busca-se a solução com o melhor custo.

Uma vizinhança de $\mathcal{N}(\sigma) \in F$ é definida como todas as soluções que podem ser geradas aplicando um conjunto de perturbações ou modificações locais partindo de alguma solução inicial. Explorando uma ou mais vizinhanças,

uma solução σ' com melhor custo pode ser obtida. O processo se repete até chegar a uma solução na qual nenhuma melhoria local seja possível. Esta solução é chamada de ótimo local, uma vez que o seu custo é o melhor possível dentre as perturbações locais.

Estudos relacionados ao TSP e suas variantes utilizam diversos tipos de vizinhanças. Estas são comumente divididas em dois grupos: vizinhanças com trocas de vértices e as com trocas de arestas.

Vizinhanças classificadas quanto à permutação de vértices também são comumente encontrados na literatura. Estes incluem movimentos simples como troca entre duas visitas, remoção e reinserção aleatória de vértices, ou movimentos um pouco mais complexos por heurísticas de remoções e reinserção criteriosas guiadas pelo impacto no custo de deslocamento ou outras restrições [7, 21, 22, 23].

A classificação de movimento quanto à modificação de arestas originalmente chamada k-Opt se refere à remoção de k arestas e adição de novas arestas. Estes procedimentos recebem essa nomenclatura pois a quantidade de movimentos possíveis e a verificação de otimalidade é da ordem de $\mathcal{O}(k!n^k)$.

Uma outra caracterização para movimentos baseados em trocas de arestas, foi proposto por Berge [24] introduzindo a ideia de ciclo alternativo. Em 1992, utilizando o conceito para o desenvolvimento de cadeias de ejeção, Glover [25] mostrou que a diferença simétrica entre duas rotas de TSP pode ser expressa como um conjunto de arestas desconectantes de ciclos alternativos e, uma vizinhança especial pode ser identificada.

Um ciclo alternativo(AC) que remove k arestas e adiciona k novas é chamado de k-AC. Este é considerado conectante se transforma uma rota arbitrária, σ, em um subgrafo conectado, e desconectante para o caso contrário. Abaixo seguem algumas observações estabelecidas por Glover [26]:

1) Uma k-AC transforma σ em uma nova rota e resulta em um movimento k-Opt, se e somente se, for conectante;
2) Um movimento k-Opt é composto por um conjunto de arestas desconectantes h-ACs para valores positivos de h que se somam a k. Estes componentes h-ACs podem incluir membros desconectantes;
3) Uma condição necessária e suficiente para o conjunto de arestas desconectantes ACs formar uma rota e resultar em um movimento k-Opt é que estas transformem σ em uma subgrafo conectado.

No caso de k = 2 são formados 2-ACs que podem pertencer a dois tipos distintos:

1) 2-AC conectante: remove arestas $(i, i+1)$ e $(j, j+1)$, adiciona (i, j) e $(i+1, j+1)$;
2) 2-AC desconectante: remove arestas $(i, i+1)$ e $(j, j+1)$, adiciona $(i, j+1)$ e $(i+1, j)$;

Vale ressaltar que, no caso de uma 2-AC desconectante, a possível atribuição $j = i+2$ faz com que a aresta adicionada $(i+1, j)$ seja $(i+1, i+2)$, que é uma aresta de σ. Isto viola a definição de ciclo alternativo no qual as arestas adicionadas não devem pertencer a subgrafo da rota. Porém, Glover [25] afirma que, por gerarem trocas de arestas válidas, estas trajetórias incorporadas à cadeia de ejeção contribuem com uma base para heurísticas úteis e também para teoremas associados sobre conectividade.

3
Estruturas de Vizinhanças

Neste trabalho, seis vizinhanças foram abordadas para o problema do TSPPD no espaço de busca de soluções válidas, ou seja, soluções que não violam a restrição de precedência.

As vizinhanças 2-Opt e Or-Opt são utilizadas como encontradas na literatura, porém uma nova vizinhança é introduzida e três outras vizinhanças modificadas para o problema do TSPPD. Na vizinhança Reposiciona Par uma implementação eficiente é proposta aproveitando características do TSPPD que permitem explorar a vizinhança em $\mathcal{O}(n^2)$, quando esta vizinhança é naturalmente avaliada em $\mathcal{O}(n^3)$. A vizinhança 2k-Opt, introduzida neste trabalho, busca a melhor combinação de movimentos 2-Opt aninhados em $\mathcal{O}(n^2)$ por programação dinâmica. Adaptações das vizinhanças 4-Opt e Balas & Simonetti, encontradas na literatura apenas para o TSP, habilitam um grande conjunto de movimentos e são avaliadas em $\mathcal{O}(n^2)$ e $\mathcal{O}(k^2 2^{k-2} n)$ respectivamente como discutido durante esse capítulo. Estas vizinhanças são divididas em dois grupos:

Estrutura de vizinhança $\mathbb{N}_1$:

- $\mathcal{N}_1$ - Reposiciona Par: os movimentos desta vizinhança reposicionam o par *p–d* na melhor posição possível. Para esta vizinhança é proposta uma maneira eficiente de buscar a melhor posição para o par em $\mathcal{O}(n)$, considerando a restrição de precedência.
- $\mathcal{N}_2$ - 2-Opt: este grupo de movimentos consiste em remover duas arestas existentes na solução e adicionar duas novas arestas.
- $\mathcal{N}_3$ - Or-Opt: consiste em movimentos que reposicionam uma sequência de visitas de tamanho máximo k na melhor posição possível. Para essa vizinhança a avaliação dos movimentos é computada em $\mathcal{O}(kn^2)$.

Estrutura de vizinhança $\mathbb{N}_2$:

- $\mathcal{N}_4$ - 2k-Opt: esta vizinhança é introduzida neste trabalho pela aplicação de múltiplos movimentos 2-Opt aninhados. A avaliação da melhor combinação de movimentos 2-Opt é obtida em $\mathcal{O}(n^2)$, considerando as restrições de precedência.

- $\mathcal{N}_5$ - 4-Opt: nesta vizinhança é proposta uma avaliação do subconjunto de movimentos 4-Opt chamados de ponte dupla e ponte torcida. A avaliação das regras de precedência propostas neste trabalho permitem explorar estes movimentos em $\mathcal{O}(n^2)$.
- $\mathcal{N}_6$ - B&S: a vizinhança Balas&Simonetti (B&S) encontra um ótimo local para uma solução inicial tal que, após o movimento, todos os vértices são reposicionados otimamente a, no máximo, k posições da posição inicial. A busca nessa vizinhança larga é feita em $\mathcal{O}(k^2 2^{k-2} n)$.

A divisão das vizinhanças em duas estruturas distintas é dada pela natureza da busca dessas vizinhanças. Embora as vizinhanças tenham complexidade de no máximo $\mathcal{O}(n^2)$, vizinhanças com buscas divisíveis são agrupadas na estrutura $\mathbb{N}_1$ enquanto vizinhanças largas com buscas implementadas utilizando programação dinâmica são agrupadas na vizinhança $\mathbb{N}_2$.

A seguir cada vizinhança é discutida quanto aos movimentos executados, complexidade computacional e pré-processamentos necessários para uma implementação eficiente.

Pré-processamentos

Durante a avaliação de movimentos, algumas vizinhanças precisam computar previamente informações sobre a rota para reduzir o processamento necessário. Basicamente duas informações são utilizadas constantemente nas avaliações de movimentos. São estas:

Informação de posições. Constrói informações sobre a ordem de visitas na rota com o propósito de informar em tempo $\mathcal{O}(1)$ em qual posição um vértice $v \in V$ está na sequência de visitas σ.

$$\sigma_v^{-1} = \text{posição do vértice } v \text{ em } \sigma. \tag{3-1}$$

Informação de dependências. Constrói informações sobre relações de dependência entre trechos da rota com o propósito de responder em tempo $\mathcal{O}(1)$ se existe relação de precedência entre trechos distintos da rota. Sejam $r_1 = \sigma_{i,j}$ e $r_2 = \sigma_{i',j'}$ dois trechos de σ, é dito que r_2 depende de r_1 se a relação $Dep(r_1, r_2)$ é satisfeita, como na Equação(3-2):

$$Dep(r_1, r_2) = \begin{cases} verdadeiro, & \exists(x \in r_1,\ y \in r_2) : x = y + n \\ falso, & \text{caso contrário,} \end{cases} \tag{3-2}$$

onde desta relação de dependência é possível formalizar também se a rota r é reversível:

$$Rev(r) = não\ Dep(r, r). \tag{3-3}$$

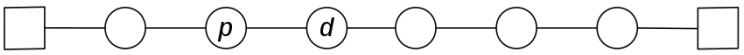
No entanto apenas algumas informações de precedência são necessárias durante a avaliação de algumas vizinhanças. Em especial, para cada trecho $\sigma_{i,j}$ é desejável obter eficientemente a maior e a menor posição σ_v^{-1} tal que, $\forall v \in \sigma_{0,i-1}, v \in P : v + n \in \sigma_{i,j}$. Outra informação relevante é se o trecho $\sigma_{i,j}$ é reversível, o que pode ser respondido pela operador $Rev(\sigma_{i,j})$ definido na Equação (3-3), uma vez que a existência de um par p–d dentro do trecho provoca violação de precedência ao inverter a ordem de visitas. Estas relações específicas podem ser avaliadas em $\mathcal{O}(n^2)$ como mostrado no Algoritmo 1.

Algoritmo 1: Informações de dependência

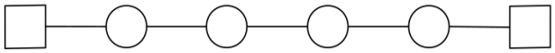
```
1  Para i = 0 até 2n + 1
2      Rev ← verdadeiro
3      Para j = i to 2n + 1
4          Se σ_j ∈ D
5              p ← σ^{-1}_{(σ_j − n)}
6              Se i ≤ p ≤ j
7                  Rev ← falso
8              senão
9                  p_min ← min(p_min, p)
10                 p_max ← max(p_max, p)
11         pred_info[i, j] = (p_min, p_max, Rev)
```

3.1 Vizinhança Reposiciona Par

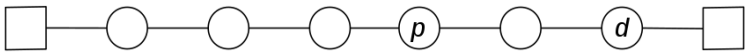
A vizinhança Reposiciona Par, a partir de uma solução σ, constrói uma nova solução σ' na qual um par de vértices da forma $(x, n+x)$ é reposicionado. Em outras palavras, seja um dado par coleta-entrega localizado nas posições i e j respectivamente, novas posições i' e j' são encontradas, onde $i'<j'$ mantendo as relações de precedência. Na Figura 3.1 é mostrado o mecanismo de construção da nova solução σ'.



3.1(a): Solução σ original.



3.1(b): Solução σ' copiada sem o par p–d.



3.1(c): Solução σ' após a reinserção do par p–d.

Figura 3.1: Passos realizados pelo movimento Reposiciona Par. (a) A solução original; (b) A solução copiada porém excluindo o par coleta-entrega avaliado; (c) O par coleta-entrega já reinserido.

A nova solução gerada pela remoção e reinserção é uma melhoria na solução inicial se a Equação (3-4) é satisfeita.

$$\Delta^{rem}(i,j) + \Delta^{add}(i',j') < 0, \tag{3-4}$$

onde $\Delta^{rem}(i,j)$ é o impacto no custo causado pela remoção do par $(x, n+x)$ das posições i e j, computado pela Equação (3-5), e $\Delta^{add}(i',j')$ é o custo de reinserção na posição i' e j' como na Equação(3-6).

$$\Delta^{rem}(i,j) = \begin{cases} c_{\sigma_{i-1},\sigma_{j+1}} - c_{\sigma_i,\sigma_j} - c_{\sigma_{i-1},\sigma_i} - c_{\sigma_j,\sigma_{j+1}}, & se\ j = i+1 \\ c_{\sigma_{i-1},\sigma_{i+1}} - c_{\sigma_{i-1},\sigma_i} - c_{\sigma_i,\sigma_{i+1}} + & \\ \quad + c_{\sigma_{j-1},\sigma_{j+1}} - c_{\sigma_{j-1},\sigma_j} - c_{\sigma_j,\sigma_{j+1}}, & se\ j > i+1 \end{cases} \tag{3-5}$$

$$\Delta^{add}(i',j') = \begin{cases} c_{x,n+x} + c_{\sigma_{i'-1},x} + c_{n+x,\sigma_{i'}} - c_{\sigma_{i'-1},\sigma_{i'}}, & se\ j' = i'+1 \\ c_{\sigma_{i'-1},x} + c_{x,\sigma_{i'}} - c_{\sigma_{i'-1},\sigma_{i'}} + & \\ \quad + c_{\sigma_{j'-1},n+x} + c_{n+x,\sigma_{j'}} - c_{\sigma_{j'-1},\sigma_{j'}}, & se\ j' > i+1 \end{cases} \tag{3-6}$$

O melhor posicionamento para um par $(x, n+x)$ pode ser obtido após a remoção tendo em mente algumas observações:

1) Se i' e j' são respectivamente as posições de x, $n+x$ após a inserção, então $i' < j'$.

2) A avaliação da inserção de menor custo para i' e j' consecutivos pode

ser avaliada trivialmente em $\mathcal{O}(n)$. Sendo para esse caso $j' = i' + 1$ considerando:

$$i' = \underset{i' \in [1, 2n-2]}{\arg\min}\ \Delta^{add}(i', i'+1). \tag{3-7}$$

3) Seja $Z^{add}(v, k)$ o custo para a inserção individual do vértice v na posição k calculado pela Equação (3-8). Para inserções não consecutivas, sendo i' uma posição arbitrária para reinserção de x, a entrega $n + x$ associada deve ser posicionada em $j' \in [i' + 2, 2n - 2]$ de forma a minimizar o custo. A Equação (3-6) para $j' > i' + 1$ é reescrita na Equação (3-9).

$$Z^{add}(v, k) = c_{\sigma_{k-1}, v} + c_{v, \sigma_k} - c_{\sigma_{k-1}, \sigma_k}. \tag{3-8}$$

$$\Delta^{add}(i, j) = Z^{add}(x, i) + Z^{add}(n + x, j). \tag{3-9}$$

Como já afirmado, para c em qualquer i' a entrega não consecutiva deve estar inserida no intervalo $j' \in [i' + 2, 2n - 2]$. Obtém-se da Equação (3-9) que o melhor custo de inserção para i' arbitrário pode ser escrito como

$$\Delta^{add}_{min}(i') = Z^{add}(x, i') + \left[\min_{j' \in [i'+2, 2n-2]} Z^{add}(n + x, j') \right]. \tag{3-10}$$

Assim, a melhor posição para a entrega é definida por:

$$j'_{min}(i') = \underset{j' \in [i'+2, 2n-2]}{\arg\min}\ Z^{add}(e, j') \tag{3-11}$$

Realizando a busca de forma descendente, logo após avaliar $\Delta^{add}_{min}(i')$, faz-se $i' - 1$ e tem-se a partir da Equação (3-10):

$$\Delta^{add}_{best}(i' - 1) = Z^{add}(x, i' - 1) + \left[\min_{j' \in [i'+1, 2n-2]} Z^{add}(n + x, j') \right], \tag{3-12}$$

ou ainda,

$$\begin{aligned} &\Delta^{add}_{best}(i' - 1) = Z^{add}(x, i' - 1) + \\ &+ \min \left\{ Z^{add}(n + x, i' + 1), \min_{j' \in [i'+2, 2n-2]} Z^{add}(n + x, j') \right\}, \end{aligned} \tag{3-13}$$

onde o termo $\min_{j' \in [i'+2, 2n-2]} Z^{add}(n + x, j')$ é valor já calculado para j'_{min} da

iteração anterior. De tal forma, o melhor custo e posição para a reinserção do vértice x para $i'-1$ é calculado em tempo $\mathcal{O}(1)$ visto que apenas é necessária a comparação de $Z^{add}(n+x, i'+1)$ com $Z^{add}(n+x, j'_{min})$. Avaliando as configurações para as n requisições pode-se obter a melhor posição em $\mathcal{O}(n)$.

No Algoritmo 2 é mostrado o pseudo-código para localizar as posições ótimas de reinserção dos vértices x e $n+xode$.

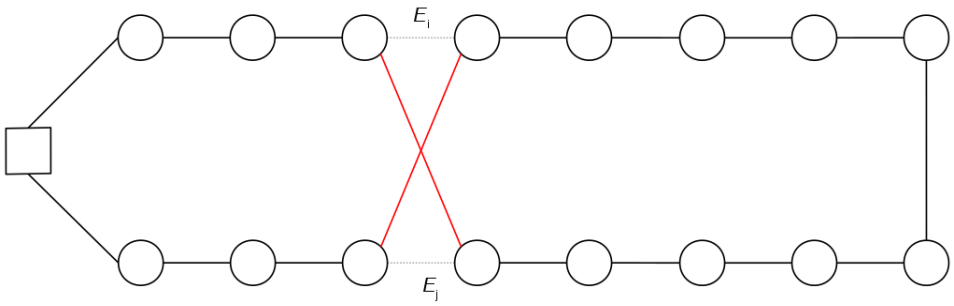
Algoritmo 2: Reposiciona Par

```
1  Δ^rem ← Custo de remoção do par (x, n + x)
2  Para k = 2n − 2 até k = 2
3      z^x ← custo para inserir x em k − 1
4      z^{n+x} ← custo para inserir n + x em k
5      Se z^{n+x} < z^{n+x}_min
6          Atualiza melhor entrega
7          z^{n+x}_min ← z^{n+x}
8          j'_min ← k
9      z ← z^x + z^{n+x}_min
10     Se z < z_min
11         Atualiza melhor movimento
12         z_min ← z
13         i' ← k − 1
14         j' ← j'_min
15 Para k = 2n − 2 até k = 1
16     z ← custo para inserir (x, n + x) consecutivamente em k
17     Se z < z_min
18         Atualiza melhor movimento
19         z_min ← z
20         i' ← k
21         j' ← k
22 return z_min + Δ^rem
```

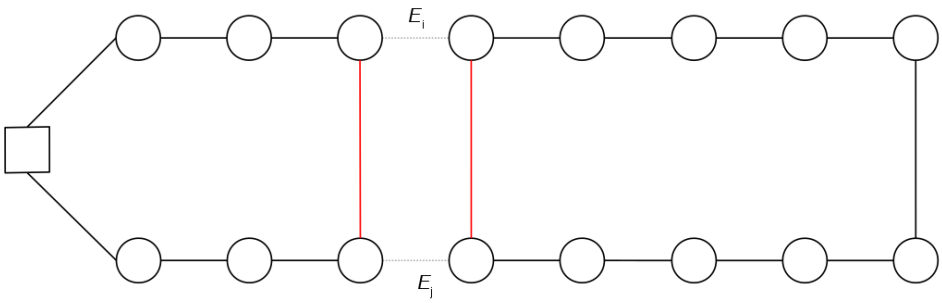
3.2 Vizinhança 2-Opt

A vizinhança 2-Opt considera todos os movimentos de substituição de duas arestas por duas outras novas. Em termos gerais, sendo σ uma solução e σ_k o k-ésimo cliente a ser visitado, as arestas (σ_i,σ_{i+1}) e (σ_j,σ_{j+1}) são substituídas pelas arestas (σ_i,σ_j) e $(\sigma_{i+1},\sigma_{j+1})$, como pode ser visto na Figura 3.2(a). A

adição das arestas (σ_i, σ_{j+1}) e (σ_{i+1}, σ_j) resultaria em uma divisão da rota, produzindo uma solução inválida, conforme mostrado na Figura 3.2(b).



3.2(a): 2-Opt conectante



3.2(b): 2-Opt desconectante

Figura 3.2: Possíveis novas arestas a serem adicionadas após remoção de E_i e E_j. (a) criação das arestas (σ_i,σ_j) e $(\sigma_{i+1}, \sigma_{j+1})$ resultando em um movimento válido. (b) rota inválida desconectada pela criação das arestas (σ_i, σ_{j+1}) e (σ_{i+1}, σ_j).

A troca de arestas 2-Opt resulta em uma melhoria local quando o custo $Z^{2opt}(i, j)$ definido pela Equação (3-14) é menor do que 0.

$$Z^{2opt}(i, j) = c_{\sigma_i,\sigma_j} + c_{\sigma_{i+1},\sigma_{j+1}} - c_{\sigma_i,\sigma_{i+1}} - c_{\sigma_j,\sigma_{j+1}}. \tag{3-14}$$

A sequência de visitas da sub-rota $\sigma_{i+1,j}$ é invertida na rota após a modificação, sendo então a existência de uma par p–d no intervalo $\sigma_{i+1,j}$ fator determinante na factibilidade do movimento uma vez que a relação de precedência é violada após a inversão do trecho (Figura 3.3). Embora a quantidade total de movimentos 2-Opt possíveis é igual ao número de combinações possíveis de duas arestas, implicando em uma complexidade $\mathcal{O}(n^2)$, em termos práticos o número de movimentos válidos diminui consideravelmente com a restrição de precedência imposta no problema de coleta e entrega.

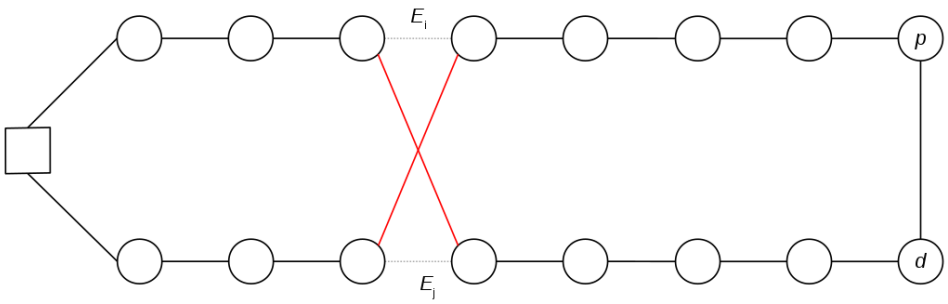


Figura 3.3: Movimento 2-Opt inválido. Após a troca das arestas pontilhadas pelas arestas em vermelho, a regra de precedência entre p–d é violada.

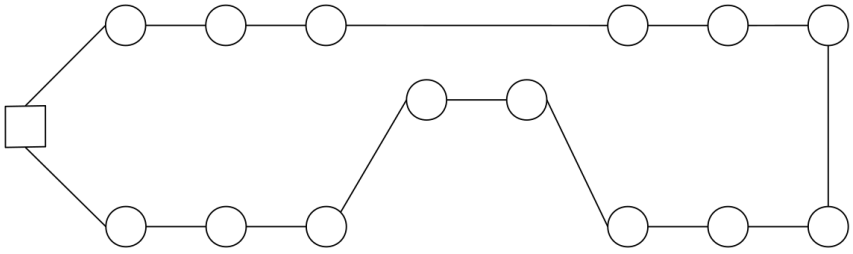
Avaliação de movimentos. No algoritmo proposto, esta vizinhança é utilizada na busca rápida. Para um dado par coleta-entrega da forma $(x, n+x)$, apenas os movimentos nos intervalos $i = \sigma_x^{-1}$ com $j \in \{i+2, \ ..., \ \sigma_{n+x}^{-1}\}$ e $i = \sigma_{n+x}^{-1}$ com $j \in \{i+2, \ ..., \ 2n\}$ são avaliados. Este subconjunto de movimentos representa o conjunto de movimentos 2-Opt iniciados na posição de coleta e entrega. A busca é realizada em tempo $\mathcal{O}(n)$ como pode ser analisado no Algoritmo 3. Além disto, o conjunto de todos os movimentos 2-Opt pode ser avaliado chamando a mesma rotina para todos os pares coleta-entrega da instância avaliada.

Algoritmo 3: 2-Opt

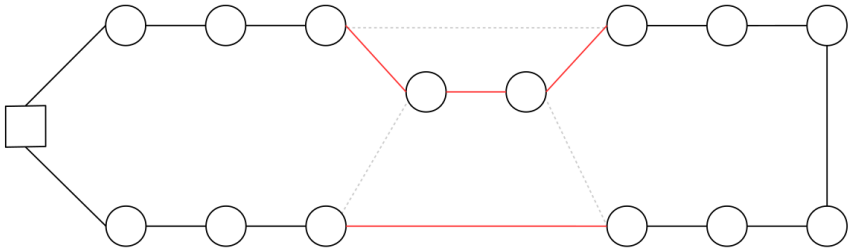
```
1  i ← σ_x^{-1}
2  Para j = i + 2 até j = σ_{(n+x)}^{-1}
3      Se σ_j ∈ D e σ_{(σ_j − n)}^{-1} ∈ {i, ..., j − 1}
4          interrompe loop
5      z ← c_{σ_i,σ_j} + c_{σ_{i+1},σ_{j+1}} − c_{σ_i,σ_{i+1}} − c_{σ_j,σ_{j+1}}
6      Se z < z_min
7          z_min ← z
8          i_min ← i
9          j_min ← j
10 i ← σ_{(n+x)}^{-1}
11 Para j = i + 2 até j = 2n + 1
12     Se σ_j ∈ D e σ_{(σ_j − n)}^{-1} ∈ {i, ..., j − 1}
13         interrompe loop
14     z ← c_{σ_i,σ_j} + c_{σ_{i+1},σ_{j+1}} − c_{σ_i,σ_{i+1}} − c_{σ_j,σ_{j+1}}
15     Se z < z_min
16         z_min ← z
17         i_min ← i
18         j_min ← j
19 return z_min; i_min; j_min
```

3.3 Vizinhança Or-Opt

A vizinhança Or-Opt [27] é um subconjunto de movimentos 3-Opt, onde três arestas são substituídas como mostrado na Figura 3.4. Neste grupo de movimentos uma sequência de vértices, com tamanho máximo k, é realocada na rota. O tempo computacional para enumerar os movimentos Or-Opt é $\mathcal{O}(kn^2)$ uma vez que entre as três arestas a serem substituídas, existe a restrição de distância k entre duas das arestas removidas, o número de combinações possíveis é igual a n^2 expressando crescimento de tempo computacional menor que $\mathcal{O}(n^3)$, da vizinhança 3-Opt.



3.4(a): Rota original



3.4(b): Rota após Or-Opt

Figura 3.4: Modificação da rota após o movimento Or-Opt. Uma cadeia de dois vértices é reposicionada na rota.

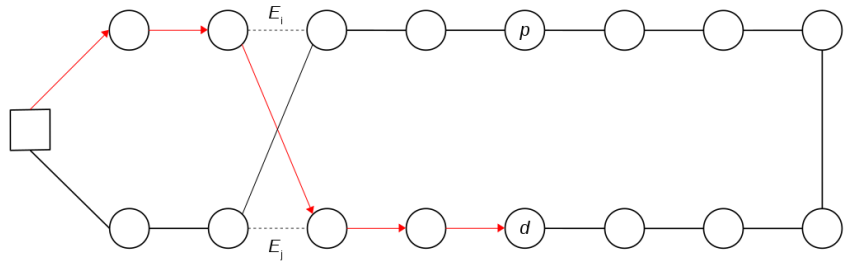
Avaliação de movimentos. Sendo k um parâmetro fixo nesta vizinhança e $w \in \{1, 2, ..., k\}$, a busca nessa vizinhança é realizada de forma trivial onde para um dado par p–d, posicionado em i e j respectivamente, as sequências de vértices $\sigma_{i,i+w}$ e $\sigma_{j,j+w}$ são avaliadas quanto ao custo de remoção e reinserção em $\mathcal{O}(n)$.

3.4 Vizinhança 2k-Opt

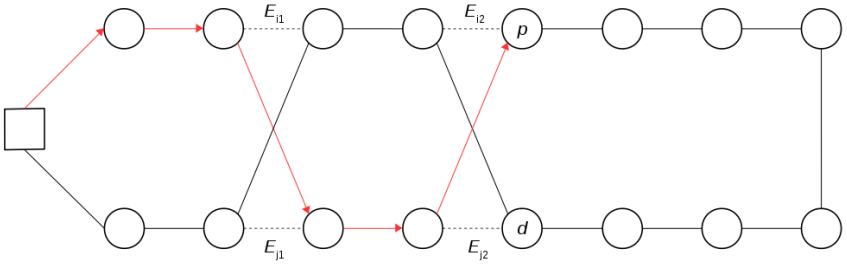
Como visto na Seção 3.2, a vizinhança 2-Opt consiste em movimentos de remoção de duas arestas existentes na rota e de adição de outras duas mantendo os vértices conectados. Estes movimentos, embora muito utilizados no TSP, se tornam muito restritos quando aplicados ao cenário do TSPPD,

no qual a regra de precedência e apenas movimentos válidos são considerados. Ainda neste contexto, é observável que apenas um par coleta-entrega pode fazer com que uma grande quantidade de movimentos não sejam considerados.

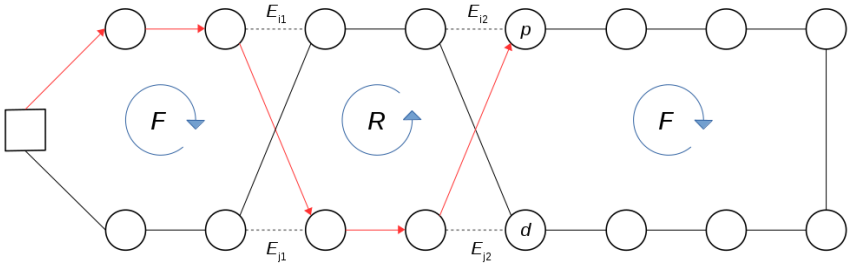
A Figura 3.5 mostra como a vizinhança 2k-Opt permite que, mesmo com a inversão de um trecho da rota, a região onde ocorre a violação da precedência possa se tornar factível se um novo movimento 2-Opt corrigir as regras de precedências violadas. Como ilustrado na Figura 3.5(c), o sentido da sub-rota sofre reversão a cada movimento aplicado sendo chamado de F quando o trecho está no sentido normal e R quando está no sentido reverso em relação a sequência de visitas da solução original.



3.5(a): 2-Opt inválido.



3.5(b): 2k-Opt válido.



3.5(c): Reversões na sequência de visitas.

Figura 3.5: O caminho percorrido na solução após movimento 2-Opt e 2k-Opt. (a) Violação de precedência após movimento 2-Opt. (b) O reparo da precedência violada após um novo 2-Opt aninhado. (c) Modificações no sentido de visitas da sub-rota mais interna a cada movimento.

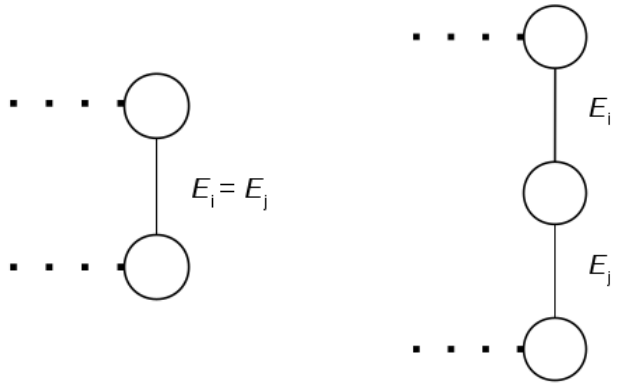
Sendo $Z^{2opt}(i,j)$ o custo do movimento 2-Opt exatamente como na Seção 3.2, "F" a formulação de recorrência para combinar movimentos quando a sub-rota $\sigma_{i,j}$ está no sentido original e "R" quando o sentido da sub-rota mais interna ao último movimento estiver invertido, as expressões podem ser escritas como:

$$F(i,j) = \begin{cases} 0, & se\ j < i+2 \\ \min\left\{ \begin{array}{l} F(i+1,j), \\ F(i,j-1) \end{array} \right\}, & \begin{array}{l} se\ \sigma_{i+1}+n \in \sigma_{i+1,j} \\ ou\ \sigma_j - n \in \sigma_{i+1,j} \end{array} \\ \min\left\{ \begin{array}{l} F(i+1,j), \\ F(i,j-1), \\ R(i+1,j-1)+ \\ \quad +Z^{2opt}(i,j) \end{array} \right\}, & caso\ contrário \end{cases} \tag{3-15}$$

$$R(i,j) = \begin{cases} 0, & \begin{array}{l} se\ j < i+2\ e \\ Rev(\sigma_{i,j+1}) \end{array} \\ \infty, & \begin{array}{l} se\ j < i+2\ e\ não \\ Rev(\sigma_{i,j+1}) \end{array} \\ F(i+1,j-1)+ \\ \quad +Z^{2opt}(i,j) \end{array}, & \begin{array}{l} se\ \sigma_{i+1}+n \in \sigma_{i+1,j} \\ e\ \sigma_j - n \in \sigma_{i+1,j} \end{array} \\ \min\left\{ \begin{array}{l} R(i+1,j), \\ F(i+1,j-1)+ \\ \quad +Z^{2opt}(i,j) \end{array} \right\}, & se\ \sigma_j - n \in \sigma_{i+1,j} \\ \min\left\{ \begin{array}{l} R(i,j-1), \\ F(i+1,j-1)+ \\ \quad +Z^{2opt}(i,j) \end{array} \right\}, & se\ \sigma_{i+1}+n \in \sigma_{i+1,j} \\ \min\left\{ \begin{array}{l} R(i+1,j), \\ R(i,j-1), \\ F(i+1,j-1)+ \\ \quad +Z^{2opt}(i,j) \end{array} \right\}, & caso\ contrário \end{cases} \tag{3-16}$$

Supondo inicialmente $i = 0$ e $j = n - 1$, ou seja, a primeira e a última arestas respectivamente, a avaliação de $F(i, j)$ consiste em definir a melhor opção entre: não aplicar o movimento 2-Opt(i, j), escolhendo o menor custo entre a avaliação $F(i + 1, j)$ e $F(i, j - 1)$, ou aplicar o movimento 2-Opt(i, j), somando o custo $Z^{2opt}(i, j)$ desse movimento à avaliação de $R(i + 1, j - 1)$.

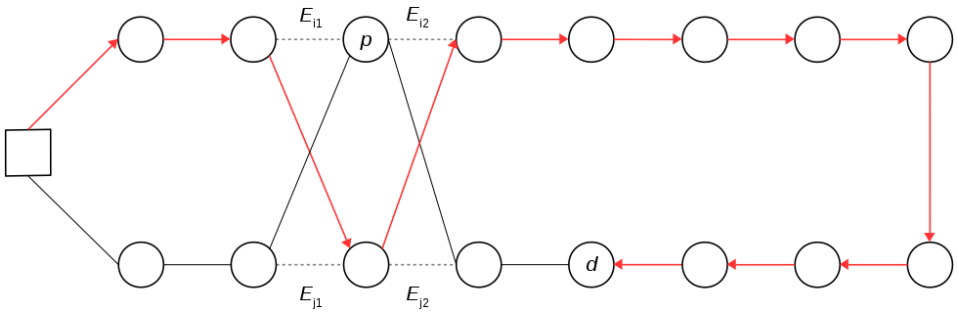
Os casos bases das funções F e R se dão quando $j = i$ ou $j = i + 1$ onde não é possível remover duas arestas e criar duas novas, conforme mostrado na Figura 3.6. No trecho invertido da rota, quando a busca atinge estas condições, o custo para estas avaliações é $R(i, j) = \infty$ se existir pelo menos um par coleta-entrega na sub-rota $\sigma_{i,j+1}$. Caso contrário, o custo é de $R(i, j) = 0$, uma vez que nenhuma mudança é realizada. No caso de alcançar estes cenários no trecho sem inversão de sentido, o custo é de $F(i, j) = 0$, uma vez que nenhuma modificação é possível e não há possibilidade de violar precedência, assumindo uma rota inicial factível.



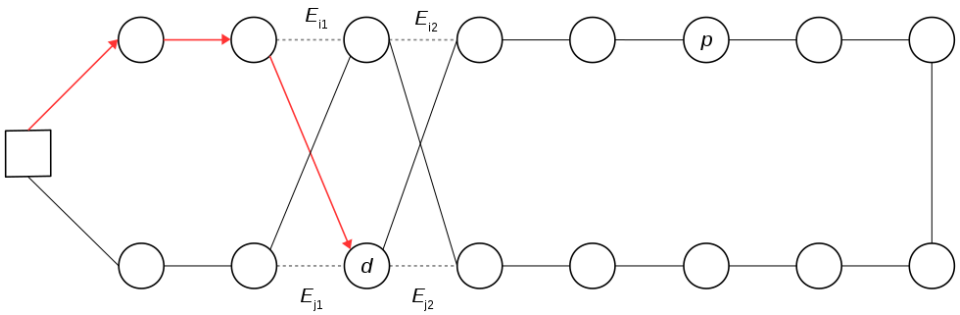
3.6(a): $j = i$ 3.6(b): $j = i + 1$

Figura 3.6: Ponto de parada da expansão da busca 2k-Opt. (a) Sendo $i = j$, não é possível realizar 2-Opt; (b) Para $j = i + 1$, o movimento 2-Opt não resulta em modificação na sequência de visitas.

Ao expandir a busca no sentido original, $F(i + 1, j - 1) + Z^{2opt}(i, j)$ só é válido se os pares das requisições σ_{i+1} e σ_j não se encontram no intervalo $\sigma_{i+1,j}$. Isto ocorre devido a, após um movimento 2-Opt, nenhum outro 2-Opt aninhado ser capaz de modificar a ordem de visita para σ_{i+1} e σ_j, provocando uma violação de precedência como mostrado na Figura 3.7. Analogamente, a busca no sentido invertido causa violação de precedência ao avançar $R(i+1, j)$, se o par de σ_{i+1} está posicionado no intervalo $\sigma_{i+1,j}$, ou ao avançar $R(i, j - 1)$, se o par de σ_j está contido em $\sigma_{i+1,j}$ (Figura 3.8).

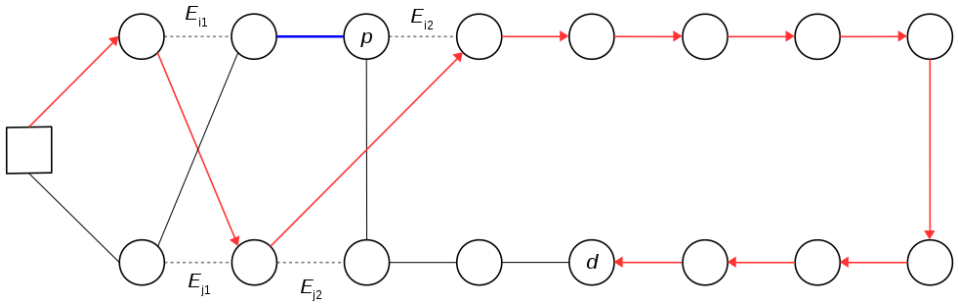


3.7(a): σ_{i+1} fixo, não sendo mais possível corrigir precedência.

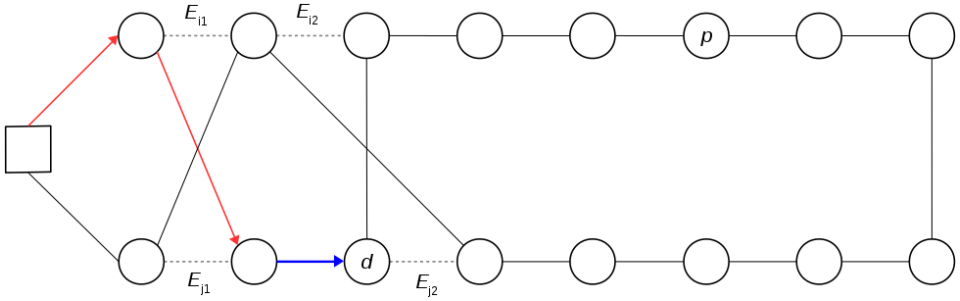


3.7(b): σ_j fixo, não sendo mais possível corrigir precedência.

Figura 3.7: Casos especiais em que um movimento 2-Opt (E_{i1},E_{j1}) provoca violação de precedência não sendo possível se tornar novamente factível.



3.8(a): σ_i mantido, não sendo mais possível corrigir precedência.



3.8(b): σ_{j+1} inalterado, não sendo mais possível corrigir precedência.

Figura 3.8: Casos especiais em que avançar uma aresta sem realizar um movimento 2-Opt provoca violação de precedências.

Avaliação de movimentos. O processo de avaliação utiliza uma matriz de memória $n \times n$ onde cada elemento armazena valores de custo para os sentidos F e R e os índices do próximo movimento 2-Opt executado. O Algoritmo 4 nos permite eliminar expansões redundantes e recuperar facilmente a melhor solução encontrada. A complexidade computacional do pré-processamento de posições é $\mathcal{O}(n)$ e, na inicialização de casos base, apenas os elementos $j = i$ e $j = i + 1$ necessitam ser avaliados tendo complexidade também $\mathcal{O}(n)$. De tal forma, a complexidade geral do algoritmo sobe para $\mathcal{O}(n^2)$ pela presença das duas repetições aninhadas e limitadas por $2n + 1$ nas linhas 1 e 2.

Algoritmo 4: 2k-Opt construtivo

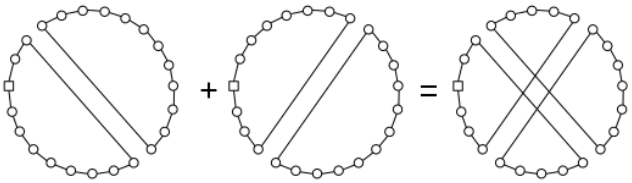
```
1  Para k = 2 to k < 2n + 1
2      Para i = 0 to i < 2n − k + 1
3          j ← i + k
4          violaParI ← σ_{i+1} ∈ P e i < σ^{-1}_{(σ_{i+1}+n)} ≤ j
5          violaParJ ← σ_j ∈ D e i < σ^{-1}_{(σ_j−n)} ≤ j
6          Se violaParI ou violaParJ
7              Se violaParI
8                  a ← ∞
9              senão
10                 a ← mem[i + 1, j].R
11             Se violaParJ
12                 b ← ∞
13             senão
14                 b ← mem[i, j − 1].R
15             mem[i, j].F ← min(mem[i + 1, j].F, mem[i, j − 1].F)
16             mem[i, j].R ← min(mem[i + 1, j − 1].F + Z^{2opt}(i, j), a, b)
17         senão
18             mem[i, j].F ←
                 min(mem[i+1, j].F, mem[i, j−1].F, mem[i+1, j−1].R+Z^{2opt}(i, j))
19             mem[i, j].R ←
                 min(mem[i+1, j].R, mem[i, j−1].R, mem[i+1, j−1].F+Z^{2opt}(i, j))
20 return mem[0, n − 1].F
```

3.5 Vizinhança 4-Opt

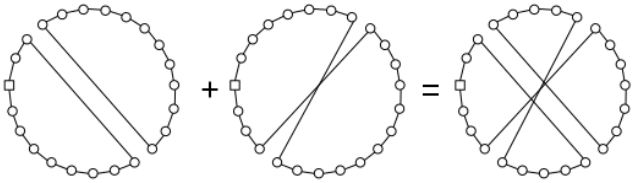
As vizinhanças 4-Opt são poderosas quanto à qualidade da solução mas não são frequentemente adotadas pelo seu elevado custo computacional. Assim, formas de reduzir o tempo computacional e restringir a busca a um subconjunto da vizinhança se fazem necessárias [9, 10, 26]. Uma classe de movimentos de 4-Opt que recebeu atenção especial na literatura TSP é a aplicação conjunta de dois 2-ACs desconectantes que se cruzam de forma a reconectar os componentes. Estes movimentos são os chamados de "super movimentos" ou "ponte dupla". Este e mais outros cinco tipos de movimentos, que também são gerados combinando 2-ACs conectantes e desconectantes, estão descritos na Figura 3.9.

Entre os tipos de movimentos construídos por dois 2-ACs, listados na Figura 3.9, apenas três produzem rotas completamente conectadas. O movimento Tipo 1 (Figura 3.9(a)), ou "ponte dupla", é obtido pela junção de dois 2-ACs desconectantes que se cruzam, reconectando os componentes desconectados. No movimento Tipo 2, a composição de 2-AC conectante com 2-AC desconectante ou 2-AC desconectante com 2-AC conectante, obtém-se uma solução completamente conectada e dois sub-trechos da rota têm as ordens de suas visitas invertidas. Já, o Tipo 6 consiste de dois 2-ACs conectantes que não se cruzam. Este movimento pode ser visto como o desdobramento em dois 2-Opt ou como um subconjunto de movimentos da vizinhança 2k-Opt mostrado na Seção 3.4. Os movimentos de Tipo 3, 4 e 5 mostrados nas Figuras 3.9(c), 3.9(d) e 3.9(e) dividem a rota em componentes desconectadas, o que não é uma solução factível para o TSP.

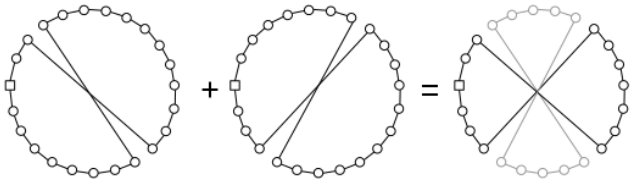
Para realizar uma busca eficiente em uma vizinhança 4-Opt para TSPPD, o procedimento de avaliação concentra a busca especialmente em movimentos do Tipo 1 e 2. A adição de testes de precedência em $\mathcal{O}(1)$ ao algoritmo permite buscar apenas movimentos que não violam a regra de precedência na vizinhança larga, além de manter a complexidade computacional original de $\mathcal{O}(n^2)$.



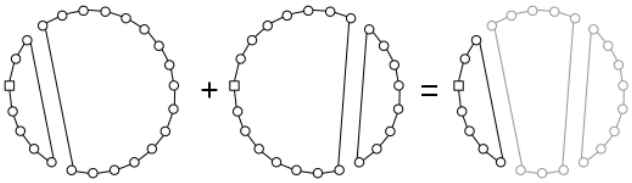
3.9(a): Tipo 1: Dois 2-ACs desconectantes que se cruzam.



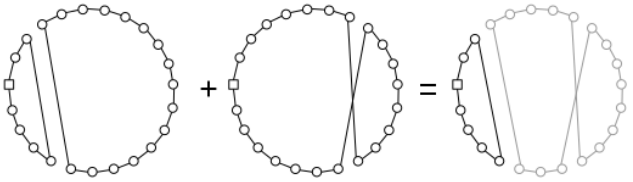
3.9(b): Tipo 2: 2-AC desconectante e 2-AC conectante que se cruzam.



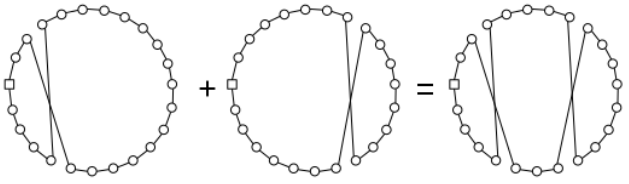
3.9(c): Tipo 3: Dois 2-ACs conectantes que se cruzam.



3.9(d): Tipo 4: Dois 2-ACs desconectantes que não se cruzam.



3.9(e): Tipo 5: 2-AC desconectante e 2-AC conectante que não se cruzam.



3.9(f): Tipo 6: Dois 2-ACs conectantes que não se cruzam.

Figura 3.9: Tipos de movimentos 4-Opt da classe de movimentos obtidos por combinações de dois 2-ACs . Em cinza, pedaços da rota original inatingíveis a partir do depósito.

Na avaliação em $\mathcal{O}(1)$ das relações de precedências, observa-se que os movimentos Tipo 1 e 2, assim como 4-Opt de forma geral, subdividem a rota em cinco sub-rotas e as reposicionam em uma diferente configuração. Sejam as arestas removidas E_{i_1}, E_{i_2}, E_{j_1} e E_{j_2}, tal que, $i_1 < i_2 < j_1 < j_2$, a solução σ é dividida nas sub-rotas 1, 2, 3, 4 e 5 definidas pelos intervalos σ_{0,i_1}, σ_{i_1+1,i_2}, σ_{i_2+1,j_1}, σ_{j_1+1,j_2} e $\sigma_{j_2+1,n}$, como é mostrado na Figura 3.10:

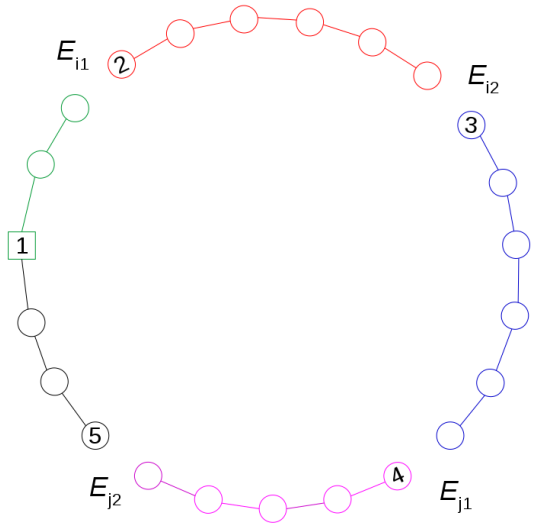
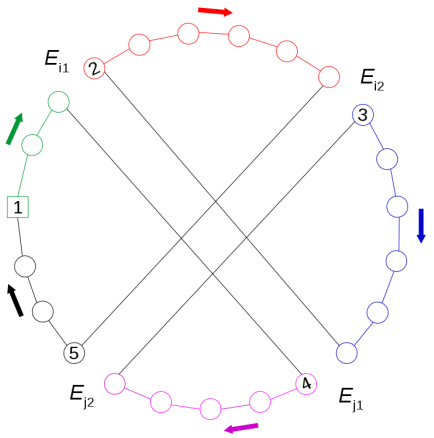
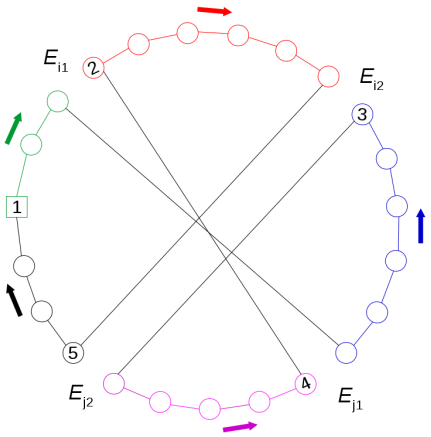


Figura 3.10: Subdivisão da rota em cinco fragmentos após remoção de quatro arestas. As sub-rotas 1, 2, 3, 4 e 5 são iniciadas em 0, i_1+1, i_2+1, j_1+1 *e* j_2+1, respectivamente.

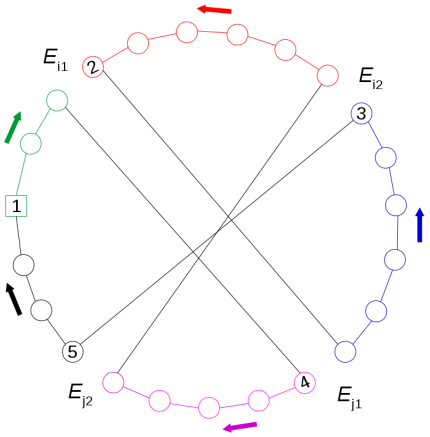
Um movimento somente é factível se todas as sub-rotas têm as suas restrições de precedência mantidas após o movimento. A configuração de visitas obtida pelos movimento de Tipo 1 é a concatenação de sub-rotas na sequência {1, 4, 3, 2, 5}, Figura 3.11(a). Analisando os movimentos de Tipo 2 como reorganização de sub-rotas: os movimentos com 2-AC conectante + 2-AC desconectante produzem visitas como a concatenação na ordem {1, 3R, 4R, 2, 5} enquanto, para 2-AC desconectante + 2-AC conectante, são concatenadas como {1, 4, 2R, 3R, 5}, Figuras 3.11(b) e 3.11(c). O sufixo R é um indicador de reversão da sub-rota. Para os três casos, as sub-rotas 2, 3 e 4 podem ter as relações de precedência violadas. Embora o conjunto de movimentos do Tipo 2 seja duas vezes a quantidade de movimentos de Tipo 1, os seus movimentos são mais suscetíveis a violação de regras de precedência uma vez que ocorrem reversões de sub-rotas.



3.11(a): Tipo 1. Reorganização de visitas para dois 2-ACs desconectante.



3.11(b): Tipo 2.1: Reorganização de visitas para 2-AC conectante + 2-AC desconectante.



3.11(c): Tipo 2.2: Reorganização de visitas para 2-AC desconectante + 2-AC conectante.

Figura 3.11: Reorganização de sub-rotas nos movimentos Tipo 1 e Tipo 2.

Regras de precedência para movimentos do Tipo 1, dois 2-AC desconectantes:

- Sub-rota 1 e 5: primeira e última sub-rotas, respectivamente, sempre factíveis por não terem as relações de precedências modificadas.
- Sub-rota 4: a segunda sub-rota na nova configuração não deve conter entregas tais que as respectivas coletas não estejam contidas na própria sub-rota ou na sub-rota 1.
- Sub-rota 3: a terceira sub-rota na nova configuração não devem conter entregas tais que as respectivas coletas não estejam contidas na própria sub-rota ou na sub-rota 1.
- Sub-rota 2: sempre factível, sendo originalmente precedida apenas pela sub-rota 1, já visitada.

Regras de precedência para movimentos do Tipo 2, 2-AC conectante e 2-AC desconectante:

- Sub-rota 1 e 5: primeira e última sub-rotas, respectivamente, sempre factíveis por não terem as relações de precedências modificadas.
- Sub-rota 3 e 4: a segunda e terceira sub-rotas na nova configuração não deve conter entregas tais que as respectivas coletas não estejam contidas na sub-rota 1. Individualmente não devem conter nenhum par coleta-entrega (ser reversível).
- Sub-rota 2: sempre factível, sendo originalmente precedida apenas pela sub-rota 1, já visitada.

Regras de precedência para movimentos do Tipo 2, 2-AC desconectante e 2-AC conectante:

- Sub-rota 1 e 5: primeira e última sub-rotas, respectivamente, são sempre factíveis por não terem as relações de precedências modificadas.
- Sub-rota 4: a segunda sub-rota na nova configuração não deve conter entregas tais que as respectivas coletas não estejam contidas na própria sub-rota ou na sub-rota 1.
- Sub-rota 2: a terceira sub-rota é factível se não contiver nenhum par coleta-entrega (ser reversível).
- Sub-rota 3: a quarta sub-rota mantém as precedências em relação a outras sub-rotas. Para que seja factível não deve conter nenhum par coleta-entrega (ser reversível).

Avaliação de movimentos. Para explicar a busca dos movimentos 4-Opt do Tipo 1 e Tipo 2 algumas notações precisam ser definidas:

- AC(i, j) é uma notação para um 2-AC que remove as arestas i e j.
- Em um AC(i, j) é dito que este "termina" em j.
- Em um AC(i_1, j_1) é dito que este "cruza" o AC(i_2, j_2) se $i_1 < i_2 < j_1 < j_2$.
- $Z^C(i, j)$ representa o custo do 2-AC que remove as arestas i e j e adiciona arestas de forma conectante.
- $Z^D(i, j)$ representa o custo do 2-AC que remove as arestas i e j e adiciona arestas de forma desconectante.

A busca do melhor movimento percorre $i_2 = (1, ..., 2n - 2)$ e $j_2 = (i_2 + 2, ..., 2n)$ onde i_2 e j_2 são relativos ao segundo AC do movimento. É possível obter uma complexidade de $\mathcal{O}(n^2)$ se o melhor AC que "cruza" é

conhecido para cada i_2 e j_2 enumerados. A obtenção do melhor cruza de forma eficiente utiliza uma memória auxiliar de tamanho n, armazenando o par i para o melhor AC terminado em cada j.

Inicialmente $i = 0$ é atribuído como melhor para os todos os ACs do tipo termina. O procedimento de iteração de i_2 e j_2, atualização de T, e melhor AC do tipo cruza podem ser vistos no Algoritmo 5. Para um i_2 fixado, a iteração de j_2, inicialmente em $i_2 + 2$, tem naturalmente como o melhor AC do tipo cruza, o melhor AC que termina em $i_2 + 1$. O estado do melhor cruza e melhor termina podem ser atualizados sem grande esforço computacional, logo após a verificação dos movimentos. O melhor cruza é atualizado para a próxima iteração de j_2 assumindo o menor valor entre o melhor cruza conhecido e o melhor termina para j_2. Após esta operação, o melhor valor para o termina em j_2 pode ser atualizado observando que o par i_2, j_2 é um AC terminal em j_2 válido para a próxima iteração de i_2.

O melhor movimento é obtido comparando o melhor movimento conhecido com o 2-Opt de $Z^C(i_2, j_2)$ e com 4-Opt, que combinam o melhor cruza conectante com o AC(i_2, j_2) desconectante e o melhor cruza desconectante com os AC(i_2, j_2) conectantes e desconectantes. Estes movimentos somente substituem o melhor movimento conhecido se as regras de precedências são mantidas.

Alguns movimentos do Tipo 1 e Tipo 2 podem não ser cobertos com as regras definidas para atualização de ACs do tipo termina e cruza. Um caso extremo pode ser observado supondo que a aresta E_{i1} tem custo muito elevado comparado às outras, Figura 3.12. Nesse caso, o melhor AC que termina em todo j mantém o par $i = 0$ durante toda a busca, resultando em alguns movimentos que não satisfazem as regras de precedência, tendo que a maioria das regras têm alguma definição baseada no primeiro trecho.

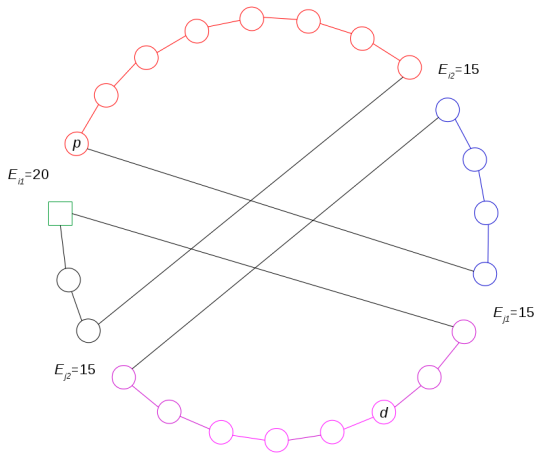


Figura 3.12: O avanço da aresta E_{i1} resultaria em um movimento válido de melhoria da rota.

Algoritmo 5: 4-Opt

```
1  i_2 ← 0
2  Para j_2 = 2 até j_2 < 2n
3      melhor_termina^x[j] ← Z^x(0, j)
4      pred_termina^x[j] ← 0
5      Verifica Melhor 2-Opt
6  Para  i_2 = 1 to i_2 < 2n − 1
7      melhor_cruza^x ← melhor_termina^x[i_2 + 1]
8      pred_cruza^x ← (melhor_termina^x[i + 1], i + 1)
9      Para  j_2 = i_2 + 2 to j_2 < 2n + 1
10         Verifica Melhor 4-Opt
11         Verifica Melhor 2-Opt
12         Se j_2 < 2n
13             Se melhor_cruza^x > melhor_termina^x[j_2]
14                 melhor_cruza^x ← melhor_termina^x[j_2]
15                 pred_cruza^x ← (pred_termina^x[j_2], j)
16             Se  melhor_termina^x[j_2] > Z^x(i_2, j_2)
17                 melhor_termina^x[j_2] ← Z^x(i_2, j_2)
18                 pred_termina^x[j_2] ← i_2
```

No Algoritmo 5, x indica o tipo de componente 2-AC, sendo $x \in \{C, D\}$ e representando componente 2-AC conectante quando $x = C$ e componente 2-AC desconectante para $x = D$.

3.6 Vizinhança Balas-Simonetti

Balas [28] introduziu uma classe de problemas TSP solucionáveis em tempo polinomial com restrições de precedência. Seja uma solução σ e dado um inteiro $k > 0$, este problema é solucionável em tempo polinomial se $\sigma_0 = 0$ e a Equação (3-17) é satisfeita.

$$\sigma_i < \sigma_j \quad \forall i, j : \; i + k \leq j. \tag{3-17}$$

O teorema de Balas mostra que qualquer TSP com esta condição pode ser solucionado em tempo $\mathcal{O}(k^2 2^{k-2} n)$ por programação dinâmica em tempo linear em n e exponencial em k [29]. Desta forma, fixando um valor de k, temos um algoritmo de tempo linear para resolver um problema de TSP se as ordens de visitas estiverem deslocadas a no máximo k posições das posições ótimas.

Quanto aos problemas que não podem ser representados por essa regra de precedência, o procedimento de programação dinâmica produz um ótimo

local em uma vizinhança larga definida pela mesma regra. Esta vizinhança, chamada então de vizinhança Balas e Simonetti (B&S), contém $2^{\Theta}(n)$ soluções e pode ser explorada linearmente para k constante ou polinomialmente quando $k = \mathcal{O}(\log n)$.

A recursão da programação dinâmica é representada por um problema de caminho mínimo em um grafo $G^* = (V^*, E^*)$ com $n + 1$ camadas. Desta forma, cada camada representa uma visita na rota começando e terminando no depósito, sendo o depósito o nó de origem e o nó terminal. Cada nó em uma camada i de G^* define a visita da posição i e o caminho percorrido nas camadas anteriores mantém a informação das visitas de 0 até $i - 1$. A notação adotada para representar eficientemente o estado em cada nó de G^* é a composição de quatro valores (i, j, S^-, S^+) onde: i é a posição na rota; j é o vértice atribuído a posição i; S^- é o conjunto de vértices originalmente visitados depois de i que foram associados a posições anteriores a i; e S^+ o conjunto de vértices originalmente visitados antes de i porém, não atribuídos a posições menores que i.

Os conjuntos S^- e S^+ trazem informações suficientes para filtrar movimentos que violariam as restrições de precedência. A transição para um nó de G^* é proibida caso nesta nova configuração exista uma entrega no conjunto S^- tal que a coleta referente esteja contida em S^+ ou igual a j (visita que será atribuída em i).

4
Metodologia

Este capítulo descreve a meta-heurística proposta para resolver o TSPPD. O algoritmo desenvolvido é baseado em um algoritmo genético híbrido (*Hybrid Genetic Search*, HGS) com melhorias locais combinadas com técnicas que permitem conciliar a qualidade com a diversidade da população de soluções. Neste método, o processo de melhoria local é realizado em dois estágios, usando duas estruturas de vizinhanças.

Um *Hybrid Genetic Search with Large Neigborhood* (HGS-LN) é proposto para solucionar o TSPPD. O algoritmo proposto é baseado no trabalho de Vidal et al. [30, 31] onde foi apresentado um controle de população balanceado em termos de qualidade da solução e contribuição quanto à diversidade. Vidal et al.[32] obtiveram bons resultados em diversos problemas de roteamento utilizando este tipo de abordagem.

Algoritmo 6: *Hybrid Genetic Search with Large Neighborhood*

```
1  Inicializa população
2  Enquanto numero de iterações sem melhoria < It_NI , e tempo < T_max
3      Seleção de pais S_1 e S_2
4      Geração de descendente C a partir de S_1 e S_2 (crossover)
5      Mutação em C
6      Educação de C (busca local)
7      Inserção de C na população
8      Se população máxima atingida
9          Seleção de sobreviventes
10     Se número de iterações sem melhoria > It_div
11         Diversificação da população
12 return melhor solução
```

O comportamento geral do HGS-LN é mostrado no Algoritmo 6. Em cada iteração, dois pais são selecionados para a aplicação do operador de *crossover* seguido do operador de mutação produzindo uma nova solução. Esta solução gerada é "educada" com uma busca local e adicionada à população. Se a população ultrapassar o tamanho máximo μ definido, o procedimento de seleção de sobreviventes elimina indivíduos que possuam clones e indivíduos

menos aptos em termos de qualidade e diversidade, até que o tamanho máximo da população seja respeitado. Caso a busca não encontre melhorias após uma quantidade determinada de iterações, o procedimento de diversificação da população elimina $\frac{2\mu}{3} + \lambda$ indivíduos da população, onde λ é a quantidade de descendentes em cada iteração. Novas 4μ soluções são adicionadas, aplicando em seguida a seleção de sobreviventes.

4.1 Representação da solução

Uma solução do TSPPD no algoritmo genético é representada pelo cromossomo que define a sequência de $2n$ visitas a todos os clientes. Desta forma, σ_k é o k-é*simo* vértice a ser visitado. Quando referido às arestas, a k-é*sima* aresta E_k está conectada aos vértices σ_k e σ_{k+1}.

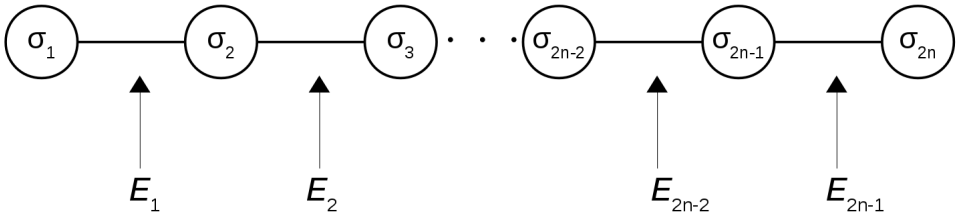


Figura 4.1: Representação da solução. A quantidade de visitas é igual a $2n$.

4.2 Avaliação da solução

O indivíduo σ é avaliado pela combinação linear do custo e sua contribuição para a diversidade da população. Esta combinação define uma métrica de qualidade da solução chamada de aptidão influenciada, ou *Biased Fitness* (BF) [30], definida pela Equação (4-1):

$$BF(\sigma) = R_c(\sigma) + \left(1 - \frac{nbElite}{nbIndiv}\right) \times R_d(\sigma), \tag{4-1}$$

onde *nbElite* é o número de indivíduos da elite que devem ser preservados, R_c é o *ranking* na população quanto à distância total da rota e R_d o *ranking* decrescente da contribuição de diversidade. A contribuição de diversidade é calculada como na Equação 4-2.

$$Div(\sigma) = \frac{1}{n_{close}} \sum_{\sigma' \in N_{close}} \delta^{RP}(\sigma, \sigma'), \tag{4-2}$$

sendo N_{close} um subconjunto da população composto pelas n_{close} soluções mais próximas de σ, onde n_{close} é um parâmetro do algoritmo. A distância entre

duas soluções $\delta^{RP}(\sigma, \sigma')$ é obtida usando *R-permutation* [33] calculada pela proporção de arestas de σ ausentes em σ'. Esta distância pode ser obtida em $\mathcal{O}(n)$ como mostrado no Algoritmo 7.

Algoritmo 7: *R-permutation*

```
1 Para i = 1 até 2n
2     sucessor[σ_{i-1}] ← σ_i
3 diferenças ← 0
4 Para i = 1 até 2n
5     Se sucessor[σ'_{i-1}] ≠ σ'_i e sucessor[σ'_i] ≠ σ'_{i-1}
6         diferenças ← diferenças + 1
7 return diferenças/(2n − 1)
```

4.3 Reprodução

O mecanismo de reprodução do HGS-LN seleciona dois pais, S^1 e S^2, e produz um único indivíduo C. A seleção dos pais é realizada através de um torneio binário onde, por duas vezes, dois indivíduos da população são escolhidos aleatoriamente em uma distribuição uniforme, sendo o mais apto mantido.

O operador de cruzamento de ordem linear (*Linear Order Crossover*, LOX) é tradicionalmente utilizado onde o cromossomo claramente tem duas extremidades, inicio e fim, bem definidas [34]. Este operador constrói a solução descendente C onde, entre dois pontos sorteados aleatoriamente, o trecho delimitado por estes pontos é copiado de S^1 para o descendente C. As lacunas formadas são preenchidas com vértices de requisições que ainda não foram visitados, obedecendo à ordem de visita encontrada em S^2.

Um exemplo de execução do LOX é visto na Figura 4.2, onde são mostrados S^1, S^2, C. Tendo $S^1 = \{0, 7, 1, 2, 3, 8, 5, 6, 4, 0\}$ e $S^2 = \{0, 3, 4, 1, 2, 6, 8, 7, 5\}$, assume-se que os dois pontos que definem o trecho a ser copiado foram escolhidos aleatoriamente como 2 e 5. A solução C, iniciada e terminada no depósito, copia a sequência $S^1_{2,5}$ para $C_{2,5}$ e tem as lacunas preenchidas pelos vértices ainda não visitados (4, 6, 7, 5), seguindo a ordem de visita de S^2.

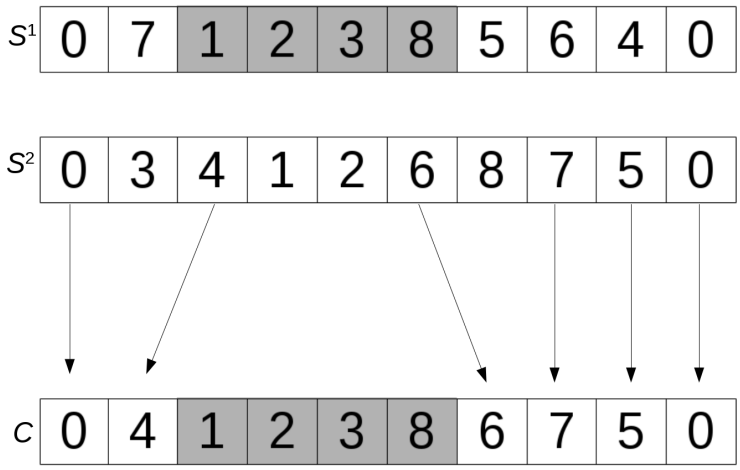


Figura 4.2: Exemplo de descendente gerado pelo operador LOX.

Esse operador pode provocar a violação de precedência supondo que, como na Figura 4.2, os vértices 3 e 4 representem uma coleta-entrega e são invertidos na ordem de visita C. O reparo das precedências remove os pares com violação de restrição e insere novamente na melhor posição possível, utilizando o Reposiciona Par descrito na Seção 3.1.

Nessas condições, o operador LOX propaga características de duas soluções existentes para a próxima geração, enquanto o reparo garante que apenas soluções factíveis são geradas colaborando com a diversificação da população pelas pequenas modificações.

4.4 Mutação

O operador de mutação tem como objetivo adicionar novas características à população. Outros trabalhos [22, 35] utilizaram o movimento "ponte dupla" como forma de causar distúrbios nos indivíduos para escapar de ótimos locais. Neste trabalho, a mutação é obtida pela combinação de dois procedimentos: primeiramente, o melhor movimento 4-Opt com duas componentes desconectantes (Tipo 1) é executado, aceitando movimentos mesmo que violando regras de precedência; em seguida, os pares que violarem a restrição de precedência são removidos e reinseridos na melhor posição possível consertando a solução, utilizando o movimento Reposiciona Par. A intuição da aplicação destas rotinas é que enquanto o super-movimento adiciona quatro novas arestas de alta qualidade, a reinserção dos pares que não respeitam precedência adiciona um distúrbio à nova solução, enquanto a torna factível.

Os movimentos 4-Opt de Tipo 2 não são considerados nesse processo por inverter parte da rota causando uma maior quantidade de violações de precedência e, consequentemente, grande número de reinserções. Isto faz com

que se perca demasiadamente o material genético dos indivíduos selecionados no processo de reprodução.

4.5 Educação

A busca local é essencial para uma rápida progressão em direção a soluções de alta qualidade. Esse procedimento tende a consumir a maior parte do esforço computacional total [31]. A busca local desenvolvida neste trabalho para o TSPPD é composta por dois estágios (Algoritmo 8). O primeiro estágio realiza modificações na solução baseadas nas vizinhanças *Reposiciona Par*, 2-Opt e Or-Opt, iterando os pares coleta-entrega, até que nenhuma melhoria possa ser alcançada. O segundo estágio, realiza modificações baseadas no melhor movimento contido nas vizinhanças 2k-Opt, 4-Opt e B&S, até que não seja mais possível melhorar a solução.

Algoritmo 8: *Busca Local*

```
1  Faça
2      σ' ← σ
3      Para v ∈ P
4          σ' ← Melhor Movimento(σ', v) ∈ {N1 ∪ N2 ∪ N3}
5      solução_melhorada ← σ' < σ
6      Se solução_melhorada
7          σ ← σ'
8  Enquanto solução_melhorada;
9  Faça
10     σ' ← Melhor Movimento(σ) ∈ {N4 ∪ N5 ∪ N6}
11     solução_melhorada ← σ' < σ
12     Se solução_melhorada
13         σ ← σ'
14 Enquanto solução_melhorada;
15 return σ
```

4.6 Gerenciamento da população

O comportamento da população complementa os outros mecanismos do algoritmo genético quanto a identificar e propagar características de boas soluções [30]. A implementação de população utilizada no HGS deve minimizar o tempo computacional das operações mais comuns como inserção, remoção e seleção de sobreviventes.

4.6.1 Inicialização

A população utilizada é controlada para conter no máximo $\mu + \lambda$ indivíduos. Durante inicialização da população são construídos 4μ novos indivíduos e, em seguida, a rotina de seleção de sobreviventes é executada, restando assim apenas μ elementos. Os passos da construção de cada solução são similares à geração de descendentes durante uma iteração do HGS-LN, modificados apenas por considerar o cromossomo inicial gerado por uma das quatro regras descritas abaixo. Após gerada a sequência de visitas inicial, a mutação é executada e, em seguida, a busca local completa a criação da solução.

Para a criação da sequência inicial, quatro regras são definidas e escolhidas com igual probabilidade. São elas:

1) De forma aleatória, pares coleta-entrega são adicionados consecutivamente ao fim da rota. O procedimento é repetido até que todos os pares sejam inseridos;
2) De forma aleatória, vértices que mantêm a relação de precedência são adicionados ao fim da rota. O procedimento é repetido até que todos os pares sejam inseridos;
3) De forma aleatória, dentre os vértices que mantêm a relação de precedência, um dos dez vértices mais próximos do vértice antecessor é adicionado ao fim da rota. O procedimento é repetido até que todos os pares sejam inseridos;
4) Iniciando a rota com um vértice de coleta aleatório, o vértice mais próximo ao anteriormente inserido, é adicionado ao fim da rota. O procedimento é repetido até que todos os pares sejam inseridos e, por fim, a reinserção das requisições com precedências violadas são resolvidas com a utilização do Reposiciona Par.

4.6.2 Diversificação

A diversificação da população é um desafio importante para evitar o problema de convergência sub-ótima prematura. Este problema é agravado pelo fato da busca local conduzir todas as soluções da população a um mínimo local, levando a uma considerável perda de diversidade na população. O procedimento de diversificação é executado quando, por It_{div} iterações, uma melhor solução não é encontrada, mantendo apenas os $\mu/3$ melhores indivíduos da população e adicionando outros 4μ novos indivíduos, como na etapa de

inicialização. A introdução do novo material genético possibilita a continuação da busca, mesmo depois da população perder grande parte de sua diversidade [30].

4.6.3 Seleção de sobreviventes

A estratégia de seleção de sobreviventes determina quais os μ indivíduos que devem ser mantidos na população para a próxima geração de forma a preservar os indivíduos que colaboram com a qualidade e diversidade. Desta forma, os λ elementos descartados devem ser prioritariamente indivíduos que possuem clones. Após a eliminação destes indivíduos, os elementos com o pior aptidão influenciada são descartados. Estes últimos são obtidos através da comparação da aptidão influenciada como mostrado no Algoritmo 9.

Algoritmo 9: Seleção de sobreviventes.

```
1 Para i = 1...λ
2     X ← indivíduos que possuem clones
3     Se X ≠ ∅
4         Remove da população o indivíduo P ∈ X com maior aptidão
            influenciada
5     senão
6         Remove da população o indivíduo P com maior aptidão
            influenciada
7     Atualiza distâncias e aptidões influenciadas
```

5
Experimentos e Análises Computacionais

Neste capítulo discutimos como os experimentos foram realizados e os resultados obtidos. Primeiramente o conjunto de instâncias usadas nos testes são descritas quanto às estruturas e formas de construção. A metodologia para seleção de parâmetros do algoritmo também é abordada nesse capítulo assim como o desempenho computacional e o comparativo com a literatura quanto qualidade das soluções e tempo de execução. Além disto, apresentamos o estudo do impacto de componentes do algoritmo desenvolvido e do tempo prático de execução com relação à complexidade computacional teórica discutida nos capítulos anteriores.

Os experimentos computacionais foram conduzidos em um computador pessoal equipado com processador Intel Core i5-6600k 3.5GHz e 16GB de memória RAM, utilizando apenas um núcleo. O código fonte foi escrito em C++ utilizando o compilador g++ 6.4.1 no sistema operacional Linux, distribuição Fedora 25 64 bits.

Para facilitar a interpretação dos resultados apresentados nas próximas seções, a qualidade das soluções são apresentadas como o erro relativo à melhor solução conhecida (MSC), como na Equação (5-1):

$$E = \frac{f - f_{msc}}{f_{msc}} \times 100, \tag{5-1}$$

onde f é o valor da função objetivo encontrada pelo algoritmo enquanto f_{msc} é o valor da função objetivo da melhor solução conhecida até então.

5.1
Instâncias

O HGS-LN foi testado em 163 instâncias com quantidade de vértices variando de 11 a 493, divididos em três conjuntos. O dois primeiros conjuntos de instâncias foram introduzidos por Renaud et al. [9] chamados de conjunto de instâncias RBO00 Classe 1 e Classe 2. O terceiro conjunto foi apresentado por Dumitrescu et al. [8], nomeado como instâncias Classe 3. Em todas as instâncias, as distâncias entre vértices são arredondadas para o inteiro mais

próximo.

O conjunto RBO00 Classe 1 contempla um total de 108 instâncias para avaliação de performance com número de vértices que variam entre 51 e 493, divididas em três subconjuntos chamados A, B e C. Estes problemas derivaram de 36 instâncias do TSPLIB [14] gerando para cada problema três instâncias diferentes de TSPPD, baseadas nas seguintes regras de definição dos pares coleta-entrega: A) O vértice de entrega é selecionado entre os 5 vértices não selecionados mais próximos da coleta. B) O vértice de entrega é selecionado entre os 10 vértices não selecionados mais próximos da coleta. C) Escolha de vértice para entrega é livre entre todos os vértices não selecionados. Nesta classe de problemas, apenas as instâncias EIL51A, EIL51B e ST69A possuem valores ótimos provados obtidos por Dumitrescu et al. [8].

Instâncias RBO00 Classe 2 são no total 20 problemas com 101 e 201 vértices, gerados a partir da solução ótima do TSP pelo algoritmo de *branch-and-cut* [36]. Os pares coleta-entrega são definidos para que não violem as regras de precedência nesta solução. O primeira visita da solução é designada ao depósito e as relações de coleta-entrega são criadas da seguinte maneira até que todos os vértices sejam mapeados: o próximo vértice visitado na solução TSP ainda não selecionado é atribuído a coleta, e também entre os vértices não selecionados, o par entrega é aleatoriamente escolhido e associado.

O algoritmo de *branch-and-cut* de Dumitrescu et al. [8] resolveu facilmente as instâncias RBO00 de Classe 2. Eles justificaram a capacidade de solução exata destas instâncias pelo fato de obter um limite inferior muito bom na raiz, uma vez que a relaxação linear das restrições de precedência levam à solução ótima do TSP. O conjunto Classe 3, com 35 instâncias, foi então proposto com número de vértices variando entre 11 e 71, onde o primeiro vértice representa o depósito e os pares coleta-entrega definidos aleatoriamente. Neste conjunto de 35 instâncias, 28 foram resolvidas otimamente.

5.2 Calibração de parâmetros

A calibração do HGS-LN, como a maioria das meta-heurísticas, depende de um conjunto de parâmetros que afetam diretamente os resultados. Os testes de calibração foram executados em seis instâncias do *dataset* RBO00: TS225A, PR123A, RD399B, PR439B, D439C e KROB199C. Estas instâncias foram escolhidas particularmente por conterem três regras de proximidade e diferentes dificuldades. Para metade destas instâncias, o trabalho mais recente na literatura não encontrou o melhor valor previamente conhecido, enquanto que para as restantes obtiveram melhorias de 2,32% a 4,62%.

Inicialmente os parâmetros considerados para o HGS-LN foram os mesmos descritos para o HGS original [30] mostrados na Tabela 5.1. Dois parâmetros foram adicionados referentes às vizinhanças Or-Opt e Balas&Simonetti com valores iniciais *or-k*= 10 e *bs-k*= 5 escolhidos de forma arbitrária.

Tabela 5.1: Parâmetros originais do HGS.

Parâmetros		
μ	Tamanho da população	25
λ	Número de descendentes por geração	1
el	Proporção de indivíduos elite, tal que $nbElit = el \times \mu$	0,4
nc	Proporção de indivíduos próximos tal que $N_{close} = nc \times \mu$	0,2

Os testes nas instâncias de calibração apresentaram resultados em média 0,46% melhores que o melhor valor conhecido. No entanto, pela observação do comportamento evolutivo do algoritmo, a população se mostra muito sensível ao elitismo com uma grande perda de material genético na população em poucas gerações. A Figura 5.1 mostra o declínio da diversidade da população durante as gerações para a instância D493C. A diversidade decresce rapidamente para aproximadamente 5% mesmo após a diversificação da população que pode ser observada próximo às gerações 1600, 2500, 3750 e 4200.

Dessa forma, os parâmetros *el*, referentes à proporção de indivíduos elite na população, os parâmetros introduzidos *or-k*, utilizado como tamanho máximo da cadeia realocada na vizinhança Or-Opt, e *bs-k*, que define a proximidade máxima considerada em Balas&Simonetti, são variáveis livres a serem calibradas. Inicialmente $el = 0,4$, *or-k* = 10 e *bs-k* = 5 são calibradas pela metodologia *one-factor-at-a-time* (OFAT) [37]. A Tabela 5.2 apresenta o intervalo de valores para cada parâmetro e o valor final selecionado.

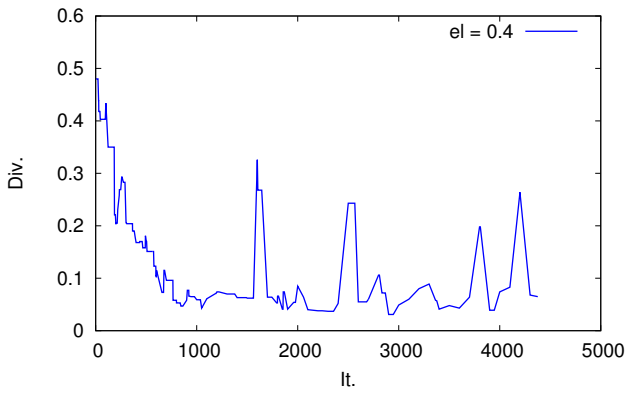


Figura 5.1: Declínio da diversidade com *el*=0,4 na instância D493C.

Tabela 5.2: Intervalo de valores para os parâmetros *el*, *or-k* e *bs-k*.

Parâmetro		Valores	Valor Final
el	Proporção de indivíduos elite	{0,1; 0,2; 0,4; 0,6; 0,8; 1.0}	0,1
or-k	Or-Opt tamanho máximo k	{2; 5; 7; 10}	10
bs-k	B&S parâmetro k	{2; 5; 7; 10}	7

A Tabela 5.3 mostra valores para os parâmetros *el, or-k* e *bs-k*, a coluna E_{med} que reporta o erro médio em 5 rodadas relativo aos melhores valores conhecidos para as instâncias de calibração e a última coluna o tempo médio de execução.

Tabela 5.3: Resultados da calibração

el	*or-k*	*bs-k*	E_{med}	t(s)
0,1	10	5	**-0,62**	105,40
0,2	10	5	-0,54	51,65
0,4	10	5	-0,46	62,53
0,6	10	5	-0,16	45,62
0,8	10	5	-0,13	50,92
1,0	10	5	0,40	52,62
0,1	2	5	-0,44	130,95
0,1	5	5	-0,61	96,15
0,1	7	5	-0,61	79,98
0,1	**10**	5	**-0,62**	105,40
0,1	**10**	2	-0,39	66,55
0,1	**10**	5	-0,62	105,40
0,1	**10**	**7**	**-0,67**	193,05
0,1	**10**	10	0,56	376,97

5.3 Resultados experimentais

Os resultados de performance computacional do HGS-LN relativos às instâncias Classe 1, Classe 2 e Classe 3 são apresentados nas Tabelas 5.4, 5.5 e 5.6, respectivamente. Pela natureza não determinística da meta-heurística proposta, os resultados do HGS-LN são relativos a 10 rodadas em cada uma das instâncias testadas.

Para comparação dos resultados são considerados os três algoritmos apresentados na literatura. Estes trabalhos consistem na heurística de deleção-

reinserção [9] apresentada por Renaud et al., posteriormente agregada às heurísticas de perturbações [10] (RBO00/02), no algoritmo de *branch-and-cut* apresentado por Dumitrescu et al. [8] (BC10) e na meta-heurística *Large Neighborhood Search* (LNS2017) introduzida por Veenstra et al. [7]. A melhor solução conhecida (MSC) é considerada a solução com menor custo entre os valores obtidos nos trabalhos conhecidos até então.

Os resultados originalmente publicados para as instâncias Classe 1 não possuem valores específicos. Para facilitar a análise, estes valores foram agrupados de acordo com a quantidade de vértices e estratégias de construção das instâncias, reportando a razão entre os valores e o melhor valor obtido para cada uma das sete perturbações. Valores individuais destes testes foram publicados mais recentemente sumarizados com os testes de BC10 e não fornecendo o tempo de máquina utilizado [7]. Por este motivo, a análise de resultados nas Tabelas 5.4 e 5.5 também resume os resultados obtidos por RBO00/02 e BC10 da mesma maneira.

Dois critérios de parada foram adotados. O tempo limite de execução T_{max} foi fixado em 2500 segundos e o número máximo de iterações sem melhorias IT_{NI} em 10000, para instâncias Classe 1, e 1000 para instâncias Classe 2 e 3. Estes valores foram arbitrariamente escolhidos para obter tempos computacionais da mesma ordem de grandeza dos trabalhos anteriores.

A análise do tempo computacional é realizada comparando os resultados obtidos com LNS2017. Os experimentos reportados para o LNS2017 foram performados em uma máquina equipada com o processador Intel Core i3-2120 operando a 3.3GHz e utilizando apenas um núcleo. Para tornar a comparação mais relevantes, os tempos mostrados nas Tabelas 5.4, 5.5 e 5.6 foram convertidos para tempos correspondentes ao processador utilizado neste trabalho, corrigindo os tempos de LNS2017 por um fator de 0,79. Este fator é calculado pela razão entre os pontos de performance informados pelo PassMark® [38] onde os processadores Intel Core i3-2120 e Intel Core i5-6600k pontuaram de 1692 e 2149, respectivamente.

Os resultados para as 108 instâncias da Classe 1 são mostrados na Tabela 5.4 de maneira sumarizada. Resultados individuais são apresentados na Tabela 5.7. O HGS-LN também obteve uma boa performance neste conjunto de instâncias. Entre as 108 instâncias, para 106 foram encontradas as melhores soluções conhecidas ou novas melhores soluções foram produzidas. Mais especificamente, a melhor solução conhecida foi encontrada em todas as 10 rodadas para 62 instâncias e a média dos valores após 10 rodadas foi inferior às melhores soluções previamente conhecidas para outras 44. As instâncias KROD99C e PR107B reproduziram os mesmos valores encontrados

em LNS2017 mas apresentaram erro de 0,08% e 0,02%, respectivamente, comparados com RBO00/02. O erro médio E_{med} é de aproximadamente -0,30% e o erro mínimo -0,34% comparado com resultados anteriores. A melhoria na qualidade das soluções é ainda mais expressiva quando o número de vértices aumenta, especialmente nas instâncias de tipos B e C, chegando a melhoras um pouco maiores que 3,7% na instância PR439C, por exemplo. O tempo computacional também supera o LNS2017.

A Tabela 5.5 apresenta os resultados para as instâncias Classe 2. O HGS-LN encontrou um erro médio E_{med} igual a 0%, o que significa que o valor ótimo foi sempre alcançado. A análise de tempo mostra que o algoritmo proposto supera a performance do LNS2017 em aproximadamente 17 vezes.

Tabela 5.4: Resultados computacionais (Instâncias Classe 1).

		BC10; RBO00/02	LNS2017			HGS-LN		
n	tipo	E_{min}	E_{med}	E_{min}	t**(s)	E_{med}	E_{min}	t(s)
	A	0,02	0,22	**0**	19,36	**-0,02**	**-0,02**	11,70
50-99	B	**0,00**	0,13	0,03	20,65	**-0,01**	**-0,01**	12,36
	C	**0,00**	0,06	0,01	21,98	0,01	0,01	12,35
	A	0,40	0,35	0,02	93,11	**-0,17**	**-0,17**	29,73
100-199	B	0,49	0,26	0	96,97	**-0,10**	**-0,10**	31,05
	C	0,48	0,37	**0**	104,34	**-0,08**	**-0,08**	34,59
	A	1,30	0,82	0,10	405,78	**-0,58**	**-0,58**	90,29
200-299	B	1,12	0,64	**0**	425,21	**-0,60**	**-0,62**	98,50
	C	1,62	0,69	**0**	461,52	**-0,52**	**-0,58**	122,18
	A	2,97	0,85	**0**	1131,12	**-0,84**	**-1,04**	350,67
300-499	B	2,78	0,86	**0**	1144,66	**-0,91**	**-1,08**	428,45
	C	1,70	1,16	0,02	1255,22	**-1,31**	**-1,57**	521,94
	A	0,84	0,46	0,02	287,00	**-0,29**	**-0,33**	86,12
All	B	0,81	0,37	0,01	293,85	**-0,28**	**-0,31**	100,94
	C	0,70	0,45	0,01	320,60	**-0,32**	**-0,37**	121,18
Média		1,01	0,43	0,01	300,48	**-0,30**	**-0,34**	102,75

*Valores ótimos provados.

**Valores de tempo ajustados.

Tabela 5.5: Resultados computacionais (Instâncias Classe 2).

		BC10; RBO00/02	LNS2017			HGS-LN	
Instância	n	f_{msc}	E_{med}	E_{min}	t**(s)	E_{med}	t(s)
N101p1	101	799*	**0**	**0**	29,39	**0**	2,56
N101p2	101	729*	**0**	**0**	30,02	**0**	1,73
N101p3	101	748*	**0**	**0**	29,94	**0**	1,95
N101p4	101	807*	**0**	**0**	29,94	**0**	2,43
N101p5	101	783*	**0**	**0**	29,94	**0**	1,93
N101p6	101	755*	**0**	**0**	30,02	**0**	2,25
N101p7	101	767*	**0**	**0**	30,18	**0**	1,72
N101p8	101	762*	**0**	**0**	29,94	**0**	2,02
N101p9	101	766*	**0**	**0**	29,86	**0**	2,14
N101p10	101	754*	**0**	**0**	30,10	**0**	2,06
N201p1	201	1039*	**0**	**0**	204,93	**0**	8,65
N201p2	201	1086*	**0**	**0**	204,14	**0**	8,44
N201p3	201	1070*	0,19	**0**	207,38	**0**	8,54
N201p4	201	1050*	**0**	**0**	204,93	**0**	7,71
N201p5	201	1052*	**0**	**0**	205,56	**0**	9,50
N201p6	201	1059*	**0**	**0**	206,98	**0**	10,42
N201p7	201	1036*	**0**	**0**	206,19	**0**	8,29
N201p8	201	1079*	**0**	**0**	205,80	**0**	8,55
N201p9	201	1050*	**0**	**0**	204,77	**0**	8,75
N201p10	201	1085*	**0**	**0**	205,48	**0**	9,87

*Valores ótimos provados.

**Valores ajustados de tempo.

Na Tabela 5.6, os resultados obtidos nas instâncias Classe 3 são mostrados. O HGS-LN encontrou a melhor solução conhecida para todas as instâncias nas 10 rodadas. Além de várias possuírem valores ótimos provados, devido a forma de construção, estas instâncias trazem características distintas comparadas às instâncias Classe 2, afirmando o poder do HGS-LN na resolução do TSPPD. O algoritmo proposto também supera os algoritmos apresentados anteriormente em qualidade das soluções e tempo computacional.

Tabela 5.6: Resultados computacionais (Instâncias Classe 3).

			BC10	LNS2017			HGS-LN	
Instance	n	f_{msc}	E_{min}	E_{med}	E_{min}	t**(s)	E_{med}	t(s)
prob5a	11	3585*	**0**	**0**	**0**	0,24	**0**	0,05
prob5b	11	2565*	**0**	**0**	**0**	0,08	**0**	0,02
prob5c	11	3787*	**0**	**0**	**0**	0,08	**0**	0,02
prob5d	11	3128*	**0**	**0**	**0**	0,08	**0**	0,02
prob5e	11	3123*	**0**	**0**	**0**	0,08	**0**	0,02
prob10a	21	4896*	**0**	**0**	**0**	0,40	**0**	0,12
prob10b	21	4490*	**0**	**0**	**0**	0,40	**0**	0,12
prob10c	21	4070*	**0**	**0**	**0**	0,40	**0**	0,11
prob10d	21	4551*	**0**	**0**	**0**	0,40	**0**	0,10
prob10e	21	4874*	**0**	**0**	**0**	0,40	**0**	0,12
prob15a	31	5150*	**0**	**0**	**0**	1,11	**0**	0,25
prob15b	31	5391*	**0**	**0**	**0**	1,11	**0**	0,23
prob15c	31	5008*	**0**	**0**	**0**	1,11	**0**	0,22
prob15d	31	5566*	**0**	**0**	**0**	1,11	**0**	0,23
prob15e	31	5229*	**0**	**0**	**0**	1,19	**0**	0,22
prob20a	41	5698*	**0**	**0**	**0**	2,53	**0**	0,41
prob20b	41	6213*	**0**	**0**	**0**	2,77	**0**	0,39
prob20c	41	6200*	**0**	**0**	**0**	2,45	**0**	0,38
prob20d	41	6106*	**0**	**0**	**0**	2,45	**0**	0,41
prob20e	41	6465*	**0**	**0**	**0**	2,53	**0**	0,36
prob25a	51	7332	**0**	**0**	**0**	4,35	**0**	0,57
prob25b	51	6665*	**0**	0,01	**0**	4,42	**0**	0,54
prob25c	51	7095*	**0**	0,07	**0**	4,42	**0**	0,51
prob25d	51	7069*	**0**	0,68	**0**	4,42	**0**	0,69
prob25e	51	6754*	**0**	**0**	**0**	4,42	**0**	0,58
prob30a	61	7309	**0**	**0**	**0**	7,35	**0**	0,69
prob30b	61	6857*	**0**	**0**	**0**	7,19	**0**	0,74
prob30c	61	7723*	**0**	**0**	**0**	7,43	**0**	0,72
prob30d	61	7310*	**0**	0,18	**0**	7,35	**0**	0,75
prob30e	61	7213	**0**	**0**	**0**	7,43	**0**	0,74
prob35a	71	7746*	**0**	**0**	**0**	11,38	**0**	0,97
prob35b	71	7904	**0**	**0**	**0**	11,14	**0**	1,00
prob35c	71	7949	**0**	0,05	**0**	11,38	**0**	0,90
prob35d	71	7905	**0**	0,42	**0**	11,22	**0**	0,96
prob35e	71	8521	0,11	0,22	**0**	11,30	**0**	1,01

*Valores ótimos provados.

**Valores de tempo ajustados.

Tabela 5.7: Resultados completos (Instâncias Classe 1).

			BC10 RBO00/02		LNS2017					HGS-LN				
Instâncias	n	f_{msc}	f_{min}	E_{min}	f_{med}	E_{med}	t**(s)	f_{min}	E_{min}	f_{med}	E_{med}	t(s)	f_{min}	E_{min}
EIL51A	51	464*	**464**	0	**464**	0	4,029	**464**	0	**464**	0	4,730	**464**	0
EIL51B	51	469*	**469**	0	**469**	0	4,266	**469**	0	**469**	0	4,950	**469**	0
EIL51C	51	488	**488**	0	**488**	0	4,345	**488**	0	**488**	0	6,100	**488**	0
ST69A	69	764*	**764**	0	**764**	0	9,085	**764**	0	**764**	0	6,950	**764**	0
ST69B	69	771	**771**	0	**771**	0	9,006	**771**	0	**771**	0	7,660	**771**	0
ST69C	69	793	**793**	0	**793**	0	10,112	**793**	0	**793**	0	7,320	**793**	0
EIL75A	75	583	**583**	0	589,3	1,081	11,692	**583**	0	**583**	0	7,870	**583**	0
EIL75B	75	601	**601**	0	601,3	0,050	12,561	**601**	0	**601**	0	10,060	**601**	0
EIL75C	75	590	**590**	0	590,1	0,017	12,956	**590**	0	**590**	0	8,290	**590**	0
PR75A	75	130531	**130531**	0	**130531**	0	12,087	**130531**	0	**130531**	0	8,090	**130531**	0
PR75B	75	128397	**128397**	0	128399	0,002	12,403	**128397**	0	**128397**	0	9,160	**128397**	0
PR75C	75	124509	**124509**	0	**124509**	0	13,272	**124509**	0	**124509**	0	7,960	**124509**	0
KROA99A	99	24980	**24980**	0	25047,2	0,269	25,201	**24980**	0	**24980**	0	12,850	**24980**	0
KROA99B	99	26552	**26552**	0	26561,3	0,035	27,413	**26552**	0	**26552**	0	14,070	**26552**	0
KROA99C	99	25769	**25769**	0	**25769**	0	28,835	**25769**	0	**25769**	0	13,160	**25769**	0
KROB99A	99	25631	**25631**	0	25687,8	0,222	25,043	**25631**	0	**25631**	0	12,580	**25631**	0
KROB99B	99	25384	**25384**	0	**25384**	0	27,176	**25384**	0	**25384**	0	12,570	**25384**	0
KROB99C	99	25795	**25795**	0	**25795**	0	29,072	**25795**	0	**25795**	0	14,530	**25795**	0
KROC99A	99	26113	**26113**	0	26215,5	0,393	25,833	**26113**	0	**26113**	0	15,370	**26113**	0
KROC99B	99	25602	**25602**	0	25640,4	0,150	26,781	**25602**	0	**25602**	0	15,140	**25602**	0

(continuação na próxima página)

Tabela 5.7: Resultados completos (Instâncias Classe 1).

			BC10 RBO00/02		LNS2017					HGS-LN				
Instâncias	n	f_{msc}	f_{min}	E_{min}	f_{med}	E_{med}	t**(s)	f_{min}	E_{min}	f_{med}	E_{med}	t(s)	f_{min}	E_{min}
KROC99C	99	26065	**26065**	0	**26065**	0	28,440	**26065**	0	**26065**	0	15,250	**26065**	0
KROD99A	99	25392	**25392**	0	25423,5	0,124	24,806	**25392**	0	**25392**	0	14,210	**25392**	0
KROD99B	99	26179	**26179**	0	26212,7	0,129	26,702	**26179**	0	**26179**	0	14,630	**26179**	0
KROD99C	99	26021	**26021**	0	26171,3	0,578	28,519	26041	0,077	26041	0,080	16,050	26041	0,080
KROE99A	99	25879	**25879**	0	25881,5	0,010	25,359	**25879**	0	**25879**	0	15,310	**25879**	0
KROE99B	99	26584	26584	0	26642,8	0,221	26,781	26591	0,026	**26542**	-0,160	17,430	**26542**	-0,160
KROE99C	99	26021	**26021**	0	26027,7	0,026	28,914	**26021**	0	**26021**	0	16,520	**26021**	0
RAT99A	99	1401	1401	0	1403,3	0,164	24,964	1401	0	**1399**	-0,140	13,630	**1399**	-0,140
RAT99B	99	1460	**1460**	0	1472	0,822	27,018	1464	0,274	**1460**	0	14,980	**1460**	0
RAT99C	99	1370	**1370**	0	1370,4	0,029	28,677	**1370**	0	**1370**	0	12,350	**1370**	0
RD99A	99	9506	9522	0,168	9523,5	0,184	24,806	9506	0	**9496**	-0,110	17,140	**9496**	-0,110
RD99B	99	9464	**9464**	0	**9464**	0	27,097	**9464**	0	**9464**	0	15,360	**9464**	0
RD99C	99	9185	**9185**	0	9186,1	0,012	28,598	**9185**	0	**9185**	0	18,290	**9185**	0
EIL101A	101	695	**695**	0	695,8	0,115	27,887	**695**	0	**695**	0	14,630	**695**	0
EIL101B	101	705	**705**	0	712	0,993	29,151	**705**	0	**705**	0	18,620	**705**	0
EIL101C	101	690	**690**	0	**690**	0	30,810	**690**	0	**690**	0	15,820	**690**	0
LIN105A	105	17791	17791	0	17836	0,253	30,099	17824	0,185	**17677**	-0,640	15,560	**17677**	-0,640
LIN105B	105	17482	**17482**	0	17486	0,023	32,469	17483	0,006	**17482**	0	16,400	**17482**	0
LIN105C	105	17813	**17813**	0	**17813**	0	34,918	**17813**	0	**17813**	0	18,030	**17813**	0
PR107A	107	51537	**51537**	0	51596,9	0,116	31,758	**51537**	0	**51537**	0	13,140	**51537**	0

(continuação na próxima página)

Tabela 5.7: Resultados completos (Instâncias Classe 1).

			BC10 RBO00/02		LNS2017					HGS-LN				
Instâncias	n	f_{msc}	f_{min}	E_{min}	f_{med}	E_{med}	t**(s)	f_{min}	E_{min}	f_{med}	E_{med}	t(s)	f_{min}	E_{min}
PR107B	107	51675	**51675**	0	51686	0,021	34,207	51686	0,021	51686	0,020	14,290	51686	0,020
PR107C	107	52657	**52657**	0	**52657**	0	36,261	**52657**	0	**52657**	0	15,990	**52657**	0
PR123A	123	75542	**75542**	0	75603,3	0,081	47,795	75575	0,044	**75542**	0	22,890	**75542**	0
PR123B	123	75389	**75389**	0	75493,9	0,139	50,876	**75389**	0	**75389**	0	19,620	**75389**	0
PR123C	123	82341	**82341**	0	82379,3	0,047	56,169	**82341**	0	**82341**	0	22,760	**82341**	0
BIER127A	127	132100	133653	1,176	133470,4	1,037	54,036	132100	0	**131974**	-0,100	30,610	**131974**	-0,100
BIER127B	127	133503	134670	0,874	133793,6	0,218	57,433	133503	0	**133378**	-0,090	30,630	**133378**	-0,090
BIER127C	127	132199	132972	0,585	132307,5	0,082	60,356	132199	0	**132020**	-0,140	38,300	**132020**	-0,140
PR135A	135	110939	111475	0,483	111519,6	0,523	62,173	**110939**	0	**110939**	0	24,230	**110939**	0
PR135B	135	110763	**110763**	0	110779,1	0,015	67,703	**110763**	0	**110763**	0	25,580	**110763**	0
PR135C	135	114232	**114232**	0	114353,6	0,106	73,549	**114232**	0	**114232**	0	29,820	**114232**	0
PR143A	143	80274	**80274**	0	**80274**	0	75,445	**80274**	0	**80274**	0	23,250	**80274**	0
PR143B	143	89472	90484	1,131	89898	0,476	79,948	**89472**	0	**89472**	0	24,900	**89472**	0
PR143C	143	91979	**91979**	0	92085,6	0,116	87,374	**91979**	0	**91979**	0	32,300	**91979**	0
KROA149A	149	30833	31467	2,056	30864,1	0,101	83,977	**30833**	0	**30833**	0	27,470	**30833**	0
KROA149B	149	31733	**31733**	0	31808,9	0,239	90,692	**31733**	0	**31733**	0	34,850	**31733**	0
KROA149C	149	32235	32351	0,360	32492,7	0,799	97,091	**32235**	0	**32235**	0	35,190	**32235**	0
KROB149A	149	31147	31360	0,684	31366,9	0,706	88,006	31147	0	**30687**	-1,480	30,540	**30687**	-1,480
KROB149B	149	31696	31995	0,943	31745,9	0,157	91,166	31696	0	**31687**	-0,030	37,850	**31687**	-0,030
KROB149C	149	31735	31771	0,113	31929,7	0,614	99,303	**31735**	0	**31735**	0	32,360	**31735**	0

(continuação na próxima página)

Tabela 5.7: Resultados completos (Instâncias Classe 1).

			BC10 RBO00/02		LNS2017					HGS-LN				
Instâncias	n	f_{msc}	f_{min}	E_{min}	f_{med}	E_{med}	t**(s)	f_{min}	E_{min}	f_{med}	E_{med}	t(s)	f_{min}	E_{min}
PR151A	151	90494	**90494**	0	**90494**	0	104,754	**90494**	0	**90494**	0	25,550	**90494**	0
PR151B	151	94937	**94937**	0	95256,6	0,337	100,567	**94937**	0	**94937**	0	30,280	**94937**	0
PR151C	151	97288	**97288**	0	**97288**	0	103,095	**97288**	0	**97288**	0	27,530	**97288**	0
U159A	159	51710	51710	0	51941	0,447	105,623	51710	0	**51686**	-0,050	29,250	**51686**	-0,050
U159B	159	50494	**50494**	0	50560,9	0,132	110,205	**50494**	0	**50494**	0	28,170	**50494**	0
U159C	159	52479	53189	1,353	53322,9	1,608	120,238	52479	0	**52187**	-0,560	36,180	**52187**	-0,560
D197A	197	15886	16032	0,919	15967,6	0,514	183,438	**15886**	0	**15886**	0	37,410	**15886**	0
D197B	197	16060	16435	2,335	16084,8	0,154	186,045	**16060**	0	**16060**	0	39,770	**16060**	0
D197C	197	16673	16724	0,306	**16673**	0	199,633	**16673**	0	**16673**	0	38,810	**16673**	0
KROA199A	199	34451	34484	0,096	34569	0,343	204,452	34451	0	**34400**	-0,150	58,760	**34400**	-0,150
KROA199B	199	34890	35259	1,058	35079,6	0,543	212,826	34890	0	**34874**	-0,050	58,720	**34874**	-0,050
KROA199C	199	35982	36251	0,748	36331,7	0,972	230,838	35982	0	**35856**	-0,350	81,640	**35856**	-0,350
KROB199A	199	34703	34774	0,205	34947,4	0,704	204,136	**34703**	0	**34703**	0	62,900	**34703**	0
KROB199B	199	34634	34806	0,497	34700,3	0,191	214,248	34634	0	**34212**	-1,220	55,030	**34212**	-1,220
KROB199C	199	35613	36784	3,288	35888,4	0,773	231,154	**35613**	0	**35613**	0	59,480	**35613**	0
PR225A	225	103652	105321	1,610	103711,7	0,058	260,779	**103652**	0	**103652**	0	68,670	**103652**	0
PR225B	225	109402	109502	0,091	109759,8	0,327	279,581	109402	0	**109324**	-0,070	53,110	**109324**	-0,070
PR225C	225	112240	113116	0,780	114198,5	1,745	299,331	**112240**	0	**112240**	0	71,100	**112240**	0
TS225A	225	156646	156646	0	158742,3	1,338	279,739	157457	0,518	153880,8	-1,770	73,800	**153834**	-1,800
TS225B	225	160404	161353	0,592	161529,5	0,702	293,643	160404	0	156923,3	-2,170	88,980	**156839**	-2,220

(continuação na próxima página)

Tabela 5.7: Resultados completos (Instâncias Classe 1).

			BC10 RBO00/02		LNS2017					HGS-LN				
Instâncias	n	f_{msc}	f_{min}	E_{min}	f_{med}	E_{med}	t**(s)	f_{min}	E_{min}	f_{med}	E_{med}	t(s)	f_{min}	E_{min}
TS225C	225	162437	165073	1,623	163254,7	0,503	309,285	162437	0	**159868**	-1,580	78,100	**159868**	-1,580
GIL261A	261	2758	2808	1,813	2775,5	0,635	436,870	2758	0	**2736**	-0,800	98,350	**2736**	-0,800
GIL261B	261	2839	2887	1,691	2876,3	1,314	457,489	2839	0	2834,4	-0,160	129,410	**2833**	-0,210
GIL261C	261	2812	2885	2,596	2839,2	0,967	486,166	2812	0	2797,3	-0,520	168,160	**2793**	-0,680
PR263A	263	60858	61805	1,556	61221,5	0,597	427,232	60858	0	**60800**	-0,100	90,610	**60800**	-0,100
PR263B	263	60360	60489	0,214	60568,1	0,345	438,292	60360	0	60302,1	-0,100	90,870	**60294**	-0,110
PR263C	263	64158	65514	2,114	64171,8	0,022	502,361	64158	0	**64143**	-0,020	95,020	**64143**	-0,020
PR299A	299	56681	57532	1,501	57526,4	1,492	624,258	56681	0	**56560**	-0,210	120,030	**56560**	-0,210
PR299B	299	57613	59342	3,001	57916	0,526	657,043	57613	0	57327,8	-0,500	130,110	**57320**	-0,510
PR299C	299	58860	59436	0,979	58989,1	0,219	710,447	58860	0	58591,8	-0,460	198,510	**58496**	-0,620
LIN317A	317	49853	51303	2,909	50361,1	1,019	613,514	49853	0	49689,3	-0,330	179,680	**49678**	-0,350
LIN317B	317	50754	51444	1,359	51319,8	1,115	635,476	50754	0	50022,9	-1,440	276,530	**50004**	-1,480
LIN317C	317	50815	51257	0,870	51147,6	0,655	703,179	50815	0	50664,8	-0,300	270,300	**50609**	-0,410
RD399A	399	17359	18101	4,274	17685,7	1,882	1102,682	17359	0	**17327**	-0,180	253,170	**17327**	-0,180
RD399B	399	18014	18451	2,426	18272,6	1,436	1120,299	18014	0	17809,4	-1,140	592,670	**17738**	-1,530
RD399C	399	18227	18839	3,358	18438	1,158	1230,267	18227	0	18073,8	-0,840	651,030	**18034**	-1,060
FL417A	417	13792	13874	0,595	13804,6	0,091	1031,898	13792	0	**13558**	-1,700	213,310	**13558**	-1,700
FL417B	417	13815	14089	1,983	13818,3	0,024	1036,796	**13815**	0	**13815**	0	193,700	**13815**	0
FL417C	417	15618	**15618**	0	15626	0,051	1099,206	**15618**	0	**15618**	0	243,060	**15618**	0
PR439A	439	105960	109424	3,269	106470,3	0,482	1183,262	105960	0	105569,8	-0,370	292,180	**104846**	-1,050

(continuação na próxima página)

Tabela 5.7: Resultados completos (Instâncias Classe 1).

			BC10 RBO00/02		LNS2017					HGS-LN				
Instâncias	n	f_{msc}	f_{min}	E_{min}	f_{med}	E_{med}	t**(s)	f_{min}	E_{min}	f_{med}	E_{med}	t(s)	f_{min}	E_{min}
PR439B	439	110241	115580	4,843	110866,9	0,568	1174,335	110241	0	110049,5	-0,170	384,070	**109979**	-0,240
PR439C	439	113514	113514	0	115747,5	1,968	1296,311	113681	0,147	109283,6	-3,730	361,690	**109237**	-3,770
PCB441A	441	60259	61289	1,709	60709,8	0,748	1314,718	60259	0	59257,2	-1,660	563,470	**59141**	-1,860
PCB441B	441	59465	61691	3,743	60386,8	1,550	1327,042	59465	0	59048,1	-0,700	503,490	**58886**	-0,970
PCB441C	441	59838	61984	3,586	61023,1	1,981	1501,158	59838	0	59193,8	-1,080	769,870	**58897**	-1,570
D493A	493	35899	37725	5,086	36217,7	0,888	1540,658	35899	0	35616,4	-0,790	602,190	**35514**	-1,070
D493B	493	36951	37806	2,314	37129,7	0,484	1573,996	36951	0	36215,4	-1,990	620,220	**36121**	-2,250
D493C	493	37895	38794	2,372	38328,8	1,145	1701,186	37895	0	37162,3	-1,930	835,710	**36908**	-2,600
Média		47914,38	48323,87	0,780	48146,22	0,428	380,359	47924,44	0,012	47710,66	-0,296	102,749	47689	-0,335

*Valores ótimos provados.

**Valores de tempo ajustados.

5.3.1
Contribuição de vizinhanças e tempo computacional

Com o propósito de entender a contribuição de cada uma das seis vizinhanças apresentadas para o HGS-LN, alguns experimentos foram executados removendo uma vizinhança por vez. Para a comparação do impacto da remoção de cada vizinhança na qualidade e tempo computacional, a Tabela 5.8 apresenta o erro médio (E_{med}) e o erro mínimo (E_{min}) em relação às melhores soluções conhecidas.

Embora todas as vizinhanças tenham contribuição para encontrar os melhores resultados, de forma geral, o algoritmo se mostra estável e não depende exclusivamente de alguma vizinhança específica para encontrar valores melhores que os mostrados anteriormente na literatura. Porém, a remoção da vizinhança Reposiciona Par ou Or-Opt ocasiona uma variação no erro médio que não é desprezível. O aumento no tempo computacional necessário ao remover estas vizinhanças se deve ao fato que durante a busca local uma maior quantidade de movimentos avaliados em $\mathcal{O}(n^2)$ são performados.

Tabela 5.8: Impacto das vizinhanças.

Configuração	E_{med}	E_{min}	t(s)
referência	-0,30	-0,34	102,75
- Reposiciona Par	-0,25	-0,33	138,46
- 2-Opt	-0,29	-0,33	98,41
- Or-Opt	-0,25	-0,32	170,42
- 2k-Opt	-0,29	-0,33	74,48
- 4-Opt	-0,28	-0,33	80,97
- B&S	-0,29	-0,33	87,86

A análise de crescimento do tempo computacional prático foi observada nas instâncias de Classe 1 por uma regressão da forma $f(n) = an^b$. Na Figura 5.2 o gráfico mostra o comportamento do tempo em segundos em função da quantidade n de vértices na instância. O tempo de execução em segundos pode ser estimado como $0,00070n^{2,16311}$, sendo o coeficiente b próximo a 2 mostrando um crescimento prático próximo ao comportamento teórico.

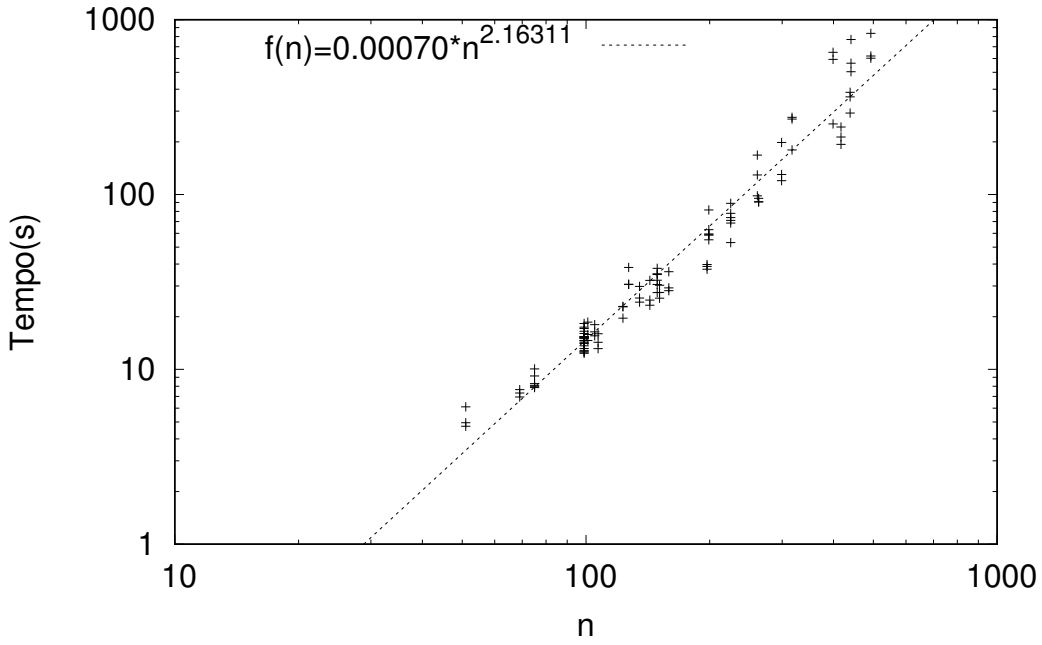


Figura 5.2: Crescimento prático do tempo de processamento em função do número de vértices nas instâncias.

6
Conclusões

Neste trabalho, tratamos o problema do caixeiro viajante com coleta e entrega, no qual o caminho de custo mínimo deve ser obtido de forma que, partindo do depósito, todas as localidades sejam visitadas sem violar a restrição de precedência e, então, retorna-se ao depósito.

Para resolver este problema, um algoritmo genético híbrido foi desenvolvido combinando busca local em vizinhanças eficientes e controles avançados de diversidade. O método proposto mostrou grande capacidade de escapar de mínimos locais e uma busca local eficiente alcançou resultados de alta qualidade. Este método superou todos os trabalhos anteriores em termo de qualidade das soluções e tempo computacional.

A eficiência do algoritmo proposto se apoia na habilidade de escapar de mínimos locais do HGS somada a um conjunto de vizinhanças largas que podem modificar significantemente a solução. Os componentes de mutação e *crossover* foram projetados para além de preservar características dos ascendentes, adicionar diversidade ao descendente com critérios que podem ajudar a escapar de mínimos locais e adicionar boas arestas à solução. O conjunto de vizinhanças também se mostrou robusto de forma que o potencial de guiar a soluções de boa qualidade não é vitalmente dependente de alguma vizinhança específica.

Como trabalhos futuros a utilização de memória pode trazer um considerável ganho em tempo computacional e algumas vizinhanças podem ter a sua cobertura de movimentos aumentada. A vizinhança 4-Opt pode aproveitar de estruturas de dados mais avançadas para completar a busca, contemplando os movimentos não cobertos pela natureza da busca atual. Or-Opt também mostra potencial de melhorias onde a utilização de programação dinâmica torna possível uma avaliação eficiente de toda a vizinhança, independente do parâmetro *or-k*. Além disto, trabalhos futuros incluem avaliar a eficiência das vizinhanças largas propostas em outros problemas de roteamento.

7
Referências bibliográficas

[1] LENSTRA, J. K.; DESROCHES, M.; SAVELBERGH, M. W.P.; SOUMIS, F. **Vehicle routing with time windows: optimization and approximation**. Vehicle routing: Methods and studies, p. 65–84, 1988.

[2] CORDEAU, J.-F.; LAPORTE, G. **The dial-a-ride problem (darp): Variants, modeling issues and algorithms**. 4OR: A Quarterly Journal of Operations Research, 1(2):89–101, 2003.

[3] CORDEAU, J.-F.; LAPORTE, G.; POTVIN, J.-Y.; SAVELSBERGH, M. W.P. **Transportation on demand**. Handbooks in operations research and management science, 14:429–466, 2007.

[4] BERBEGLIA, G.; CORDEAU, J.-F.; GRIBKOVSKAIA, I.; LAPORTE, G. **Static pickup and delivery problems: a classification scheme and survey**. Top, 15(1):1–31, 2007.

[5] PARRAGH, S. N.; DOERNER, K. F.; HARTL, R. F. **A survey on pickup and delivery models part ii: Transportation between pickup and delivery locations**. Journal für Betriebswirtschaft, 58(2):81–117, 2008.

[6] PARRAGH, S. N.; DOERNER, K. F.; HARTL, R. F. **A survey on pickup and delivery problems**. Journal für Betriebswirtschaft, 58(1):21–51, 2008.

[7] VEENSTRA, M.; ROODBERGEN, K. J.; VIS, I. F. A.; COELHO, L. C. **The pickup and delivery traveling salesman problem with handling costs**. European Journal of Operational Research, 257(1):118–132, 2017.

[8] DUMITRESCU, I.; ROPKE, S.; CORDEAU, J.-F.; LAPORTE, G. **The traveling salesman problem with pickup and delivery: polyhedral results and a branch-and-cut algorithm**. Mathematical Programming, 121(2):269, 2010.

[9] RENAUD, J.; BOCTOR, F. F.; OUENNICHE, J. **A heuristic for the pickup and delivery traveling salesman problem**. Computers & Operations Research, 27(9):905–916, 2000.

[10] RENAUD, J.; BOCTOR, F. F.; LAPORTE, G. **Perturbation heuristics for the pickup and delivery traveling salesman problem**. Computers & Operations Research, 29(9):1129–1141, 2002.

[11] LOKIN, F.C.J. **Procedures for travelling salesman problems with additional constraints**. European Journal of Operational Research, 3(2):135–141, 1979.

[12] KALANTARI, B.; HILL, A. V.; ARORA, S. R. **An algorithm for the traveling salesman problem with pickup and delivery customers**. European Journal of Operational Research, 22(3):377–386, 1985.

[13] SAVELSBERGH, M. W.P.; SOL, M. **The general pickup and delivery problem**. Transportation science, 29(1):17–29, 1995.

[14] REINELT, G. **Tsplib - a traveling salesman problem library**. ORSA Journal on Computing, 3(4):376–384, 1991.

[15] REINELT, G. **Optimal solutions for symmetric tsps**, 2018. Acesso em: Fevereiro de 2018.

[16] APPLEGATE, D. L.; BIXBY, R. E.; CHVÁTAL, V.; COOK, W.; ESPINOZA, D. G.; GOYCOOLEA, M.; HELSGAUN, K. **Certification of an optimal tsp tour through 85,900 cities**. Operations Research Letters, 37(1):11–15, 2009.

[17] LITTLE, J. D.C.; MURTY, K. G.; SWEENEY, D. W.; KAREL, C. **An algorithm for the traveling salesman problem**. Operations research, 11(6):972–989, 1963.

[18] RULAND, K.S.; RODIN, E.Y. **The pickup and delivery problem: Faces and branch-and-cut algorithm**. Computers & mathematics with applications, 33(12):1–13, 1997.

[19] GLOVER, F. W.; KOCHENBERGER, G. A. **Handbook of metaheuristics**, volumen 57. Springer Science & Business Media, 2006.

[20] PAPADIMITRIOU, C. H.;STEIGLITZ, K. **Combinatorial optimization: algorithms and complexity**. Courier Corporation, 1998.

[21] ROPKE, S.; PISINGER, D. **An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows**. Transportation science, 40(4):455–472, 2006.

[22] CARRABS, F.; CORDEAU, J.-F.; LAPORTE, G. **Variable neighborhood search for the pickup and delivery traveling salesman problem with lifo loading**. INFORMS Journal on Computing, 19(4):618–632, 2007.

[23] LI, H.; LIM, A. **A metaheuristic for the pickup and delivery problem with time windows**. International Journal on Artificial Intelligence Tools, 12(02):173–186, 2003.

[24] BERGE, C. **Graphs and Hypergraphs**. Elsevier Science Ltd., Oxford, UK, UK, 1985.

[25] GLOVER, F. **Ejection chains, reference structures and alternating path methods for traveling salesman problems**. Verão reduzida publicada em Discrete Applied Mathematics (1996), (65):223–253, 1992.

[26] GLOVER, F. **Finding a best traveling salesman 4-opt move in the same time as a best 2-opt move**. Journal of Heuristics, 2(2):169–179, 1996.

[27] OR, Í. **Traveling salesman-type combinatorial problems and their relation to the logistics of regional blood banking**. PhD thesis, Northwestern University, 1976.

[28] BALAS, E.; SIMONETTI, N. **Linear time dynamic-programming algorithms for new classes of restricted tsps: A computational study**. Journal on Computing, 13(1):56–75, 2001.

[29] BALAS, E. **New classes of efficiently solvable generalized traveling salesman problems**. Annals of Operations Research, 86:529–558, 1999.

[30] VIDAL, T.; CRAINIC, T. G.; GENDREAU, M.; LAHRICHI, N.; REI, W. **A hybrid genetic algorithm for multidepot and periodic vehicle routing problems**. Operations Research, 60(3):611–624, 2012.

[31] VIDAL, T.; CRAINIC, T. G.; GENDREAU, M.; PRINS, C. **A hybrid genetic algorithm with adaptive diversity management for a large class of vehicle routing problems with time-windows**. Computers & operations research, 40(1):475–489, 2013.

[32] VIDAL, T.; CRAINIC, T. G.; GENDREAU, M.; PRINS, C. **A unified solution framework for multi-attribute vehicle routing problems**. European Journal of Operational Research, 234(3):658–673, 2014.

[33] CAMPOS, V.; LAGUNA, M.; MARTÍ, R. **Context-independent scatter and tabu search for permutation problems**. Journal on Computing, 17(1):111–122, 2005.

[34] PRINS, C. **A simple and effective evolutionary algorithm for the vehicle routing problem**. Computers & Operations Research, 31(12):1985–2002, 2004.

[35] HERNÁNDEZ-PÉREZ, H.; RODRÍGUEZ-MARTÍN, I.; SALAZAR-GONZÁLEZ, J.-J. **A hybrid heuristic approach for the multi-commodity pickup-and-delivery traveling salesman problem**. European Journal of Operational Research, 251(1):44–52, 2016.

[36] PADBERG, M.; RINALDI, G. **A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems**. SIAM review, 33(1):60–100, 1991.

[37] CZITROM, V. **One-factor-at-a-time versus designed experiments**. The American Statistician, 53(2):126–131, 1999.

[38] CPUBENCHMARK.NET. **Passmark - cpu mark, single thread performance**, 2018. Acesso em: Fevereiro de 2018.