Odivaney Ramos Pedro

Uma abordagem de Busca Tabu para o Problema do Caixeiro Viajante com Coleta de Prêmios

Minas Gerais
2013

Odivaney Ramos Pedro

Uma abordagem de Busca Tabu para o Problema do Caixeiro Viajante com Coleta de Prêmios

Dissertação apresentada ao Departamento de Engenharia Elétrica da Universidade Federal de Minas Gerais, para a obtenção de Título de Mestre em Engenharia Elétrica, na Área de Otimização.

Orientador: Rodney Rezende Saldanha

Co-Orientador: Ricardo Saraiva de Camargo

Minas Gerais
2013

Aluno, Odivaney Ramos Pedro.
Uma abordagem de Busca Tabu para o Problema do Caixeiro Viajante com Coleta de Prêmios
109 páginas
Dissertação (Mestrado) - Universidade Federal de Minas Gerais. Departamento de Engenharia Elétrica.

1. Busca Tabu
2. PCVCP

Universidade Federal de Minas Gerais. Departamento de Engenharia Elétrica.

Comissão Julgadora:

Prof. Dr. Alexandre Xavier Martins

Prof. Dr. Lucas de Souza Batista

Prof. Dr. Thiago Ferreira de Noronha

Prof. Dr. Ricardo Saraiva de Camargo
Co-Orientador

Prof. Dr. Rodney Rezende Saldanha
Orientador

Dedico este trabalho à Deus e à meus pais, Odilon e Vanda.

Provérbios de Salomão

O coração do homem pode fazer planos,
mas a resposta certa vem dos lábios do Senhor.
Todos os caminhos do homem são puros aos seus olhos,
mas o Senhor pesa o espírito.
Confia ao Senhor as tuas obras,
e os teus desígnios serão estabelecidos.
Quanto melhor é adquirir a sabedoria do que o ouro!
E mais excelente, adquirir a prudência do que a prata.
O que atenta para o ensino acha o bem,
e o que confia no Senhor este é feliz!
O sábio de coração é chamado de prudente,
e a doçura no falar aumenta o saber.
O entendimento para aqueles que o possuem, é fonte de vida.
O coração do sábio é mestre de sua boca e aumenta a persuasão nos seus lábios.
Coroas de honras são as cãs,
quando se acham no caminho da justiça.
A sorte se lança no regaço, mas do Senhor procede toda decisão.

Provérbios, Capítulo 16, Bíblia Sagrada.

Agradecimentos

Agradeço muito a Deus por toda a misericórdia e amor que ele sempre teve comigo, por ter sido o meu sustento em todos os momentos da minha vida e por ter planejado este momento desde quando eu ainda nem havia sido formado no ventre de minha mãe.

Agradeço aos meus pais Odilon e Vanda, por terem me ensinado o que de mais valioso aprendi, os meus valores e princípios, por serem meus grandes amigos e protetores, e por todo amor e apoio que me deram. Em especial agradeço a minha mãe pelos calos em seus joelhos. Também agradeço ao restante de minha família por me apoiar e sonhar comigo.

Agradeço a minha namorada, Gecirlene, por todo amor, carinho e cuidado dado a mim neste período da minha vida, por ter sido mais que uma namorada, por ter sido minha melhor amiga, e por ter sonhado junto comigo a todo momento.

Ao Prof. Rodney Rezende agradeço por ter aceitado o meu pedido de orientação, pela paciência que teve comigo ao longo deste período, por todo conhecimento que me transmitiu e pelo exemplo que se tornou.

Agradeço ao Prof. Ricardo Camargo pelo apoio acadêmico, orientação e acompanhamento desde a minha graduação, e também pelos ensinamentos dados no intuito de melhorar o homem que excede o aluno, foram muitas as lições aprendidas com o Professor e com o Homem.

Agradeço também aos meus amigos, em especial a Bruno Nonato e a Theo Lins, grandes companheiros e parceiros acadêmicos que colaboraram diretamente para esta conquista.

Agradeço também a Liquigás pelo suporte financeiro no período do mestrado, e por ter me permitido fazer grandes amizades: Dyerre, Iran, Nilza, Rafael, Wellington entre outros.

Enfim agradeço a Comissão Julgadora pela disponibilidade em avaliar este trabalho.

Resumo

O objetivo principal deste trabalho é o desenvolvimento de um procedimento heurístico para resolução do Problema do Caixeiro Viajante com Coleta de Prêmios (PCVCP), o procedimento aqui proposto tem como base principal o método Busca Tabu. Devido a complexidade de resolução do PCVCP de forma exata optou-se pelo desenvolvimento de uma abordagem heurística para sua resolução. Além do método Busca Tabu, utilizado como método de busca local no algoritmo proposto, são utilizadas outras duas heurísticas, o GENIUS e as heurísticas K-Optimal. A primeira é empregada para geração de uma solução inicial e a segunda para refinamento de soluções geradas pelo procedimento de busca local. O método Busca Tabu (TS) utilizado neste trabalho passou por algumas adaptações de forma a melhor lidar com as peculiaridades do PCVCP, tais como a não necessidade de visitação de todas as cidades pelo caixeiro viajante. O TS armazena informações sobre as soluções geradas durante o processo de busca e estas informações são usadas em estratégias de diversificação e intensificação de soluções, assim como para formação de uma lista com movimentos tabus. O algoritmo proposto foi testado em problemas-teste disponíveis na literatura, apresentando na média boas soluções para o PCVCP. Dentre os resultados alcançados destacaram-se os relativos a instâncias com maior quantidade de cidades, que são as de maior complexidade de resolução. Para estas instâncias na maioria dos casos o algoritmo obteve os melhores resultados reportados na literatura para o problema.

Palavras-chave: PCVCP, Busca Tabu, Genius, K-Optimal

Abstract

The main objective of this work is the development of a heuristic procedure to solve the Price Collecting Travelling Salesman Problem (PCTSP), this procedure is based mainly on the method Tabu Search. Due to the complexity of solving the PCTSP in the exact way was chosen a development of a heuristic approach to their resolution. Beyond the Tabu Search method, used as a method of local search in proposed algorithm, two other heuristics are used, the GENIUS and the K-Optimal heuristics. The first is employed to construction of initial solutions, and the second to the refinement of generated solutions by the local search procedure. The Tabu Search used in this work went through some adaptations, in order to cope better with the peculiarities of the PCTSP, such as the non necessity of visitation of all cities by the traveling salesman. This method stores informations about the generated solutions during the search process, and this informations is used by strategies of diversification and intensification of solutions, so as to form a list of movements "taboos". The proposed algorithm is tested in test-problems available in the literature, and showed the capacity of generate good solutions to PCTSP. From among the results achieved stands out the results obtained to instances with larger amount of cities, which are of greater complexity of solving. To this instances, the proposed algorithm obtained the best results reported in literature.

Keywords: PCTSP, Tabu Search, Genius, K-Optimal

Lista de Figuras

Lista de Tabelas

Sumário

Capítulo 1

INTRODUÇÃO

Este trabalho tem como objetivo principal o desenvolvimento de técnicas heurísticas que buscam a resolução do PCVCP (Problema do Caixeiro Viajante com Coleta de Prêmios). Problema este que pertence à classe de problemas provenientes do largamente conhecido e estudado TSP (Travelling Salesman Problem)(Nils, 1982).

O PCVCP, do inglês Prize Collector Travelling Salesman Problem (PCTSP), pode ser associado a um caixeiro viajante que recebe um prêmio positivo P_k, por cada cidade K que ele visita, e paga uma penalidade γ_k por cada cidade não visitada, sendo que para realizar o deslocamento entre as cidades o caixeiro tem um custo de deslocamento C_{ij}. O problema consiste em minimizar o somatório do custos de deslocamentos mais as penalidades pagas por não visitar cidades. Contudo o percurso do caixeiro tem que ser definido de forma que seja incluída em sua rota uma quantidade suficiente de cidades que permitam o recolhimento de uma quantia mínima de prêmios P_{min}.

O PCVCP foi inicialmente definido por Balas and Martin (1984) com o intuito de modelar a programação diária de um laminador de aço. Laminação de aços é o processo onde placas de aço são transformadas em lâminas. A laminação é a compressão destas placas por rolos laminadores. Como entradas para o processo de laminação existem diversas placas de aço e diversos pedidos de lâminas, sendo que cada um dos pedidos

de lâminas possui uma especificação de tamanho, comprimento e espessura da lâmina. Baseado nas características físicas e dimensões, as placas são agrupadas em células. A saída deste processo é o inventário de células.

Devido ao desgaste do processo de laminação os rolos precisam ser trocados regularmente. O tempo de troca entre rolos varia de acordo com algumas características físicas (rigidez, flexibilidade, dureza e outras) do aço laminado. Os materiais a serem laminados são classificados de acordo com as suas características, e o ideal é que após as trocas de rolos sejam laminados materiais de mesma classificação. Da mesma forma que os aços a serem laminados, os rolos também variam entre si de acordo com suas propriedades físicas, permitindo a especificação de quais tipos de rolos podem ser utilizados para laminar determinadas células.

A tarefa de programação (escalonamento de tarefas) consiste em definir, a partir dos rolos que serão utilizados, quais células serão laminadas, e qual a ordem de processamento das células. A ordem de processamento interfere na qualidade do aço laminado e na eficiência do processo. Além destes fatores os pedidos de lâminas tem um tempo máximo para serem entregues, o que também deve ser levado em conta na programação do laminador. Outro ponto influenciado pelo sequenciamento de laminação é o desgaste dos rolos, visto que determinadas células devem ser laminadas por rolos com baixo nível de desgaste.

A importância do estudo do PCVCP é justificada por sua similaridade com problemas enfrentados no cotidiano de muitas empresas, principalmente das empresas que são das áreas de logística e distribuição, como transportadoras de cargas, e das empresas cujo sucesso de seus negócios depende da eficiência destas áreas, tais como distribuidoras em geral.

Como exemplo prático de problemas do cotidiano destas empresas, pode ser citada a entrega de um produto qualquer, por um distribuidor a seus revendedores. Há casos em que a demanda do produto por parte dos revendedores é maior que a capacidade de

entrega do veículo do distribuidor. Neste caso para cada revendedor que não receber o produto, o distribuidor paga uma multa contratual, de valor variável de acordo com o contrato firmado com cada revendedor.

A definição da rota a ser percorrida por um veículo de entregas deve procurar a minimização do custo de deslocamento (C_{ij}) entre os revendedores que receberão as entregas e do custo relativo às multas pagas pelas entregas não realizadas (γ_k). A empresa ainda pode definir que todo veículo, usado para distribuição, tenha uma taxa de utilização acima de um determinado percentual (prêmio mínimo) (ex: todo caminhão de entregas deve sair com no mínimo 90 % da sua capacidade total).

Outro exemplo são situações enfrentadas pelas transportadoras de cargas que prestam serviços de frete para produtores. Em momentos de pico de consumo dos produtos (ex: consumo de bebidas em véspera de datas festivas) a capacidade total de entrega da transportadora, pode ser menor que a demanda total de pedidos de frete por parte dos produtores. Devido a detalhes da relação comercial entre transportador e produtor (tempo de relacionamento, potencial de produção, formas de pagamentos e outros) cada produtor pode pagar valores diferenciados pelo serviço de frete. Neste tipo de situação a penalidade, considerada pelo transportador, pode ser o número de pedidos rejeitados. Nesse caso a minimização da penalidade resultaria em uma maior quantidade de entregas realizadas.

Assim sendo deve se definir uma rota que minimize o número de pedidos rejeitados (γ_i) e o custo de deslocamento entre as entregas (C_{ij}). A transportadora pode definir ainda uma receita mínima (P_{min}) para que uma rota seja realizada, ou seja o somatório dos valores recebidos pelas entregas realizadas deve ser maior que esse valor mínimo. Esse tipo de restrição evita que a transportadora realize rotas economicamente inviáveis.

O PCVCP tem alto custo computacional de resolução, principalmente devido ao seu elevado número de soluções possíveis. Assim como será mostrado no próximo capítulo, considerando N como o número total de cidades, e assumindo que a distância

entre qualquer cidade i e qualquer cidade j seja simétrica, isto é, que D_{ij}=D_{ji}, o número total de soluções possíveis é $(N-1)!/2$, sendo classificado na literatura como NP-difícil, ou seja, não sao conhecidos algoritmos que resolvam o problema em tempo polinomial. Apesar dos avanços tecnológicos, como desenvolvimento de processadores mais eficientes, para valores elevados de N é inviável enumerar todas as soluções. Em alguns casos uma resposta viável deve ser gerada em um curto espaço de tempo, como no caso de uma transportadora que frente a uma alta demanda de serviços para realizar não pode aguardar a certeza de ter encontrado a solução ótima. A complexidade do problema pode inviabilizar a geração de soluções viáveis em tempo hábil. Mesmo representando diversos problemas, são relativamente poucos os trabalhos na literatura relacionados ao estudo da resolução do PCVCP, se comparado a outros problemas provenientes do PCV. Isso indicia a possibilidade de avançar a fronteira do conhecimento para o problema através de seu estudo detalhado.

Este trabalho tem como objetivo contribuir com o desenvolvimento de um método Busca Tabu específico para resolver o PCVCP. Além deste método também se faz uso de heurísticas de refinamento para aprimorar as soluções geradas e de um método específico para construir a solução inicial do método Busca Tabu aqui proposto.

A meta-heurística principal utilizada neste trabalho é a Busca Tabu. O TS é um método de busca local proposto por Glover (1989). Uma das principais características desse método é que ele busca o ótimo global permitindo movimentos de piora. Dessa forma, o TS abre espaço no conjunto de soluções de forma a percorrer o espaço de busca de maneira mais abrangente. O método faz isto armazenando a melhor solução encontrada até o momento, e utilizando uma lista de movimentos tabu. Os movimentos desta lista ficam bloqueados, não podendo ser novamente realizados por algumas iterações, a não ser que este movimento satisfaça algum critério pré-determinado (ex: geração da melhor solução encontrada). A utilização desta lista evita a ciclagem, a forma como a ciclagem é evitada será explicada mais adiante neste trabalho. A Figura 1.1 demonstra

o que são ótimos globais e ótimos locais. É possível visualizar na figura que caso um método de busca tente escapar de um ótimo local, em alguns casos isso só é possível através da exploração de regiões não promissoras.

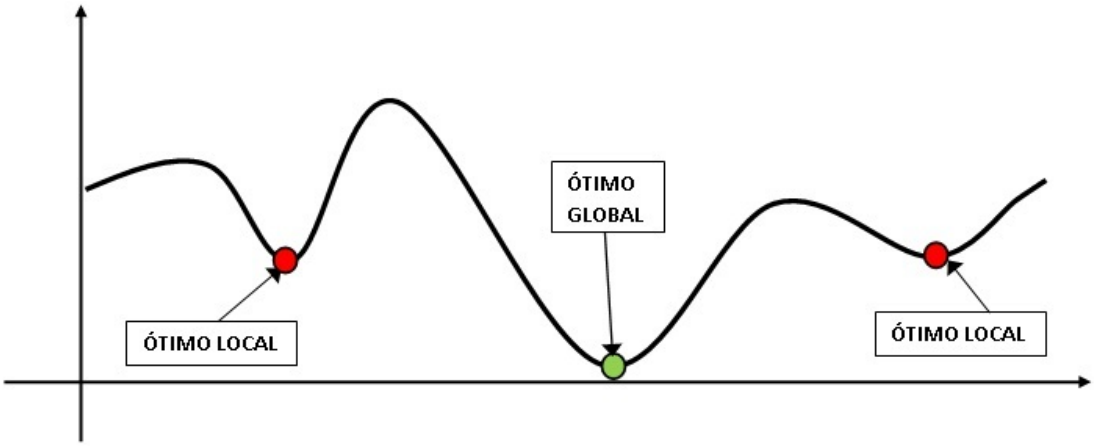

Figura 1.1: Representação de Ótimos.

No período de desenvolvimento optou-se inicialmente por utilizar métodos gulosos e semi-gulosos para construção da solução inicial. Devido a baixa qualidade das soluções iniciais geradas, e a influência destas no desempenho do método de busca utilizado, o estudo de um melhor método de construção foi aprofundado. Baseado na literatura do PCV, empregou-se o método GENIUS (Gendreau et al., 1992) para resolver o PCVCP, apresentando melhoras significativas nos resultados obtidos. Esta heurística inicialmente foi desenvolvida para resolver o problema do PCV, sendo composta de uma fase construtiva GENI (Generalized Insertion), na qual os nós vão sendo acrescentados gradativamente à solução, e outra fase de refinamento US (Unstringing and Stringing), onde a retirada desses nós da solução gerada é avaliada. Ambas as fases são melhor explicadas mais adiante neste trabalho.

Os métodos de busca local do tipo K-optimal (Steiglitz and Weiner, 1968) foram utilizados como estratégias de melhoria das soluções geradas após os movimentos TS. Estas heurísticas tem a vantagem de sua fácil implementação. Ao aplicar-se uma heu-

rística do tipo k-optimal em uma solução, todas as trocas entre K arcos são avaliadas e, caso alguma troca gere melhorias no valor da função objetivo associada a solução, a troca entre os K arcos é realizada. A medida que maiores valores de K são utilizados o espaço de soluções é percorrido de forma mais completa, fazendo com que o custo computacional cresça de forma acentuada, visto a complexidade deste algoritmo $O_{(nk)}$ (Goldbarg and Luna, 2000), onde n é o número de vértices. Neste trabalho foi utilizado valores de K igual a 2 e a 3, $K = 2$ entre as iterações da busca Tabu e $K = 3$ como método de refinamento da solução final, isto será detalhado mais adiante neste trabalho. A combinação dos métodos acima descritos se mostrarão interessantes para a resolução do PCVCP. Para avaliar a eficiência do método proposto foram utilizados os melhores resultados conhecidos, a partir da literatura do problema, e um software de modelagem matemática para resolução exata do problema.

Para instâncias de grande porte, as quais são inviáveis de serem resolvidas, os resultados obtidos pelo método proposto superam os demais resultados obtidos na literatura do problema, isto será mostrado na sequência deste trabalho. O método também se mostrou versátil para solucionar instâncias de menor porte, chegando ao valor ótimo das mesmas na maioria das vezes, mantendo sempre pequenos valores de desvios médios em relação ao menor valor encontrado. Os resultados são melhor detalhados no capítulo 5.

Além da comparação dos resultados do algoritmo proposto com os resultados de outros métodos da literatura, este trabalho tambem aborda a influência de alguns parâmetros no desempenho da Busca Tabu proposta. Este trabalho também aborda a utilização do método 3-OPT ao fim do processo de busca tabu. No decorrer deste trabalho são mostrados de forma detalhada os resultados obtidos pela aplicação deste método para a resolução do PCVCP.

Este trabalho está dividido em seis seções, iniciando-se por esta introdução. No capítulo 2 é realizada uma revisão da literatura caracterizando o PCVCP, sua origem

e problemas correlatos. São revistos também os principais métodos de resolução para o problema, e seus correlatos, e aplicações práticas para o mesmo. Na capítulo 3 são apresentadas as metodologias utilizadas para resolver o PCVCP neste e em outros trabalhos, e também são apresentadas as justificativas para a escolha da metodologia utilizada. No capítulo 4 os algoritmos utilizados neste trabalho são apresentados de forma detalhada, e os algoritmos utilizados pelos outros trabalhos da literatura aqui comparados. No capítulo 5, os resultados obtidos e sua análise são apresentados. No capítulo 6 é feita uma avaliação geral de tudo que foi feito, bem como são apresentados às contribuições obtidas por este trabalho e possíveis trabalhos futuros.

Capítulo 2

EXAME DA LITERATURA

Este capítulo aborda o referencial teórico que foi utilizado como base para o desenvolvimento deste trabalho. O capítulo é dividido em três partes: a primeira parte apresenta o PCV (problema que originou o PCVCP) e alguns de seus provenientes relacionados ao problema em questão. Em seguida são apresentados os principais métodos exatos e heurísticos já utilizados para resolver o problema em questão. Por fim na terceira parte é abordado com mais detalhes o PCVCP através de suas referências na literatura, sua modelagem matemática e suas aplicações práticas.

2.1 O PROBLEMA DO CAIXEIRO VIAJANTE

O Problema do Caixeiro Viajante PCV (Nils, 1982) é um dos problemas de otimização combinatória mais estudados da literatura, devido a sua aplicação prática e grande complexidade de resolução. Sua origem, assim como dito por Dantzig et al. (1954) não é completamente certa. O nome *Caixeiro Viajante* surgiu pela primeira vez em um livro publicado na Alemanha no ano de 1832 (***The Traveling Salesman, how he should be and what he should do to get the comissions and to be successful in his business. By an veteran Traveling Salesman***). Anos mais tarde o jogo

Around the World, proposto por Willian Rowan Hamilton, feito sobre um dodecaedro em que cada vértice estava associado a uma cidade, o qual tinha como desafio a definição de uma sequência incluindo todos os vértices com um custo mínimo e passando apenas uma vez por cada vértice, já apresentava uma estrutura muito similar ao PCV. Devido ao nome do autor do jogo mais tarde seria feita a definição de **ciclo hamiltoniano**. Modernamente credita-se a Hassler Whitney a primeira menção do problema, em um trabalho realizado na Princeton University. A definição do problema consiste basicamente em:

> ***Dado um conjunto de N cidades e a matriz de distâncias entre elas, determinar uma sequência de visitas que passe por todas as cidades, e somente uma vez em cada uma delas, e volte à origem (ciclo hamiltoniano) de maneira que alguma função de custo seja minizada.***

De acordo com Lawler et al. (1985) o PCV pode ser representado através da estrutura de um grafo completo e ponderado. Completo pelo fato de que a partir de um vértice é possível chegar diretamente a qualquer outro. E ponderado devido ao fato de que são associados custos as arestas ou aos vértices. A Figura 2.1 mostra dois grafos ponderados (com uso de peso associados às arestas), sendo que o primeiro grafo é completo e o segundo incompleto, pois os vértices 2 e 3 não são ligados diretamente. De acordo com a definição acima do PCV, a partir do grafo completo mostrado na figura a seguir, podemos definir uma sequência ótima de visitação como sendo (1,4,2,3), ou ainda esta mesma sequência mas em uma ordem de visitação inversa (3,2,4,1).

Dado um grafo completo e ponderado $G' = (V,E)$, sem perda de generalidade para representar o PCV pode-se associar os vértices às cidades a serem visitadas, e as arestas aos custos de deslocamento entre as cidades. A Figura 2.2 contém um grafo representando um problema do caixeiro viajante com 3 cidades, com o custo de deslocamento 3 entre as cidades A e C.

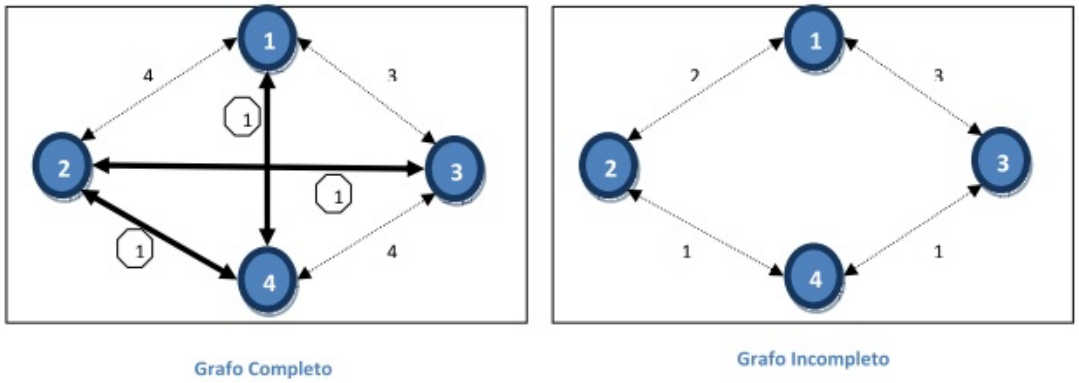

Figura 2.1: Representação de grafo completo e grafo incompleto.

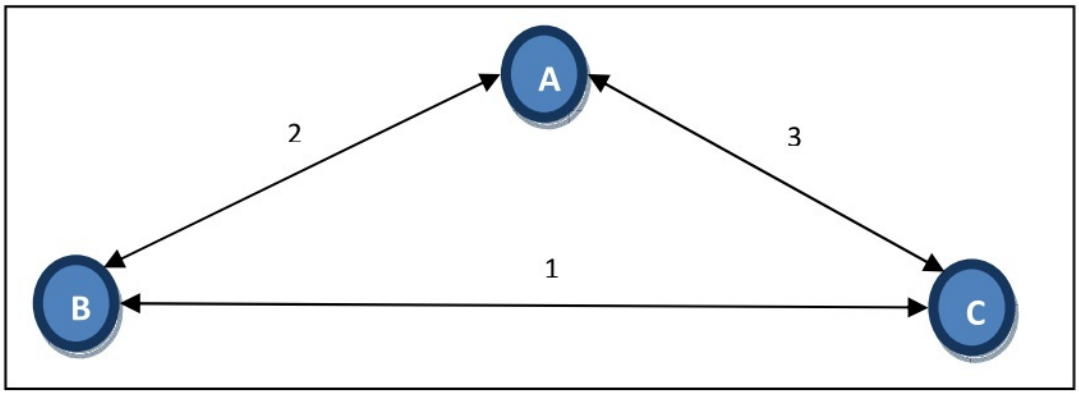

Figura 2.2: Exemplo de PVC simétrico com 3 cidades.

A Figura 2.2 mostra um caso específico do PCV, o simétrico, no qual o custo de deslocamento de qualquer vértice i para qualquer vértice j tem o mesmo valor de deslocamento de j para i. Existe ainda a variação assimétrica do PCV, na qual o custo de deslocamento entre duas cidades varia de acordo com o sentido, assim como mostrado na Figura 2.3.Na figura em questão podemos notar que o custo de deslocamento entre A e C, partindo de A, é 3, mas partindo de C é 2.

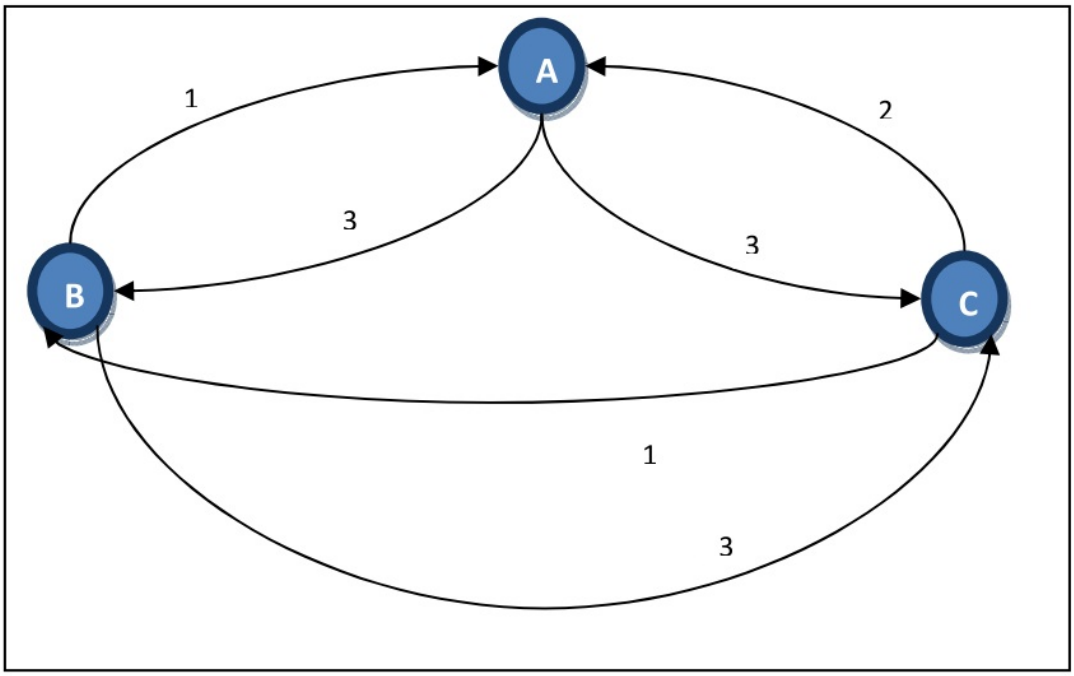

Figura 2.3: Exemplo de PVC assimétrico com 3 cidades.

Como pode ser visto na Figura 2.4, que utiliza os dados apresentados na Figura 2.3 como base, uma mesma sequência de cidades se visitada em sentido inverso pode resultar em um valor diferente de custos no caso do PCV assimétrico.

Na Figura 2.4 é possível notar que, ao contrário do caixeiro viajante simétrico, a ordem de visitação é determinante na avaliação de custo, sendo que uma mesma solução se visitada em ordem inversa pode resultar em valores de função objetivo completamente diferentes, como pode ser visto pela comparação de custos da rota (1) com a rota (2).

Figura 2.4: Cálculo de custo do PCV assimétrico.

Este fato consequentemente resulta em um maior espaço de soluções para ser avaliado na resolução do problema assimétrico, e um consequente maior custo de resolução. O problema assimétrico do PCV pode ser representado pelo uso de grafos direcionados, enquanto o simétrico através do uso de grafos não direcionados.

Dado a sua aplicação e semelhança com diversos problemas do cotidiano diversas generalizações do *PCV* foram apresentadas na literatura, normalmente considerando um segundo objetivo associado à sequência de visitas e/ou acrescentando alguma restrição específica. Entre as generalizações do PCV que são problemas similares ao PCVCP, podemos citar:

- PCVS-C: O problema do caixeiro viajante simétrico com agrupamento (PCVS-C) é um caso especial de PCV simétrico, em que existem agrupamentos que possuem restrições que os obrigam a estar em uma determinada sequência de atendimento (Goldbarg and Luna, 2000).

- PCVG: proposto por Ong (1982), o PCV Generalizado é semelhante ao PCVS-C, mas cada grupamento formado deve contribuir com certo número de nós para o ciclo hamiltoniano. A versão "equality"exige que apenas um vértice em cada cluster seja visitado. Ou seja, deseja-se encontrar a menor rota parcial através de pelo menos um vértice em cada agrupamento (Goldbarg and Luna, 2000).

- PCVB: o termo Backhauls significa operação de coleta com retorno. O PCV Backhauls pode ser considerado um caso especial de PCVG onde os nós de G são particionados em dois grupamentos denominados normalmente de L (nós li-

nehauls) e B (nós backhauls). A matriz de custo do problema atende as condições da norma Euclidiana. Uma versão do problema determina que os nós de L sejam visitados inicialmente e, posteriormente, os nós de B. A estratégia dessa versão é dar preferência ao descarregamento dos veículos para, posteriormente, realizar o carregamento em direção ao depósito (nó inicial) (Gendreau et al., 1996).

- PCV-B: o PCV com bônus foi proposto por Fischetti and Toth (1988), o PCV-B associa um bônus a cada nó do grafo e procura uma rota hamiltoniana de menor comprimento dado a condição restritiva, que consiste na obtenção de um bônus total maior ou igual a certo valor mínimo (Goldbarg and Luna, 2000).

- PCV-S: o PCV Seletivo foi proposto por Laporte and Martello (1990). O PCV-S aborda a situação em que, para cada vértice k do grafo, este possui um prêmio wk, não negativo. O problema consiste na construção de uma rota através de um subconjunto destes vértices, maximizando o total de prêmios coletados pela rota, onde o comprimento não deve exceder a um certo valor R, (Goldbarg and Luna, 2000).

- PCVJT: o PCV com janela de tempo define que dado um conjunto de N vértices, com uma distância d_{ij}, e um tempo de viagem t_{ij} entre os vértices i e j para todos pares de i, $j \in N$, e uma janela de tempo $[a_i, b_i]$ para cada vértice i, onde o vértice i não pode ser visitado antes de a_i e depois de b_i. Se o vértice i for visitado antes de a_i, será necessário esperar um tempo w_i até a_i; mas se o vértice i for visitado depois de b_i, a rota fica inviável. O objetivo é minimizar o custo da rota, onde o custo da rota pode ser a distancia total percorrida (neste caso o tempo de espera é ignorado) ou o tempo total gasto para completar a rota (neste caso o tempo de espera w_i é adicionado ao tempo de viagem t_{ij})(Balas and Simonetti, 2001).

- PCV-G ou MinMax: o PCV com gargalo ou bottleneck (PCV-G), também conhecido como MinMax, foi formalizado por Burkard et al. (1998), e consiste em obter

um ciclo hamiltoniano tal que seu arco de maior comprimento seja mínimo. Esses problemas são tradicionais no contexto de otimização, por estarem associados a situações de deslocamento de facilidades (Goldbarg and Luna, 2000).

- PCVM: o PCV múltiplo é uma generalização do caixeiro viajante onde é necessário usar mais de um caixeiro viajante, ou seja, devem ser encontradas R rotas, todas iniciando e terminando em um determinado nó, associados a x carteiros, onde o custo total é mínimo. Cada caixeiro deve viajar por uma sub-rota de nós a qual inclui um depósito comum, e cada nó, exceto o depósito, deve ser visitado exatamente por um caixeiro (Larson and Odoni, 1981).

- PCVE: no PCV estocástico as janelas de tempo, os nós, os tempos e os custos das arestas são elementos aos quais podemos associar distribuições de probabilidades que lhes definirão existência ou valor. Algumas variantes tratam os clientes como estocásticos, onde em determinados casos alguns clientes não demandavam visitas, sendo assim as rotas definidas deveriam excluir tais clientes (Goldbarg and Luna, 2000).

- OP: o problema de orientação, do inglês Orienteering Problem, define que dado um grafo não direcionado, com pesos associadas as arestas e prêmios aos vértices. O problema busca um simples ciclo compostos por diversas arestas, cujo peso total não exceda um limite, enquanto visita um subconjunto de vértices com bônus máximo (Fischetti et al., 1998).

- PTP: o problema de rotas lucrativas, do inglês Profitable Tour Problem, é o caso onde um único veículo precisa atender a demanda de diversos vértices, sendo que cada um desses vértices tem associado a si uma penalidade caso não ocorra à visitação pelo veículo, existe ainda uma matriz de custos que define os custos de deslocamento entre quaisquer dois vértices que podem fazer parte da rota. O

objetivo é a minimização do custo total de deslocamento e o somatório de penalidades pagas por não visitações. Algumas variações deste problema trabalham com prêmios por visitações, ao invés de trabalhar com penalidades por não visitações (Dell'Amico et al., 1998).

- MVP: o problema do vendedor com multiobjetivo, do inglês MultiObjective Vending Problem, define que cada nó tenha associado a si um prêmio, e uma matriz de custo de deslocamento entre vértices. O problema é definir uma rota que maximize o somatório dos prêmios coletados e minimize o somatório dos custos de deslocamento (Keller and Goodchild, 1988).

- QTSP: o PCV por quotas, do inglês Quota Traveling Salesman Problem, é variante do PCV na qual o caixeiro não precisa visitar todas os vértices, mas para cada cidade visitada uma quantidade de mercadoria é vendida. O objetivo do caixeiro é realizar uma quota de vendas através da visitação de um mínimo de vértices, e então retornar ao nó origem. Basicamente o problema é definir uma rota que o caixeiro possa realizar seu objetivo de forma mais rápida (Awerbuch et al., 1999).

Este trabalho aborda especificamente uma variante do PCV, o problema do caixeiro viajante com coleta de prêmios, o PCVCP. Assim como o PCV, o PCVCP tem como objetivo formar rotas que minimizem custos de deslocamento. Com uma importante diferença entre ambos os problemas: nem todos os vértices devem necessariamente ser visitados pelo caixeiro. Isso resulta em outro objetivo para o PCVCP: a escolha de quais vértices deverão ser visitados.

Uma das aplicações práticas deste problema é a definição da rota a ser percorrida por um helicóptero que deve visitar plataformas de petróleo, em alto mar, para coletar ou entregar pessoas ou suprimentos nas mesmas. Normalmente estas rotas devem ter uma distância mínima, e em certos casos algumas plataformas devem ser visitadas antes que outras (ex: no caso em que o suprimento coletado em uma plataforma ter

que ser entregue em outra). Esse é o caso de condições de precedência para o PCV. Em outros casos o helicóptero não tem como visitar todas as plataformas em uma única rota, devido a restrições como: combustível, tempo máximo de voo ininterrupto entre outras. Nesses casos algumas plataformas podem ser priorizadas através da associação de prêmios as mesmas.

A perfuração de solos por máquinas perfuratrizes é outra aplicação prática que pode ser descrita, pelo PCVCP. Estas máquinas perfuram o solo através do uso de ferramentas chamadas brocas, as quais se desgastam constantemente e precisam ser afiadas para continuar a serem utilizadas. Esse desgaste é influenciado pelas características físicas do solo que estiver sendo perfurado, pois a perfuração ocorre através do atrito entre broca e solo. O objetivo nesse caso é definir uma rota de perfuração, que tenha uma distância mínima, que possa ser perfurada com a mesma broca, sem que seja necessária a afiação da mesma.

Além das aplicações anteriormente descritas, e da laminação de aços (explicada no capítulo introdutório) ainda podemos citar a distribuição de alimentos perecíveis. Nestes casos pode ocorrer de determinadas mercadorias precisarem ser entregues, aos respectivos clientes, em determinados horários ou até mesmo com prioridades de entregas diferentes devido a questões contratuais ou ao fato de umas serem mais perecíveis que outras.Além das aplicações descritas existem outras aplicações práticas que podem ser formuladas e resolvidas através do uso do PCVCP.

No próximo item são descritos os métodos, exatos e heurísticos, apresentados na literatura para resolução do PCVCP. Antes será apresentado um breve estudo justificando o uso de heurísticas e metaheurísticas para a resolução deste problema e de outros problemas similares.

2.2 MÉTODOS EXATOS E HEURÍSTICOS

O PCV é um problema NP-difícil, não sendo conhecidos algoritmos de resolução em tempo polinomial (Dantzig et al., 1954). Neste contexto, para o PCV não são conhecidos algoritmos que o resolva em tempo polinomial. Para o PCV, no caso assimétrico, o número total de soluções possíveis é $(N-1)!$, onde N é o total de vértices. Sendo assim, à medida que N cresce o tempo de resolução cresce exponencialmente, tornando sua resolução exata inviável, seja avaliando recursos computacionais ou tempo de resolução. O mesmo ocorre no caso simétrico, que tem $(N-1)!/2$ soluções possíveis, pois neste caso não é necessário avaliar as variações de rota em sentido contrário. Esta complexidade justifica o estudo e desenvolvimento de heurísticas e metaheurísticas para solucionar o problema de forma viável, que apesar de não garantir a resolução do problema de forma exata, alcançam soluções finais de boa qualidade de forma rápida.

O PCVCP, assim como o PCV, pode ser representado por um grafo $G = (V,A)$ onde os vértices V representam as cidades, e os arcos A os custos de deslocamento entre estes vértices. Contudo, diferentemente do PCV, na solução ótima do PCVCP o caixeiro não precisa visitar todos os vértices. Cada um dos vértices tem associado a si um prêmio (coletado caso o vértice seja visitado pelo caixeiro) e uma penalidade (paga caso o vértice NÃO seja visitado pelo caixeiro).

A solução ótima deve minimizar o somatório dos custos de deslocamento, da rota realizada pelo caixeiro, e também minimizar o somatório das penalidades pagas por não visitação de vértices. Outra diferença entre o PCV e o PCVCP, é que este adiciona uma restrição de prêmio mínimo a ser coletado ($BenMin$), isto é o somatório dos prêmios associados aos vértices pertencentes à solução deve ser maior que $BenMin$. Demais restrições, e as demais características do PCVCP, são abordadas no item Formulação Matemática 2.3.1.

Assim como o PCV, o PCVCP é um problema NP-difícil (Bienstock et al., 1993).

Devido às características do PCVCP, fica intuitivo perceber que o mesmo apresenta uma complexidade maior que a do PCV, pois o espaço de soluções do PCVCP tem desde soluções de tamanho 0, até soluções de tamanho N. Como o espaço de soluções do PCV contém apenas soluções de tamanho de rota igual a N, o espaço de soluções do PCV é um subconjunto do espaço de soluções do PCVCP.

Devido a alta complexidade de resolução do PCVCP, poucos trabalhos foram realizados de forma a solucionar o mesmo de forma exata, e estes trataram instâncias menos complexas. A maior parte das contribuições apresentadas no meio acadêmico consistiram de métodos de resolução utilizando heurísticas, metaheurísticas e combinação entre as mesmas. Heurísticas e metaheurísticas são técnicas desenvolvidas com o intuito de resolver, satisfatóriamente, problemas complexos de se resolver exatamente, tais como o PCVCP (estas técnicas são explicadas detalhadamente de forma geral no próximo capítulo).

A seguir é realizada uma breve descrição de trabalhos relativos ao PCVCP, não tendo sido realizada separação entre abordagens exatas e heurísticas, possibilitando a análise da evolução do conhecimento do problema ao longo do tempo.

- Balas and Martin (1984) produziram um software para apoiar a programação da laminação de uma usina de bobinas de aço, problema de programação que mais tarde seria usado como base para a definição do PCVCP. Este software tinha o seu funcionamento baseado na união de várias heurísticas para procurar boas soluções para a programação dos laminadores.

- Fischetti and Toth (1988) desenvolveram procedimentos para estabelecer limites para o problema baseando-se em diferentes relaxações do problema, especificamente relaxação Lagrangiana e relaxação baseada em disjunção. No mesmo trabalho os autores propuseram um algoritmo Branch and Bound para obter a solução ótima para pequenas instâncias do PCVCP. Devido a complexidade e o

custo, tal método se aplicou apenas a estas instâncias.

- Balas (1989) introduziu o problema de forma definitiva no meio acadêmico. Apresentou um novo trabalho a respeito do problema, desta vez definindo algumas propriedades estruturais do problema e duas formulações matemáticas para o mesmo, sendo que uma foi a simplificação da primeira.

- Bienstock et al. (1993) apresentou um algoritmo MLP(Modified Linear Programming Relaxation). Neste trabalho são consideradas apenas instâncias simétricas PCVCP, que obedecessem à igualdade triangular e sem trabalhar com restrição de prêmio mínimo. No MLP é realizada a resolução de uma relaxação linear do problema juntamente com o algoritmo Christofides, largamente utilizado para resolução do TSP. O MLP apresentou resultados nos piores casos de 2,5 vezes a solução ótima do problema.

- Goemans and Williamson (1995) desenvolveram um algoritmo 2-aproximativo para uma versão do PCVCP, versão esta que se difere da estudada neste trabalho pelo fato de que a mesma não trabalha com a restrição de prêmio mínimo a ser coletado.

- Dell'Amico et al. (1998) utilizou relaxação Lagrangiana para obter limitantes inferior para o PTP e o PCVCP, através da relaxação do prêmio mínimo e aplicando o método de subgradientes para resolver o problema dual. Utilizou-se as relaxações em conjunto com a heurística de inserção mais barata (Cheapest Insertion Heuristic), clássica para o PCV, obtendo um método que a partir de uma solução relaxada se alcançasse uma solução viável. Esta técnica realiza iterativamente a inserção de vértices com melhores economias na rota, até que o prêmio mínimo seja coletado. Posteriormente ainda propuseram utilizar uma heurística para refinamento da solução viável obtida, heurística esta chamada de Extensão e Colapso (Extension and Collapse Heuristic).

- Awerbuch et al. (1999) propôs um algoritmo aproximativo para o QTSP com complexidade $O(\lg_2 n)$. Este método utiliza-se do agrupamento de pontos do problema dentro de componentes, similar ao algoritmo de Kruskal, e procura interligar os dois componentes que apresenta a menor razão de distância e o número mínimo dentro de dois componentes.

- Melo and Martinhon (2004) apresentam uma meta-heurística híbrida para solucionar o PCVCP. Utilizam uma variação do método GRASP, o GRASP Progressivo, em conjunto com o VNS (Variable Neighborhood Search - VNS)(Mladenovic and Hansen, 1997).

- Chaves et al. (2007a) e Chaves et al. (2007b) apresentam dois trabalhos abordando a resolução do PCVCP. Em (Chaves et al., 2007a) o trabalho conteve duas modelagens para o PCVCP, uma modelagem matemática resolvendo o problema de forma ótima para instâncias menores, e outra sendo utilizada para resolver o problema através das heurísticas GRASP e VNS. Em (Chaves et al., 2007b) novamente foi utilizado GRASP com métodos de vizinhança variável (VNS e VND) para resolver o PCVCP.

- Ausiello et al. (2008) apresentam uma versão online do PCVCP, na qual os pedidos chegam em tempo real e o caixeiro (servidor) precisa decidir quais pedidos atender e em que ordem atender, sem ainda conhecer a sequência completa de pedidos. O objetivo é, similarmente ao PCVCP offline, coletar um prêmio mínimo enquanto minimiza a soma do tempo para completar a rota e as penalidades associadas aos pedidos não atendidos na rota. Os autores desenvolveram um algoritmo 7/3 aproximativo para esta versão do PCVCP.

- Tang and Wang (2008) propuseram uma variação do PCVCP, chamada Capacitated Prize-Collecting Travelling Salesman Problem. Nesta versão o objetivo

é minimizar os custos de viagem e as penalidades pagas aos clientes não visitados, coletando uma quantidade de prêmios suficientemente grande e considerando que a demanda dos clientes visitados não exceda a capacidade do caixeiro. Os autores utilizaram a metaheurística de Busca Local Iterativa (ILS) para resolver esse problema. Através de testes em instâncias criadas aleatoriamente obtiveram resultados melhores que os obtidos por métodos de Busca Tabu, que utilizavam memória baseada em frequência. Também obtiveram melhores resultados quando compararam o ILS com métodos de programação utilizados em problemas reais.

- Chaves and Lorena (2007) apresentaram novamente um trabalho contendo duas abordagens de solução para o PCVCP. Utilizaram o conceito de algoritmo híbrido, combinando as metaheurísticas híbridas utilizadas pelo mesmo em seus trabalhos anteriores com um processo de agrupamento de soluções em subespaços de busca (Cluster), procurando encontrar regiões promissoras. Na primeira abordagem foi utilizado o procedimento de agrupamento chamado de ECS (Evolutionary Clustering Search) que foi proposto por Oliveira and Lorena (2004). No mesmo trabalho foi apresentada uma adaptação de tal procedimento. Por último, em Chaves and Lorena (2008) os autores apresentam uma nova versão do algoritmo utilizando novamente o conceito de "clusterização"de agrupamentos, mas associado com o GRASP, para geração de soluções iniciais. No mesmo trabalho ainda são usados os métodos VND e VNS para refinamento das soluções geradas. Em Chaves and Lorena (2008) os autores também testam a eficácia do algoritmo para instâncias com o prêmio mínimo variando entre 20%,50% e 80%, validando o método proposto para diferentes tipos de problema. Os resultados obtidos por (Chaves and Lorena, 2008) são utilizados como base comparativa para validar o algoritmo proposto neste trabalho. A escolha deste trabalho se deve ao notável histórico de conhecimento dos autores e excelente nível dos resultados alcançados.

2.3 MODELOS E FORMULAÇÃO MATEMÁTICA

Como dito anteriormente a complexidade do PCVCP torna o problema computacionalmente inviável de se resolver para instâncias maiores, pelo menos em tempo hábil e contando com os recursos tecnológicos atuais. Mesmo assim, para validarmos o método híbrido aqui proposto, a modelagem matemática será necessária, pois esta modelagem e as heurísticas já utilizadas para resolver o problema irão compor a base de comparação para os resultados que serão apresentados. A seguir a formulação matemática do problema é apresentada.

2.3.1 FORMULAÇÃO MATEMÁTICA

Nesta seção a formulação matemática do problema, e todos os seus componentes, são apresentados detalhadamente.Considere que o PCVCP possa ser representado por um grafo completo e não direcionado $G = (V,E)$. Define-se a priori:

- V: Conjunto de vértices.
- E: Conjunto de arcos (i,j).
- C_{ij}: Custo de deslocamento entre i e j, $(i,j \in V)$.
- P_i: penalidade associada a não visitação de cada vértice $i \in V$;
- W_i: prêmio associado a visitação de cada vértice $i \in V$;

Supõe-se que um vértice qualquer $i \in V$, sem perda de generalidade, seja o depósito ou a cidade de origem do caixeiro. Este vértice deve ter prêmio nulo ($P_i = 0$), e penalidade tendendo ao infinito $\gamma_x \to \infty$. A seguir é apresentada a formulação utilizada, sendo que esta foi proposta originalmente por Balas, Balas (1989). Contudo algumas alterações foram inseridas para eliminação de subrotas, assim como será explicado adi-

ante.

$$Minimizar \sum_{i \in N} \sum_{j \in N-\{1\}} C_{ij} X_{ij} + \sum_{i \in N} P_i(1 - \gamma_i) \tag{2.1}$$

Sujeito à:

$$\sum_{j \in N-\{i\}} X_{ij} = \gamma_i \qquad \forall i = 1,...,N \tag{2.2}$$

$$\sum_{i \in N-\{j\}} X_{ij} = \gamma_j \qquad \forall j = 1,...,N \tag{2.3}$$

$$\sum_{i \in N} (W_i * \gamma_i) \geq BenMin \tag{2.4}$$

$$\sum_{i \in N} \sum_{j \in N} f_{ij} - \sum_{i \in N} \sum_{j \in N} f_{ji} = \gamma_i \qquad \forall i = 1,...,N \quad i \neq 1 \tag{2.5}$$

$$f_{ij} \leq (N-1) X_{ij} \qquad \forall (i,j) \in E \tag{2.6}$$

$$X_{ij} \in [0,1] \qquad \forall (i,j) \in E \tag{2.7}$$

$$\gamma_j \in [0,1] \qquad \forall i = 1,...,N \tag{2.8}$$

A seguir é mostrado o significado das váriaveis de decisão utilizadas na formulação acima:

1. X_{ij}:
   - 1, se a aresta (i,j) pertence à rota solução.
   - 0, caso contrário.

2. γ_i:

   - 1, se o vértice (i) está incluido na rota solução.
   - 0, caso contrário.

A função objetivo 3.4 é composta basicamente por dois componentes, o primeiro computando o custo total de deslocamento e o segundo os custos de penalidades pagos por não visitação de vértices. Na formulação anterior as restrições (2.2) e (2.3) são para garantir que em todo vértice da rota tenha apenas um arco chegando e um arco saindo. A restrição (2.4) é a restrição chave do problema, que define o principal critério de viabilidade de uma solução colete um prêmio maior ou igual ao prêmio mínimo. As restrições (2.5) e (2.6) garantem que a diferença de fluxo que chega e que sai do vértice seja igual a 1 caso o vértice seja visitado, 0 caso contrário e que o fluxo máximo que pode passar por uma aresta é $N-1$ (número máximo de arestas possíveis), respectivamente. Já as restrições (2.7) e (2.8) são relativas aos limites de valores que podem ser assumidos pelas váriaveis de decisão, váriaveis que são influenciadas pela presença ou não do vértice na rota.

Capítulo 3

METODOLOGIA

3.1 INTRODUÇÃO

O PCVCP é um problema de grande aplicação prática, que pode apresentar alto grau de complexidade de resolução para diversos casos, sendo que em alguns desses casos é inviável resolver o problema de forma exata, devido a restrições tais como recursos computacioniais ou tempo de resposta disponível. Como exemplo claro de situação em que a resolução exata do PCVCP não é possível, pode ser citado a definição do sequenciamento de trabalho de uma laminadora de aços (problema este que pode ser modelado pelo PCVCP). Em ambientes industriais estas laminadoras trabalham de forma inimterrupta, e ao terminar uma sequência de laminação, a próxima sequência a ser laminada já precisa estar definida. Caso o tempo necessário para encontrar, de forma exata, a próxima sequência de laminação seja superior ao tempo de laminação da sequência que estiver sendo laminada, a resolução exata do problema não atenderia as necessidades de resolução do problema.Em casos como estes é interessante o estudo e o desenvolvimento de métodos que gerem boas soluções (mesmo que não sejam ótimas) de forma rápida. Heurísticas e metaheurísticas são técnicas desenvolvidas com a finalidade de resolver satisfatoriamente problemas com estas características.

Na sequência é realizada uma breve revisão sobre heurísticas e metaheurísticas, e logo após algumas premissas serão introduzidas, por serem necessárias para o entendimento das explicações e das técnicas que são apresentadas mais adiante neste trabalho. Ainda neste capítulo tambem são mostradas as técnicas heurísticas e metaheurísticas que são utilizadas neste trabalho, e logo são expostas as principais técnicas já aplicadas e apresentadas em trabalhos similares, inclusive nos trabalhos que são utilizados como base comparativa para avaliar a eficácia da metodologia apresentada neste trabalho.

3.2 HEURÍSTICAS E METAHEURÍSTICAS

Heurísticas e metaheurísticas são técnicas desenvolvidas para resolver, de forma satisfatória, diversos problemas. Estas técnicas não garantem que a solução ótima seja encontrada, nem informam quão perto a solução encontrada está da otimalidade, mas encontram boas soluções geralmente com um custo de resolução menor que os metódos exatos.

3.2.1 HEURÍSTICAS

Heurísticas são técnicas que empregam estratégias, procedimentos e métodos aproximativos no processo de busca de boas soluções para diversos tipos de problemas. Estas técnicas são amplamente utilizadas na busca de boas soluções para problemas que apresentam alto grau de complexidade associado a sua resolução exata. Exemplos de problema deste tipo são aqueles que têm grande espaço de soluções (ex: problemas combinatórios). Nestes casos pode ser inviável percorrer todo o espaço de soluções para encontrar a solução ótima do problema. E esta inviabilidade acontece devido aos recursos computacionais que seriam necessários (tempo, memória computacional, capacidade de processamento, entre outros).

A seguinte definição de heurística é realizada em Rich and Knight (1993)

"Para resolver eficientemente muitos problemas difíceis, geralmente é necessário comprometer as exigências de mobilidade e sistematicidade e construir uma estrutura de controle que não garanta encontrar a melhor resposta, mas que quase sempre encontre uma resposta muito boa... a heurística é uma técnica que melhora a eficiência de um processo de busca, possivelmente sacrificando pretensões de completeza."

Heurísticas podem ser classificadas quanto a seu objetivo principal, sendo de construção ou de refinamento. As de construção, normalmente utilizadas no início do processo de busca, buscam construir uma solução elemento a elemento, segundo algum critério de otimização, até que todos os elementos façam parte da solução ou até que algum critério de parada seja satisfeito.

As heurísticas de refinamento tentam melhorar uma solução, passada como parâmetro de entrada. Procuram melhorias através de mudanças na solução de entrada, ou seja, estudam a vizinhança local da solução procurando por vizinhos que tenham melhores valores de função objetivo. Isto é feito até que não seja mais possível melhorias através das alterações ou até que algum outro critério de parada seja satisfeito (ex: número de iterações do processo de busca, tempo máximo de busca entre outras).

Neste trabalho foram utilizadas tanto heurísticas de construção como de refinamento, as quais são explicadas posteriormente. Além das heurísticas, neste trabalho também são utilizadas as metaheurísticas.

3.2.2 METAHEURÍSTICAS

As metaheurísticas são procedimentos heurísticos que guiam outras heurísticas, usualmente de busca local, experimentando o espaço de soluções além da vizinhança local, ou seja as heurísticas estudam os vizinhos de uma solução, já as metaheurísticas tem a capacidade de verificar os vizinhos dos vizinhos de uma solução. Também buscam utilizar características das boas soluções já encontradas para explorar regiões promissoras de vizinhanças ainda não exploradas, permitindo desta forma a fuga das regiões onde se

encontram os ótimos locais. Esta é uma das principais diferenças entre metaheurísticas e heurísticas, sua maior capacidade de fugir de ótimos locais.

Existem várias formas de classificar as metaheurísticas, sendo que as mais comuns são explicitadas a seguir:

- **Uso de memória:**

  As metaheurísticas com memória, como a Busca Tabu (Glover and Laguna, 1997), apresentam estruturas de dados que armazenam temporariamente características ou soluções inteiras de bom aproveitamento. Algumas outras não utilizam nenhum tipo de memória, como o Simulated Annealing (Kirkpatrick et al., 1983) e o GRASP (Feo and Resende, 1995).

- **Técnicas Populacionais e Não Populacionais:**

  As metaheurísticas não populacionais exploram somente um elemento da vizinhança a cada nova iteração gerando somente uma trajetória de soluções. Essa característica torna estas técnicas de mais fácil entendimento devido a menor complexidade no processo de busca. A Busca Tabu e o Simulated Annealing são exemplos de técnicas que percorrem apenas um elemento da vizinhança por vez.

  Os algoritmos populacionais, tais como os Algoritmos Genéticos (Goldberg, 1989), Algoritmos Meméticos (Moscato, 1989) e Busca Dispersa (Glover and Laguna, 1997), prevêem a exploração de várias regiões do espaço de busca em uma mesma iteração, tornando as técnicas mais complexas mas expandindo a possibilidade de selecionar boas soluções. Diversas boas soluções são encontradas e então são combinadas, gerando novas soluções através dos operadores de variação. Os operadores e as estruturas de aplicação dos mesmos variam de acordo com a técnica utilizada.

3.3 REPRESENTAÇÃO DA SOLUÇÃO

Assim como em outros problemas combinatórios, neste trabalho optou-se por representar computacionalmente as soluções do PCVCP através de vetores de números inteiros, no qual cada índice armazenado representa um cidade. O vetor solução S contém todas as cidades candidatas a fazer parte da rota (o termo N será utilizado para representar o total de cidades candidatas). A ordenação dos itens neste vetor indica o sequenciamento de visitação das cidades pelo do caixeiro viajante (ex: na figura 3.1 o vértice A é o primeiro a ser visitado, seguido pela cidade C). Esse vetor tem associado a si um indicador de tamanho da solução (TAM), que indica quantos itens do vetor realmente fazem parte da rota, ou seja quantas cidades serão visitadas. Como cada cidade somente pode ser visitada uma única vez, o valor máximo que TAM pode assumir é N, no caso de todas as cidades candidatas fazerem parte da rota . É importante lembrar que mesmo que S tenha referência a todas as cidades candidatas, nas soluções encontradas para o PCVCP nem sempre todas as cidades fazem parte da solução.

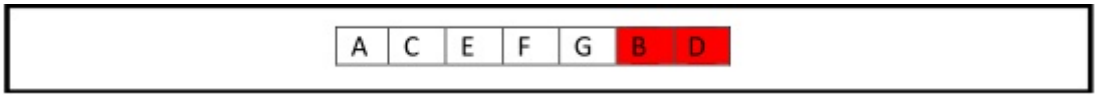

Figura 3.1: Representação de uma solução do PCVCP (TAM=5 e N=7).

O exemplo da Figura 3.1 representa, através de um vetor, uma solução com N igual a 7, e TAM igual a 5. A solução representada é composta pelos arcos: $(A;C)$, $(C;E)$, $(E;F)$, $(F;G)$ e $(G;A)$. Os vértices B e D (em cor vermelha) não fazem parte da rota obtida como solução. A Figura 3.2 mostra graficamente a solução representada por um vetor. A Figura 3.1, na Figura 3.2 cada vértice representa uma cidade candidata.

A Figura 3.2 inclui os dados de penalidades e prêmios associadas à cada um dos vértices. No PCVCP toda solução tem associada a si uma prêmio total (que é o somatório dos prêmios associados às cidades visitadas) e uma penalidade total (que é o somatório das penalidades associadas às cidades **NÃO** visitadas). Considerando que

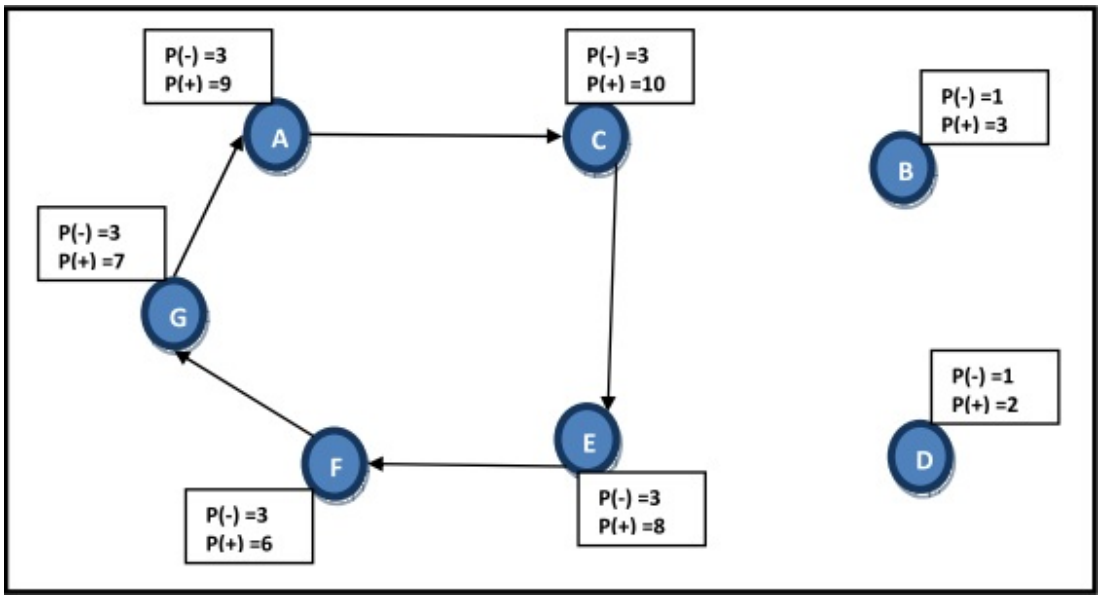

Figura 3.2: Representação gráfica de uma solução do PCVCP (TAM=5 e N=7).

na representação gráfica, da Figura 3.2 o valor de penalidade associada a cada cidade seja expressa por $P(-)$, e o prêmio associado seja representado por $P(+)$, a penalidade total da solução representada seria calculada pela soma das penalidades associadas às cidades B e D, e o prêmio total seria composto pela soma dos prêmios associados às cidades: A, C, E, F e G. No caso representado na Figura 3.2 teriamos um penalidade total igual a 2, e uma premio total igual a 40.

O PCVCP define um valor mínimo de prêmio total a ser recolhido pelo caixeiro para que uma rota definida seja considerada viável. Novamente sem perda de generalidade, daqui em diante esse valor mínimo será referenciado por P_{min}. O termo S, ou $rota$, será utilizado para referenciar o conjunto de vértices ou cidades selecionadas para visitação pelo caixeiro, e TAM ou $Tamanho$ será usado para referenciar a quantidade de vértices em uma solução.

3.4 ESTRUTURAS DE VIZINHANÇA

A técnica de busca local utilizada neste trabalho emprega a estratégia de busca em vizinhança para percorrer o espaço de soluções, com o objetivo de melhorar as soluções correntes. Para que fosse possível o estudo da vizinhança, três movimentos básicos foram definidos:

- **M1**: trocar um vértice que faça parte da rota, por um que não faça.
- **M2**: excluir um vértice da rota.
- **M3**: adicionar um vértice a rota.

O movimento **M2** pode inviabilizar uma solução, pois a exclusão de um vértice qualquer pode fazer com que o prêmio mínimo (**BenMin**) não seja alcançado. Para lidar com estas soluções e permitir uma análise mais completa do espaço de busca, adicionou-se uma penalidade variável associada a soluções inviáveis, de forma a permitir assim que a busca percorra por áreas aparentemente não promissoras, e consequentemente aumentando a capacidade do método de fugir de ótimos locais. Esta caracaterística será melhor explicada na Seção 3.5. Outra característica importante adquirida com os movimentos implementados é o aumento da flexibilidade e eficiência da busca, pois a ocorrência do movimento de troca, ou de um simples movimento de exclusão com uma posterior inclusão pode melhorar de forma significativa uma solução. As Figuras 3.3, 3.4 e 3.5 mostram respectivamente os movimentos **M1, M2 e M3**.

Além dos movimentos básicos, ideais para estudar os vizinhos diretos de S, também foi implementado um quarto movimento $M4$, para ajuste de tamanho da solução. Este movimento, assim como $M2$, tem como base a exclusão de vértices. Contudo, diferentemente de $M2$, vários vértices podem ser excluídos em um único movimento $M4$. As Figuras 3.6 e 3.7 mostram as representações de uma solução S' antes e depois do

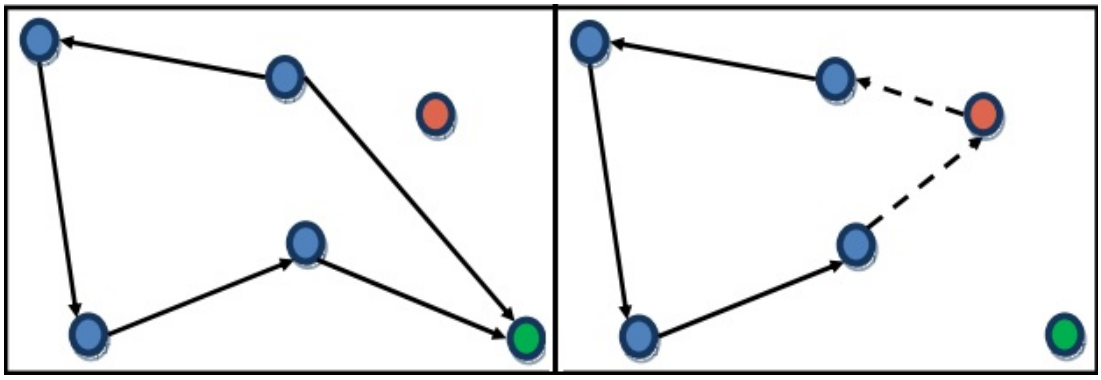

Figura 3.3: Movimento M1 (troca com um vértice não visitado).

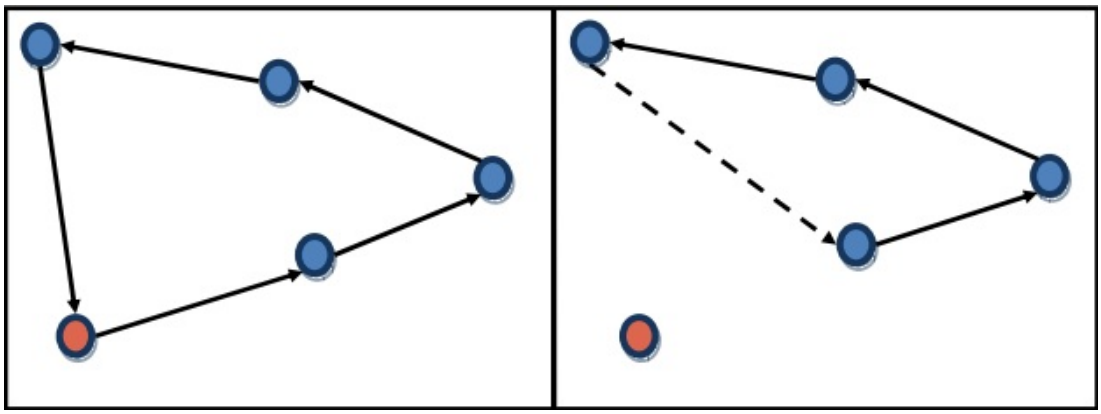

Figura 3.4: Movimento M2 (exclusão de um vértice).

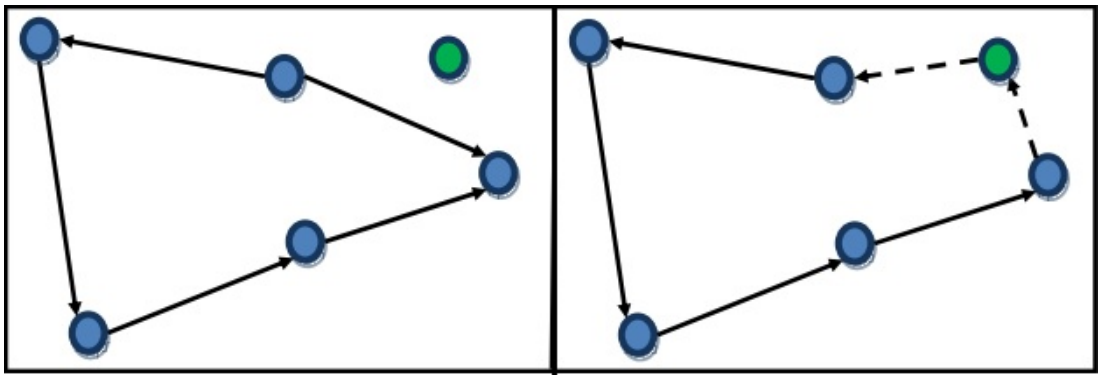

Figura 3.5: Movimento M3 (inclusão de um vértice).

movimento de ajuste de tamanho ($M4$). Nas figuras os vértices que não fazem parte da rota estão destacados em cor vermelha.

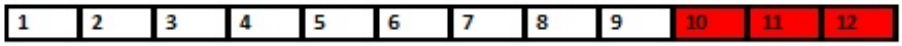

Figura 3.6: Solução S' antes do movimento M4 (Ajuste de tamanho).

Figura 3.7: Solução S' depois o movimento M4 (Ajuste de tamanho).

$M4$ foi inserido devido a característica do PCVCP de ter vários ótimos locais, e alguns desses ótimos são intransponíveis utilizando apenas os movimentos básicos. $M4$ funciona dentro do método Busca Tabu como estratégia de diversificação, pois neste trabalho as soluções iniciais geralmente tem tamanho próximos a N, mas os ótimos locais destas soluções podem ter tamanho próximo de 0. Suponha por exemplo um problema com $N = 1000$, mas que a solução ótima tenha tamanho igual a 50. O processo até chegar a esta solução seria muito custoso, e talvez o método nem chegasse à solução ótima usando apenas os movimentos básicos. Dessa forma $M4$ acelera o processo de convergência para o tamanho ótimo (daqui adiante assuma que Ω define a quantidade de vértices a serem excluídos em uma execução de $M4$). O processo de ajuste de tamanho, e a definição de Ω é melhor explicado no próximo capítulo.

Outros movimentos foram implementados durante a fase de desenvolvimento (ex:troca dois vértices da rota com dois não visitados, exclusão de dois vértices, entre outros), no entanto tais movimentos não melhoraram a qualidade das soluções obtidas, e ainda au-

mentaram o uso dos recursos computacionais e o tempo de execução. Esse aumento se deve a um maior quantidade de avaliações do espaço de vizinhança, o que se justificaria apenas por uma relação custo/benefício atrativa do aumento de recursos e melhoras nas soluções alcançadas, porém isto não foi verificado para estes movimentos.

3.5 FUNÇÃO OBJETIVO

A função objetivo utilizada neste trabalho (FO) se baseia na função apresentada em 2.3.1, contudo novos termos foram incorporados para permitir a implementação da estratégia de diversificação utilizada no método TS (4.1.2) proposto neste trabalho. Para melhor entendimento, a FO é detalhada por partes, sendo que cada uma destas partes são relacionadas a uma entrada específica do problema (distâncias entre os vértices, penalidades e premiação). A fórmula geral da FO é mostrada a seguir, e as partes que compõe a FO são explicadas na sequência.

$$FO = Minimizar \sum_{i \in N} \sum_{j \in N-\{1\}} C_{ij} X_{ij} + (\phi(\sum_{i \in N} P_i(1-\gamma_i))) + (\alpha max(0, BenMin - \sum_{i \in N}(W_i * \gamma_i))) \tag{3.1}$$

3.5.1 CUSTOS DE DESLOCAMENTO

São computados em FO os custos de deslocamento de todos os arcos $(i,j) \in V$.

$$\sum_{i \in N} \sum_{j \in N-\{1\}} C_{ij} X_{ij} \tag{3.2}$$

3.5.2 PENALIDADES

Assim como na função objetivo apresentada em 3.5, na avaliação de uma solução S é computado em FO o custo total das penalidades (γ_i) pagas por não visitação de

quaisquer vértices $i \in V$. Porém na FO aqui utilizada é adicionada uma variável dinâmica (ϕ) que altera o peso do somatório de γ_i em FO. A variável ϕ assume valores entre [0,1] durante o processo de busca. Isto é realizado para viabilizar a estratégia de diversificação que é utilizada. No entanto, mesmo que durante o processo de busca, soluções S sejam avaliadas com ($\phi < 1$), para fins de resposta do algoritmo, as soluções S geradas são avaliadas com o valor de ϕ igual a 1. Esse processo ´r melhor explicado no próximo capítulo.

$$\phi(\sum_{i \in N} P_i(1 - \gamma_i)) \tag{3.3}$$

3.5.3 PREMIAÇÃO

Para permitir que o método de busca percorra pelo espaço de soluções inviáveis, adicionou-se a FO um termo para penalizar soluções S que não coletarem o benefício mínimo requerido ($BenMin$). No entanto é importante ressaltar que para fins de resposta do algoritmo, somente são aceitas soluções S que alcancem o prêmio mínimo. Ou seja, é possível percorrer por soluções inviáveis durante o período de busca, através da computação na função objetivo do valor de prêmio que faltar recolher para o alcance do prêmio mínimo $BenMin$. No entanto para fins de resposta do algoritmo, estas soluções não são aceitas. A váriavel α utilizada mantém o seu valor variando entre [0,1] durante a execução, computando a inviabilidade nos casos em que soluções não atinjam $BenMin$. Esse processo é melhor explicado no próximo capítulo.

$$\alpha max(0, BenMin - \sum_{i \in N}(W_i \gamma_i)) \tag{3.4}$$

3.6 GENIUS

A heurística GENIUS foi apresentada por Gendreau et al. (1992) para resolver o PCV. De acordo com os autores uma das características principais do método é sua eficiência em gerar rapidamente soluções de alta qualidade. Esta heurística foi utilizada para construir a solução inicial do algoritmo proposto neste trabalho, principalmente devido ao fato da sua larga aplicação em outros problemas provenientes do PCV, e também devido a sua simplicidade de implementação.

Basicamente esta heurística é composta por duas fases, uma de construção e a outra de refinamento. As duas fases do método serão explicadas a seguir, antes porém algumas definições são necessárias:

- E: Conjunto de vértices candidatos.
- S: Solução (rota) parcial ou completa.
- P: Parâmetro que define a quantidade de vizinhos mais próximos a serem avaliados nas inserções e remoções.
- $N_p(v)$: Conjunto dos p vértices, pertencentes à rota, mais próximas de v.

A solução de entrada para a fase de construção GENI, deve conter pelo menos três vértices, que são aleatóriamente introduzidos no ínicio do método GENIUS. No algoritmo utilizado neste trabalho foi utilizado o valor 3 para P. A seguir o algoritmo que demonstra o funcionamento do método GENIUS é exibido.

3.6.1 GENI

A primeira fase, procedimento de inserção generalizada - GENI (do inglês Generalized Insertion), realiza inserção de todos vértice $v \in E$ em uma solução S. De acordo com seu idealizadores esta fase funciona tão bem para problemas simétricos, quanto para problemas assimétricos.

Algoritmo 1: *GENIUS*

Data: (E)
Result: (S)
1 **inicio**
2 $S \leftarrow \varnothing$;
3 **repita**
4 Selecione aleatoriamente um vértice $v \in E, \notin S$ e insira-o em S;
5 **até** S *conter três vértices*;
6 $S \leftarrow$ GENI(E,S);
7 $S \leftarrow$ US(E,S);
8 **retorna** (S);

Nesta heurística um vértice $v \notin S$ é inserido em S e após a sua inserção S passa por uma reorganização. Nem sempre a inserção de um vértice v é feita entre dois vértices (v_i,v_j) que são adjacentes, este é um dos fatores para a eficiência do método, pois o mesmo percorre um maior espaço de vizinhança da solução corrente através de movimentos mais complexos.

A fase GENI insere vértices a partir de dois tipos de inserção (Tipo 1 e Tipo 2). Suponha que se queira inserir um vértice v entre dois vértices (v_i e v_j) que já façam parte da rota. Para uma dada orientação da rota, entenda-se que v_k é um vértice no caminho entre v_j e v_i, e v_l um vértice no caminho entre v_j e v_i. Para qualquer vértice v_h na rota, entenda-se que v_{h-1} é seu predecessor e v_{h+1} seu sucessor. A seguir as duas formas são explicadas.

INSERÇÃO TIPO 1

Assumindo $v_k \neq v_i$ e $v_k \neq v_j$. A inserção de v na rota resulta na exclusão dos arcos (v_i,v_{i+1}), (v_j,v_{j+1}) e (v_k,v_{k+1}), e na inserção dos arcos (v_i,v), (v,v_j),(v_{i+1},v_k) e (v_{j+1},v_{k+1}). Estas operações resultam na inversão dos caminhos $(v_{i+1}, ..., v_j)$ e $(v_{j+1}, ..., v_k)$. Repare que se $(j = i + 1)$ e $(k = J + 1)$ tem-se o procedimento de inserção padrão. O Algoritmo 2 e a Figura 3.8 mostram uma rota antes e depois da inserção de um vértice qualquer, através da inserção TIPO 1.

Algoritmo 2: *GENI: Inserção Tipo 1*

Data: (v,v_i,v_j,v_k,S)
Result: (S')
inicio
  $S' \leftarrow S$;
  se $((v_k \neq v_i)$ *e* $(v_k \neq v_j))$ **então**
    Remova os arcos $(v_i,v_{i+1}),(v_j,v_{j+1}),((v_k,v_{k+1}))$ *de* S';
    Insira os arcos $(v_i,v),(v,v_j),(v_{i+1},v_k),(v_{j+1},v_{k+1})$ *de* S';
  retorna (S');

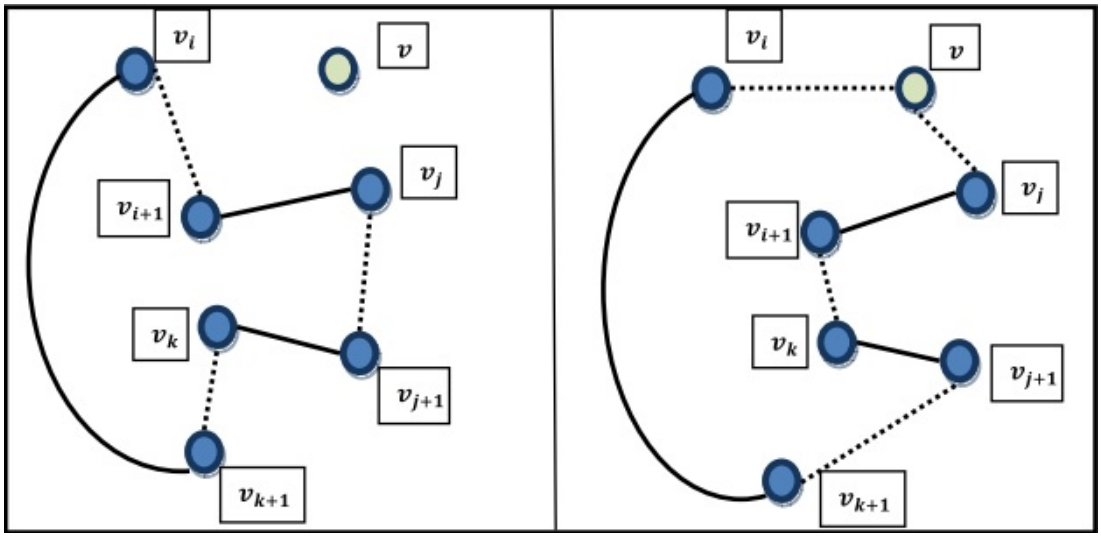

Figura 3.8: INSERÇÃO TIPO 1 de um vértice v entre dois vértices (v_i e v_j).

INSERÇÃO TIPO 2

Assumindo $v_k \neq v_j$ e $v_k \neq v_{j+1}$,$v_l \neq v_i$ e $v_l \neq v_{i+1}$. A inserção de v na rota resulta na exclusão dos arcos (v_i,v_{i+1}), (v_{l-1},v_l),(v_j,v_{j+1}) e (v_{k-1},v_k), e na inserção dos arcos (v_i,v), (v,v_j),(v_l,v_{j+1}) e (v_{k-1},v_{l-1}). Estas operações resultam na inversão dos caminhos $(v_{i+1},...,v_{l-1})$ e $(v_l,...,v_j)$. O Algoritmo 3 e a Figura 3.9 mostram uma rota antes e depois da inserção de um vértice qualquer através da inserção Tipo 2.

O método de construção GENI inicia com uma solução S vazia como entrada e iterativamente vai inserindo vértices na solução. O vértice a ser inserido (v) é escolhido aleatoriamente a cada iteração. Logo após as duas possíveis orientações da rota para

Algoritmo 3: *GENI: Inserção Tipo 2*

Data: (v,v_i,v_j,v_k,v_l,S)
Result: (S')
inicio
  $S' \leftarrow S$;
  se $((v_k \neq v_j)\ e\ (v_k \neq v_{j+1})\ e\ (v_l \neq v_i)\ e\ (v_l \neq v_{i+1}))$ **então**
    Remova os arcos $(v_i,v_{i+1}),(v_j,v_{j+1}),(v_{k-1},v_k),(v_{l-1},v_l)$ *de* S';
    Insira os arcos $(v_i,v),(v,v_j),(v_l,v_{j+1}),(v_{k-1},v_{l-1}),(v_{i+1},v_k)$, *de* S';
  retorna (S');

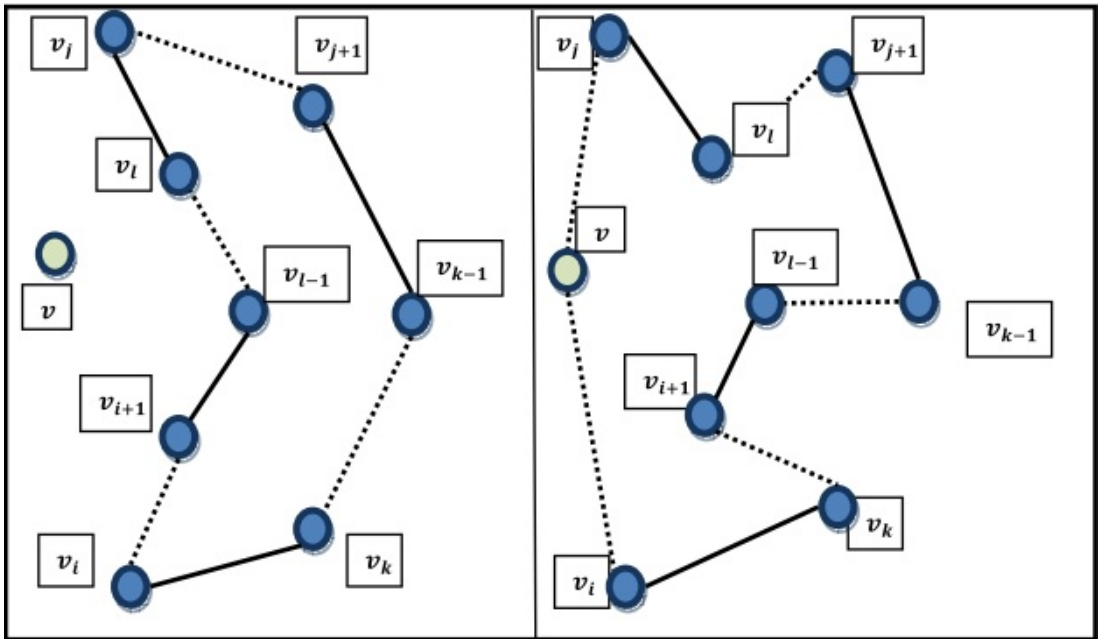

Figura 3.9: INSERÇÃO TIPO 2 de um vértice v entre dois vértices (v_i e v_j).

cada inserção são consideradas (TIPO 1 e TIPO 2). A partir da definição do parâmetro p (quantidade de vizinhos próximos que são analisados) são selecionados v_i e $v_j \in N_{(p)}(v)$, $v_k \in N_{(p)}(v_i{+}1)$ e $v_l \in N_{(p)}(v_j{+}1)$ e todas as combinações com o vértice v são analisadas, sendo escolhida a que resultar no menor custo de função objetivo.

A cada iteração o método de inserção GENI avalia as duas possibilidades de inserção (Tipo 1 e Tipo 2), inserindo o vértice (escolhido aleatoriamente) através daquela que tiver o valor de função mais atrativo. O algoritmo 4 mostra o funcionamento do método

de inserção GENI.

Algoritmo 4: *Inserção GENI: Método de inserção completo (Tipo 1 + Tipo 2)*

```
   Data: (v,S)
   Result: (S')
1  inicio
2      f* ← ∞;
3      para todo (v_i,v_j) ∈ N_p(V)S faça
4          para todo k ∈ N_p(V+1)S faça
5              S' ← insercao tipo1 (v,v_i,v_j,v_k,S);
6              S'' ← insercao tipo2 (v,v_i,v_j,v_k,v_l,S);
7              se (f' < f'') e (f' < f*) então
8                  S* ← S';
9              senão se (f'' < f') e (f'' < f*) então
10                 S* ← S'';
11     retorna (S*);
```

Este processo é realizado iterativamente até que todos os vértices $v \in E$ tenham sido inseridos em S. Sinteticamente o algoritmo GENI fica como descrito no pseudocódigo 5 .

Algoritmo 5: *GENI: Visão Geral*

```
   Data: (E,N)
   Result: (S*)
1  inicio
2      S* ← ∅;
3      enquanto (|E| − |S*|) > 0 faça
4          Selecione aleatóriamente um vértice v ∈ E, ∉ S*;
5          S* ← Inserção GENI(v,S*);
6      retorna (S*);
```

3.6.2 US

Em complementação a fase de inserção, GENI, foi desenvolvida uma segunda fase, que consiste de um procedimento de remoção de vértices, US (do inglês Unstringing and

Stringing).Esse procedimento consiste basicamente da remoção e posterior reinserção de um vértice em uma solução S, é importante ressaltar que esta remoção altera a ordem dos vértices na solução, através da exclusão e adição de arcos. Similarmemte a fase de inserção dois métodos são propostos para realizar a remoção de um vértice v de S. Logo após a remoção são realizadas tentativas de inserção de outro vértice v_x em S pelos dois métodos de inserção da fase GENI. A seguir os dois métodos de remoção são explicados de forma detalhada.

REMOÇÃO TIPO 1

Assuma $v_j \in N_p(V_{i+1})$, $v_k \in N_p(V_{i-1})$, uma dada orientação de rota, e que v_k seja um vértice no caminho entre v_{i+1} e v_{j-1}. Excluem-se os arcos (v_{i-1},v_i),(v_i,v_{i+1}),(v_k,v_{k+1}) e (v_j,v_{j+1}). E adiciona-se os arcos (v_{i-1},v_k),(v_{i+1},v_j) e (v_{k+1},v_{j+1}). Essas alterações fazem com que os caminhos $(v_{i+1}...v_k)$ e (v_{k+1},v_j) sejam invertidos. O algoritmo 6 e a Figura 3.10 mostram uma rota antes e depois de uma remoção de um vértice qualquer, através da remoção TIPO 1.

Algoritmo 6: *US: Remoção Tipo 1*

Data: (v_i,v_j,v_l,S)
Result: (S')
1 **inicio**
2 $S' \leftarrow S$;
3 *Remova os arcos* (v_{i-1},v_i),(v_i,v_{i+1}),(v_k,v_{k+1}),(v_j,v_{j+1}) *de* S';
4 *Insira os arcos* (v_{i-1},v_k),(v_{i+1},v_j),(v_{k+1},v_{j+1}) *de* S';
5 **retorna** (S');

REMOÇÃO TIPO 2

Assuma $v_j \in N_p(V_{i+1})$, uma dada orientação de rota, $v_k \in N_p(V_{i-1})$, e que v_k seja um vértice no caminho entre v_{j+1} e v_{i-2}. Assuma também que $v_l \in N_p(V_{k+1})$, e que v_k é um vértice no caminho entre v_j e v_{k-1}.

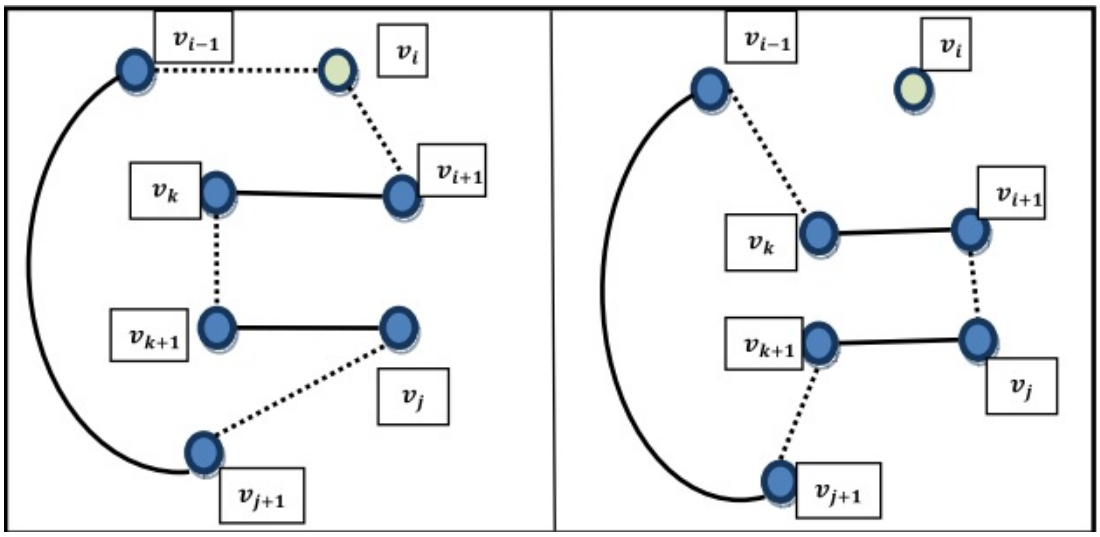

Figura 3.10: REMOÇÃO TIPO 1 de um vértice v_i

Excluem-se os arcos (v_{i-1},v_i),(v_i,v_{i+1}),(v_{j-1},v_j),(v_l,v_{l+1}) e (v_k,v_{k+1}). Adiciona-se os arcos (v_{i-1},v_k),(v_{l+1},v_{j-1}),(v_{i+1},v_j) e (v_l,v_{k+1}). Essas alterações fazem com que os caminhos $(v_{l+1}...v_k)$ e (v_{i+1},v_{j-1}) sejam invertidos. O algoritmo 7 e a Figura 3.11 mostram uma rota antes e depois de uma remoção de um vértice qualquer, através da remoção TIPO 2.

Algoritmo 7: *US: Remoção Tipo 2*

Data: (v_i,v_j,v_k,v_l,S)
Result: (S')
inicio
  $S' \leftarrow S$;
  Remova os arcos (v_{i-1},v_i),(v_i,v_{i+1}),$((v_{j-1},v_j))$,$((v_l,v_{l+1}))$,$((v_k,v_{k+1}))$ *de* S';
  Insira os arcos (v_{i-1},v_k),(v_{l+1},v_{j-1}),(v_{i+1},v_j) *de* S';
  retorna (S');

O funcionamento do método US basicamente é avaliar, para todo o vértice v pertencente à uma solução S, o custo de sua remoção através dos dois métodos US, e realizar a remoção que resultar no melhor valor de função objetivo. Caso a solução gerada por este processo (S') tenha melhor valor de função objetivo do que a solução S, S deve re-

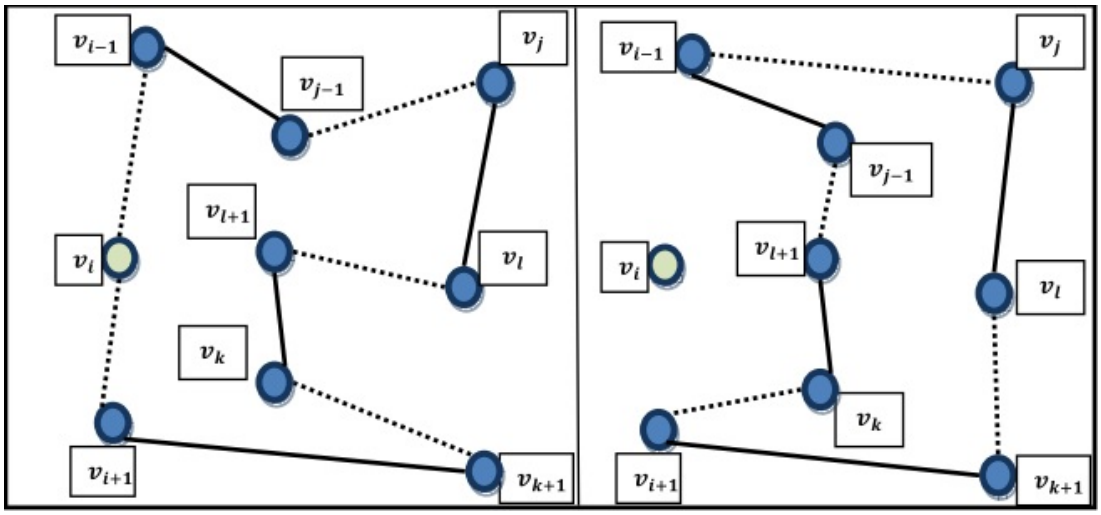

Figura 3.11: REMOÇÃO TIPO 2 de um vértice v_i

ceber S' e o método US deve reiniciar a partir do primeiro vértice. O método GENIUS utilizado neste trabalho sofreu pequenas alterações como pode ser visto no capítulo 4. O Algoritmo 8 mostra o pseudocódigo do método Remoção US e o algoritmo 9 mostra o pseudocódigo da fase de refinamento (US+GENI).

Algoritmo 8: *Remoção US: Método de remoção completo (Tipo 1 + Tipo 2)*

Data: (v_i,S)
Result: (S^*)
inicio
  $f^* \leftarrow \infty$;
  para todo $(v_k) \in N_p(v_{i+1})$ **faça**
    para todo $v_j \in N_p(V_k + 1)$ **faça**
      $S' \leftarrow remocao\ tipo1\ (v_i,v_j,v_l,S)$;
      $S'' \leftarrow remocao\ tipo2\ (v_i,v_j,v_k,v_l,S)$;
      se $(f' < f'')$ *e* $(f' < f^*)$ **então**
        $S^* \leftarrow S'$;
      senão se $(f'' < f')$ *e* $(f'' < f^*)$ **então**
        $S^* \leftarrow S''$;
  retorna (S^*);

Algoritmo 9: *Fase de Refinamento*

```
   Data: (S)
   Result: (S*)
1  inicio
2      f* ← f_(S);
3      para todo v_i ∈ S faça
4          S' ← US(v_i,S);
5          S'' ← GENI(v_i,S');
6          se (f'' < f*) então
7              S* ← S'';
8      retorna (S*);
```

3.7 K-OPTIMAL

Croes (1958) e Lin (1965) foram os primeiros a introduzir as heurísticas K-OPTIMAL (K-OPT) para melhorar soluções do PCV. A ideia central destas heurísticas é avaliar a estrutura de vizinhança de uma solução, buscando melhores soluções. Nesta heurística uma solução (S') é dita vizinha de outra solução (S") se esta pode ser obtida através da exclusão de K arestas em uma rota e a substituição por outras K arestas factíveis, em que K é chamado de dimensão da vizinhança. Este método deve executar iterativamente até que não haja melhores soluções na dimensão da vizinhança estudada. O objetivo da heurística K-OPT é diminuir os custos de deslocamento em S, através da substituição de determinadas arestas por outras de menor custo. A Figura 3.12 e o algoritmo 10 mostram o funcionamento de uma heurística K-OPT, com K assumindo o valor 2.

Maiores valores de K possibilitam uma maior probabilidade de se encontrar a solução ótima, conduto isso também acarreta em um custo computacional maior devido a complexidade deste algoritmo, que é $O_{(nk)}$ de acordo com Goldbarg and Luna (2000).

Neste trabalho foram utilizados heurísticas K-optimal com valores K igual a 2 e a 3, sendo o método $2-OPT$ utilizado após a geração da solução inicial e sobre todos os vizinhos gerados pelo método de busca local. O método $2-OPT$ é executado diversas vezes devido à sua rapidez e baixo custo computacional. Já o método $3-OPT$

Algoritmo 10: *Algoritmo 2-OPT*

```
   Data: (S)
   Result: (S*)
1  inicio
2      f* ← f(S);
3      para todo (vértice a ∈ S) faça
4          para todo vértice (b > a) ∈ S faça
5              S' ← S;
6              Inverta caminho entre (a,b) em S';
7              se (f(S') < f*) então
8                  S* ← S';
9      retorna (S*);
```

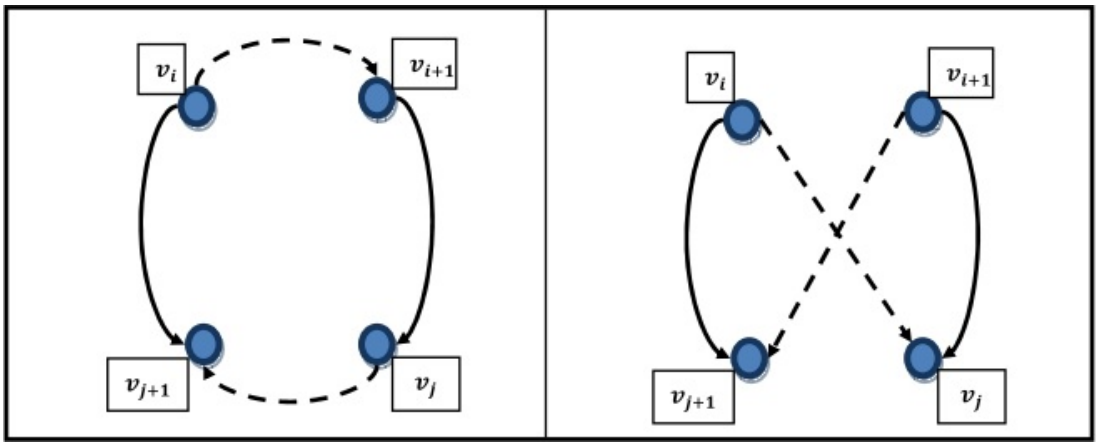

Figura 3.12: 2-OPT

foi utilizado para o refinamento da solução final, tendo menor custo computacional quando aplicado em uma solução já otimizada pelo método de busca local. O algoritmo e outros detalhes do processo de utilização destas heurísticas são mostrados no próximo capítulo deste trabalho.

3.8 BUSCA TABU

Proposto por Glover (Glover, 1989), o método Busca Tabu (do inglês Tabu Search) é uma metaheurística não estocástica que tem a capacidade de escapar de ótimos locais.

Nesta seção será utilizado o termo TS para referenciar o método de Busca Tabu. A capacidade de fuga de ótimos locais do TS é possível através da aceitação de movimentos não aprimorantes, junto com estruturas de memória com informações coletadas durante o processo de busca. Estas características permitem o método percorrer o espaço de soluções de forma mais diversa, escapando de ótimos locais e consequentemente aumentando a probabilidade de encontrar soluções ótimas. O método TS tem uma fase exaustiva em seu funcionamento. Nesta fase movimentos gulosos, de melhora, são realizados, partindo-se de uma solução S para uma solução S' que tenha melhor valor de função objetivo. Esta fase ocorre até que se encontre um ótimo local, ou seja, até o momento em que nenhum vizinho seja melhor que a solução corrente. Apenas após este momento movimentos de piora são permitidos. Basicamente durante o TS a busca avança sempre de uma solução para o seu melhor vizinho, porém um grande problema pode acontecer neste processo, a **ciclagem de soluções**. O processo de ciclagem de soluções é explicado no próximo tópico.

3.8.1 CICLAGEM DE SOLUÇÕES

Para explicar este processo será utilizada a Figura 3.13. Esta figura mostra um processo simples de busca gulosa por melhores vizinhos e a ocorrência do processo de ciclagem de soluções. Na Figura 3.13 é possível ver o processo de ciclagem de soluções em uma busca gulosa por melhores vizinhos.

No momento 1 a busca inicia-se a partir de uma solução a. No segundo momento da busca o método parte para o melhor vizinho de a (solução b). No terceiro momento outro movimento de melhora é realizado, movimentando-se de b para a solução c. Como pode ser visto na figura a solução c é um ótimo local. No quarto momento a busca movimenta-se para o melhor vizinho da solução c, que é a solução d. Neste momento ocorre o primeiro movimento de piora do procedimento de busca. No quinto momento a busca movimenta-se novamente para o melhor vizinho da solução corrente d, que no

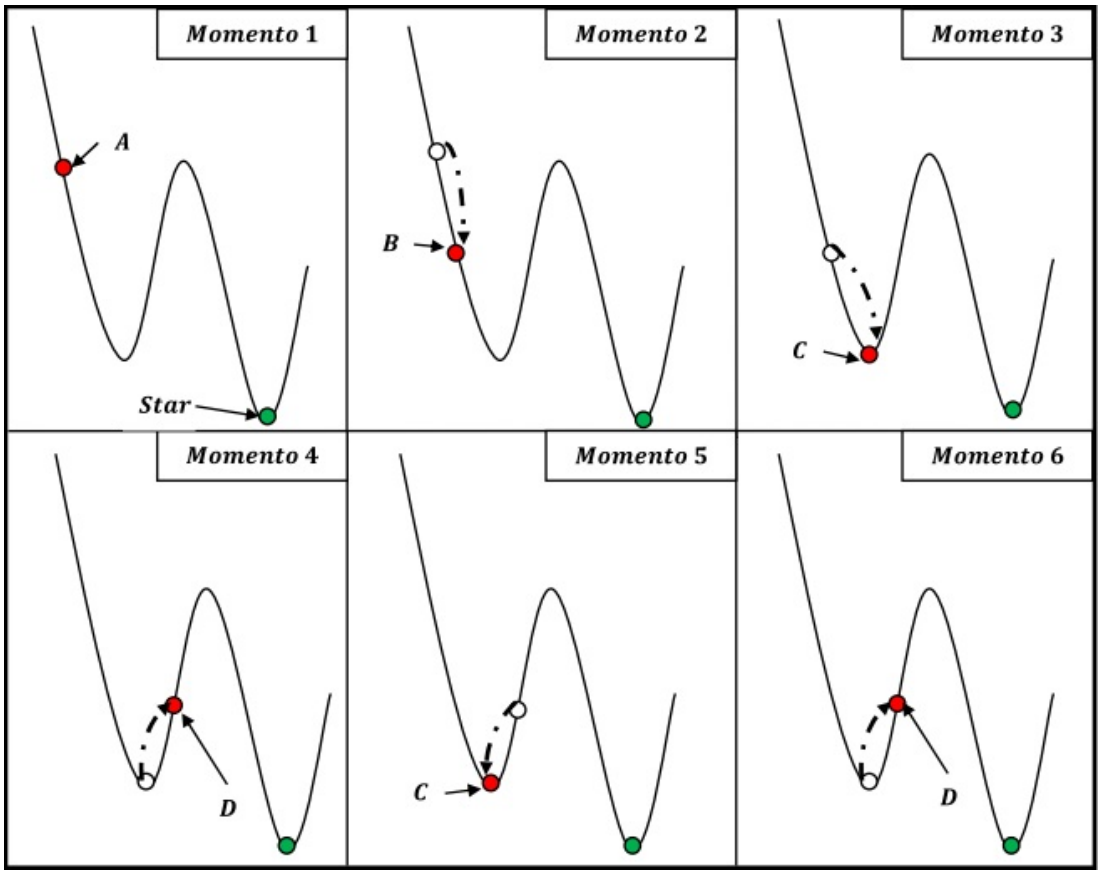

Figura 3.13: Ciclagem de Soluções

caso é novamente a solução c. Consequentemente, o processo de busca fica preso a este ótimo local (c). Esse é o processo de ciclagem de soluções. No processo simples de busca gulosa por melhores vizinhos sempre ocorrerá ciclagem entre uma solução localmente ótima e seu melhor vizinho. Neste método a solução ótima jamais é alcançada, se o metódo se prender a um ótimo local.

3.8.2 LISTA TABU

Para evitar a ciclagem de soluções o TS utiliza um conceito de **memória de curto prazo** chamado de Lista Tabu. A lista guarda informações sobre as últimas soluções exploradas. Essa lista diminui a probabilidade da geração de ciclos durante o processo de busca. O método armazena apenas algumas das características principais das soluções, normalmente são armazenados os movimentos que geraram estas soluções. As soluções completas não são armazenadas, pois isto teria um custo computacional enorme. O TS utiliza a Lista Tabu para impedir que movimentos recentes voltem a acontecer por algumas iterações, assim diminui a probabilidade da busca voltar para soluções anteriormentes pesquisadas.

O **tamanho** da lista tabu é um importante parâmetro do método, pois ele define quantos movimentos farão parte desta lista. Se o tamanho for muito grande, o espaço de busca explorado tende a ser mais diverso, contudo listas com tamanhos muito grandes são muito proibitivas. Tamanhos pequenos de Lista Tabu são menos proibitivos, mas aumentam a probabilidade de ciclos. Além do tamanho da lista, outro parâmetro a ser definido pelo método é a quantidade de iterações necessárias para que um movimento seja considerado tabu, conceito este chamado de **tenure**. A definição do valor do **tamanho da lista** e de **tenure** são completamente relacionadas, visto que, por exemplo, não faria diferença uma Lista Tabu ter tamanho igual a 100, mas o tenure desta mesma lista ter valor igual a 2, pois sendo assim esta lista nunca ficaria completamente preenchida de movimentos tabus. O parâmetro Tenure pode ser estático, alterado de-

terministicamente durante a execução do método ou aleatoriamente dinâmicos. Glover and Laguna (1997) afirma que o uso de valor de tenure dinâmico produz melhores resultados que valores estáticos ou alterados deterministicamente, além de afirmar que valores estáticos e pequenos de tenure fazem o método convergir para ótimos locais. Um simples exemplo de Tenure dinâmico seria definir aleatoriamente o seu valor entre um intervalo de dois números, no método proposto neste trabalho o cálculo de tenure é realizado desta forma.

Um problema com o uso do método TS é que a criação de uma lista de movimentos tabu pode bloquear alguns movimentos que se executados novamente gerariam soluções melhores que as já encontradas, pois nem sempre os mesmos movimentos irão gerar as mesmas soluções, visto que normalmente são armazenadas informações dos movimentos e não das soluções.Para lidar com esta situação, o método TS utiliza o conceito de **critério de aspiração**. Este critério define uma condição de aceitação para um movimento tabu, isto é, uma condição que se satisfeita permite a execução de um movimento mesmo que este esteja na lista tabu. Um exemplo de uso do Critério de Aspiração é o caso em que a execução de um movimento tabu gere uma solução que tenha valor de função objetivo melhor do que todas as outras soluções já encontradas. Por último, outro critério do TS é o **critério de parada**, ou seja, até quando o método deve executar. Número de iterações, tempo de execução, número de iterações sem melhora, são alguns exemplos de critérios de parada para o TS. Segundo Gendreau (2003) os critérios de paradas mais utilizados são:

- Quantidade de iterações.
- Quantidade de iterações sem melhoras na função objetivo.
- Quando a função objetivo alcança um limite pré-determinado.

Uma lista tabu para o PCVCP pode ser composta por vários itens, onde cada item armazena informações sobre um movimento bloqueado. Estes itens podem ser com-

postos por dois campos: um campo chamado “movimento”, que armazena informações dos movimentos considerados tabu (ex: troca, inclusão, exclusão...); E o outro campo o tenure, que define por quantas iterações o movimento deve ser considerado tabu. A Figura 3.14 mostra a estrutura de uma possível simples lista tabu de tamanho igual a 3 para o PCVCP.

MOVIMENTO	***TENURE***
INSERÇÃO (v)	**4**
REMOÇÃO (v)	**3**
INSERÇÃO (v_i)	**1**

Figura 3.14: Lista Tabu

O algoritmo 11 mostra o processo que deve ocorrer em toda iteração da busca tabu, a atualização da lista. Basicamente este processo deve diminuir em uma unidade o “tenure” de todos os itens da lista. Se o “tenure” de um item assumir o valor zero este item deve sair da lista, ou seja, o movimento deixa de ser tabu. O algoritmo 12 mostra o funcionamento básico do TS. Para os algoritmos 11 e 12 considere uma lista tabu ordenada pelo campo tenure, isto é os primeiros itens são os compostos pelos movimentos que serão tabus por mais tempo, e que no momento de inserção de um item o mesmo já seja inserido ordenadamente nesta lista.

Algoritmo 11: *AtualizaListaTabu*

Data: $(LT, TamanhodaLT)$
1 **inicio**
2 **para todo** *(item* $A \in LT$*)* **faça**
3 $A.tenure \leftarrow (A.tenure - 1)$;
4 **se** *(*$A.tenure \leq 0$*)* **então**
5 $A.movimento \leftarrow \varnothing$

Algoritmo 12: *Busca Tabu*

Data: $(S, N, MaxIter, TamMaxTabu, Tenure)$
Result: (S^*)
1 **inicio**
2 $LT \leftarrow \varnothing$;
3 $Iter \leftarrow 0$;
4 $S^* \leftarrow S$;
5 $S_{(corrente)} \leftarrow S$;
6 **enquanto** *(*$Iter < MaxIter$*)* **faça**
7 $S' \leftarrow S_{(corrente)}$;
8 gerarvizinho(S',*movimento*);
9 **se** *(*$f_{(S')} < f_{(S^*)}$*)* **então**
10 $S^* \leftarrow S'$;
11 $S_{(corrente)} \leftarrow S'$;
12 Insere *movimento* em LT;
13 **senão se** *(*$movimento \notin LT$*)* **então**
14 $S_{(corrente)} \leftarrow S'$;
15 AtualizaListaTabu($LT, TamMaxTabu$);
16 **retorna** (S^*);

3.8.3 INTENSIFICAÇÃO

Outro tipo de memória que pode ser utilizada pelo TS é a **memória de médio prazo**. Basicamente o TS armazena informações das melhores soluções encontradas (regiões promissoras), e utiliza estas informações para gerar novas soluções. Esta estratégia se apoia na ideia de que as melhores soluções tem similaridade entre si. Um exemplo de intensificação pode ser vista no método Reconexão por Caminhos.

A Reconexão por Caminhos foi introduzida por Glover and Laguna (1997). Este mé-

todo de intensificação define uma maneira de explorar regiões consideradas promissoras, mas ainda não exploradas durante a busca, estabelecendo trajetórias entre soluções já conhecidas. Basicamente este método transforma a solução corrente de forma iterativa, e a cada iteração incorpora, na solução corrente, as características mais aprimorantes de uma **solução guia** (uma boa solução já conhecida). Esta estratégia possibilita encontrar soluções melhores que muito provavelmente não seriam encontradas em um processo de busca normal.

Contudo em alguns casos os mínimos locais encontrados podem não ter similaridades com os ótimos globais, e assim a intensificação das características desses mínimos diminuiria a possibilidade de se encontrar o ótimo, reduzindo a eficácia da busca.

A sequência de figuras a seguir mostra um exemplo em que uma estratégia de intensificação funcionou para encontrar um ótimo global. Para o exemplo em questão considere que a solução ótima é composta pelos vértices (1,2,3,4,5). A Figura 3.15 mostra a melhor solução encontrada pela método de busca até o momento (esta solução represente um ótimo local). Já a Figura 3.16 mostra a solução atual do processo de busca.

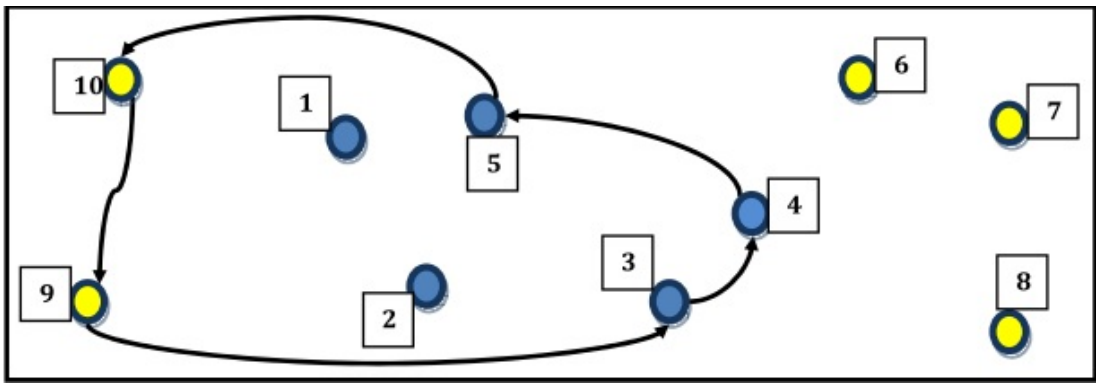

Figura 3.15: Representação da solução ótima local.

A Figura (3.17) mostra a solução corrente após um processo de intensificação. Neste processo uma característica da melhor da solução (Figura 3.15) foi inserida na solução

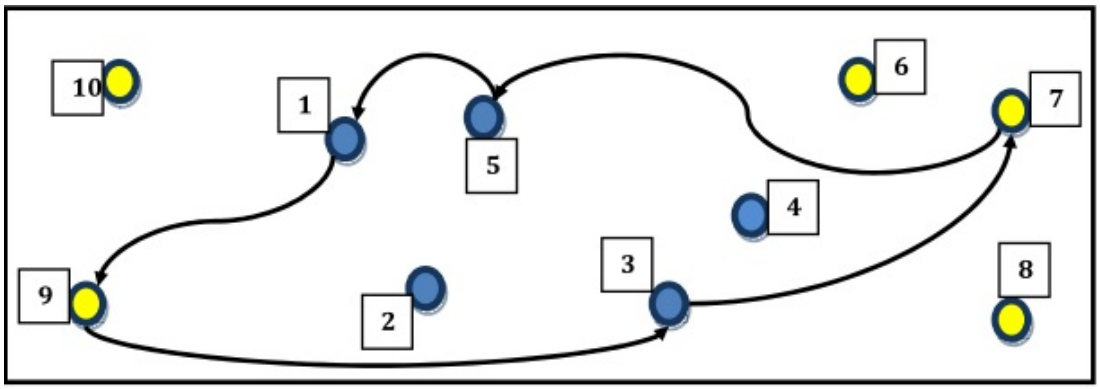

Figura 3.16: Representação da solução corrente.

corrente (Figura 3.16). Neste caso específico foi a troca do vértice 7 pelo vértice 4.

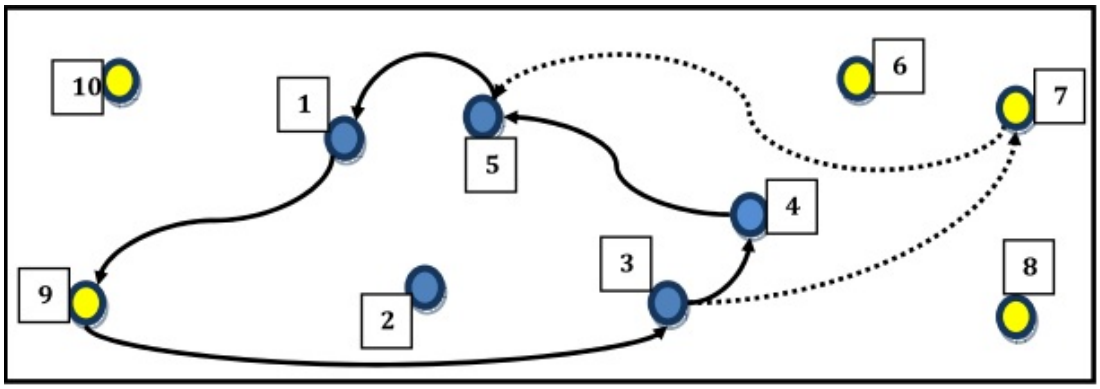

Figura 3.17: Processo de Intensificação.

Após a intensificação o processo de busca continua, e na vizinhança da nova solução corrente (3.17) o ótimo global (3.18) é encontrado através do movimento de troca do vértice 9 pelo vértice 2.

Neste trabalho foi utilizado uma estratégia de intensificação similar ao descrito nestas figuras. A intensificação foi apoiada na frequência de cada vértice $v \in E$ nas soluções que melhoraram o valor de S^* durante o processo de busca. Essa intensificação se baseia na ideia de que se um vértice v faz parte de muitas soluções ótimas locais a probabilidade v fazer parte da solução ótima global é maior. O próximo capítulo apresenta mais detalhes da aplicação desta estratégia no algoritmo aqui proposto.

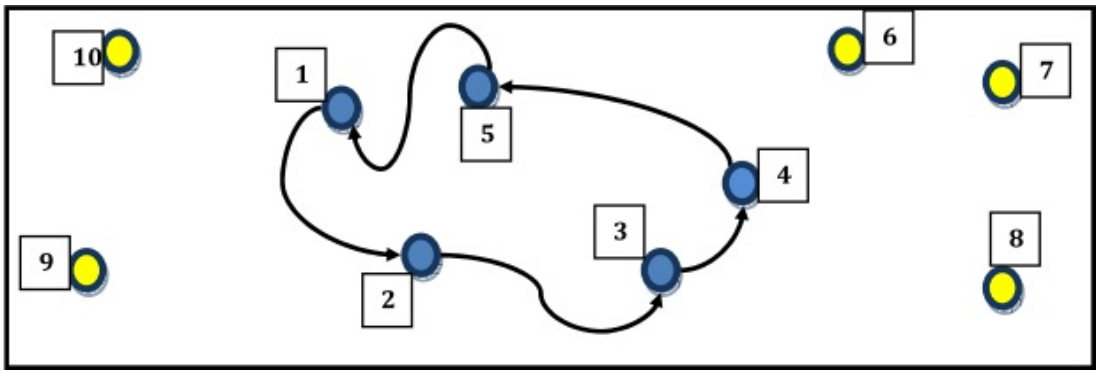

Figura 3.18: Representação de solução ótima global.

3.8.4 DIVERSIFICAÇÃO

Além dos usos de memória de curto e médio prazo, a TS pode utilizar de forma complementar a **memória de longo prazo**, de acordo com Glover and Laguna (1997). Com o objetivo inverso às estratégias de intensificação, a memória de longo prazo busca usar as informações das soluções já encontradas para percorrer um espaço de busca ainda não explorado, ou seja, encontrar uma solução diversa das já encontradas. Segundo Gendreau et al. (1992) a diversificação é um mecanismo que tenta forçar a busca por áreas não exploradas no espaço de busca. Essa estratégia é comunmente utilizada quando o processo de busca tabu não evolui, ficando nitidamente preso ao redor de ótimos locais. Nestes casos o processo fica iterativamente gerando soluções de qualidade igual ou inferior às melhores já encontradas pelo método durante a sua execução.

De acordo com Glover and Laguna (1997) as técnicas de diversificação são classificadas geralmente em duas categorias.

- **DIVERSIFICAÇÃO CONTÍNUA (Continuous Diversification)**:adiciona na função objetivo um componente referente a frequência dos atributos, ou seja, insere informação de memória na avaliação dos custos dos movimentos.

- **DIVERSIFICAÇÃO DESCONTÍNUA (Restart Diversification)**:em algum momento do método de busca tabu a função de diversificação é ativada. Esta

função altera a solução corrente, fazendo isto através da combinação com característica de outras soluções, utilizando esta solução alterada como novo ponto inicial para a busca.

Um tipo clássico de diversificação Descontínua é a **Diversificação por Reinício**. Neste tipo de diversificação em um determinado momento da busca (ex: um momento em que após várias iterações não houve melhoria no valor da função objetivo) o método força atributos raramente utilizados na solução atual, e reinicia a busca a partir da solução obtida com as modificações. A Figura 3.19 mostra a aplicação dessa estratégia de diversificação.

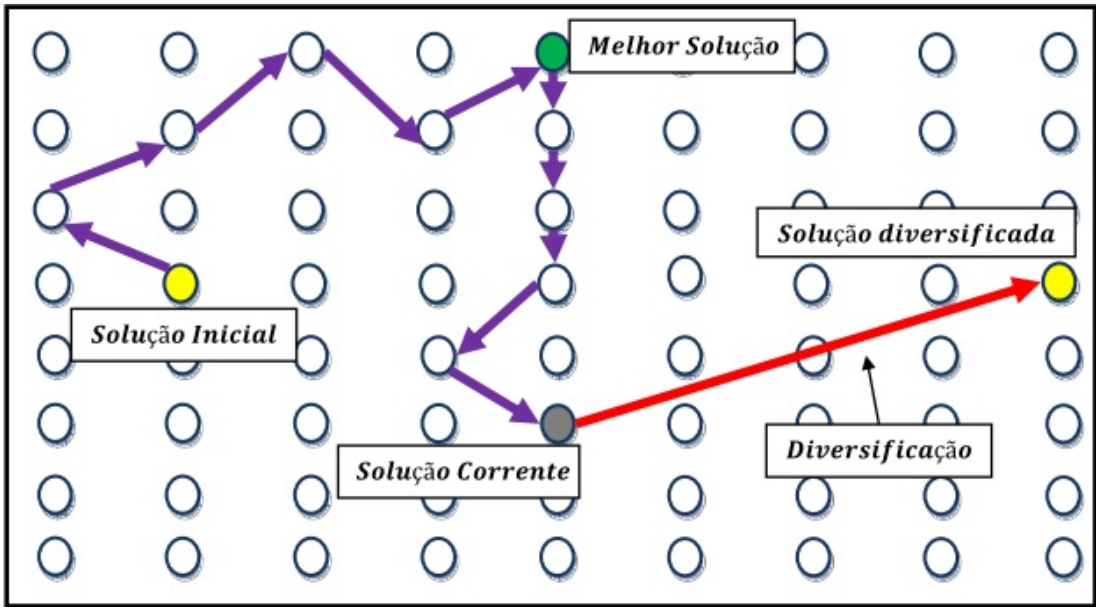

Figura 3.19: Exemplo de Diversificação por Reínicio.

É importante verificar que quando a Diversificação por Reinício é aplicada, em uma solução, uma grande área do espaço de busca deixa de ser explorada momentaneamente, embora futuramente esta mesma região possa ser explorada.

No algoritmo apresentado neste trabalho duas estratégias de diversificação descontínuas são utilizadas. A primeira é a realização do movimento $M4$, conforme mencionado

na seção 3.4. A segunda é realizada através do uso de ϕ e α dinâmicos. Ambas estratégias utilizadas neste trabalho são menos bruscas que o exemplo da Figura 3.19, evitando dessa forma que uma grande parte do espaço de busca não seja explorada. A atualização das váriaveis dinâmicas ocorre apenas após um determinado limiar de iterações em que o método não apresente melhoras. Esta atualização força o método a diversificar as soluções geradas, mas este processo ocorre de forma gradual, pois a solução corrente é alterada a cada iteração. Outra característica do processo de diversificação utilizado neste trabalho é que as variáveis dinâmicas somente podem ter seu valor alterado durante um certo número de alterações, evitando assim que o método se perca no espaço de buscas. Mais detalhes da diversificação utilizada são apresentados no próximo capítulo.

3.9 AGRUPAMENTO DE SOLUÇÕES

A metaheurística Agrupamento de Soluções do inglês Clustering Search (CS), introduzida por Oliveira and Lorena (2004), busca regiões promissoras no espaço de busca, fazendo isto através do agrupamento de soluções geradas por alguma outra metaheurística específica. A partir do momento que uma região promissora do espaço de busca é encontrada, continua-se a busca com algum método específico de busca local, que seja menos custosos e mais eficazes para refinar soluções.

Duas atividades são executadas de forma simultânea no CS: uma metaheurística (que gera boas soluções) e um processo de agrupamento iterativo (que tem a função de identificar soluções ou grupo de soluções promissoras). As soluções identificadas como promissoras são posteriormente submetidas a um processo de refinamento através de alguma método de busca local.

O método utiliza o conceito de clusters ($\boldsymbol{G}$) para agrupar as regiões promissoras. Um agrupamento é formado pelos seguintes componentes:

- β **(Estratégia de Busca)**: É uma estratégia específica que visa explorar as soluções que compõe um cluster, com o objetivo de gerar novas soluções.

- C **(Centro de Busca)**: O centro C, representado por uma solução, identifica um cluster no espaço de busca. A princípio os centros são gerados aleatoriamente, no entanto ao longo do processo estes centros tendem a se movimentar para as regiões que se mostram mais promissoras no espaço de busca.

- r **(Raio de uma região de busca)**: assim como o nome indica, especifica uma distância entre o centro e o limite de uma região, estabelecendo a distância máxima que uma solução pode estar do centro para ser associada a um cluster. Para o PCVCP, por exemplo, esta distância pode ser definida como a quantidade de movimentos necessários para transformar uma solução em outra, ou ainda a quantidade de vértices iguais presentes em duas soluções.

O CS é composto basicamente pelos 4 componentes mostrados a seguir:

1. ***Metaheurística (ME)***: É um processo que trabalha de forma contínua para a geração de novas soluções que são associadas aos clusters. Normalmente é uma heurística ou metaheurística que tem baixo custo computacional, em função da quantidade de soluções que deve gerar.

2. ***Agrupador Iterativo (AI)***:É um processo paralelo que recebe as soluções geradas pela ME e faz a associação das mesmas ao agrupamento de acordo com a similaridade da solução gerada e do centro do cluster. Para evitar a criação ilimitada de agrupamentos é definido o m_c, que é um máximo para o número de agrupamentos. Ao se associar uma solução a um agrupamento gera-se uma perturbação que atualiza o centro deste agrupamento. Essa pertubação é chamada de assimilação. Para fazer esta assimilação pode ser usada alguma estratégia de intensificação como a reconexão por caminhos, apresentada na 3.8.3.

3. ***Analisador de Agrupamentos (AA)***: é um processo que ocorre a cada intervalo de tempo pré-determinado, ou no momento que o centro de um agrupamento é atualizado. Tem objetivo de buscar agrupamentos promissores e de eliminar agrupamentos não promissores. Um agrupamento i é avaliado como promissor ou não de acordo com sua densidade (γ_i: quantidade de soluções que compõe um cluster). O **AA** entende que se um agrupamento i tem alta densidade, ele deve ser melhor analisado pois sua alta densidade evidencia uma maior probabilidade de se encontrar o ótimo nesta região. A eliminação de um agrupamento, não significa a eliminação das soluções que compõe o mesmo, pois estas soluções são reagrupadas em outros agrupamentos.

4. ***Otimizador Local (AO)***: O AO é associado à estratégia de busca β, para refinamento das soluções promissoras.

A Figura 3.20, adaptada de Chaves and Lorena (2008), mostra a forma como os componentes do CS interagem entre si.

3.10 GRASP

A metaheurística GRASP (Greedy Randomized Adaptive Search Procedure) é um algoritmo multi-partida que tem sido frequentemente aplicado a problemas de otimização combinatória (Feo and Resende, 1995).O método consiste em criar uma solução inicial e depois efetuar uma busca local para melhorar a qualidade da solução. Este processo pode ocorrer diversas vezes até atingir algum critério de parada, sendo mantida a melhor solução encontrada. O algoritmo 13 mostra o funcionamento básico do método. A ideia principal do método é utilizar as boas características dos algoritmos puramente gulosos aliados à aleatoriedade na fase de construção de soluções viáveis, que posteriormente são submetidas a um processo de refinamento. A fase construtiva do GRASP se diferencia dos métodos puramente gulosos devido a inserção do conceito da **Lista**

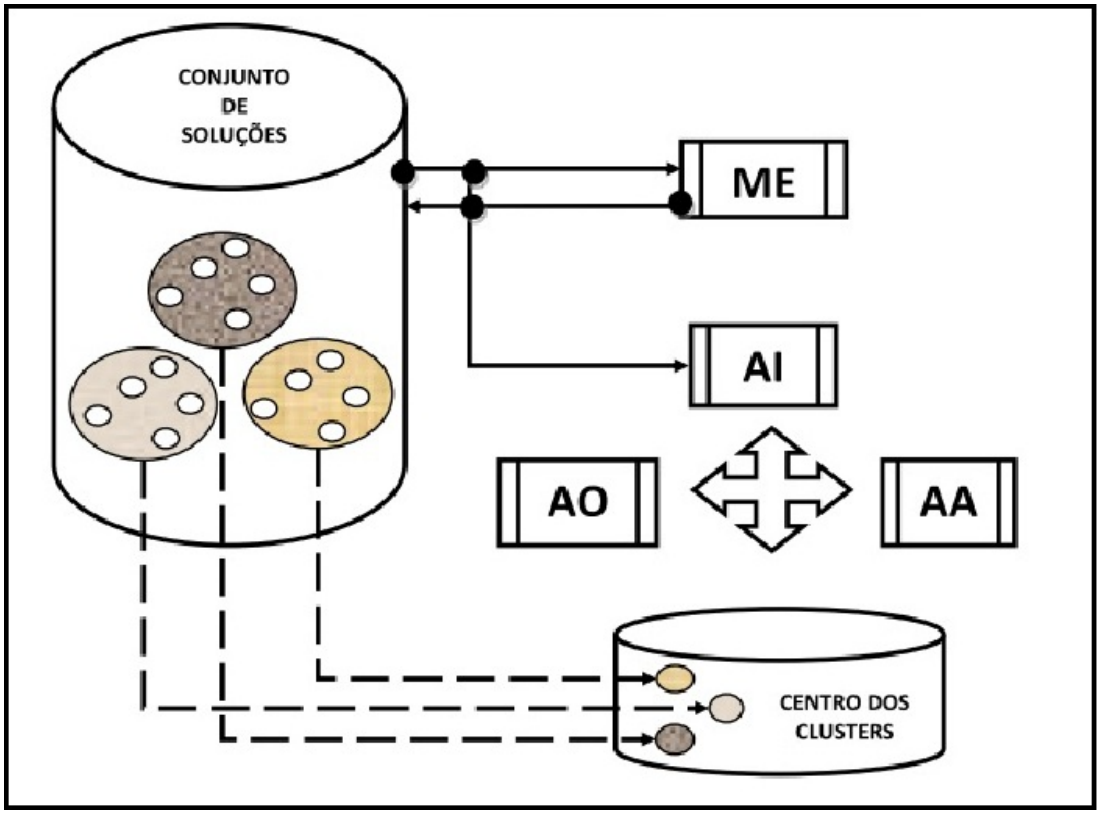

Figura 3.20: Diagrama conceitual do CS.

de Candidatos e da aleatoriedade. Esta lista contêm todos os vértices candidatos a inserção em uma iteração do método. Essa lista é então ordenada segundo alguma função gulosa adaptativa. A aleatoriedade do método está na escolha do vértice a ser inserido, o qual é selecionado de forma aleatória em um subconjunto da lista. Esse subconjunto contém os vértices mais atrativos a serem inseridos na iteração segundo a avaliação da função gulosa adaptativa.

A seguir podemos ver um exemplo prático do conceito de Lista Restrita de Candidatos (LRC). Na Figura 3.21 temos uma lista de candidatos, contendo todos os candidatos a inserção em um determinado momento, sendo que cada um desses candidatos tem a si associado um valor de função específico. Na Figura 3.22 temos uma lista contendo os mesmos candidatos, porém em uma ordenação de acordo com o valor da função de avaliação. Definindo-se que o candidato a ser escolhido deve estar entre os 25% melhores da lista, temos que o candidato deverá sair da LRC apresentada na Figura 3.23.

LISTA DE CANDIDATOS INICIAL								
Ordem dos vértices	1	2	3	4	5	6	7	8
Função de avaliação	5	7	3	6	8	2	1	0

Figura 3.21: Representação de LISTA DE CANDIDATOS.

LISTA DE CANDIDATOS ORDENADA								
Ordem dos vértices	5	2	4	1	3	6	7	8
Função de avaliação	8	7	6	5	3	2	1	0

Figura 3.22: Representação de LISTA DE CANDIDATOS ORDENADA.

LISTA DE CANDIDATOS RESTRITA (25%)		
Ordem dos vértices	5	2
Função de avaliação	8	7

Figura 3.23: Representação de LISTA RESTRITA DE CANDIDATOS.

Algoritmo 13: *GRASP*

```
   Result: (S*)
1  inicio
2      S* ← ∅;
3      enquanto (critério de parada não for atingido) faça
4          S' ← ∅;
5          Construir Solução (S');
6          Refinar Solução (S');
7          se (f_S' < f_S*) então
8              S* ← S';
9      retorna (S*);
```

3.11 PESQUISA EM VIZINHANÇA VARIÁVEL

O VNS (*Variable Neighborhood Search*) é um método de pesquisa em vizinhança variável (Mladenovic and Hansen, 1997). Basicamente este método gera vizinhos de uma solução e aplica algum método de busca local sobre os mesmos. Diferentemente de outros métodos de busca local, o VNS não segue necessariamente um trajetória em seu processo de busca, mas procura diversificar o espaço de soluções percorrido, somente aprofundando em determinada área se esta representa melhoras na busca. A solução inicial do método é gerada aleatoriamente de forma a evitar a ciclagem proveniente de alguma regra de construção deterministica. O algoritmo 14 mostra o funcionamento do método VNS.

3.12 DESCIDA EM VIZINHANÇA VARIÁVEL

O Método de Descida em Vizinhança Variável (Variable Neighborhood Descent, VND), foi proposto por Mladenovic and Hansen (1997). É também um método de busca local que procura encontrar melhores soluções através do estudo de vizinhança. No VND, diferentemente do VNS que escolhe a vizinhança através de aleatoriedade, existe uma ordem prédeterminada de estudo da vizinhança, de acordo com alguns movimentos. Se

Algoritmo 14: *VNS*

```
   Data: (R: número de vizinhanças.)
   Result: (S)
1  inicio
2      S ← ∅;
3      enquanto (critério de parada não for atingido) faça
4          k ← R;
5          enquanto (K > 0) faça
6              S' ← Gerar vizinho de (S);
7              Busca Local (S');
8              se (f_S' < f_(S)) então
9                  S ← S';
10                 K ← R;
11             senão
12                 K + +;
13     retorna (S);
```

a execução de qualquer movimento sobre uma solução resultar em melhoras na mesma, o método atualiza a solução corrente com a solução gerada e volta ao ínicio da ordem de movimentos para o completo estudo da vizinhança da solução récem-gerada. Trabalhos posteriores reforçam a possibilidade de que a ordem dos movimentos executados para geração da vizinhança seja atualizada a cada iteração de acordo com algum critério, pois isso diminuiria a tendência do método em convergir para ótimos locais (Mine et al., 2010). O algoritmo 15 mostra o funcionamento básico do método VND.

3.13 BUSCA LOCAL ITERATIVA

A busca local iterativa (Iterated Local Search, ILS) foi proposta por Lourenço et al. (2003). Esse método procura focar a busca em áreas promissoras, provocando perturbações em soluções que são ótimos locais. O método gera uma solução inicial qualquer e a submete então a um método de busca local. Após o refinamento, a solução é perturbada de alguma forma e é novamente submetida a um processo de busca local. Fazendo isso

Algoritmo 15: *VND*

```
   Data: (R: número de vizinhanças.)
   Result: (S)
1  inicio
2      S ← ∅;
3      k ← R;
4      enquanto (K > 0) faça
5          S' ← Melhor vizinho de (S);
6          Busca Local (S');
7          se (f_S' < f_(S)) então
8              S ← S';
9              K ← R;
10         senão
11             K + +;
12     retorna (S);
```

de forma iterativa, ao fim da iteração verifica se a solução gerada satisfaz um critério de aceitação (ex: melhora de FO). Caso positivo ela passa a ser o parâmetro para gerar perturbação em outras soluções.

Uma importante característica que influencia na eficiência do método é a perturbação a qual as melhores soluções são submetidas, pois se esta perturbação for muito intensa a solução gerada vai herdar poucas características da solução geradora, e se a perturbação for muito fraca a solução gerada vai tender a ficar presa ao ótimo local da solução geradora. Existem trabalhos apresentando bons resultados com o uso de perturbação de intensidade váriavel (Blum et al., 2008). Similarmente à outros métodos de busca local, durante a ILS pode ocorrer ciclagem de soluções. Uma boa alternativa para se evitar que isto aconteça é o uso de critérios de perturbação aleatórios ou semi-determínisticos. O algoritmo 16 mostra o funcionamento básico do método ILS.

Algoritmo 16: *ILS*

```
   Result: (S)
1  inicio
2      Crie uma solução S qualquer;
3      S ← Busca Local (S);
4      enquanto (Critério de parada não satisfeito) faça
5          S' ← Perturbação de (S);
6          Busca Local(S');
7          Critério de Aceitação (S',S);
8      retorna (S);
```

Capítulo 4

ALGORITMOS

Neste capítulo é apresentado com maior riqueza de detalhes o algoritmo proposto para resolver o PCVCP, o qual se baseia em alguns dos métodos já apresentados no capítulo anterior. Também neste capítulo é realizada uma breve revisão de outros algoritmos, propostos em diferentes trabalhos da literatura, que apresentaram bons resultados para o problema. Entre estes algoritmos estão os que são usados como base comparativa para os resultados obtidos neste trabalho.

4.1 ALGORITMO BUSCA TABU PROPOSTO

O algoritmo apresentado neste trabalho é uma combinação entre métodos de uso consolidado na literatura do PCV. Utilisou-se o metaheurística busca tabu como método de busca local (seção 3.8), para a construção da solução inicial foi utilizado o GENIUS (seção 4.1.1), e para refinamento de soluções durante a execução do algoritmo foram utilizados algoritmos K-OPT (seção 3.7). O pseudocódigo do algoritmo busca tabu proposto pode ser visto no Algoritmo 17.

Para entendimento das explicações posteriores considere as seguintes definições:

- E: Conjunto composto por N vértices candidatos.

Algoritmo 17: *AlgoritmoProposto*

Data: ($BenMin$,N,E,$TamMaxTS$,$Tenure$,$MaxIter$,$StopAlt$,$StopInv$,$StopRuim$)
Result: (S)

```
1 inicio
2     S ← Filtragem de Solução Inicial(BenMin,N,E);
3     Aplicar 2-OPT em S;
4     S ← BuscaTabu
      (S,E,BenMin,MaxIter,StopAlt,StopInv,StopRuim,ϕ,α,TamMaxTS,Tenure);
5     Aplicar 3-OPT em S;
6     retorna (S);
```

- S: Solução (rota) parcial ou completa.
- LT: Lista contendo informações sobre os movimentos bloqueados/tabu.
- S^*: Melhor solução viável encontrada durante o processo de busca. Considere que uma solução somente pode se tornar S^* se ela é viavel, e sua função objetivo foi calculada com θ igual a 1;
- $S2^*$: Solução com menor valor de FO encontrada, que não seja viável, ou que sua FO tenha sido calculada com θ menor que 1.
- $FREQ$: Vetor contendo informações de frequência de todos os vértices v nas soluções que melhoraram S^*.
- Solução Viável: Toda solução S que o somatório dos premios P_i ($\forall i \in S$) seja maior ou igual ao $BenMin$ (Prêmio Mínimo).
- Solução Inviável: Toda solução S que o somatório dos prêmios P_i ($\forall i \in S$) seja menor que $BenMin$ (Prêmio Mínimo).

Na sequência, detalhes da implementação do algoritmo proposto (**TS'**) são mostrados.

4.1.1 SOLUÇÃO INICIAL

Considerando que S seja uma solução inicialmente vazia, podemos descrever o funcionamento do método construtivo proposto da seguinte forma: inicialmente seleciona-se de forma aleatória 3 vértices v pertencentes a E, e os insere sequencialmente em S. Após as inserções iniciais aleatórias é realizado a fase de inserção inteligente de vértices. Esta fase consiste em um procedimento iterativo para a inserção de vértices $v \in E, \notin S$ através do método GENI. A cada iteração um vértice $v \in E$ ainda não inserido em S é escolhido de forma aleatória, e sua inserção em S é realizada pelo metódo GENI. Este procedimento iterativo deve ser executado enquanto o tamanho da solução S for menor que o parâmetro Δ. Este parâmetro foi adicionado para permitir o processo de filtragem cujo objetivo é gerar soluções iniciais de tamanho diferente escolher a que tiver o menor valor de função objetivo como entrada para o método de busca local.

Após a fase de inserção, a solução S é submetida a um processo iterativo de refinamento através da remoção de vértices, sendo que a cada iteração deste processo um vértice v pertencente a S é escolhido de forma aleatória, e a sua remoção é realizada de acordo com o método US. É importante ressaltar que diferentemente do método GENIUS original, o algoritmo implementado não tenta reinserir os vértices v que são removidos. Esta opção foi escolhida devido ao fato de uma solução do PCVCP poder ser composta por uma quantidade de vértices menor do que N. O procedimento de remoção de vértices ocorre enquanto as remoções realizadas estiverem melhorando o valor da função objetivo de S, e S continuar viável. Ao fim do processo a solução S é retornada como saída.

O algoritmo 18 mostra o funcionamento do método de construção proposto.

FILTRAGEM DA SOLUÇÃO INICIAL

A passagem de boas soluções iniciais como parâmetros de entrada para métodos de busca em vizinhança é um fator decisivo para a qualidade das soluções retornadas

Algoritmo 18: *Construção da Solucao Inicial*

Data: $(BenMin,N,E,\Delta)$
Result: (S^*)

```
1  inicio
2      S ← ∅;
3      Escolha aleatoriamente três vértices v e os insira sequencialmente em S;
4      enquanto Tamanho de S for menor que Δ faça
5          Selecione aleatoriamente um vértice v ∈ E, ∉ S;
6          Geni(v,S,Qtde);
7      enquanto ((continuar = sim) e (S é viável) ) faça
8          S' ← S;
9          Selecione aleatoriamente um vértice v_i ∈ S';
10         RemocaoUs(v_i,S');
11         se ( (f_(S') < f_(S)) e (S' é viável) ) então
12             S ← S';
13             continuar = sim;
14         senão
15             continuar = não;
16     retorna (S*);
```

por estes métodos. Devido a este fato foi adicionado um procedimento de filtragem para melhorar a qualidade das soluções iniciais geradas no algoritmo aqui proposto. A melhoria da qualidade é alcançada através da geração de diversas soluções iniciais de tamanhos diferentes, sendo escolhida como ponto de partida a melhor solução S viável gerada.

Inicialmente, durante a fase de desenvolvimento deste trabalho, o método de construção da solução inicial implementado retornava uma solução inicial S que continham todos os vértices v, no entanto em alguns casos a escolha destas soluções de tamanho igual a N como soluções iniciais podem prejudicar o desempenho do método de busca posterior. Considere por exemplo um problema com 500 vértices candidatos e que tenha um benefício mínimo igual a 20% do valor de $BenMin$, considere ainda que a solução ótima para este caso tenha tamanho igual a 100. Neste exemplo, caso a busca inicie em uma solução S de tamanho 500, o método de busca gastará boa parte do tempo

ajustando o tamanho da solução S através da exclusão de vértices. Além do maior consumo de recursos, outra dificuldade enfrentada é a possibilidade de ótimos locais que o método de busca não consiga superar.

Para lidar com estas dificuldades, relacionadas ao tamanho da solução inicial S, diversas soluções iniciais S' são geradas através do método de construção da solução inicial (Algoritmo 18). Sendo escolhida como ponto de partida (S), para o método de busca posterior, a solução S' viável que tenha o menor valor de função objetivo.

Durante a fase de construção da solução inicial são geradas de forma iterativa 10 soluções iniciais candidatas S', sendo que cada uma destas soluções é gerada com um valor do parâmetro Δ diferente. Considere *iter* com a variável que define a iteração do procedimento de filtragem. O valor de Δ é atualizado iterativamente durante o procedimento de filtragem de acordo com a seguinte fórmula:

$$\Delta = iter * (\frac{N}{10}) \tag{4.1}$$

O algoritmo 19 mostra o funcionamento básico do procedimento de filtragem de soluções iniciais.

Algoritmo 19: *Filtragem de Solução Inicial*

Data: $(BenMin,\phi,\alpha,N,E,\Delta)$
Result: (S)

```
1  inicio
2      iter ← 1;
3      S ← ∅;
4      f_(S) ← ∞;
5      enquanto iter ≤ 10 faça
6          Δ ← (iter * (N/10));
7          S' ← Construção da Solução Inicial (BenMin,φ,α,N,E,Δ);
8          se ( (f_(S') < f_(S)) e (S' é viável) ) então
9              S ← S';
10         iter ← iter + 1;
11     retorna (S);
```

A solução S obtida após o procedimento de filtragem é submetida a um processo de refinamento através do método K-OPT (com $k = 2$), detalhado na seção 4.1.3.

Após o refinamento, a solução inicial S é submetida a um procedimento de melhorias através do estudo de vizinhança baseado no método Busca Tabu, procedimento este que será detalhado na seção 4.1.2.

4.1.2 BUSCA TABU

A solução inicial S obtida é submetida a um processo de melhorias através do metódo de busca local Busca Tabu (TS). Como já visto na seção 3.8, o funcionamento básico do TS consiste em um processo de busca de melhores soluções através do estudo da vizinhança de uma solução, fazendo isto com a possibilidade de escape de ótimos locais através da aceitação de vizinhos não aprimorantes. Os movimentos utilizados para gerar os vizinhos de uma solução são os já descritos na seção 3.4.

Foram adicionadas novas estratégias de diversificação e intensificação no método de busca tabu desenvolvido neste trabalho (TS'). As estratégias de diversificação baseiam-se na atualização das variáveis α e ϕ adicionadas na FO (seção 3.5), e também na realização do movimento $M4$ para ajuste de tamanho. Já a intensificação do método baseia-se na reinserção de um vértice v que tenha sido excluído de S. É escolhido para ser reinserido o vértice $v \notin S$ que durante o processo de busca mais vezes tenha feito parte de soluções S que tenham melhorado o valor de S^*. As estratégias de diversificação e intensificação são explicadas com detalhes na seção 10 e na seção 41. Outra diferença do método proposto é que os movimentos de inserção e remoção tem prioridade sobre os movimentos de troca, isto é explicado na seção 41. Além das estratégias citadas anteriormente, o TS' se difere de um método de busca convencional por ter dois critérios de aceitação, sendo eles:

- **Critério de Aceitação 1 (C1)**: Se $(f_{(S)} < f_{(S^*)})$ e (S for viável) e $(\theta = 1)$;

- **Critério de Aceitação 2 (C2)**: Se ($f_{(S)} < f_{(S2^*)}$);

No caso de C1 ser satisfeito, S^* é atualizada recebendo S. Já no caso de C2 ser satisfeito, apenas a solução corrente é atualizada, e na próxima iteração a busca continua a partir da mesma. O critério C2 se torna muito útil no caso das váriaveis ϕ e α assumirem com valor abaixo de 1. O uso desse critério permiti que em determinados momentos o método de busca explore soluções inviáveis ou alteradas, atingindo assim uma maior diversidade do espaço de buscas explorado.

O algoritmo 20 mostra o funcionamento básico de TS'.

MOVIMENTOS PRIORITÁRIOS

Devido ao fato de que soluções do PCVCP tem vários ótimos locais, podendo cada um destes ter tamanhos diferentes, optou-se por priorizar movimentos de inserção e remoção ($M2$,$M3$) de um vértice v em relação a um movimento de troca ($M1$). Dessa forma, primeiro o método busca encontrar um bom tamanho de solução, para então definir quais os vértices devem fazer parte da mesma. Antes da implementação da priorização o método claramente se perdia buscando ao mesmo tempo encontrar um bom tamanho e definir quais vértices deviam fazer parte da solução. Por exemplo, em uma solução S com tamanho igual a Tam, a inserção de um vértice v pode ser atraente. Assuma que a inserção de v em S seja realizada. Na continuação do processo de busca, x vértices são removidos e o tamanho da solução diminua para $Tam - x$. Após estas alterações, a remoção do vértice v que tinha sido inserido em S pode voltar a ser atraente. Para problemas com maiores valores de N (ex:500), claramente o método de busca se perde. A priorização de movimentos evita que isso ocorra. Sendo assim, um movimento de troca só pode ser realizado se os movimentos de inserçao ($M2$) e remoção ($M3$) não gerarem um vizinho que seja a melhor solução viável já encontrada (S^*).

A ideia é que primeiramente o método encontre um bom tamanho, e somente depois ele procure os melhores vértices. Contudo depois de alguns movimentos de troca,

os movimentos de inserção e remoção podem voltar a ser realizados. Os resultados mostram a eficácia da priorização dos movimentos que alteram o tamanho, pois estes deixam o processo de busca mais claro e coeso.

DIVERSIFICAÇÃO

Como dito anteriormente duas estratégias de diversificação foram utilizadas no TS' proposto, uma através da atualização de variáveis dinâmicas (ϕ e θ) e outra através do ajuste de tamanho de uma solução. A explicação das duas será realizada na sequência, para questão de entendimento utilizaremos os termos $D1$ e $D2$ para falar das duas estratégias separadamente.

Diversificação D1 : A estratégia de diversificação $D1$ de TS' se baseia na atualização das variáveis dinâmicas ϕ e θ, sendo que ambas variáveis são iniciadas com valor igual a 1. O funcionamento básico de $D1$ consiste em zerar as variáveis ϕ e θ quando o método de busca atingir determinado número de iterações ($StopRuim$) sem melhorias em S^*.

Quando θ é zerado, as penalidades deixam de ser computadas no cálculo da função objetivo, e a tendência do procedimento de busca passa a ser a remoção de todos os vértices em S, pois dessa forma teoricamente a distância percorrida tende a ser minimizada. Para evitar que todos os vértices sejam removidos, adiciona-se um limite de iterações ($IterAlterada$) em que θ pode valer menos que 1. Quando esse limite é atingido, θ tem o seu valor novamente igualado a 1.

Quando ϕ é zerado, independente de uma solução S gerada ser ou não viável, esta função não é penalizada no cálculo da função objetivo. Dessa forma o processo de busca tende a remover todos os vértices $v \in S$, novamente para diminuir a distância percorrida e o valor de FO. Assim o método tende a aprofundar cada vez mais no espaço de busca contendo soluções inviáveis. Da mesma forma que

foi feito com θ, adiciona-se um limite de iterações inviáveis($IterInviavel$): se esse limite é atingido (ou seja, se o método estiver retornando muitas soluções inviáveis) ϕ volta a valer 1, e as soluções inviáveis voltam a ser penalizadas em FO com o intuito de parar a tendência de remoção de vértices do método. É importante frisar que mesmo que a função objetivo de uma solução seja calculada com ϕ igual a 0, isso não quer dizer que esta solução seja inviável. Sendo assim, mesmo que $IterAlterada$ e $IterInviavel$ recebam o mesmo valor, não necessariamente as variáveis ϕ e θ voltariam a valer 1 ao mesmo tempo. Essa estratégia de perturbação adotada por TS' se mostra válida no sentido de modificar a direção da busca ou o espaço de soluções a explorado, possibilitando fuga de ótimos locais através do caminho pelo espaço de soluções inviáveis.

A forma como as variáveis (ϕ e θ) e os contadores ($IterInviavel, IterAlterada$) são atualizados é vista no algoritmo 21.

Diversificação D2 : Em toda iteração do método TS' uma tentativa de ajuste de tamanho ($M4$) da solução corrente é realizada. Esse ajuste de tamanho é realizado através da exclusão de Ω vértices, sendo Ω um parâmetro definido como $\frac{1}{5}$ do tamanho da solução corrente. A escolha deste valor ocorreu após sucessivos testes, a calibragem de parâmetros será melhor explicada no próximo capítulo. Os vértices que são escolhidos para serem excluídos são os Ω últimos vértices de uma solução, o uso de inteligência adicional para a escolha de quais vértices devem ser excluídos não apresentou melhorias nos testes realizados. Este movimento não verifica a Lista Tabu, visto que ele só é efetivado se a solução gerada for a melhor viável até o momento. As figuras 4.1 e 4.2 mostram a realização do movimento $M4$ em uma solução corrente.

Após a exclusão dos Ω vértices, verifica-se então se a solução gerada S' é viável, no caso de S' ser **inviável** o movimento não é realizado. No caso de ser **viável**,

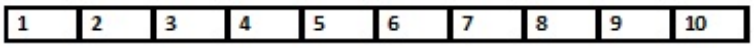

Figura 4.1: Solução S' de $Tam = 10$.

Figura 4.2: Solução S' depois do movimento M4 com $\Omega = (Tam/5)$.

S' é então submetida ao processo de refinamento através do método $2-OPT$, se após o refinamento S' for a melhor solução já gerada, S^* é atualizada e o processo de busca TS' continua a partir da mesma.

INTENSIFICAÇÃO

A estratégia de intensificação utilizada por TS' baseia-se na reinserção de vértices v em uma solução S, através da estratégia Geni. É escolhido para ser reinserido em S o vertice $v \in E, \notin S$ que durante o processo de busca mais vez fez parte de soluções que melhoraram o valor de f^*, firmado na idéia de que um vértice que muitas vezes faz parte de um ótimo local, tem alta probabilidade de fazer parte de um ótimo global. Na implementação, os dados de frequência foram armazenados em um vetor $FREQ$, e toda vez que S^* é atualizado, $FREQ$ também é atualizado, com os dados dos vértices $v \in S^*$. Os algoritmos 22 e 23 mostram a manipulação básica de $FREQ$.

É importante observar que a intensificação ocorre somente quando em uma iteração de TS', não é possível a geração de uma solução viável, e nem de uma solução inviável S tal que $f_{(S)}$ seja menor que $f_{(S2^*)}$. Isto ocorre normalmente quando o processo de busca

diverge após as atualizações dos valores das variáveis ϕ e θ, ficando com tendência à remoção contínua de todos os vértices em S. Sendo assim, depois da intensificação as variáveis ϕ e θ são novamente igualadas a 1.

4.1.3 K-OPT

A heurística de refinamento K-OPT foi utilizada em três pontos do algoritmo busca tabu proposto. Esta heurística se mostrou eficiente para a melhoria da solução inicial, das soluções geradas pelos operadores de TS', e no refinamento da solução final.

O primeiro ponto no algoritmo que se aplicou a heurística K-OPT foi após a construção da solução inicial. Neste ponto considerou-se $K = 2$ devido a rapidez e baixo custo computacional do método. O intuito da aplicação do método neste ponto foi passar como entrada para o método posterior (TS') uma solução ótima local já refinada.

Depois do primeiro ponto a heurística K-OPT é executada em toda iteração do processo de busca do TS'. Diferentemente do método TS convencional, onde os vizinhos gerados pelos movimentos são avaliados logo após a realização dos movimentos, no TS' antes de serem avaliados todos os vizinhos são submetidos ao processo de refinamento através do método K-OPT (com $K = 2$), como pode ser visto nos algoritmos 25, 26, 24 e 27. Somente após a execução do método K-OPT (com $K = 2$) é que é realizada a escolha do melhor vizinho segundo o algoritmo 20.

Por último a heurística K-OPT com $k = 3$ é utilizada para refinar a solução final retornada pelo método TS'.

4.2 AGRUPAMENTO DE SOLUÇÕES

Chaves and Lorena (2008) apresentam um trabalho no qual foi proposto uma metaheurística híbrida para resolver o PCVCP. Esta heurística utiliza o conceito de Agrupamento de Soluções (CS), do inglês Clustering Search,z para estudar áreas promissoras

do espaço de busca. Uma das principais diferenças entre o trabalho de Lorena e Chaves e o método CS original (ECS), Oliveira and Lorena (2004), é que o CS original foi proposto com a utilização de algoritmos evolucionários na construção de soluções, enquanto o algoritmo proposto por Lorena e Chaves faz uso de métodos de estudo de vizinhança GRASP/VNS.

As áreas promissoras são identificadas através da criação e agrupamento de soluções, e posterior estudo dos agrupamentos criados. O diagrama conceitual do método pode ser visto na seção 3.20. Neste trabalho os autores validam o uso do CS através da apresentação dos resultados obtidos pelo método GRASP/VNS sem o processo de clusterização, e do GRASP/VNS com o processo de clusterização, mostrando nitidamente uma melhora nos valores dos resultados obtidos com o uso do CS para a maioria dos testes. No mesmo trabalho os autores utilizaram o "solver"CPLEX 10.01.1 para resolver a formulação matemática exata do problema.

Na sequência detalhes dos componentes do CS implementado são apresentados.

4.2.1 GERAÇÃO DE SOLUÇÕES (ME)

No ECS um algoritmo evolucionário é executado simultaneamente ao processo de clusterização, o qual é responsável pela geração de soluções. Já no CS proposto em (Chaves and Lorena, 2008), o algoritmo evolucionário é substituído pelo uso das heurísticas GRASP/VNS, sendo o GRASP responsável pela criação de soluções e o VNS pelo refinamento das mesmas. Estes processos são explicados na sequência.

GRASP

Como dito no capítulo anterior, a ideia principal do método GRASP é utilizar as boas características dos algoritmos puramente gulosos aliados a aleatoriedade na fase de construção de soluções viáveis. O GRASP constrói uma solução iterativamente através da inserção de vértices.

No trabalho em questão o procedimento Adding-Nodes (Dell'Amico et al., 1998) foi utilizado para criar a lista de candidatos, a qual posteriormente é ordenada através da função $G_{(k)}$, que avalia a economina de inserção de um vértice k entre dois vértice (i,j).

$$G_{(k)} = C_{(i,j)} + \gamma_{(k)} - C_{(i,k)} - C_{(k,j)} \tag{4.2}$$

O procedimento de construção somente deve parar se nenhum vértice k apresentar $G_{(k)}$ positivo e se o prêmio mínimo $(Pmin)$ já tiver sido coletado.

VNS

Como explicado, o funcionamento básico do VNS é gerar um vizinho de uma solução S e então submetê-lo a um processo de refinamento através de uma busca local. A solução gerada pelo procedimento construtivo anterior é passada como entrada para o método VNS, que no trabalho em questão gera aleatoriamente um vizinho em uma das seguintes estruturas de vizinhança:

- **M1**: adição de um vértice v em S.
- **M2**: remoção de um vértice v de S.
- **M3**: troca de posição entre dois vértices (i,j) em S.
- **M4**: adição de dois vértice (i,j) em S.
- **M5**: remoção de dois vértice (i,j) de S.
- **M6**: troca de posição entre quatro vértices (i,j,k,l) em S.
- **M7**: adição de três vértices (v,j,k) em S.
- **M8**: remoção de três vértices (v,j,k) de S.
- **M9**: troca de posição entre seis vértices (i,j,k,l,m,n) em S.

O método VNS interrompe a sua execução quando nenhum vizinho gerado pelas estruturas descritas apresenta melhorias se comparado à solução corrente. No trabalho citado, os autores utilizam o método VND como procedimento de busca local nos vizinhos gerados. No VND, usado pelo CS, três heurísticas de refinamento são realizadas em busca de gerar um melhor vizinho de uma solução. Estas heurísticas utilizam os movimentos *Add-Step* e *Drop-Step*, propostos por Oliveira e Lorena (Oliveira and Lorena, 2004). Basicamente, o movimento *Add-Step* consiste em selecionar, dentre todos os vértices que não fazem parte de uma solução, o vértice que apresentar a maior economia de inserção e então inserí-lo na mesma. Já o movimento *Drop-Step* consiste em selecionar, dentre todos os vértices da solução, aquele que apresentar a maior economia na função da solução após a sua remoção e então removê-lo da mesma.

No método VND do trabalho, os dois movimento são utilizados pelas seguintes heurísticas:

- **SeqAdd**: adição de vértices em uma solução S ocorre enquanto as adições de vértices melhorarem o valor da função objetivo de S (adição sequencial de vértices em uma solução).

- **SeqDrop**: remoção de vértices de uma solução S, ocorre enquanto as remoções melhorarem o valor da função objetivo de S (remoção sequencial de vértices de uma solução).

- **AddDrop**: executar o SeqAdd e em sequência o SeqDrop em S (adição e remoção sequencial de vértices em uma solução).

A adição e a remoção de vértices, executadas pelos três movimentos do VND, são realizadas de acordo com dois movimentos Seq-Add e Drop-step. O método VND executa até que nenhum dos movimentos possam melhorar o valor de S.

4.2.2 CLUSTERIZAÇÃO (AI)

Paralelamente ao processo de geração de soluções, ocorre o processo de clusterização. As soluções geradas por ME são enviadas ao componente AI (agrupador iterativo), que agrupa estas soluções ao agrupamento mais próximo. Para se definir qual o agrupamento mais próximo, a distância considerada pelos autores foi a seguinte: o número de diferentes arcos presentes em uma solução e o centro de um cluster (representado por uma solução).

Como visto no capítulo anterior, ao se inserir uma solução em um agrupamento, o centro do mesmo passa por uma perturbação. No CS proposto por Chaves and Lorena (2008) o processo de reconexão de caminhos foi utilizado par realizar esta pertubação no centro do agrupamento.

4.2.3 ANALISADOR DE AGRUPAMENTOS (AA)

Sempre que o centro de um cluster é atualizado, o componente AA verifica se este cluster pode ser considerado promissor. No CS proposto por Chaves and Lorena (2008), um cluster é considerado promissor quando este alcança uma densidade igual ou superior a 2.5. A seguinte fórmula é usada para o cálculo da densidade D_i de um cluster i:

$$D_{(i)} = PD * \frac{NS}{clus} \tag{4.3}$$

Na fórmula de densidade o termo PD é um parâmetro de entrada relativo a densidade desejada, $clus$ é o número de clusters e NS é o número de soluções que foram geradas no intervalo de análise.

4.2.4 OTIMIZADOR LOCAL (OL)

No CS proposto, o processo de otimização do centro de um cluster i, é realizado através do método de busca local K-OPT, com $K = 2$.

4.2.5 PARÂMETROS

Os autores definiram os seguintes parâmetros para o método:

- **NS** = 200: número de soluções geradas a cada análise de cluster.
- **NC** = 20: número máximo de clusters.
- α = 20%: percentual utilizado para formação de LRC.

Algoritmo 20: *TS'*

Data: $(S,E,N,BenMin,MaxIter)$
Data: $(StopAlt,StopInv,StopRuim,\phi,\theta,TamMaxTabu,Tenure,\Omega)$
Result: (S^*)

inicio
  $ListaTabu \leftarrow \varnothing$;
  $S^*,S2^* \leftarrow S$;
  AtualizaFreq($S^*,FREQ$);
  $Iter,IterRuim,IterInviavel,IterAlterada \leftarrow 0$;
  enquanto *($Iter < MaxIter$)* **faça**
    AtualizaParametros($StopAlt,StopInv,StopRuim,\phi,\theta,IterRuim,IterAlterada,IterInviavel$);
    $S2 \leftarrow$ Inserção $(f_{(S^*)},f_{(S2^*)},ListaTabu)$;
    $S3 \leftarrow$ Remoção $(f_{(S^*)},f_{(S2^*)},ListaTabu)$;
    se *($S2$ ou $S3$ for viável e melhor que S^*)* **então**
      $f_{(S1)} \leftarrow \infty$
    senão
      $S1 \leftarrow$ Troca $(f_{(S^*)},f_{(S2^*)},ListaTabu)$;
    $Saux \leftarrow \text{argmin}(f_{(S1)},f_{(S2)},f_{(S3)})$;
    $S4 \leftarrow$ Ajuste $(f_{(S^*)},\Omega)$;
    se *($f_{(S4)} < f_{(S^*)}$) e ($f_{(S4)} < f_{(Saux)}$)* **então**
      $S^*,S2^*,S \leftarrow S4$;
      AtualizaFreq($S^*,FREQ$);
      $IterRuim \leftarrow 0$;
    senão
      se *($Saux$ satisfizer C1)* **então**
        Insere movimento realizado em LT;
        $IterRuim \leftarrow 0$;
        $S^*,S2^*,S \leftarrow Saux$;
        AtualizaFreq($S^*,FREQ$);
      senão se *(($Saux$ for viável) e (movimento $\notin LT$) e ($\theta = 1$))* **então**
        Insere movimento em LT;
        $IterRuim++$;
        $S \leftarrow Saux$;
      senão se *(C2 for satisfeito)* **então**
        $IterRuim++$;
        Insere movimento que gerou $Saux$ em LT;
        AtualizaContadores($IterAlterada,\theta,IterInviavel,\phi,Saux$);
        $S,S2^* \leftarrow Saux$;
      senão
        $v \leftarrow$ SelecionarFrequente($S,FREQ$);
        Inserção GENI(v,S);
        $\theta,\phi \leftarrow 1$;
      AtualizaListaTabu($LT,TamMaxTabu$);
    $Iter++$;
  retorna (S^*);

Algoritmo 21: *AtualizaParametros*

Data: (ϕ,α,$IterRuim$,$StopRuim$,$IterAlterada$,$StopAlterada$,$IterInviavel$,$StopInviavel$)

```
1  inicio
2      se (IterAlterada ≥ StopAlterada) então
3          φ ← 1;
4          IterAlterada ← 0;
5      se (IterInviavel ≥ StopInviavel) então
6          α ← 1;
7          IterInviavel ← 0;
8      se (IterRuim ≥ StopRuim) então
9          φ,α ← 0;
10         IterRuim,IterAlterada,IterInviavel ← 0;
```

Algoritmo 22: *AtualizaFreq*

Data: (S,$FREQ$)

```
1  inicio
2      para todo v ∈ S faça
3          FREQ[v] + +;
```

Algoritmo 23: *SelecionarFrequente*

Data: (S,$FREQ$)
Result: v

```
1  inicio
2      Maior ← 0;
3      para todo (a ∈ FREQ) faça
4          se (FREQ[a] > Maior) e (a ∉ S) então
5              v ← a;
6      retorna (v);
```

Algoritmo 24: *M1 (Troca)*

Data: $(f_{(S^*)}, f_{(S2^*)}, LT)$
Data: $(S, E, BenMin, \alpha, \phi)$
inicio
  $x^*, x2^* \leftarrow -1$;
  $y^*, y2^* \leftarrow -1$;
  $f_{(x^*)}, f_{(x2^*)} \leftarrow \infty$;
  para todo $x \in E, \notin S$ **faça**
    para todo $y \in S$ **faça**
      $Saux \leftarrow S$;
      Retirar y de $Saux$;
      Colocar x em $Saux$;
      $Saux \leftarrow$ OPT2($Saux$,$Distancias$);
      $ben \leftarrow$ (Beneficios recolhidos por $Saux$);
      se *(*$ben > Benmin$*) e (*$\phi = 1$*)* **então**
        se *(* $(f_{(Saux)} < f_{(x^*)})$ *e* $(f_{(Saux)} < f_{(S^*)})$ *)* **então**
          $x^* \leftarrow x$;
          $y^* \leftarrow y$;
          $f_{(x^*)} \leftarrow f_{(Saux)}$;
        senão se $(f_{(Saux)} < f_{(x^*)})$ **então**
          se *(Trocar x com y não estiver em LT)* **então**
            $x^* \leftarrow x$;
            $y^* \leftarrow y$;
            $f_{(x^*)} \leftarrow f_{(Saux)}$;
      senão se $(x^* = -1)$ **então**
        se $(f_{(Saux)} < f_{(S2^*)})$ **então**
          $x2^* \leftarrow x$;
          $y2^* \leftarrow y$;
          $f_{(x2^*)} \leftarrow f_{(Saux)}$;
  se $(x^* > -1)$ **então**
    Colocar x^* em S;
    Retirar y^* em S;
    $A \leftarrow x^*$;
    $B \leftarrow y^*$;
    $movimento \leftarrow troca$;
  senão se $(x2^* > -1)$ **então**
    Colocar $x2^*$ em S;
    Retirar $y2^*$ em S;
    $A \leftarrow x2^*$;
    $B \leftarrow y2^*$;
    $movimento \leftarrow troca$;
  Aplicar 2-OPT em S;
  retorna (S);

Algoritmo 25: *M3 (Inserção)*

```
   Data: (f_(S*),f_(S2*),LT)
   Data: (S,E,BenMin,α,φ)
 1 inicio
 2     x*,x2* ← −1;
 3     f_(x*),f_(x2*) ← ∞;
 4     para todo x ∈ E faça
 5         se x ∉ S então
 6             Saux ← S;
 7             Colocar x em S_(aux);
 8             Aplicar 2-OPT em S_(aux);
 9             ben ← (Beneficios recolhidos por Saux);
10             se (ben > Benmin) e (φ = 1) então
11                 se ( (f_(Saux) < f_(x*)) e (f_(Saux) < f_(S*)) ) então
12                     x* ← x;
13                     f_(x*) ← f_(Saux);
14                 senão se (f_(Saux) < f_(x*)) então
15                     se (Inserir x não estiver em LT) então
16                         x* ← x;
17                         f_(x*) ← f_(Saux);
18             senão se (x* = −1) então
19                 se (f_(Saux) < f_(S2*)) então
20                     x2* ← x;
21                     f_(x2*) ← f_(Saux);
22     se (x* > −1) então
23         Colocar x* em S;
24     senão se (x2* > −1) então
25         Colocar x2* em S;
26     Aplicar 2-OPT em S;
27     retorna (S);
```

Algoritmo 26: *M2 (Remoção)*

Data: $(f_{(S^*)}, f_{(S2^*)}, LT)$
Data: $(S, E, BenMin, \alpha, \phi)$

```
1  inicio
2      x*, x2* ← −1;
3      f_(x*), f_(x2*) ← ∞;
4      para todo x ∈ S faça
5          Saux ← S;
6          Retirar x de S_(aux);
7          Aplicar 2-OPT em S_(aux);
8          ben ← (Beneficios recolhidos por Saux);
9          se (ben > Benmin) e (φ = 1) então
10             se ( (f_(Saux) < f_(x*)) e (f_(Saux) < f_(S*)) ) então
11                 x* ← x;
12                 f_(x*) ← f_(Saux);
13             senão se (f_(Saux) < f_(x*)) então
14                 se (Remover x não estiver em LT) então
15                     x* ← x;
16                     f_(x*) ← f_(Saux);
17         senão se (x* = −1) então
18             se (f_(Saux) < f_(S2*)) então
19                 x2* ← x;
20                 f_(x2*) ← f_(Saux);
21     se (x* > −1) então
22         Retirar x* de S;
23     senão se (x2* > −1) então
24         Retirar x2* de S;
25     Aplicar 2-OPT em S;
26     retorna (S);
```

Algoritmo 27: *M4 (Ajuste)*

Data: (S,E,$BenMin$,α,ϕ,Ω)

inicio
  $Saux \leftarrow S$;
  Exclua os Ω últimos vértices de $Saux$;
  $ben \leftarrow$ (Beneficios recolhidos por $Saux$);
  se *(*$ben > Benmin$*) e (*$\phi = 1$*)* **então**
    Aplicar 2-OPT em $Saux$;
    retorna ($Saux$);
  retorna (S);

Capítulo 5

EXPERIMENTOS COMPUTACIONAIS

O algoritmo proposto foi codificado em C++ utilizando a ferramenta de desenvolvimento Dev C++ 4.9.9.2. Os testes foram realizados em um microcomputador com processador Core i5-2520M (2.50 GHz), com 4 GB de memória ram e sistema operacional Windows 7 Ultimate de 64 bits. Apesar do computador ter cinco núcleos o algoritmo não foi implementado fazendo uso de multiprocessamento.

O intuito dos testes é validar a abordagem proposta como um bom método heurístico para solucionar o PCVCP, principalmente para grandes instâncias em que a resolução exata é inviável.

A biblioteca de testes utilizada se encontra disponível em http://www.lac.inpe.br/ lorena/intancias.html. Foram testados problemas com $N = \{40,60,80,100,200,300,400,500\}$. Para facilitar a análise dos resultados, os mesmos são classificados em três grupos diferentes $\{A,B\ e\ C\}$, sendo cada um dos grupos com características específicas, como mostrado na sequência:

Grupo A:

- Custos de Deslocamento $C_{i,j} \in [1{,}1000], \forall i,j \in V$.
- Penalidades $\gamma_i \in [1{,}100], \forall i \in V$.
- Prêmios $W_i \in [1{,}100], \forall i \in V$.

Grupo B:

- Custos de Deslocamento $C_{i,j} \in [1{,}10000], \forall i,j \in V$.
- Penalidades $\gamma_i \in [1{,}1000], \forall i \in V$.
- Prêmios $W_i \in [1{,}100], \forall i \in V$.

Grupo C:

- Custos de Deslocamento $C_{i,j} \in [1{,}10000], \forall i,j \in V$.
- Penalidades $\gamma_i \in [1{,}100], \forall i \in V$.
- Prêmios $W_i \in [1{,}100], \forall i \in V$.

Cada um dos grupos acima foram testados com valores de benefício mínimo ($Pmin$) variando no intervalo $\{0.2; 0.5; 0.8\}$. A biblioteca de testes foi a mesma utilizada por (Chaves and Lorena, 2008), sendo este o trabalho o qual contém os resultados exatos e heurísticos usados como base comparativa nesta dissertação. Para calibragem dos parâmetros foram realizados diversos testes, os testes de calibragem foram realizados apenas com amostragem de 30% do total das instâncias. Os parâmetros do TS' foram calibrados da seguinte forma:

- TamMaxTS = N;
- Tenure = valor aleatório sorteado no seguinte intervalo [1,10];
- Máximo de iterações (MaxIter)= 2000.

- Tempo Máximo de execução= $2 * N$.
- $\Omega = \frac{1}{5}$ do tamanho de uma solução.
- $P = 3$. (Quantidade de vizinhos avaliados na inserção do método GENI).
- StopRuim (TS')= 150.
- StopAlt (TS')= $\frac{N}{2}$.
- StopInv (TS')= $\frac{N}{2}$.

5.1 TS' x OUTROS MÉTODOS

Nesta seção os resultados obtidos pelo algoritmo proposto são comparados com outros métodos da literatura. Para fins de validação do TS aqui proposto, são utilizados os resultados do método GRASP/VNS com e sem agrupamento de soluções, e também os dados obtidos através da resolução exata do problema.As tabelas 5.1 a 5.9 apresentam os resultados obtidos pelo método TS' para o PCVCP, nestas tabelas são apresentados os resultados exatos e heurísticos obtidos no trabalho de (Chaves and Lorena, 2008). Nas tabelas tambem são apresentados os resultados obtidos para cada uma das instâncias (A,B e C), sendo que para cada uma das instâncias é avaliado o desempenho do algoritmo com entrada de benefício mínimo diferente, com valor de $Pmin$ variando entre $\{0.2; 0.5; 0.8\}$. Todos os testes foram executados 30 vezes, sendo que as tabelas reportam os melhores resultados obtidos e os resultados médios. É importante observar que reporta-se também o tempo médio de execução, mas devido ao fato dos testes terem sido rodados em máquinas com configurações diferentes fica díficil analisar precisamente estes dados, apesar de que os maiores desvios são facilmente visualizados. Ainda em relação ao tempo de execução nos testes realizados por (Chaves and Lorena, 2008) o tempo máximo de execução do método CPLEX foi de 100,000.00 segundos.

Os valores de desvio médio (DM e DMin) apresentados nas tabelas foram calculados a partir das seguintes fórmulas:

$DM = (FOmed - FOmin)/FOmin\ x\ 100.$

$DMin = (FOmed - FO^*)/FO^*\ x\ 100.$

O valor de FO^* é definido como o menor valor de solução encontrada pelos três métodos (exato, CS e TS').

A seguir o significado das colunas das tabelas é mostrado:

- Número de vértices N(quantidade de cidades).
- O valor inteiro da melhor solução encontrada pelo CPLEX (FOmin), o desvio médio em relação ao menor valor de função objetivo encontrado (DMin) e a quantidade de segundos do tempo de execução (T).
- Melhor solução (FOmin), valor médio das soluções (FOmed), tempo médio de execução (TM) encontrado pelo TS', e os valores de desvio médio em relação ao menor valor de solução encontrado pelo TS' e em relação ao menor valor de solução encontrado pelos três métodos (CPLEX, TS' e CS).
- Melhor solução (FOmin), valor de solução médio (FOmed), tempo médio de execução (TM) encontrado pelo CS, e os valores de desvio médio em relação ao menor valor de solução encontrado pelo CS e em relação ao menor valor de solução encontrado pelos três métodos (CPLEX, TS' e CS).

A tabela 5.1 mostra os resultados relativos aos testes das instâncias do grupo A com $Pmin = \{0.2\}$. A partir destes resultados é possível observar um desempenho superior do método TS' para gerar boas soluções para as instâncias com valores de N a partir de 200, que naturalmente são as de maior complexidade. Por exemplo para a instância com $N = 500$ o valor médio de função objetivo encontrado pelo TS' fica cerca de 30% menor que o valor médio obtido pelo CS. Para as menores instâncias o CS

apresenta desempenho superior ao TS', com uma pequena vantagem de desempenho (valores próximos a 2%).

Apesar da impossibilidade da análise direta em relação ao tempo de execução é nítido um melhor desempenho do CS para menores instâncias quando comparado ao TS' e ao CPLEX, enquanto para grandes instâncias o TS' tem nitidamente um desempenho melhor do que o CS, enquanto CPLEX não consegue nem mesmo gerar soluções viáveis para grandes instâncias em tempo hábil.

Tabela 5.1: Conjunto de dados A com (Benmin= 20%)

	CPLEX			TS'					CS				
N	FOmin	DMin	T(s)	FOmin	FOmed	TM(s)	DM	DMin	FOmin	FOmed	TM(s)	DM	DMin
40	996	0.00	20.46	996	996.0	11.47	0.00	0.00	996	996.0	2.28	0.00	0.00
60	1314	0.00	474.94	1314	1319.3	75.29	0.40	0.40	1314	1314.0	13.73	0.00	0.00
80	1384	0.00	26692.93	1386	1416.3	160.22	2.18	2.33	1386	1392.8	83.03	0.49	0.63
100	1514	0.39	100,000.00	1514	1557.6	200.22	2.87	3.28	1508	1526.4	196.10	1.22	1.22
200	-	-	-	1822	1844.4	402.94	1.22	1.56	1816	1834.4	502.94	1.01	1.01
300	-	-	-	1999	2090.7	620.46	4.58	4.58	2281	2313.0	1069.11	1.40	15.7
400	-	-	-	2307	2422.25	913.26	4.99	4.99	2504	2554.2	1212.85	2.00	10.71
500	-	-	-	2400	2534.65	1276,24	5.61	5.61	3233	3281.1	1355.62	1.48	36.71
				Média	DMin	TS'= 2,84%			Média	DMin	TS'= 8,25%		

A tabela 5.2 mostra os resultados relativos aos testes das instâncias do grupo A com $Pmin = \{0.5\}$. Os resultados da tabela 5.2 se mostram muito similares aos resultados apresentados na tabela 5.1. Novamente o TS' apresenta melhores resultados para maiores instâncias (valores de N iguais ou acima de 200), chegando novamente a valores próximos de 30% de diferenças de desvio médio obtidos se comparados ao CS. Em contrapartida, uma superioridade do CS é notado na resolução das menores instâncias, para estas novamente é observado menores diferenças entre os resultados das duas abordagens (TS' x CS), ficando a diferença dos desvios médios sempre menor do que 3% quando avaliadas as menores instâncias.

Nestes testes fica mais nítido a diferença de desempenho em relação aos tempos de execução, ficando o TS' com tempos médios muito acima para menores instâncias, e muito abaixo para maiores instâncias quando comparado ao CS. Novamente o CPLEX somente conseguiu resolver menores instâncias e com um tempo médio maior que o TS' e que o CS.

Tabela 5.2: Conjunto de dados A com (Benmin= 50%)

	CPLEX			TS'					CS				
N	FO	DMin	T(s)	FOmin	FOmed	TM(s)	DM	DMin	FOmin	FOmed	TM(s)	DM	DMin
40	996	0.00	21.84	996	996.0	4.36	0.00	0.00	996	996.0	2.34	0.00	0.00
60	1314	0.00	468.51	1314	1322.2	81.31	0.62	0.62	1314	1314.0	11.42	0.00	0.00
80	1384	0.00	32121.21	1385	1412.2	160.12	1.96	2.04	1388	1396.6	72.74	0.61	0.91
100	1514	0.00	100,000.00	1514	1577.6	202.29	4.20	4.26	1513	1534.6	181.00	1.42	1.42
200	-	-	-	1815	1919.4	417.16	5.75	5.75	1816	1844.2	572.46	1.55	1.60
300	-	-	-	1999	2093,1	613.92	4.70	4.70	2171	2250.5	1213.63	3.66	12.58
400	-	-	-	2326	2396,2	917.13	3.02	3.02	2489	2579.7	1490.49	3.64	10.90
500	-	-	-	2400	2524.9	1150.56	5.20	5.20	3159	3200.7	1784.58	1.32	33.36
				Média	DMin	TS'= 3,20%			Média	DMin	TS'= 7,60%		

A tabela 5.3 mostra os resultados relativos aos testes das instâncias do grupo A com $Pmin$ igual a $\{0.8\}$. Os resultados da tabela 5.3 se mostram muito similares aos resultados apresentados nas tabelas 5.1 e em 5.2. Novamente o TS' apresenta melhores resultados para maiores instâncias (valores de N iguais ou acima de 200), chegando novamente a valores próximos de 30% de diferenças de desvio médio obtidos se comparados ao CS, mostrando uma grande vantagem do TS para resolução das grandes instâncias do grupo A. Novamente o CS encontrou na média os melhores resultados para as menores instâncias, com diferença de desvios menores do que 2% quando comparado ao TS'. Mais uma vez fica nítido a diferença de desempenho em relação aos tempos de execução, ficando o TS' com tempos médios de execução acima para menores instâncias, e muito abaixo para maiores instâncias quando comparado ao

CS. Novamente o CPLEX somente conseguiu resolver apenas menores instâncias e com um tempo médio maior que o TS' e que o CS.

Tabela 5.3: Conjunto de dados A com (Benmin= 80%)

	CPLEX			TS'					CS				
N	FO	DMin	T(s)	FOmin	FOmed	TM(s)	DM	DMin	FOmin	FOmed	TM(s)	DM	DMin
40	1129	0.00	44.69	1129	1135.6	18.17	0.58	0.58	1129	1137.0	2.95	0.70	0.70
60	1319	0.00	474.80	1319	1348.2	117.45	2.21	2.21	1319	1344.2	17.46	1.91	1.91
80	1384	0.00	27498.29	1385	1410.6	160.12	1.84	1.92	1396	1400.2	63.86	0.30	1.17
100	1575	3.68	100,000.00	1519	1567.2	200.94	3.17	3.17	1519	1537.6	186.54	1.22	1.22
200	-	-	-	1769	1870.4	403.47	5.73	5.79	1768	1797.2	805.01	1.65	1.65
300	-	-	-	1999	2095.1	609.22	4.80	4.80	2148	2213.0	1528.29	3.02	10.70
400	-	-	-	2274	2399.2	915.87	5.50	5.50	2455	2494.3	1668.90	1.60	14.08
500	-	-	-	2400	2542.4	1306.14	5.93	5.93	3214	3324.2	1708.36	3.42	38.50
				Média	DMin	TS'= 3,74%			Média	DMin	TS'= 8,74%		

As tabelas 5.4, 5.5 e 5.6 mostram os resultados relativos aos testes das instâncias do grupo B com $Pmin$ variando no intervalo $\{0.2; 0.5; 0.8\}$. As três tabelas apresentam resultados similares, sendo novamente possível observar uma maior capacidade do TS' em gerar boas soluções para maiores instâncias (valores de N acima de 200), tanto em questão de valores médio quanto de menores valores de função objetivo alcançado. Para maiores instâncias a diferença dos resultados obtidos novamente chega a superar a casa dos 30% se compararmos os resultados obtidos pelo TS' com os obtidos pelo CS. Além de gerar melhores soluções para as grandes instâncias, o TS' faz isto em tempo médio muito menor do que o CS. Novamente nota-se um melhor desempenho do CS para gerar soluções para as menores instâncias, mas com uma vantagem muito menor do que a obtida pelo TS' para as grandes instâncias. Novamente o CPLEX conseguiu gerar soluções em tempo hábil apenas para menores instâncias, fazendo isto com um tempo execução em uma ordem de grandeza muito maior do que o tempo gasto pelo TS' e pelo CS.

Tabela 5.4: Conjunto de dados B com (Benmin= 20%)

	CPLEX			TS'					CS				
N	FO	DMin	T(s)	FOmin	FOmed	TM(s)	DM	DMin	FOmin	FOmed	TM(s)	DM	DMin
40	10776	0.00	21.97	10776	10776.6	19.20	0.00	0.00	10776	10776.0	3.19	0.00	0.00
60	14236	0.00	1151.37	14243	14391.8	120.08	1.04	1.09	14243	14314.1	7.11	0.49	0.73
80	14484	0.00	68464.35	14690	14951.6	160.13	1.78	3.22	14609	14760.9	114.86	1.03	1.91
100	14841	8.96	100,000.00	13768	14189.3	200.48	3.05	4.17	13620	14015.0	104.17	2.90	2.90
200	-	-	-	15467	16111.6	402.56	4.16	5.28	15303	15628.2	528.53	2.12	2.12
300	-	-	-	20477	21329.8	616.67	4.16	4.16	21869	22158.0	662.13	1.32	8.20
400	-	-	-	19093	20267.5	897,25	6.15	6.15	24390	25099.6	1354.40	2.90	31.45
500	-	-	-	23050	23901.4	1393,30	3.69	3.69	31090	31558.7	1643.15	1.50	36.91
				Média	DMin	TS'= 03,47		%	Média	DMin	TS'= 10,53		%

Tabela 5.5: Conjunto de dados B com (Benmin= 50%)

	CPLEX			TS'					CS				
N	FO	DMin	T(s)	FOmin	FOmed	TM(s)	DM	DMin	FOmin	FOmed	TM(s)	DM	DMin
40	10776	0.00	21.95	10776	10776.6	9.89	0.00	0.00	10776	10776.0	5.58	0.00	0.00
60	14236	0.00	1152.99	14300	14422.0	120.10	0.85	1.30	14349	14421.9	13.73	0.50	1.30
80	14484	0.00	68464.35	14690	14929.8	160.10	1.63	3.07	14512	14830.0	111.92	1.02	2.38
100	14841	9.78	100,000.00	13518	14162.8	200.56	4.76	4.76	13900	14089.1	118.35	1.36	4.22
200	-	-	-	15383	15970.9	402.47	3.82	5.14	15190	15440.4	664.17	2.30	2.30
300	-	-	-	20477	21285.6	610.13	3.94	3.94	22731	23211.7	696.75	2.11	13.35
400	-	-	-	19820	20517.5	977.60	3.51	3.51	23898	24525.3	1755.43	2.62	23.74
500	-	-	-	22717	23730.5	1194.08	4.46	4.46	30275	30842.0	2173.11	1.87	35.76
				Média	DMin	TS'= 03,27		%	Média	DMin	TS' = 10,38		%

Tabela 5.6: Conjunto de dados B com (Benmin= 80%)

	CPLEX			TS'					CS				
N	FO	DMin	T(s)	FOmin	FOmed	TM(s)	DM	DMin	FOmin	FOmed	TM(s)	DM	DMin
40	10776	0.00	31.32	10776	10784.2	14.33	0.00	0.00	10776	10776.0	4.66	0.00	0.00
60	14864	0.00	18508.17	14864	15085.0	111.80	1.48	1.48	14864	15017.1	32.74	1.03	1.03
80	14484	0.00	70205.22	14712	14957.4	160.10	1.66	3.26	14740	14793.9	92.77	0.36	2.13
100	17316	28.09	100,000.00	13518	14071.8	200.35	4.09	4.09	13704	13971.9	95.31	2.04	3.35
200	-	-	-	15383	16022.0	402.60	4.15	5.40	15200	15376.2	716.94	1.15	1.15
300	-	-	-	20589	21396.9	613.78	3.92	3.92	22168	22467.7	1496.77	1.35	9.12
400	-	-	-	19547	20376.2	879.53	4.24	4.24	22790	23688.3	1953.06	3.94	21.18
500	-	-	-	23214	24062.7	1362.83	3.65	3.65	30385	30707.0	2336.85	1.05	32.27
				Média	DMin	TS'= 03,26		%	Média	DMin	TS' = 8,78		%

As tabelas 5.7, 5.8 e 5.9 mostram os resultados relativos aos testes das instâncias do grupo C com $Pmin$ variando no intervalo $\{0.2; 0.5; 0.8\}$. As três tabelas apresentam resultados similares, sendo novamente possível observar uma maior capacidade do TS' em gerar boas soluções para maiores instâncias (valores de N acima de 200), tanto em questão de valores médio quanto de menores valores de função objetivo alcançados. Para maiores instâncias a diferença dos resultados obtidos novamente chega a superar a casa dos 30% se compararmos os resultados obtidos pelo TS' com os obtidos pelo CS. Além de gerar melhores soluções para as grandes instâncias o TS' faz isto em tempo médio muito menor do que o CS. Novamente é possível avaliar um melhor desempenho do CS para gerar soluções para as menores instâncias, mas com uma vantagem muito menor do que a obtida pelo TS' para as grandes instâncias. Novamente o CPLEX conseguiu gerar soluções em tempo hábil apenas para menores instâncias, fazendo isto com um tempo execução em uma ordem de grandeza muito maior do que o tempo gasto pelo TS' e pelo CS.

A tabela 5.7 mostra os resultados relativos aos testes das instâncias do grupo C com $Pmin = \{0.2\}$. Novamente é possível verificar uma maior capacidade do TS' em gerar

boas soluções para grandes instâncias (com valores de N superiores a 200) quando comparado ao CS, contudo com desvios médios variando sempre com uma diferença menor do que 3%, diferentemente dos testes para as instâncias dos outros grupos onde esse valor chegou a superar a casa dos 30%. Para menores instâncias o CS novamente alcançou melhores resultados, com exceção da intância com valor de $N = 100$, pois nesta o TS' apresentou resultados ligeiramente melhores do que o CS. Diferentemente dos resultados obtidos para os outros grupos, para as instâncias do grupo C é possível observar menores valores de tempo médio de execução do CS, apesar das diferenças das máquinas onde os testes foram realizados.

Tabela 5.7: Conjunto de dados C com (Benmin= 20%)

	CPLEX			TS'					CS				
N	FO	DMin	T(s)	FOmin	FOmed	TM(s)	DM	DMin	FOmin	FOmed	TM(s)	DM	DMin
40	3506	0.00	26.06	3506	3508.1	1.99	0.00	0.00	3506	3531.6	0.67	0.71	0.71
60	4251	0.00	130.31	4251	4289.5	14.98	0.90	0.90	4277	4287.2	14.86	0.23	0.85
80	4903	0.00	2226.51	4906	5100.5	107.62	3.96	4.02	4903	5044.3	22.61	2.88	2.88
100	5635	0.00	10824.22	5666	5789.1	152.27	2.36	2.93	5702	5802.0	59.71	1.75	2.96
200	-	-	-	8931	9474.8	401.65	6.08	6.08	9035	9129.0	303.41	1.04	2.21
300	-	-	-	13600	14139.6	665.89	3.96	3.96	14592	14875.2	319.14	1.94	9.37
400	-	-	-	16203	17027.9	816.99	5.09	5.09	16651	16850.3	640.17	1.19	3.99
500	-	-	-	19131	20793.7	1024.33	8.69	8.69	20612	21305.7	836.68	3.36	11.36
				Média	DMin	TS'= 03,96		%	Média	DMin	TS' = 04,29		%

As tabelas 5.8 e 5.9 apresentam os resultados obtidos para os testes das instâncias do grupo C com benefício mínimo $Pmin$ valendo {0.5} e {0.8} respectivamente. Diferentemente dos testes anteriores é possível observar uma maior capacidade do CS em gerar boas soluções para praticamente todas as instâncias, com exceção das instâncias com valores de N igual a 40 e da instância com $N = 500$ e benefício mínimo $Pmin = 0.8$. Para a maior parte das instâncias testadas nas tabelas 5.8 e 5.9 o TS' apresentou claramente um maior tempo médio de execução quando comparado ao CS, com exceção

para as maiores instâncias (valores de $N >= 200$) da tabela 5.9. Da mesma forma que nos testes anteriores o CPLEX mostrou capacidade de gerar boas soluções em tempo hábil apenas para instâncias menores, no entanto com um tempo de execução muito maior do que o TS' e do que o CS.

Tabela 5.8: Conjunto de dados C com (Benmin= 50%)

	CPLEX			TS'					CS				
N	FO	DMin	T(s)	FOmin	FOmed	TM(s)	DM	DMin	FOmin	FOmed	TM(s)	DM	DMin
40	4694	0.00	59.86	4694	4694.0	7.65	0.00	0.00	4694	4694.0	4.29	0.00	0.00
60	6120	0.00	700.35	6232	6542.0	109.99	4.97	6.89	6232	6361.7	14.05	2.08	3.94
80	6319	0.00	72518.87	6480	6769.5	160.09	4.46	7.12	6528	6628.1	63.69	1.53	4.89
100	6869	0.00	69562.82	7293	7615.9	200.00	4.42	10.87	7710	7833.7	119.52	1.60	14.04
200	-	-	-	10293	10592.1	403.39	2.90	2.90	10293	10578.0	438.80	2.76	2.76
300	-	-	-	15321	16615.1	615.32	8.44	8.51	15312	15698.3	549.65	2.52	2.52
400	-	-	-	17936	18942.6	822.45	5.61	9.72	17263	17535.7	841.70	1.57	1.57
500	-	-	-	20631	22410.2	1038.63	8.62	8.62	20896	21263.7	1056.08	1.75	3.06
				Média	DMin	TS'= 06,83		%	Média	DMin	TS' = 04,10		%

Tabela 5.9: Conjunto de dados C com (Benmin= 80%)

	CPLEX			TS'					CS				
N	FO	DMin	T(s)	FOmin	FOmed	TM(s)	DM	DMin	FOmin	FOmed	TM(s)	DM	DMin
40	9070	0.00	43.26	9070	9113.3	35.91	0.47	0.47	9070	9171.6	7.55	1.12	1.12
60	9459	0.00	10854.96	9904	10114.4	120.10	2.12	6.92	9664	9810.5	32.77	1.51	3.71
80	9699	0.00	98073.17	10102	10574.2	160.13	4.67	9.02	9991	10048.0	97.02	0.57	3.59
100	10002	0.00	100,000.00	10491	10921.8	200.26	4.10	4.10	10641	10724.1	157.96	0.78	2.22
200	-	-	-	13024	13573.4	406.18	4.21	7.29	12650	13024.0	616.84	2.95	2.95
300	-	-	-	17783	19358.1	607.29	8.85	8.85	18253	18740.8	1100.30	2.67	5.38
400	-	-	-	19485	21113.8	852.82	8.35	14.12	18501	18955.7	1574.56	2.45	2.45
500	-	-	-	21494	22858.4	1109.08	6.34	6.34	23234	23590.3	1920.92	1.53	9.75
				Média	DMin	TS'= 07,14		%	Média	DMin	TS' = 03,90		%

A partir da análise das tabelas acima fica claro uma maior capacidade do TS' em gerar melhores soluções para maiores instâncias do que o CS. No entanto o TS'

obteve uma menor eficiência quando analisados os resultados para as menores instâncias. Quando comparado com a resolução de forma exata, o método TS' se mostrou mais eficiente, gerando boas soluções para todas as instâncias, enquanto o CPLEX somente conseguiu gerar soluções para as menores instâncias.

Nitidamente o TS' mostrou melhor desempenho para as instâncias do grupo A e B, enquanto obteve um pior desempenho para as instâncias do grupo C. A partir das características dos grupos verifica-se que os grupos A e B têm uma maior importância de penalidade no valor da função objetivo. Considerando a relação entre o maior valor possível de penalidade de um vértice i e o maior valor de custo de deslocamento entre dois vértices i e j, para estes grupos a relação é de $\frac{1}{10}$. A definição deste mesmo valor de relação para o grupo C é de $\frac{1}{100}$. Entre os testes do grupo C, o TS' apresentou melhores resultados nos casos em que o valor de benefício mínimo foi igual a 0.2. A análise dos resultados indicia uma capacidade do método TS' em definir os vértices a serem inclusos na rota, devido a alta qualidade das soluções obtidas para instâncias com BenMin igual a 20% e 50%. Diminuindo dessa forma os custos de penalidade, e assim gerando boas soluções nos casos em que o custo de penalidade tenha grande importância no cálculo de função objetivo.

Também com o intuito de comparar a capacidade dos dois algoritmos (TS' e CS) em gerar boas soluções para o PCVCP, na figura 5.1 é mostrado o gráfico boxplot criado a partir dos dados desvios médios (DM e DMin)dos algoritmos TS' e CS. A primeira caixa no gráfico foi construída com os dados de DM do algoritmo TS', e a segunda dos dados de DMin do TS'. Similarmente a terceira caixa do gráfico foi construída com os dados de DM do algoritmo CS, e por último, a quarta caixa foi construída com os dados de DMin do CS.

A figura 5.1 mostra que tanto em relação ao DM (caixas 1 e 3), quanto em relação ao DMin (caixas 2 e 4), o método CS obteve valor de mediana mais baixo que o TS', em ambos os casos a diferença dos valores de mediana obtidos ficou abaixo da casa de

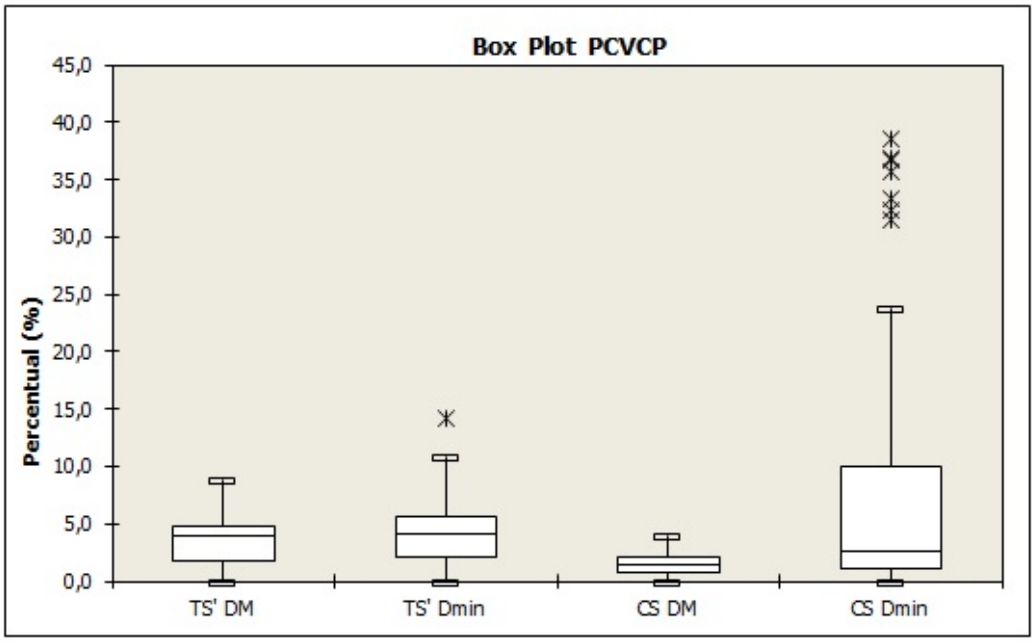

Figura 5.1: BoxPlot Desvios TS'x CS.

2%, o que indica um desempenho similar dos métodos. Este fato pode ser comprovado ao se analisar o primeiro quartil, que novamente mostra uma ligeira superioridade de desempenho do CS em relação ao TS', em ambos os casos menos que 1% de diferença. Ao analisarmos o terceiro quartil vemos melhores valores do CS em relação ao DM, mas em relação ao DMin vemos uma grande diferença favorável ao TS', fato que pode ser facilmente justificado pelas altas diferenças de valores de função objetivo obtidos para as maiores instâncias.

Ainda com o intuito de comparar a eficiência dos dois algoritmos é mostrada na figura 5.2 o gráfico com os valores médios de DMin de cada um dos métodos (TS' e CS). Os dados de entrada deste gráfico podem ser vistos na última linha de cada uma das nove tabelas mostradas anteriormente. Diferentemente do gráfico boxplot (Figura 5.1), no gráfico a seguir não é utilizado a Mediana para se avaliar a eficiência dos dois métodos, e sim a Média dos DMin obtidos em cada uma das tabelas acima. A partir

da análise da média podemos ver com mais clareza o impacto dos resultados extremos obtidos pelos dois algoritmos.

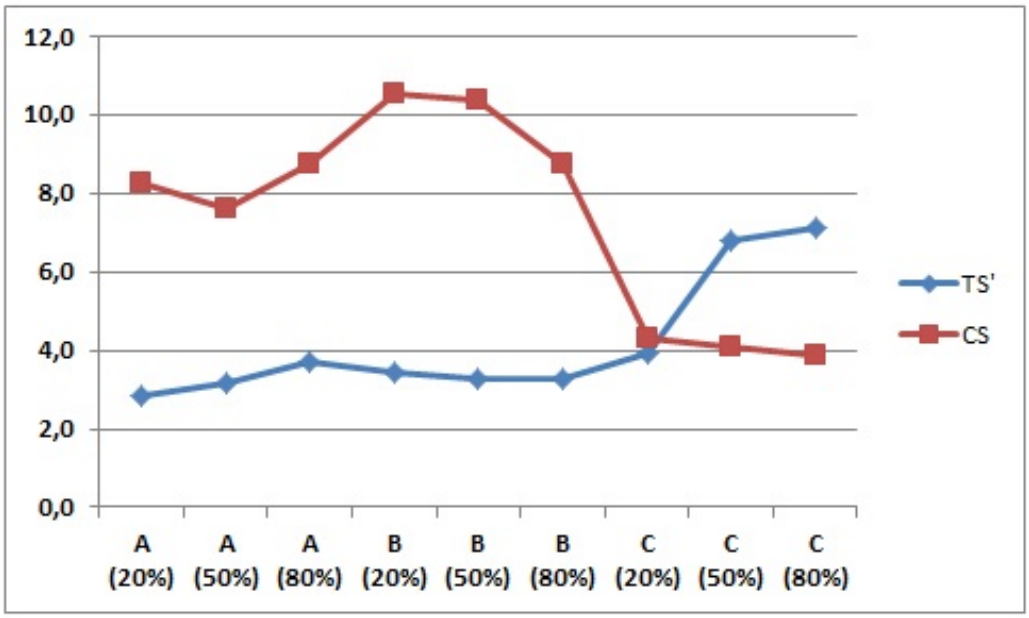

Figura 5.2: Valores Médios de DMin - TS'x CS.

Como pode ser visto na figura 5.2 o método TS' obteve menores valores de função objetivo médios para as instâncias do grupo A e para as instâncias do grupo B. Isto ocorre com todos os valores de benefício mínimo testado $\{0.2; 0.5; 0.8\}$. Isto também ocorre para as instâncias do grupo C com valor de benefício mínimo igual a 0.2. Essa superioridade de desempenho média se justifica principalmente devido aos altos valores de função objetivo obtidos pelo CS para as maiores instâncias destes grupos. Esses valores extremos impactaram significativamente nos valores médios obtidos. Para as instâncias do grupo C com benefício mínimo igual a $\{0.5; 0.8\}$, o método CS apresentou um melhor desempenho médio. Para as instâncias do grupo B o método TS' apresenta larga superiodade se comparado ao método CS, chegando ao extremo de apresentar valores de desvio médio 7.1% menores que o CS para as instâncias do grupo B com

benefício mínimo igual a 0.5. Já para as instâncias do grupo C é possível notar uma maior proximidade de desempenho de ambos os métodos, com uma nítida vantagem de desempenho para o método CS.

5.2 PRIORIZAÇÃO DE INSERÇÃO E REMOÇÃO

Uma das mais importantes características do PCVCP, já explicadas neste trabalho, é o fato de que uma boa heurística para resolvê-lo deve ter a capacidade de gerar soluções de tamanhos diferentes durante o processo de busca, pois diferentemente do PCV este problema pode ter soluções viáveis de diversos tamanhos. Durante a realização dos testes preliminares, para avaliação do método TS', ficou evidente que o processo de busca perdia muito tempo, principalmente no ínicio da busca, avaliando trocas entre vértices, enquanto um bom tamanho de solução ainda não havia sido encontrado. Este fato ficava ainda mais claro durante a execução das maiores instâncias com baixo $Pmin$, pois na maioria das vezes o tamanho da solução final era diferente do tamanho da solução inicial do processo de busca. Para sanar tal dificuldade foi avaliado e implementado a priorização de movimentos de inserção e remoção sobre movimentos de troca. Sendo assim durante o processo de busca em uma iteração um movimento de troca somente pode ser realizado caso nem o movimento de inserção, nem o movimento de remoção gere algum vizinho que melhore o valor de FO^*

Para mostrar a eficácia da priorização de movimentos realizou-se a análise de probabilidade empírica, apresentada na Figura 5.3. Para a realização desta análise foi utilizada uma instância do grupo C com valor de N igual a 40, e com $Pmin$ igual a 0.2. Como critério de parada foi utilizado o encontro do valor de função objetivo ótima da instância, encontrada pelo CPLEX, ou a realização do número máximo de iterações durante o processo de busca tabu.

Para a geração do gráfico de distribuição de probabilidade em relação ao tempo

(Figura 5.3) empregou-se um método de análise empírica utilizado por Aiex et al. (2002). Basicamente para fazer a análise da distribuição empírica o método foi executado 100 vezes, e os tempos de execução foram ordenados de forma crescente $T = (t_1, t_2,...,t_{100})$, e cada um dos testes foi associado a um valor i correspondente a posição do teste após a ordenação. Em seguida foi calculada a probabilidade acumulada $Prob_i = (i - 1/2)/100$ associada ao i-ésimo tempo de execução. Após isto os 100 pontos $Z_i = t_i, Prob_i$ foram plotados na Figura 5.3.

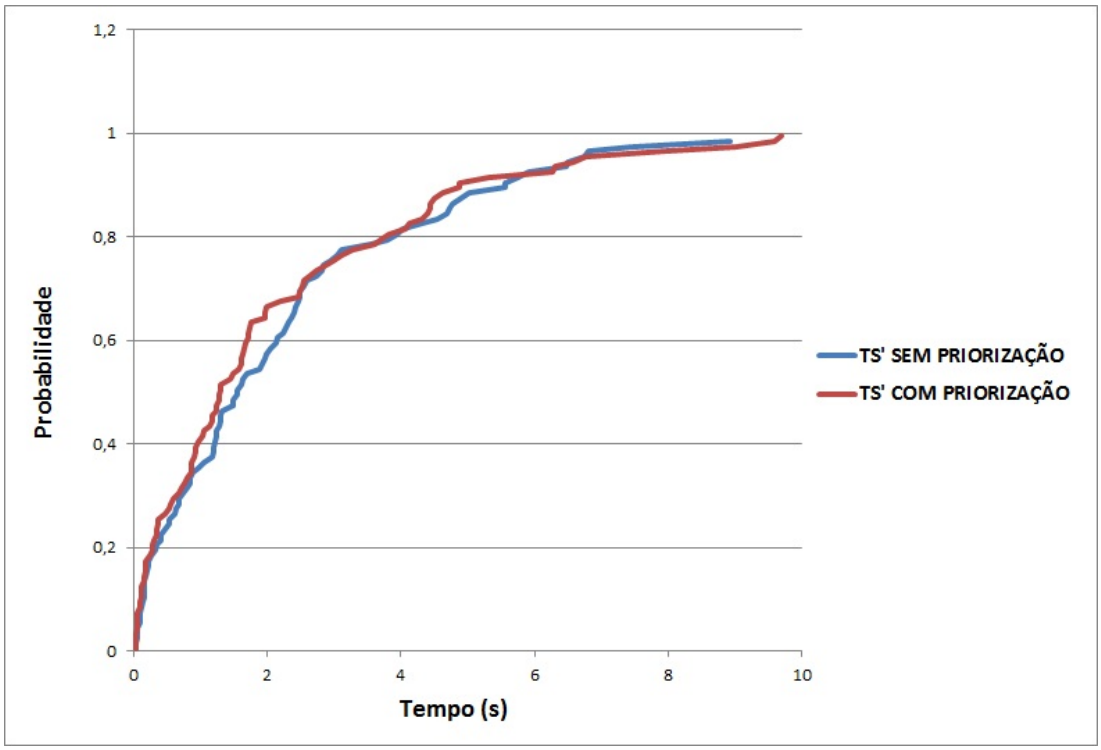

Figura 5.3: Gráfico comparativo TS' *com*/*sem* priorização de movimentos (Distribuição Empírica).

Ao se analisar a figura 5.3 é possível notar que o método com priorização de movimentos (linha vermelha) apresenta uma probabilidade, ligeiramente maior, de encontrar a solução mais rapidamente do que o método sem priorização. Isto se deve ao fato de que o método primeiramente prioriza a busca por um melhor tamanho, para somente após iniciar procedimentos de trocas de vértices. É importante ressaltar que durante as in-

serções e remoções realizadas o método de refinamento k-Optimal é executado em todas as iterações, fazendo com que a ordenação dos vértices na solução seja otimizada, além da busca por um bom tamanho. Apesar da baixa variação de desempenho observada na figura 5.3 entre os dois métodos, quando a comparação de desempenho é realizada avaliando os resultados obtidos para grandes instâncias a diferença amplifica-se, mostrando a priorização de movimentos ser uma excelente característica para a resolução das instâncias de grande porte (não foi feito a análise de distribuição empírica para grandes instâncias devido a falta de tempo hábil para realizar tais testes).

Capítulo 6

CONSIDERAÇÕES FINAIS

Este trabalho teve foco no desenvolvimento de uma metaheurística baseada na Busca Tabu para ser aplicada como método de resolução do Problema do Caixeiro Viajante com Coleta de Prêmios (PCVCP). O PCVCP é derivado do Problema do Caixeiro Viajante (PCV), se diferindo do problema original basicamente devido ao fato de que o caixeiro não é obrigado a visitar todas as cidades, mas apenas uma quantidade suficiente que lhe garanta a coleta de um prêmio mínimo. Sendo assim o objetivo do mesmo pode ser resumido em visitar uma quantidade de cidades que garanta o prêmio mínimo, mas fazendo isso de forma que o somatório dos custos de deslocamento e penalidades, pagas por não visitação, seja minimizado.

Devido ao fato, já explicado, da complexidade e custo computacional do PCVCP, é inviável resolvê-lo exatamente no caso de grandes instâncias, surgindo daí a necessidade e a possibilidade do desenvolvimento de abordagens viáveis de resolução, em especial as heurísticas. Como citado algumas abordagens heurísticas já foram propostas para solucionar o PCVCP. Este trabalho tem como foco principal enriquecer o universo das heurísticas apresentadas para resolver o problema.

O algoritmo proposto neste trabalho foi baseado na junção de três heurísticas de uso consagrado na literatura, o método GENIUS para construção da solução inicial, os

métodos K-optimal para refinamento de soluções e a metaheurística Busca Tabu como método de Busca Local. A Busca Tabu e o método Genius sofreram adaptações para serem aplicados ao PCVCP.

O método GENIUS foi adaptado basicamente para poder gerar soluções iniciais com tamanhos menores do que a quantidade de cidades candidatas, devido ao fato de que soluções ótimas do PCVCP podem ter esta característica.

A escolha da Busca Tabu (TS) como método de busca foi devido ao fato dos bons resultados relatados na literatura sobre sua aplicação em outros problemas oriundos do PCV, porém pelo que foi revisado foram raros os trabalhos relatando a aplicação da TS ao PCVCP. As adaptações propostas ao método incluem a priorização de movimentos de inserção e remoção. Outras adaptações propostas foram as duas técnicas de diversificação: uso de variáveis dinâmicas e de um movimento auxiliar de ajuste de tamanho. As mudanças mostraram ser adequadas ao problema, pois elas auxiliam o método a encontrar bons tamanhos para soluções, antes de realizar a otimização de vértices presentes nas mesmas.

Para validação do método proposto foi utilizado uma biblioteca pública de testes. As instâncias desta biblioteca são adaptações de instâncias do PCV. As instâncias originais do PCV, que foram adaptadas, são largamente utilizadas para validar métodos propostos na literatura do problema.

Também para validar o método aqui propostoos resultados obtidos foram comparados com um método da literatura que tem os melhores resultados reportados na literatura do problema, o que pode ser visto no capítulo anterior. O método usado como base comparativa é conhecido como CS (Clustering Search), e tem seu funcionamento baseado em agrupamento de soluções. Além dos conceito de agrupamento este método utiliza as heurísticas VNS/VND e K-Optimal para construir e refinar soluções.

Além da comparação com outro método heurístico (CS), também foi realizado comparações com os resultados obtidos por um software de resolução exata (CPLEX).

Tanto os resultados do CS quanto os do CPLEX foram retirados a partir do trabalho de Chaves and Lorena (2008).

Além das comparações com outros métodos e algoritmos da literatura, foi realizada a comparação do método TS' aqui proposto com e sem a característica de priorização de movimentos, comparação que validou a aplicação do algoritmo com esta característica.

O método TS' demostrou capacidade de gerar boas soluções para o PCVCP, fazendo isto de forma consideravelmente rápida, principalmente quando analisadas grandes instâncias. O método TS' alcançou um desvio médio de 4,1% (DM) em relação a melhor solução encontrada na literatura (incluindo a do própio TS'), e com um desvio médio interno de 3,9% (DMin). Quando comparado a resolução exata do problema (CPLEX), o método TS' se mostrou mais eficiente, gerando boas soluções para todas as instâncias enquanto o CPLEX solucionou apenas as instâncias menores e consumindo muito mais tempo para encontrar as soluções. Quando comparado ao CS o método TS se mostrou mais eficiente para gerar boas soluções para as maiores instâncias (N a partir de 300), e menos eficiente para gerar soluções para as menores instâncias.

Mesmo apresentando um menor desvio médio em relação a melhor solução encontrada na literatura (2,5%), o método CS gerou uma maior quantidade de instâncias que ficaram muito distantes deste menor valor. Enquanto o CS gerou 18 instâncias que tiveram um desvio médio, em relação ao mínimo, superior a 10% (outlier's), apenas em duas instâncias o TS' gerou soluções que superaram este limite, fato que pode ser visualmente verificado através do gráfico BoxPlot (Figura 5.1).

Outro fato que pode ser visto neste trabalho é o benefício gerado pelo inclusão da priorização de movimentos na busca tabu proposta, caracaterística que possibilita ao método um maior escape de ótimos locais e uma maior rapidez na geração da solução ótima, o que tambem pode ser visualmente verificado na distribuição empírica mostrada no capítulo anterior.

Em resumo, a adaptação de busca tabu (TS') se mostrou eficiente para solucionar o

PCVCP, principalmente em se tratando de instâncias de grande porte(com valores de N iguais ou superiores a 300), gerando na maioria das vezes as melhores soluções relatadas na literatura para tais problemas. Quanto as menores instâncias, mesmo não gerando os melhores resultados da literatura o método proposto gerou na grande maioria das vezes soluções com baixo desvio médio em relação ao mínimo conhecido. A partir da ideia que menores soluções podem ser resolvidas exatamente, e que o grande desafio é solucionar problemas que são inviáveis de se resolver dessa forma, a adaptação do método de busca tabu aqui proposta pode ser considerada uma das mais competitivas metodologias heurística já proposta para solucionar o PCVCP.

Diversos trabalhos futuros podem ser sugeridos, dentre eles: estudo da técnica de ajuste de tamanho buscando melhorias, por exemplo testar o ajuste dinâmico de Ω; aplicação do método proposto à problemas similares (PCVCP online, QTSP, entre outros) e em variantes do PCVCP que tenham funções objetivo e restrições diferentes das utilizadas neste trabalho; outras formas de se definir tamanho de soluções iniciais para o procedimento de busca (ex: definição de função relacionando os custos de deslocamento e penalidades), pois isto permitiria que o processo de busca fosse melhor guiado; a avaliação de paralelizar o algoritmo para aproveitar melhor os recursos computacionais dísponíveis (multi-processadores) dentre outras.

Referências Bibliográficas

Aiex, R. M., Resende, M. G. C., and Ribeiro, C. C. (2002). Probability distribution of solution time in grasp; an experimental investigation. *Journal of heuristics*, 8:343–373.

Ausiello, G., Bonifaci, V., and Laura, L. (2008). The online prize-collecting traveling salesman problem. *Information Processing Letters - IPL*, 107:199–204.

Awerbuch, B., Azar, Y., Blum, A., and Vempala, S. (1999). New approximation guarantees for minimum-weight k-trees and prize-collecting salesman. *SIAM Journal on Computing*, 28:254–262.

Balas, E. (1989). The prize collecting traveling salesman problem. *Networks*, 19:621–636.

Balas, E. and Martin, G. (1984). Roll-a-round: Software package for scheduling the rounds of a rolling mill.

Balas, E. and Simonetti, N. (2001). Linear time dynamic-programming algorithms for new classes of restricted tsps: A computational study. *INFORMS Journal on Computing*, 13:56–75.

Bienstock, D., Goemans, M., Simchi-Levi, D., and D., W. (1993). A note on the prize-collecting traveling salesman problem. *Mathematical Programming*, pages 413–420.

Blum, C., Aguilera, M., Roli, A., and Sampels, M., editors (2008). *Hybrid Metaheuristics, An Emerging Approach to Optimization*, volume 114 of *Studies in Computational Intelligence*. Springer.

Burkard, E., Deineko, V., Van Dal, R., Van Der Veen, J. A. A., and Woeginger, G. (1998). Well-solvable special cases of the traveling salesman problem: A survey. *SIAM Rev.*, 40:496–546.

Chaves, A. A., Biajoli, F. L., M., M. O., and Souza, M. J. F. (2007a). Modelagens exata e heurística para resolução de uma generalização do problema do caixeiro viajante. In *Anais 36 do Simpósio Brasileiro de Pesquisa Operacional*, pages 1367–1378, São João Del Rei, Brazil.

Chaves, A. A., Biajoli, F. L., Mine, O. M., and Souza, M. J. F. (2007b). Metaheurísticas híbridas para resolução do problema do caixeiro viajante com coleta de prêmios. *Produção*, 17:263 – 272.

Chaves, A. A. and Lorena, L. A. N. (2007). AplicaÇÃo do algoritmo clustering search aos traveling salesman problems with profits. In *Anais 39 do Simpósio Brasileiro de Pesquisa Operacional*, pages 1472–14830, Fortaleza, Brazil.

Chaves, A. A. and Lorena, L. A. N. (2008). Hybrid metaheuristic for the prize collecting travelling salesman problem. In Hemert, J. and Cotta, C., editors, *Evolutionary Computation in Combinatorial Optimization*, volume 4972 of *Lecture Notes in Computer Science*, pages 123–134. Springer Berlin Heidelberg.

Croes, G. (1958). A method for solving travelling salesman problems. *Operations Research*, 6:791–812.

Dantzig, G., Fulkerson, R., and Johnson, S. (1954). Solution of a large-scale traveling salesman problem. *Anals of Operations Research*, 2(1):393–410.

Dell'Amico, M., Maffioli, F., and Sciomanchen, A. (1998). A lagrangian heuristic for the prize collecting traveling salesman problem. *Anals of Operations Research*, 81(1):289–306.

Feo, T. A. and Resende, M. (1995). Greedy randomized adaptive search procedures. *Journal of Global Optimization*, 6:109–133.

Fischetti, M., Gonzales, J., and P., T. (1998). Solving the orienteering problem through branch-and-cut. *INFORMS Journal on Computing*, 10:133–148.

Fischetti, M. and Toth, P. (1988). *An additive approach for the optimal solution of the prize collecting traveling salesman problem*, pages 319–343. GOLDEN, B. L. and ASSAD, A. A.,North-Holland.

Gendreau, M. (2003). *An introduction to Tabu Search*, pages 37–54. Kluwer Academics Publishers,Boston, USA.

Gendreau, M., Hertz, A., and Laporte, G. (1992). New insertion and post optimization procedures to the traveling salesman problem. *Operations Research*, 40:1086–1094.

Gendreau, M., Hertz, A., and Laporte, G. (1996). The traveling salesman problem with backhauls. *Computers & Operations Research*, 23:501–508.

Glover, F. (1989). Tabu search - part i. *ORSA Journal on Computing*, 1(3):190–206.

Glover, F. and Laguna, M. (1997). *Tabu Search*. Kluwer Academic Publishers.

Goemans, M. X. and Williamson, D. P. (1995). Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *J. ACM*, 42:1115–1145.

Goldbarg, M. C. and Luna, H. P. (2000). *Otimização Combinatória e Programação Linear: Modelos e Algoritmos*. Editora Campos, third edition.

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison Wesley Longman, first edition.

Keller, C. P. and Goodchild, M. (1988). The multiobjective vending problem: a generalization of the traveling salesman problem. *Environment and Planning B: Planning and Design*, 15:447–460.

Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, 220:671–680.

Laporte, G. and Martello, S. (1990). The selective traveling salesman problem. *Discrete Applied Mathematics*, 26:193–207.

Larson, R. and Odoni, A. (1981). *Urban operations research*. Number Monograph.

Lawler, E., Lenstra, J., Rinnooy, K. A., and Shmoys, D. (1985). The traveling salesman problem. *NETWORKS*, 18:253–254.

Lin, S. (1965). Computer solutions of the traveling-salesman problem. *Bell System Technology Journal*, 44:2245–2269.

Lourenço, H. R. D., Martin, O. C., and Stutzle, T. (2003). *Iterated Local Search*, pages 321–353. Kluwer Academics Publishers,Boston, USA.

Melo, V. A. and Martinhon, C. A. (2004). Metaheurísticas híbridas para o problema do caixeiro viajante com coleta de prêmios. In *Anais 36 do Simpósio Brasileiro de Pesquisa Operacional*, pages 1295–1306, São João Del Rei, Brazil.

Mine, M. T., Silva, M. S. A., Ochi, L., and Souza, M. (2010). O problema de roteamento de veículos com coleta e entrega simultânea: Uma abordagem via iterated local search e genius. In *Transporte em Transformação XIV: Trabalhos Vencedores do Prêmio CNT de Produção Acadêmica 2009*, pages 59–78, Brasilia, Brazil. Editora Positiva.

Mladenovic, N. and Hansen, P. (1997). Variable neighborhood search. *Computers & Operations Research*, 24:1097–1100.

Moscato, P. (1989). On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms.

Nils, J. N. (1982). *Principles of Artificial Inteligence*. Morgan kaufmann, first edition.

Oliveira, A. C. M. and Lorena, L. A. N. (2004). Detecting promising areas by evolutionary clustering search. In *Brazilian Symposium on Artificial Intelligence - SBIA*, pages 385–394, Maranhão, Brazil.

Ong, H. (1982). Aproximate algorithms for the traveling purchaser problem. *Operationals Research Letters*, 1:201–205.

Rich, E. and Knight, K. (1993). *Inteligência Artificial*. Makron.

Steiglitz, K. and Weiner, P. (1968). Some improved algorithms for computer solution of the traveling salesman problem. In *Proceedings of the Sixth Allerton Conference on Circuit Theory*, pages 814–821, Monticello, EUA.

Tang, L. and Wang, X. (2008). An iterated local search heuristic for the capacitated prize-collecting travelling salesman problem. *Journal of The Operational Research Society - J OPER RES SOC*, 59:590–599.

Apêndice A

PRODUTOS

São listados a seguir os trabalhos oriundos desta pesquisa que foram publicados, ou já tiveram sua publicação aceita, em eventos, conferências, periódicos ou outros meios de divulgação de trabalhos acadêmicos.

Título:A tabu search approach for the Prize Collecting Traveling Salesman Problem
Autores:Ricardo Saraiva de Camargo, Rodney Rezende Saldanha e Odivaney Ramos Pedro
Evento:International Network Optimization Conference 2013
Período: 20/05/2013 to 23/05/2013
Local:Tenerife, Spain

Título:A tabu search approach for the Prize Collecting Traveling Salesman Problem
Autores:Ricardo Saraiva de Camargo, Rodney Rezende Saldanha e Odivaney Ramos Pedro
Periódico:Electronic Notes in Discrete Mathematics
Ano: 2013