UNIVERSIDADE FEDERAL FLUMINENSE

MARCOS ANTONIO GUERINE RIBEIRO

Metaheurística Híbrida com Mineração de Dados: Explorando uma Nova Categoria de Problemas

Niterói

2013

UNIVERSIDADE FEDERAL FLUMINENSE

MARCOS ANTONIO GUERINE RIBEIRO

Metaheurística Híbrida com Mineração de Dados: Explorando uma Nova Categoria de Problemas

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

Orientador:
Isabel Cristina Mello Rosseti

Co-orientador:
Alexandre Plastino de Carvalho

Niterói

2013

MARCOS ANTONIO GUERINE RIBEIRO

Metaheurística Híbrida com Mineração de Dados: Explorando uma Nova Categoria de Problemas

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

Aprovada em Agosto de 2013.

BANCA EXAMINADORA

Profª. Isabel Cristina Mello Rosseti - Orientador, UFF
(Presidente)

Prof. Alexandre Plastino de Carvalho - Co-orientador, UFF

Prof. Celso da Cruz Carneiro Ribeiro, UFF

Profª. Simone de Lima Martins, UFF

Prof. Haroldo Gambini Santos, UFOP

Niterói
2013

À minha mãe.

Agradecimentos

Aos meus pais, Glaucio e Gracinha, pela base familiar que sempre tive. Vocês são, com toda certeza, uma inspiração pra mim.

Aos meus irmãos, em especial, ao Glaydston, que muito me incentivou durante todos os períodos.

A Jaqueline, minha namorada, pela paciência e apoio incondicional, mesmo à distância.

Aos meus orientadores, Isabel e Plastino, pelos ensinamentos, orientação, confiança e, não menos importante, pelas broncas recebidas. Tenho certeza que aprendi lições, não só acadêmicas, mas também para a vida. Muito obrigado!

Aos demais docentes do IC/UFF, pelo aprendizado.

Às secretárias da pós, em particular a Teresa, que muito contribuiu para a minha permanência no IC.

A todos que, direta ou indiretamente, participaram dessa etapa. Seria injusto da minha parte se não mencionasse os principais: Adriana Guerini, Tias Norma e Zinha Guerini, Rafael, Roger, Jivago, Eyder, Julliany, Pedro, Gustavo Zanatta, Luiz, Mariana Bernardo, Roberto Menezes e todos os membros da *OptHouse*.

Ao CNPq, pelo apoio financeiro.

Acima de tudo, agradeço a Deus.

Resumo

Nas últimas décadas, algoritmos baseados em metaheurísticas ganharam notoriedade por apresentarem soluções de boa qualidade para diversos problemas de otimização combinatória em um tempo computacional aceitável. Mais recentemente, conceitos e processos de outras áreas de pesquisa têm sido utilizados, com bons resultados, na construção de metaheurísticas híbridas.

Técnicas de Mineração de Dados (MD) vêm sendo aplicadas com sucesso para aperfeiçoar heurísticas já consolidadas na literatura. A ideia principal da utilização de MD consiste em, a partir de um conjunto de soluções de boa qualidade, extrair padrões que representem boas características dessas soluções e utilizá-los para guiar a busca no espaço de soluções.

A combinação de metaheurísticas e MD obteve resultados relevantes tanto em termos de qualidade de solução, quanto em termos de tempo computacional, quando aplicada aos problemas: empacotamento de conjuntos, diversidade máxima, replicação de servidores para transmissão *multicast* confiável, p-medianas, e mais recentemente ao problema de projeto de redes a 2-caminhos.

Todos os problemas citados anteriormente possuem uma característica em comum: suas soluções são representadas por conjuntos de elementos e não levam em consideração a ordem desses elementos. Entretanto, em alguns problemas de otimização combinatória, essa ordem tem papel importante, como é o caso do problema do caixeiro viajante com coleta e entrega envolvendo único tipo de produto (em inglês, *One-Commodity Pickup-and-Delivery Traveling Salesman Problem*, 1-PDTSP).

Assim, este trabalho tem por objetivo a introdução de um módulo de mineração de dados em uma heurística já existente para o 1-PDTSP, baseada na metaheurística GRASP e na estratégia de busca local *Variable Neighborhood Descent* (VND), proposta por Hernández-Pérez *et al.* (2009). Pretende-se mostrar, como principal contribuição deste trabalho, que a hibridização de metaheurísticas com MD pode ser aplicada com sucesso não somente a problemas cujas soluções são representadas por conjuntos, mas também a problemas cujas soluções são representadas pela ordem dos seus elementos. Resultados computacionais mostraram que as versões híbridas com MD desenvolvidas foram capazes de obter melhores resultados tanto em termos de qualidade das soluções quanto em termos de tempo computacional, quando comparadas com a versão original da metaheurística.

Palavras-chave: GRASP, Metaheurística híbrida, 1-PDTSP, Mineração de Dados

Abstract

Over the last decades, algorithms based on metaheuristics have been proposed to solve a large set of hard optimization problems, achieving very good solutions in a acceptable computational time. More recently, concepts and process of others research areas were successfully combined with metaheuristic.

Some data mining (DM) techniques has been used to improve several state-of-art heuristics in different optimization problems. The main idea of this hybridization is to extract a subsets of elements that frequently occurs in a given set of high quality solutions and use them to guide the search in the solution space.

The combination of metaheuristics with DM achieved very promising results both in terms of quality of solution and computational time, when applied to the following problems: set packing problem, the maximum diversity, the efficient server replication for reliable multi-cast, the p-median and recently to the 2-path network design problem (2PNDP).

All these applications has one property in common: theirs solutions are represented by a subset of elements which its order is not considered. However, there are some optimization problems that this order is essential, which is case of the one-commodity pickup-and-delivery traveling salesman problem (1-PDTSP).

This work presents the incorporating of a data mining technique to a existing heuristic for the 1-PDTSP, based on the GRASP metaheuristic and the local improvement *Variable Neighborhood Descent* (VND), proposed by Hernández-Pérez *et al.* [40]. We intend to show, as the main contribution of this work, that the hybridization of metaheuristic with DM is not only successfully applied to problems which solutions are represented by a subset of elements, but also to problems which solutions are represented by the order of the elements. Extensive computational analysis shows that the two hybrid heuristics with DM outperforms the original algorithm both in terms of the solution quality and computational efforts.

Keywords: GRASP, Hybrid Metaheuristic, 1-PDTSP, Data Mining

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

GRASP	:	*Greedy Randomized Adaptative Search Procedures*;
DM-GRASP	:	Metaheurística GRASP Híbrida com Mineração de Dados;
LRC	:	Lista Restrita de Candidatos (*Restricted Candidate List*);
MCF	:	Mineração de Conjuntos Frequentes (*Frequent Itemset Mining*);
MD	:	Mineração de Dados (*Data Mining*);
MRA	:	Mineração de Regras de Associação (*Association Rule Mining*);
PCV	:	Problema do Caixeiro Viajante;
1-PDTSP	:	PCV com coleta e entrega para único tipo de produto;
RA	:	Regras de Associação;
VND	:	*Variable Neighborhood Descent*;
CE	:	Conjunto Elite;
CC	:	Componente Conexa;

Sumário

Capítulo 1

Introdução

Otimização Combinatória (OC) é um ramo da Ciência da Computação e da Matemática Aplicada que se refere ao estudo de problemas em que se busca minimizar ou maximizar uma função, por meio de escolhas sistemáticas dos valores das variáveis desses problemas.

Problemas de OC podem ser definidos por um conjunto base $E = \{1, \ldots, n\}$, um conjunto de soluções viáveis dado por $F \subseteq 2^E$ e uma função objetivo $f : 2^E \rightarrow R$, tal que $f(S) = \sum_{e \in S} c(e), \forall S \in 2^E$, onde $c(e)$ é o custo associado à inclusão do elemento $e \in E$ na solução S [24]. Na versão de minimização, procura-se uma solução ótima $S^* \in F$ tal que $f(S^*) \leq f(S), \forall S \in F$. O conjunto E, o vetor de custos c e o conjunto de soluções viáveis F são definidos para cada problema específico [15].

Para os problemas de OC conhecidos como $\mathcal{NP}$-Difíceis, a obtenção de soluções ótimas exatas, considerando pelo menos um critério de otimização, exige o emprego de algoritmos de complexidade superpolinomial que consomem elevados tempos de processamento, inviabilizando assim a resolução exata de problemas de grande porte que ocorrem em situações práticas reais.

Assim sendo, torna-se evidente, nesse contexto, a necessidade da aplicação de algoritmos aproximados que, ou podem ser combinados com os métodos exatos para acelerar seu tempo de processamento, ou eventualmente utilizados de maneira isolada para se obter soluções aproximadas, podendo ser ótimas, para esses problemas. O processo de busca de uma boa solução aproximada consiste em aplicar heurísticas projetadas de acordo com cada problema específico.

Nas últimas décadas, algoritmos baseados em metaheurísticas ganharam notoriedade por conseguirem resolver diversos problemas de OC de maneira quase ótima em tempo de execução consideravelmente menor que os tempos exigidos por métodos exatos.

Metaheurísticas podem ser definidas como procedimentos genéricos para a solução aproximada de problemas de otimização combinatória computacionalmente difíceis [57]. A adaptação de uma metaheurística para um determinado problema fornece uma heurística para esse problema.

As metaheurísticas tornaram-se populares por serem mais flexíveis, fáceis de entender e, ainda, por utilizarem, em alguns casos, comportamentos e ideias da natureza para justificarem suas abordagens. Na literatura, existem diversas metaheurísticas já consolidadas e comprovadamente eficientes, destacando-se: *Greedy Randomized Adaptive Search Procedure* (GRASP) [19, 63], Busca Tabu [26], Algoritmos Genéticos (AG) [31, 41], *Scatter Search* (SS) [25], *Simulated Annealing* (SA) [13, 43], *Variable Neighborhood Search* (VNS) [54] e *Iterated Local Search* (ILS) [9, 49]. Cada uma delas é baseada em um paradigma distinto e oferece diferentes mecanismos que permitem escapar de ótimos locais.

Mais recentemente, conceitos e processos de outras áreas de pesquisa foram utilizados em conjunto com metaheurísticas. Técnicas de Mineração de Dados (MD) — que consistem basicamente na extração de conhecimento, na forma de regras e padrões, de bases de dados de forma automática [34] — vêm sendo aplicadas com sucesso para aperfeiçoar metaheurísticas já consolidadas na literatura [61, 69]. Regras de associação, padrões sequenciais, agrupamento de dados e conjuntos frequentes são exemplos de regras e padrões extraídos em processos de MD, que podem ser usados em favor de heurísticas de maneiras distintas.

A hibridização de metaheurísticas com MD foi inicialmente proposta por Ribeiro *et al.* [65, 66] através da incorporação de mineração de conjuntos frequentes à metaheurística GRASP [19]. Essa proposta obteve resultados relevantes quando aplicada ao problema do empacotamento de conjuntos. O desenvolvimento dessa abordagem híbrida com MD está baseado na hipótese de que padrões extraídos de soluções subótimas representam características dessas soluções e, por isso, podem ser usados para guiar a busca por melhores soluções. Essa abordagem foi denominada DM-GRASP (*Data Mining* GRASP) e também foi aplicada com êxito em outros problemas [7, 52, 60, 61, 70, 71].

Todas essas aplicações da estratégia DM-GRASP possuem uma característica em comum: as soluções dos problemas de OC abordados são representadas por conjuntos de elementos que não levam em consideração a ordem em que esses elementos estão dispostos. Em [61], no problema das p-medianas, por exemplo, a ordem das facilidades escolhidas não influencia no custo que será calculado. Entretanto, em alguns problemas de OC, a ordem tem papel importante, como é o caso do problema de roteamento de veículos (PRV).

O PRV, um dos problemas mais conhecidos e estudados em OC, consiste em atender um conjunto interligado de consumidores por meio de uma frota de veículos, que partem de um ou mais pontos denominados depósitos. O objetivo é encontrar um conjunto de rotas que satisfaçam a critérios e restrições, buscando a minimização do custo total, que pode estar associado, por exemplo, à soma das distâncias percorridas pelos veículos. Existem diversas variantes do PRV que envolvem diferentes critérios e restrições, tais como: janelas de tempo [74], coleta e entrega [42, 55], múltiplos depósitos [16], entre outros. Os problemas de roteamento de veículos, em sua maioria, são problemas reais que grandes empresas de logística se deparam e não encontram soluções que satisfaçam aos seus interesses.

O problema do caixeiro viajante com coleta e entrega envolvendo um único tipo de produto (em inglês, *one-commodity pickup and delivery traveling salesman problem*, 1-PDTSP), é uma variante do PRV que possui rota única, apenas um veículo com capacidade limitada e clientes com demandas, podendo ser classificados como clientes de coleta (demanda positiva) ou de entrega (demanda negativa). O 1-PDTSP é um problema $\mathcal{NP}$-Difícil [36] e tem grande aplicabilidade em problemas reais de logística, em particular, no contexto de reposicionamento de produtos.

Este trabalho tem por objetivo inserir um módulo de mineração de dados em uma heurística já existente para o 1-PDTSP baseada na metaheurística GRASP e na estratégia de busca local *Variable Neighborhood Descent* (VND), proposta por Hernández-Pérez *et al.* [40]. Pretende-se, dessa forma, como principal contribuição deste trabalho, mostrar que a hibridização de metaheurísticas com MD pode ser aplicada com sucesso não somente a problemas cujas soluções são representadas por conjuntos, mas também a problemas cujas soluções são representadas pela ordem dos seus elementos, como é o caso do 1-PDTSP.

O restante desta dissertação está organizado da seguinte forma. O Capítulo 2 introduz conceitos da metaheurística GRASP e de técnicas de mineração de dados, em particular sobre mineração de conjuntos frequentes e sua combinação com o GRASP, além de apresentar uma compilação de trabalhos onde essa abordagem foi aplicada. O Capítulo 3 apresenta o 1-PDTSP e sua formulação matemática, bem como uma revisão bibliográfica de trabalhos que tratam esse problema, além de uma breve revisão de problemas correlacionados.

O Capítulo 4 descreve o GRASP/VND proposto em [40] para o 1-PDTSP e apresenta como a técnica de mineração foi inserida nessa heurística. Em seguida, são apresentadas

duas estratégias híbridas que combinam o GRASP/VND com MD: a primeira, que aplica o módulo de mineração uma única vez, denominada DM-GRASP/VND, e a segunda, que aplica o módulo de mineração mais de uma vez, denominada MDM-GRASP/VND.

O Capítulo 5 contém os experimentos computacionais realizados, comparando e analisando o comportamento das três estratégias. Finalmente, o Capítulo 6 contém as conclusões deste trabalho, juntamente com indicações de trabalhos futuros.

Capítulo 2

Metaheurística GRASP e Mineração de Dados

Problemas de otimização combinatória podem possuir alta dificuldade de resolução quando a quantidade de combinações de soluções é extremamente grande. No processo de busca pela solução ótima, examinar todas as soluções acaba se tornando inviável. Para tais problemas, algoritmos baseados em metaheurísticas são propostos para examinar o espaço de busca de modo a encontrar soluções próximas a ótima (ou até mesmo a ótima) em um tempo computacional viável.

Neste capítulo, serão apresentados os conceitos da metaheurística GRASP e de mineração de dados (MD), a fim de facilitar o entendimento da hibridização do GRASP com as técnicas de MD.

2.1 Metaheurística GRASP

Greedy Randomized Adaptive Search Procedure (GRASP) é uma metaheurística proposta por Feo e Resense [19] que já foi aplicada com sucesso a diversos problemas de otimização [20, 21]. O GRASP é composto por um processo iterativo, no qual cada iteração é dividida em duas fases: construção e busca local. A cada iteração, uma solução viável s é construída por um procedimento guloso e aleatório e avaliada por uma função $f(s)$. Essa função fornece o custo da solução construída e deve ser definida de acordo com o problema abordado. Em seguida, a solução s é submetida a uma busca no espaço vizinho de soluções, com o objetivo de melhorá-la até encontrar um ótimo local. A melhor solução geral é mantida e retornada como resultado final.

O pseudocódigo da versão original proposta do GRASP [19] pode ser visualizado no

Algoritmo 1. O primeiro passo do algoritmo é inicializar as soluções s^* (melhor solução) e s (solução corrente) como vazias e o valor do custo de s^*, representado por $f(s^*)$, com valor infinito. Assim, o laço principal começa e uma solução s é construída (linha 4). Essa solução é submetida à heurística de busca local (linha 5) e, caso o custo da solução s seja melhor[1] (no caso, menor) que o custo da melhor solução s^*, então s^* é atualizada com a solução corrente s (linha 7). Esses passos se repetem até que se completem $maxIter$ iterações.

Algoritmo 1: Pseudocódigo do GRASP original

```
1: GRASP(α, E, maxIter)
2: s* ← s ← ∅; f(s*) ← ∞
3: Para iter = 1 até maxIter faça
4:    s ← ConstruçãoGulosaAleatória(α, E)
5:    s ← BuscaLocal(s)
6:    Se f(s) < f(s*) então
7:       s* ← s
8:    Fim-se
9: Fim-para
10: Retorne s*
```

A etapa de construção é também iterativa e baseada em um procedimento guloso e aleatório, que considera o conjunto de todos os elementos E. Inicialmente, todos os itens $i \in E$ que podem ser inseridos na solução são avaliados de acordo com uma função $c(i)$, que determina o quanto aquele elemento contribui para a solução s. Assim, são inseridos em uma lista denominada Lista Restrita de Candidatos (LRC) todos os elementos i que atendem ao seguinte critério:

$$c(i) \in \left[c^{min}, c^{min} + \alpha\left(c^{max} - c^{min}\right)\right] \tag{2.1}$$

com $\alpha \in [0, 1]$ e considerando c^{min} e c^{max}, respectivamente, os valores máximo e mínimo de contribuição dentre os elementos avaliados. Após a criação da LRC, um dos elementos é escolhido aleatoriamente e inserido na solução. Em seguida, repetem-se esses mesmos passos até que todos os elementos possíveis sejam inseridos. Na Equação 2.1, pode-se notar que o parâmetro α funciona como uma componente de aleatoriedade. Quando α se aproxima de zero, a escolha tende a ser mais gulosa, ao passo que quando se aproxima de um, a seleção tende a ficar mais aleatória. O pseudocódigo da construção pode ser visualizado no Algoritmo 2.

O Algoritmo 2 representa o pseudocódigo da fase de construção do GRASP. A solução

[1] O termo *melhor* depende do problema tratado, se esse é de maximização ou de minimização. Nesta dissertação, o problema abordado é de minimização e por isso será usado o termo menor.

Algoritmo 2: Pseudocódigo da fase de construção

1: **ConstruçãoGulosaAleatória**(α, E)
2: $s \leftarrow \emptyset$
3: Avaliar $c(i)\ \forall\ i \in E$
4: **Enquanto** $\neg$ SoluçãoCompleta(s) **faça**
5: $LRC \leftarrow$ ConstroiLRC(α)
6: $i \leftarrow$ EscolhaAleatória(LRC)
7: $s \leftarrow s \cup \{i\}$
8: Avaliar $c(i)\ \forall\ i \in E \backslash s$
9: **Fim-enquanto**
10: **Retorne** s

a ser construída s é inicializada como uma solução vazia (linha 2) e, após avaliar cada elemento i, $\forall\ i \in E$ (linha 3), constrói-se a LRC (linha 5) de acordo a função $c(i)$ e o parâmetro α, presentes na Equação 2.1. Dentre os elementos da LRC, um é escolhido aleatoriamente (linha 6) e inserido na solução (linha 7). Esses passos (linhas 5, 6, 7 e 8) se repetem até que uma solução seja construída.

Após a construção, a solução s é submetida a uma intensificação, com o objetivo de melhorá-la. A fase de busca local é composta por uma estratégia de aprimoramento iterativa, que busca soluções de melhor qualidade em um conjunto de soluções próximas a s. Esse conjunto é denominado vizinhança de s, representado por $N(s)$, e pode ser obtido aplicando simples operações que alteram s. Assim, a busca local visa encontrar o melhor vizinho de s com respeito a uma vizinhança $N(s)$.

O Algoritmo 3 apresenta o pseudocódigo da fase de busca local para o GRASP. A linha 2 define o conjunto H, composto por todas as soluções y presentes na vizinhança $N(s)$ que possuem qualidade melhor que a solução s, de acordo com a função de avaliação $f(s)$. Assim, a cada iteração, seleciona-se um vizinho s' que seja melhor que s (linha 4) e um novo conjunto H é estruturado com base na solução s' (linha 5). A busca local termina quando não existirem vizinhos capazes de melhorar a qualidade da solução corrente s'. Nesse caso, afirma-se que s' é ótimo local com respeito à vizinhança $N(s)$.

Algoritmo 3: Pseudocódigo da fase de busca local

1: **BuscaLocal**(s)
2: $H = \{y \in N(s) | f(y) < f(s)\}$
3: **Enquanto** $|H| > 0$ **faça**
4: Selecionar $s' \in H$
5: $H = \{y \in N(s') | f(y) < f(s')\}$
6: **Fim-enquanto**
7: **Retorne** s'

Na seção seguinte, serão apresentados os conceitos e técnicas de mineração de dados utilizados em trabalhos anteriores na construção do GRASP híbrido com MD.

2.2 Mineração de Dados

O conceito de Mineração de Dados (MD) surgiu no final da década de 80 e consiste na extração de conhecimento de grandes bases de dados, de forma automática, na forma de padrões ou regras [34]. Esses padrões e regras, tais como conjuntos frequentes, padrões sequenciais e regras de associação (RAs), podem ser utilizados para entender comportamentos e auxiliar na tomada de decisões.

As RAs podem ser consideradas como uma das principais formas de conhecimento extraídas de bases de dados [34]. Minerar RAs é uma tarefa descritiva de MD, que tem como objetivo buscar associações entre itens que ocorrem com frequência dentro da base de dados.

A tarefa de minerar RAs é geralmente dividida em duas etapas. A primeira, que demanda maior esforço computacional [30], é denominada mineração de conjuntos frequentes (MCF) e consiste em, a partir de um conjunto de transações θ, identificar todos os subconjuntos de itens que ocorrem com frequência nesse conjunto de transações. Um conjunto λ é considerado frequente se sua medida de suporte, que indica o número de transações da base de dados que os elementos de λ estão presentes, é maior ou igual a um suporte mínimo predefinido, ou seja, se $suporte(\lambda) \geq sup_{min}$.

A segunda etapa da mineração de RAs fornece, para cada conjunto frequente extraído na primeira etapa, um conjunto de regras de associação. Cada RA consiste em uma implicação na forma $X \Rightarrow Y$, com $X \subset \Phi$, $Y \subset \Phi$, $X \neq \emptyset$, $Y \neq \emptyset$ e $X \cap Y = 0$, sendo Φ o conjunto de todos os itens da base de dados. Considera-se que a regra $X \Rightarrow Y$ é minerada se: i) o número de transações da base que contém os itens do conjunto $X \cup Y$ for pelo menos igual a sup_{min}, e ii) dentre as transações que contém X, o número de transações que contém Y for pelo menos igual a $conf_{min}$, parâmetro conhecido como confiança mínima. No entanto, para a hibridização com a metaheurística GRASP, somente a MCF importa e será considerada.

Para exemplificar alguns conceitos, a Tabela 2.1 apresenta uma base de dados exemplo com seis transações, envolvendo o conjunto de itens $\Phi = \{A, B, C, D, E\}$. É possível observar que o conjunto $\{A, B\}$ está presente em três transações (1, 2 e 4) das seis possíveis e, portanto, tem suporte 3 (ou 50%). Já o conjunto $\{C, D, E\}$ tem suporte 1 (ou 16.6%),

pois ocorre apenas na transação 5.

ID	Transações
1	$\{A, B, C, D\}$
2	$\{A, B\}$
3	$\{B, C, E\}$
4	$\{A, B, C\}$
5	$\{B, C, D, E\}$
6	$\{C, D\}$

Tabela 2.1: Banco de dados de transações.

Além disso, considera-se conjunto frequente maximal aquele conjunto frequente que não é subconjunto de outro conjunto frequente. Assim, assumindo sup_{min} igual a 3, o conjunto $\{A, B\}$ é frequente e também maximal, pois possui suporte três e não existe nenhum conjunto frequente que o contenha.

Em 1993, O problema de MCF foi introduzido por Agrawal *et al.* em [1], onde os autores propuseram o algoritmo AIS, que evita avaliar todos os $2^{|\Phi|}$ subconjuntos possíveis, considerando apenas os elementos que ocorrem em alguma transação. Assim, foi reduzido o esforço computacional dessa etapa, que ainda assim era muito grande.

No ano seguinte, o algoritmo *Apriori* [2] foi proposto e obteve grandes melhorias em relação a esse esforço computacional, utilizando a seguinte propriedade: todo conjunto de itens que contém um subconjunto não frequente, também não é frequente. Dessa maneira, diminuiu-se consideravelmente a quantidade de subconjuntos a serem avaliados.

Desde então, muitos algoritmos propostos para a MCF foram baseados no *Apriori*, aprimorando suas características negativas. Tais algoritmos, segundo Han *et al.* [35], ainda tem como principal ponto negativo a geração exagerada de conjuntos a serem avaliados, especialmente quando o suporte mínimo é baixo.

No entanto, existem alguns algoritmos que extraem os conjuntos frequentes sem a geração de conjuntos candidatos e, consequentemente, evitam o ponto negativo do *Apriori*, tais como o *PatriciaMine* [30, 59], *FP-growth* [35] e *FP-growth** [29, 30, 32], cuja variação conhecida como *FPmax** [29, 30, 32] fornece conjuntos de itens frequentes maximais. O algoritmo *FPmax** tem sido a principal técnica de MCF utilizada nos trabalhos que envolvem o DM-GRASP, a versão híbrida do GRASP com mineração de dados [7, 60, 61, 65, 66, 70, 71].

A seção a seguir apresenta duas estratégias que combinam GRASP com mineração de dados, além de uma revisão bibliográfica de trabalhos que as utilizaram.

2.3 GRASP com Mineração de Dados

O GRASP, em sua forma original [19], tem como característica não possuir qualquer tipo de memória a longo prazo, com exceção da melhor solução que é mantida ao longo das iterações. Isto se deve ao fato de o GRASP ser iterativo e multipartida, com iterações totalmente independentes que, a cada partida, iniciam uma nova e aleatória execução do processo de busca [4]. Ideias de manter informações de iterações anteriores foram sendo investigadas, como técnicas baseadas em memória adaptativa [22, 47, 48, 67] e construção de vocabulário [4, 11, 27].

Dentre essas estratégias, Laguna e Martí [44] adaptaram a técnica de reconexão por caminhos, proposta por Glover [25, 28], para utilizá-la em conjunto com o GRASP. Essa técnica de intensificação pode ser entendida como uma memória a longo prazo quando inserida ao GRASP, pois mantém um conjunto elite de soluções durante toda a execução do algoritmo. A reconexão por caminhos explora todo o espaço intermediário entre duas soluções s_i e s_f, escolhidas desse conjunto. Em s_i, são introduzidos iterativamente atributos de s_f de maneira a transformar s_i em s_f. As soluções intermediárias são avaliadas e, eventualmente, contribuem com a busca por melhores soluções.

Essas abordagens, caracterizadas pela utilização de conjuntos de boas soluções, reforçam a ideia de que padrões extraídos desses conjuntos podem auxiliar na busca por melhores soluções. As próximas seções apresentam os conceitos das estratégias DM-GRASP e MDM-GRASP, duas abordagens que combinam a metaheurística GRASP com a técnica de mineração de conjuntos frequentes.

2.3.1 DM-GRASP

Proposto por Ribeiro *et al.* [64, 65, 66], a estratégia *Data Mining GRASP* (DM-GRASP) baseia-se na hipótese de que padrões minerados a partir de um conjunto de soluções subótimas podem ser utilizados em procedimentos de construção adaptados, para guiar a exploração do espaço de busca, visando melhores soluções.

O DM-GRASP é dividido em duas etapas. A primeira, denominada fase de geração do conjunto elite (CE), consiste em executar k iterações do GRASP original e, a cada iteração, manter atualizado um conjunto com as d melhores soluções encontradas. Após essa etapa, o CE é submetido a um procedimento de mineração baseado na técnica de mineração de conjuntos frequentes, detalhada anteriormente. O CE funciona como uma base de dados de soluções e, assim, a mineração retorna $numP$ maiores padrões que ocorreram com uma

frequência prefixada. Em outras palavras, cada padrão é um conjunto de partes frequentes dessas soluções.

Após a mineração, a segunda etapa, denominada etapa híbrida, consiste em executar mais k iterações do GRASP, mas agora substituindo a heurística construtiva original por uma heurística construtiva híbrida, adaptada para utilizar os padrões minerados na primeira etapa.

O Algoritmo 4 representa a construção adaptada para usar os padrões minerados. Um padrão p, dentre os padrões minerados para guiar a construção, é passado como parâmetro para a construção. A solução inicial s recebe p como solução parcial (linha 2) e em seguida, a construção prossegue da mesma forma que a construção original, detalhada na Seção 2.1.

Algoritmo 4: Pseudocódigo da construção adaptada

```
1: ConstruçãoAdaptada(α, E, p)
2: s ← p
3: Avaliar c(i) ∀ i ∈ E\s
4: Enquanto ¬ SoluçãoCompleta(s) faça
5:    LRC ← ConstroiLRC(α)
6:    i ← EscolhaAleatoria(LRC)
7:    s ← s ∪ {i}
8:    Avaliar c(i) ∀ i ∈ E\s
9: Fim-enquanto
10: Retorne s
```

O processo de utilização do padrão (linha 2 do Algoritmo 4) depende do problema que está sendo tratado e de como sua solução é representada. Na Seção 4.2, será descrito em detalhes como a etapa de construção adaptada foi implementada nesta dissertação.

O Algoritmo 5 mostra o pseudocódigo geral do DM-GRASP, que é formado por dois laços com k iterações, onde o primeiro laço representa a primeira etapa do DM-GRASP, e o segundo laço representa a segunda fase. No primeiro, além da construção e da busca local (linhas 5 e 6, respectivamente), há também a construção do conjunto elite (linha 10), que é atualizado sempre que a solução corrente for melhor que a pior das suas soluções. Entre os laços, acontece o processo de mineração (linha 12).

No segundo laço, um padrão p é selecionado da *ListaPadrões* e aproveitado pela construção adaptada (linha 15), que substitui a construção original. A fase de busca local, no entanto, permanece idêntica à primeira etapa (linha 16).

A heurística DM-GRASP já foi avaliada com sucesso em alguns trabalhos existentes na literatura. Inicialmente aplicada ao problema do empacotamento de conjuntos [64, 65, 66],

Algoritmo 5: Pseudocódigo do DM-GRASP

```
1: DM-GRASP (α, E, k, sup_min, d, numP)
2: f(s*) ← f(s) ← ∞
3: CE ← ∅
4: Para iter = 1 até k faça
5:    s ← ConstruçãoGulosaAleatória(α, E)
6:    s ← BuscaLocal(s)
7:    Se f(s) < f(s*) então
8:       s* ← s
9:    Fim-se
10:   AtualizaCE(s, CE, d)
11: Fim-para
12: ListaPadrões ← ExecutaMineração(CE, sup_min, numP)
13: Para iter = 1 até k faça
14:   p ← EscolhePadrão(ListaPadrões)
15:   s ← ConstruçãoAdaptada(α, E, p)
16:   s ← BuscaLocal(s)
17:   Se f(s) < f(s*) então
18:      s* ← s
19:   Fim-se
20: Fim-para
21: Retorne s*
```

essa abordagem obteve resultados relevantes tanto em termos de qualidade de solução quanto em tempo de execução quando comparada com o GRASP original para o referido problema.

Em seguida, Santos *et al.* [68, 70, 71] aplicaram o DM-GRASP ao problema da maximização da diversidade e também ao problema de replicação de servidores para transmissão *multicast* confiável, conseguindo também melhorar a qualidade das soluções e reduzir o tempo computacional, novamente em relação ao GRASP original.

Em [8, 52, 60, 61], os autores aplicaram o DM-GRASP para o problema das p-medianas, novamente conseguindo melhorar a qualidade das soluções obtidas, além de diminuir o tempo computacional, em relação ao GRASP original.

Recentemente, Barbalho *et al.* [7] apresentaram uma heurística que combina GRASP, reconexão por caminhos e mineração de dados para o problema de projeto de redes a 2-caminhos (2-PNDP, sigla em inglês). Essa heurística demostrou que a inserção da MD é benéfica tanto à metaheurística GRASP pura, quanto ao GRASP com reconexão por caminhos, em termos de qualidade de solução e também de tempo computacional.

A próxima seção apresenta outra abordagem que combina MD com GRASP, porém aplicando a mineração de padrões mais de uma vez durante a execução do algoritmo.

2.3.2 MDM-GRASP

Na proposta híbrida DM-GRASP, o processo de mineração é executado somente uma vez, entre a primeira e a segunda etapa, exatamente na metade das iterações do algoritmo. Apesar dos bons resultados [8, 52, 60, 61, 68, 70, 71], Plastino *et al.* [60] acreditaram que aplicar a mineração mais de uma vez poderia ser ainda mais eficiente, pois, a cada aplicação, os padrões seriam cada vez mais refinados, uma vez que seriam extraídos de CEs compostos por melhores soluções. Para tal, os autores definiram que a mineração seria aplicada assim que o conjunto elite se tornasse estável, ou seja, que não fosse modificado num intervalo predefinido de iterações.

Dessa maneira, a aplicação da mineração está condicionada a estabilidade do CE, podendo ser aplicada até mesmo antes da metade das iterações, diferentemente do DM-GRASP. Essa estratégia foi denominada pelos autores como *Multi Data Mining* GRASP (MDM-GRASP).

O Algoritmo 6 mostra os passos do MDM-GRASP. No primeiro laço (linhas 4-12), são realizadas as etapas de construção e busca local (linhas 5 e 6, respectivamente), e o CE é preenchido com as d melhores soluções (linha 10). Esse laço somente é finalizado quando não há alterações no CE por um determinado número de iterações. Em seguida, inicia-se outro laço (linhas 13-25), em que a mineração é executada (linha 15) sempre que o CE se torna estável.

Dessa maneira, na primeira iteração desse segundo laço, a mineração é sempre executada devido ao critério de parada do primeiro laço (por estabilidade do CE). Após a execução da mineração, seleciona-se um padrão (linha 17) que é usado na construção adaptada (linha 18) e, em seguida, aplica-se a busca local (linha 19). Por fim, o CE é atualizado sempre que a solução corrente for melhor que a pior solução presente no CE.

A estratégia MDM-GRASP foi inicialmente avaliada no problema das p-medianas [61], conseguindo superar o DM-GRASP (que, por sua vez, já havia superado o GRASP) em termos de qualidade de solução e também em tempo de execução. Em seguida, o módulo MDM foi aplicado ao problema 2PNDP [7], novamente conseguindo superar ambas as versões GRASP e DM-GRASP com reconexão por caminhos. Nos dois casos, a versão MDM foi superior tanto em termos de qualidade de solução quanto em termos de tempo computacional. Além disso, para algumas instâncias, a heurística MDM-GRASP com reconexão por caminhos ainda obteve os melhores resultados da literatura para o 2-PNDP.

O Capítulo 3, a seguir, apresenta o problema 1-PDTSP, que será utilizado como base

Algoritmo 6: Pseudocódigo do MDM-GRASP

```
1: MDM-GRASP (α, E, maxIter, sup_min, d, numP)
2: f(s*) ← f(s) ← ∞; iter ← 1
3: CE ← ∅
4: Enquanto CE ¬ estável faça
5:    s ← ConstruçãoGulosaAleatória(α, E)
6:    s ← BuscaLocal(s)
7:    Se f(s) < f(s*) então
8:       s* ← s
9:    Fim-se
10:   AtualizaCE(s,CE,d)
11:   iter ← iter +1
12: Fim-enquanto
13: Enquanto iter ≤ maxIter faça
14:   Se CE está estável então
15:      ListaPadrões ← ExecutaMineração(CE,sup_min, numP)
16:   Fim-se
17:   p ← SelecionaPróximoPadrão(ListaPadrões)
18:   s ← ConstruçãoAdaptada(α,E,p)
19:   s ← BuscaLocal(s)
20:   Se f(s) < f(s*) então
21:      s* ← s
22:   Fim-se
23:   AtualizaCE(s,CE,d)
24:   iter ← iter +1
25: Fim-enquanto
26: Retorne s*
```

para avaliar o comportamento das estratégias híbridas com MD propostas neste trabalho. Além disso, alguns conceitos acerca desse problema são detalhados e uma formulação matemática é descrita. Por fim, são listados alguns problemas correlatos ao 1-PDTSP.

Capítulo 3

Problema 1-PDTSP e Variantes

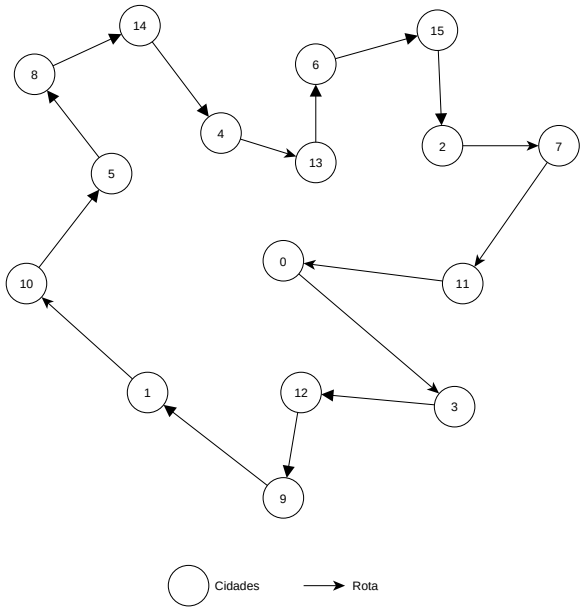
O problema do caixeiro viajante (PCV), em inglês *Traveling Salesman Problem (TSP)*, é um problema antigo que surgiu em meados do século XIX da necessidade de um vendedor visitar um conjunto de n cidades, sem repetição, retornando à primeira cidade visitada após passar por todas as outras, de forma que a distância percorrida fosse minimizada [72]. Quando o número de cidades é pequeno, a solução é simples e trivial. Porém, à medida que a quantidade de cidades aumenta, esse simples problema se torna extremamente difícil de se resolver. Existem algumas variantes do PCV, como por exemplo o PCV com janelas de tempo, onde o caixeiro deve visitar cada cidade dentro de um intervalo de horário, tornando o problema ainda mais específico.

O problema do caixeiro viajante com coleta e entrega envolvendo um único tipo de produto (1-PDTSP), estudado neste trabalho, é uma generalização do PCV onde cada cidade (ou no caso, cliente) possui uma demanda de um produto específico. O caixeiro viajante (ou no caso, o veículo), além de visitar todos os clientes, também deverá entregar ou coletar uma quantidade de um determinado produto, de forma que todos os clientes sejam atendidos com relação à sua demanda. É importante ressaltar que o veículo tem capacidade limitada, que deve ser respeitada ao longo da rota.

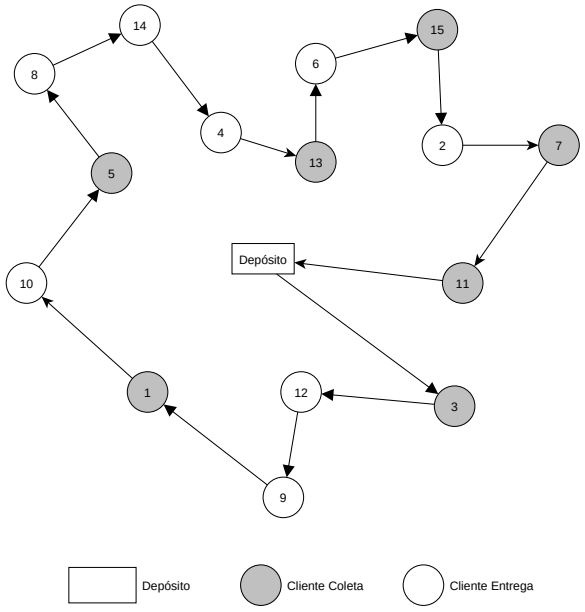
O objetivo do 1-PDTSP, portanto, consiste em encontrar a rota de menor custo que visite todas os clientes uma única vez. O trajeto do veículo começa em um ponto inicial, denominado depósito, e retorna ao ponto de partida não repetindo nenhum cliente, estabelecendo assim um circuito hamiltoniano. Por se tratar de um único produto, a quantidade coletada em um cliente pode ser utilizada para suprir a demanda de outro.

A Figura 3.1(a) representa uma possível solução para o PCV clássico, com uma rota visitando todas as 15 cidades que compõem um problema exemplo. Observando a Fi-

gura 3.1(b), que denota o PCV com coleta e entrega, é possível perceber que as principais mudanças na ilustração do problema são: a diferenciação entre clientes de coleta e de entrega, e um depósito no ponto inicial. Além dessas, no PCV com coleta e entrega, o veículo possui capacidade limitada.



(a) PCV Clássico



(b) PCV com Coleta e Entrega

Figura 3.1: Soluções do PCV Clássico e do PCV com Coleta e Entrega

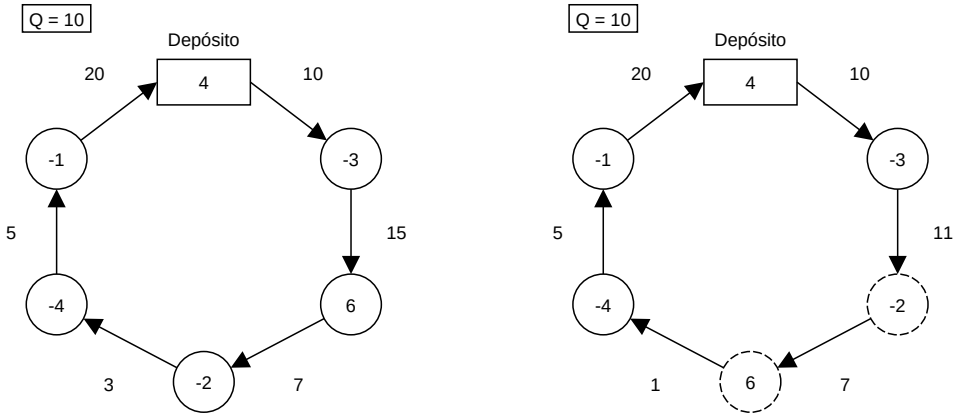
No 1-PDTSP, as soluções sempre contém todos os clientes. A ordem em que eles estão organizados define a viabilidade e a qualidade da solução. Em uma rota, a simples inversão de dois clientes pode influenciar no atendimento de toda a rota, podendo provocar

até a inviabilidade da solução, seja por estourar a capacidade do veículo ou por deixar algum cliente sem atendimento.

Quando a capacidade do veículo é suficientemente grande, o 1-PDTSP coincide com o PCV e, portanto, é considerado $\mathcal{NP}$-Difícil. Além disso, examinar se existe uma solução viável para uma instância do 1-PDTSP é um problema $\mathcal{NP}$-Completo. Por outro lado, dada uma solução para o 1-PDTSP, a tarefa de verificar se essa rota é viável pode ser realizada em $\mathcal{O}(n)$ [36].

A Figura 3.2 apresenta uma pequena instância do 1-PDTSP, que contém um conjunto de seis clientes representados por círculos, com exceção do depósito que está representado por um retângulo. Considera-se que a capacidade do veículo seja $Q = 10$ e denota-se a carga parcial acumulada no veículo por L. O valor no interior dos clientes corresponde à sua demanda e o valor acima de cada arco é o custo de transporte associado entre os clientes. As setas indicam a direção da rota.

A Figura 3.2(a) apresenta uma solução viável para a instância pois o veículo percorre todos os nós, coletando e entregando todas as demandas, sem que exceda a capacidade do veículo ($L > Q$) ou que algum cliente não seja atendido ($L < 0$). O custo da rota apresentada é a soma dos valores dos arcos que, no caso, é igual a 60.



(a) Solução viável (b) Solução inviável

Figura 3.2: Duas soluções diferentes envolvendo os mesmos clientes

Por outro lado, com o intuito de diminuir o custo da solução, pode-se trocar as posições do segundo e do terceiro cliente, dando origem à solução representada na Figura 3.2(b). A nova rota obtida possui menor custo, igual a 54. Entretanto, a viabilidade dessa nova solução deve ser conferida. O veículo parte do depósito com carga acumulada $L = 4$ e atende a demanda do primeiro cliente (-3), entregando três unidades do produto trans-

portado e fica com carga acumulada $L = 1$. No momento em que for atender o próximo cliente, que possui demanda -2, a carga acumulada passa a ser $L = -1$ e a solução se torna inviável, pois o veículo não possui duas unidades do produto necessárias para atender esse cliente. Além disso, outra maneira não ilustrada de inviabilizar a solução seria coletar uma quantidade de produto de maneira que $L > Q$, excedendo assim a capacidade do veículo.

O 1-PDTSP tem algumas aplicações reais [36] como, por exemplo, o reposicionamento de estoque de uma empresa varejista, onde uma filial da empresa necessita de um produto que outra filial tenha em excesso. Pode ocorrer então um reposicionamento daquele produto, transferindo-o do local que esteja sobrando para o local onde há sua falta.

Outro exemplo acontece em terminais de saque de dinheiro. Quando um terminal acusa falta de notas de dinheiro em estoque, o banco pode optar por reposicionar as notas entre seus terminais, migrando daqueles que tem menor frequência para aqueles com alta frequência de saques.

Um terceiro exemplo seria a distribuição de um determinado produto, por exemplo leite, para mercados ou residências, utilizando fazendas de produção de leite como fornecedores. Assim, um veículo distribuidor pode percorrer a cidade coletando caixas de leite das fazendas, e entregando-as em locais de consumo ou revenda.

Por fim, cita-se o problema da construção de grandes imóveis, no qual o nível dos terrenos utilizados nem sempre é o ideal, exigindo a remoção ou preenchimento de determinados lotes com grandes quantidades de terra. Dessa maneira, pode-se definir locais de remoção e de preenchimento e, assim, estabelecer uma rota de realocação das porções de terra, minimizando os custos de transporte e, consequentemente, os custos da construtora.

Na seção seguinte, o 1-PDTSP é definido formalmente e uma formulação matemática é apresentada para esse problema.

3.1 Formulação Matemática

Hernández-Pérez e Salazar-González em [36] introduziram um modelo matemático de programação linear inteira para o 1-PDTSP. Para defini-lo formalmente, considera-se $G = (V, A)$ um grafo completo, onde $V = \{1, \ldots, n\}$ é o conjunto de vértices que representam os clientes, e $A = \{(i, j) : i, j \in V\}$ é o conjunto de arcos. Cada vértice $i \in V$ possui uma demanda q_i associada, sendo cliente de entrega, quando $q_i < 0$, e cliente de

coleta, quando $q_i > 0$. Para cada par (i, j), a distância (ou custo) entre os clientes i e j é dado por c_{ij}. Para cada subconjunto $S \subset V$, define-se $\delta^+(S) = \{(i, j) \in A : i \in S, j \notin S\}$ e $\delta^-(S) = \{(i, j) \in A : i \notin S, j \in S\}$.

Seja Q a capacidade do veículo e q_1 a demanda do cliente definido como depósito. O último pode ser considerado um consumidor que absorve ou fornece uma determinada quantidade de produto para garantir que a equação $q_1 = -\sum_{i=2}^{n} q_i$ seja satisfeita.

A Equação 3.1 garante a conservação do fluxo no 1-PDTSP.

$$\sum_{\forall i \in V : q_i > 0} q_i + \sum_{\forall i \in V : q_i < 0} q_i = 0 \tag{3.1}$$

Seja x_{ij} a variável binária de decisão que informa se o arco (i, j) está ($x_{ij} = 1$) ou não ($x_{ij} = 0$) na solução e f_{ij} uma variável não negativa que indica o fluxo do arco $(i, j) \in A$. A formulação matemática para o 1-PDTSP é apresentada a seguir.

$$min \sum_{(i,j) \in A} c_{ij} x_{ij} \tag{3.2}$$

sujeito a:

$$\sum_{(i,j) \in \delta^+(\{i\})} x_{ij} = 1 \qquad \forall i \in V \tag{3.3}$$

$$\sum_{(i,j) \in \delta^-(\{i\})} x_{ij} = 1 \qquad \forall i \in V \tag{3.4}$$

$$\sum_{(i,j) \in \delta^+(S)} x_{ij} \geq 1 \qquad \forall S \subset V \tag{3.5}$$

$$\sum_{(i,j) \in \delta^+(\{i\})} f_{ij} - \sum_{(i,j) \in \delta^-(\{i\})} f_{ij} = q_i \qquad \forall i \in V \tag{3.6}$$

$$0 \leq f_{ij} \leq Q x_{ij} \qquad \forall (i, j) \in A \tag{3.7}$$

A função objetivo, representada na Equação 3.2, é caracterizada pela minimização da soma dos custos de viagem. As restrições (3.4) e (3.3) indicam a obrigatoriedade de

visita única em cada cliente. A restrição (3.5) é a desigualdade que proíbe a formação de subciclos ou rotas desconexas e a restrição (3.6) assegura a conservação do fluxo. Por fim, a restrição (3.7) define o domínio das variáveis de fluxo.

3.2 Revisão da Literatura

O problema do caixeiro viajante envolvendo um único produto (1-PDTSP) foi apresentado por Hernández-Pérez e Salazar-González em [36] e, no mesmo trabalho, foi desenvolvido um algoritmo *branch-and-cut* para resolvê-lo, utilizando plano de corte baseado em decomposição de Benders [10] e em desigualdades utilizadas no PCV. Uma heurística de construção baseada na inserção mais próxima e heurísticas de busca local foram usadas inicialmente para obter uma solução viável com o objetivo de acelerar a ramificação e poda no algoritmo *branch-and-bound*. Esse algoritmo conseguiu soluções ótimas para instâncias com até 60 clientes, variando a capacidade do veículo em $10, 15, 20, 25$ e 30.

Em seguida, os mesmos autores propuseram duas heurísticas em [37] a fim de tratar instâncias maiores, sendo que a primeira consiste em uma heurística de busca local desenvolvida para fornecer limitantes superiores primais ao *branch-and-cut* descrito em [36] e a outra consiste em aplicar o algoritmo *branch-and-cut* considerando apenas um subconjunto de variáveis (associadas a arestas promissoras), reduzindo assim o espaço de busca. Testes computacionais apresentaram resultados próximos aos ótimos para as instâncias menores, além de conseguir obter soluções de boa qualidade para as instâncias maiores de até 500 clientes, que até então não haviam sido tratadas.

Uma nova versão do algoritmo *branch-and-cut* foi proposta posteriormente em [38], com a adição de um novo conjunto de restrições para o 1-PDTSP, baseadas em desigualdades válidas utilizadas no problema de roteamento de veículos capacitado. Esse algoritmo conseguiu resolver instâncias com até 100 clientes, variando a capacidade do veículo em 10[1], $15, 20, 25$ e 30.

Martinovic *et al.* [51] propuseram uma heurística baseada em *Simmulated Annealing* (SA) modificada e iterativa, que utiliza uma construção gulosa com aleatoriedade. Essa estratégia foi originalmente proposta para o problema de roteamento de veículos com coleta e entrega para um único tipo de produto, mas também foi aplicada ao 1-PDTSP.

Hernández-Pérez *et al.* [40] desenvolveram uma heurística híbrida com componentes da metaheurística GRASP e da heurística *Variable Neighborhood Descent* (VND) para

[1]Exceto para instâncias com $n = 90$ e $n = 100$

resolver o 1-PDTSP. A solução inicial é construída iterativamente, selecionando um novo cliente de acordo com a lista de candidatos restritos, que é ordenada por um critério guloso e adaptativo atualizado a cada iteração. A fase de busca local do GRASP é então substituída pela heurística VND, que utiliza procedimentos de aprimoramento baseados nas trocas 3-opt e 2-opt, utilizadas em problemas de roteamento de veículos. O Capítulo 4 apresenta detalhadamente essa heurística híbrida.

Zhao *et al.* [76] apresentaram um Algoritmo Genético (AG) composto por uma heurística construtiva mais eficiente que a adotada em [40] e uma heurística de busca local para que o algoritmo convergisse mais rapidamente. Além disso, os autores propuseram um novo operador de *crossover* que utiliza informações locais e globais para gerar a população inicial. Testes computacionais melhoraram os resultados obtidos em [40] em termos de qualidade solução.

Hosny e Mumford [42] introduziram uma heurística híbrida baseada nas metaheurísticas VNS e SA. São realizadas várias execuções do VNS e a melhor solução encontrada em cada execução é usada como solução inicial na próxima. A heurística é também adaptativa, no sentido de que o tamanho da vizinhança permitida em cada execução VNS não é fixa, sendo determinada de acordo com o avanço da busca. Durante cada execução do VNS, um critério de aceitação de solução similar ao do SA é aplicado a fim de escapar de ótimos locais e explorar melhor o espaço de busca.

Em [58], Paes *et al.* propuseram uma abordagem que combina GRASP (na fase construtiva), ILS (como método principal) e VND com ordem aleatória de vizinhanças (na fase de busca local) para resolver o 1-PDTSP. Essa heurística híbrida foi testada em 90 instâncias disponíveis na literatura e, em termos de qualidade de soluções, conseguiu melhorar alguns resultados obtidos em [76]. Em geral, o tempo computacional total dessa heurística foi superior ao tempo dos trabalhos anteriores. Por outro lado, o tempo para obter as melhores soluções da literatura era substancialmente menor do que o tempo total do algoritmo.

Recentemente, Mladenović *et al.* [55] propuseram um algoritmo baseado na metaheurística VNS que usa uma nova e eficiente forma de verificar a viabilidade das soluções, que é baseada em uma árvore binária indexada que armazena informações específicas das soluções, ao contrário da estratégia linear até então aplicada. Esse método reduz bastante o esforço computacional necessário nas heurísticas de busca local e, consequentemente, o tempo total do algoritmo. Os autores avaliaram a heurística em instâncias de até 500 clientes, disponíveis na literatura, e melhoraram os resultados obtidos em [76]. Além

disso, o algoritmo foi testado em novas instâncias de 1000 clientes, propostas no mesmo trabalho.

Além desses trabalhos, existem outros na literatura diretamente relacionados ao problema 1-PDTSP, mas que o apresentaram com nomes diferentes. Anily e Bramel [5] trataram o 1-PDTSP como PCV com coleta e entrega capacitado (em inglês, *Capacited Traveling Salesman Problem with Pickups and Deliveries*) fixando quantidade de entrega e coleta igual a um e propuseram dois algoritmos aproximativos com tempo polinomial, analisando os limitantes obtidos nos piores casos desses algoritmos. Chalasani e Motwani [14] consideram o *k-Delivery TSP* como um caso especial do 1-PDTSP, também com quantidade unitária de produto, e propõem algoritmos aproximativos como métodos de solução.

Na seção seguinte, são apresentados alguns problemas bem similares ao 1-PDTSP, mas que possuem outras características, tais como demandas estocásticas, múltiplos produtos, múltiplos veículos, janelas de tempo, entre outros.

3.3 Problemas Correlacionados

Na literatura, também são encontradas algumas variantes do PRV muito similares ao 1-PDTSP. Por exemplo, a utilização de mais de um produto resulta em uma generalização direta do 1-PDTSP, denominada problema do caixeiro viajante com coleta e entrega com múltiplos produtos (m-PDTSP, sigla em inglês), onde cada cliente agora possui uma demanda de m produtos diferentes [39]. Um caso particular desse problema surge quando há somente uma origem e um destino para cada tipo de produto, denominado *one-to-one* m-PDTSP [39].

Outra variante decorre da obrigatoriedade de realizar a coleta (ou a entrega, ou ambas) para determinado cliente. O PRV com coleta e entrega simultânea foi introduzido por Min [53] e um exemplo mais próximo ao 1-PDTSP que possui essa característica é o PCV com coleta seletiva e entrega obrigatória (1-TSP-SELPD, sigla em inglês) [45]. O 1-TSP-SELPD é uma generalização do 1-PDTSP onde nem todos os clientes devem ser visitados e pode ser aplicado ao problema real de reparação de cobertura baseada em transporte de sensores sem fio e rede de robôs. Esse problema consiste em, dados uma rede de sensores fixos que cobrem uma determinada área, divididos em sensores ativos e passivos, e um robô móvel com capacidade limitada, deve-se determinar uma rota para esse robô que colete todos os sensores passivos e reposicione-os em locais cuja cobertura foi prejudicada

pela falha de um sensor ativo.

Outra variação do problema é o PCV com demandas estocásticas [50], onde a demanda dos clientes agora está associada à variável aleatória com probabilidade discreta de distribuição, ou seja, podendo variar em determinadas situações. O PCV com coleta e entrega em ordem de carregamento LIFO é uma variação que considera que a coleta realizada em um cliente é sempre a próxima a ser entregue. Esse problema surge naturalmente quando o veículo considerado possui espaço reduzido de carregamento, ou ainda quando não é possível rearranjar os produtos no veículo de maneira que facilite a entrega no cliente seguinte [17]. Outrossim, o PCV com coleta e entrega para um único produto em um caminho ou árvore [75] é mais uma variação, que recebe como dados de entrada grafos cujos clientes estão, respectivamente, organizados na forma de caminhos e árvores no 1-PDTSP.

O problema de *Swapping*, introduzido por Anily e Hassin [6], é um problema relacionado no qual diversos produtos devem ser transportados por um veículo capacitado de muitas origens para muitos destinos, sem necessariamente passar por todos os clientes, e com a propriedade de que alguns produtos podem ser armazenados temporariamente em um cliente intermediário. O PCV com *backhauls* [23] é o problema em que um veículo não capacitado deve visitar todos os clientes de entrega e, somente após isso, visitar o conjunto de clientes de coleta.

O problema de *Dial-a-Ride* [62] é um problema que envolve transporte de passageiros com locais de origem e destino predeterminados e tem como objetivo criar rotas de custo mínimo que atendam as restrições dos passageiros, sem exceder a capacidade do veículo. Variantes desse problema consideram algumas características específicas, como veículo capacitado, diversos veículos, janelas de tempo e até requisições de carona em tempo real.

O Capítulo 4, a seguir, apresenta o algoritmo proposto neste trabalho. Para isso, inicialmente revisa-se a heurística GRASP/VND proposta em [40], detalhando suas fases de construção e de busca local. Em seguida, apresenta-se como a MD foi inserida no GRASP/VND, dando origem às duas heurísticas híbridas com MD propostas neste trabalho.

Capítulo 4

Proposta do Trabalho

Esta dissertação tem como proposta principal mostrar que metaheurísticas híbridas com MD podem ser aplicadas com sucesso não somente a problemas cujas soluções são representadas por conjuntos, mas também a problemas cujas soluções são representadas pela ordem dos seus elementos.

O problema escolhido foi o 1-PDTSP (apresentado no capítulo anterior) e, como método de solução para esse problema, escolheu-se a heurística proposta por Hernández-Pérez *et al.* [40]. Essa heurística, que usa componentes da metaheurística GRASP e da estratégia de busca local VND, foi usada como base da proposta híbrida do presente trabalho por ser uma estratégia competitiva para o 1-PDTSP e por ser uma estratégia baseada em GRASP. que tem sido utilizado com sucesso em hibridização com MD.

Na próxima seção, será revisado o GRASP/VND proposto por Hernández-Pérez *et al.* [40] para o 1-PDTSP. Nas seções seguintes, são apresentadas as suas versões híbridas com MD.

4.1 Heurística GRASP/VND

A heurística híbrida apresentada em Hernández-Pérez *et al.* [40] possui estrutura semelhante a da metaheurística GRASP clássica, como mostra o Algoritmo 7. A estratégia é composta por um laço principal cujo critério de parada é o número máximo de iterações. Dentro desse laço, ocorrem as fases de construção (linha 4) e de busca local (linha 5). Após o fim desse laço, uma nova busca local é aplicada à melhor solução encontrada (etapa chamada de "otimização final") com o intuito de melhorá-la (linha 10).

Algoritmo 7: Heurística Híbrida para o 1-PDTSP

1: **GRASP/VND** (α, E, *maxIter*)
2: $f(s^*) \leftarrow \infty$
3: **Para** $iter = 1$ **até** *maxIter* **faça**
4: $s \leftarrow$ ConstruçãoGulosaAleatória(α, E)
5: $s \leftarrow VND_1(s)$
6: **Se** s é viável e $f(s) < f(s^*)$ **então**
7: $s^* \leftarrow s$
8: **Fim-se**
9: **Fim-para**
10: $s^* \leftarrow VND_2(s^*)$
11: **Retorne** s^*

4.1.1 Construção

O Algoritmo 8 ilustra a fase de construção. Inicialmente, um cliente é selecionado de maneira aleatória para representar o depósito (linha 3). A seguir, novos clientes são selecionados e inseridos iterativamente na solução, utilizando o conceito de inserção mais próxima, da seguinte forma: para cada iteração, os clientes que não inviabilizam a solução são ordenados de maneira crescente de acordo com a sua distância para o último cliente inserido, e, na linha 6, os α primeiros são incluídos na Lista Restrita de Candidatos (LRC).

Algoritmo 8: Heurística construtiva

1: **ConstruçãoGulosaAleatória** (α, E)
2: $s \leftarrow \emptyset$
3: *depósito* $\leftarrow$ EscolhaAleatória(E)
4: $s \leftarrow s \cup$ *depósito*
5: **Enquanto** $\neg$ SoluçãoCompleta(s) **faça**
6: $LRC \leftarrow$ ViáveisMaisPróximos(E,s,α)
7: **Se** $Vazia(LRC)$ **então**
8: $LRC \leftarrow$ MaisPróximos(E,s,α)
9: **Fim-se**
10: $i \leftarrow$ EscolhaAleatória(LRC)
11: $s \leftarrow s \cup i$
12: **Fim-enquanto**
13: $s \leftarrow s \cup$ *depósito*
14: **Retorne** s

Se nenhum cliente puder ser inserido sem inviabilizar a solução, a LRC é então composta pelos α clientes mais próximos do último inserido (linha 8). No algoritmo, o parâmetro α assume o valor mínimo entre 10 e o número de clientes que puderem ser inseridos na LRC. Por fim, um cliente é escolhido aleatoriamente da LRC e inserido no fim da solução em construção (linha 10). Essa etapa termina quando todos os clientes são

inseridos.

Durante a construção, a distância *dist* entre os clientes i e j são redefinidas com o objetivo de penalizar arcos que conectem clientes do mesmo tipo (coleta → coleta ou entrega → entrega), de acordo com a Equação 4.1. Nessa equação, Q é a capacidade do veículo, que é um dado de entrada do problema, e C é uma constante definida em [37] como $C = \frac{(\sum_{i \in V: q_i > 0} q_i - Q) \sum_{(i,j) \in A} c_{ij}}{(10nQ)}$, que depende dos dados de entrada da instância (custos, demandas e da capacidade do veículo).

$$dist_{ij} = \begin{cases} c_{ij} + C(2Q - |q_i - q_j|), & se \quad |q_i + q_j| \leq Q \\ \infty, & \text{caso contrário.} \end{cases} \tag{4.1}$$

É possível perceber que quanto maior for a distância das demandas $|q_i - q_j|$, menor será o valor acrescentado ao custo original de c_{ij} e, consequentemente, menor será o valor de $dist_{ij}$. Segundo Hernández-Pérez *et al.* [37], essa ideia favorece a construção de soluções viáveis, embora não garanta que isso ocorra.

4.1.2 Busca Local

Após a primeira etapa, a solução construída é submetida a um procedimento de busca local. Esse procedimento, denominado VND_1, é baseado na heurística VND [54], que consiste em aplicar sistematicamente algumas estruturas de vizinhança na solução de entrada, buscando soluções de custo menor. A ordem em que essas vizinhanças são aplicadas é predefinida, e sempre que há melhoria na solução corrente, deve-se retornar o processo de busca à primeira vizinhança.

O Algoritmo 9 mostra um pseudocódigo genérico para o VND. Inicialmente, deve-se escolher um conjunto de vizinhanças N, já na ordem em que essas serão aplicadas. Em seguida, a solução s de entrada é submetida à primeira vizinhança $N_1(s)$ (linha 4). Se essa vizinhança melhorar o custo de s, então deve-se atualizar s com a nova solução s' (linha 6) e retornar à primeira vizinhança da lista (linha 7).

Caso a vizinhança N_k aplicada não obtiver melhoria, então deve-se seguir com a busca na próxima vizinhança N_{k+1}, até que não haja mais vizinhanças a serem exploradas (critério de parada).

O procedimento VND_1 tem como vizinhanças variações das heurísticas clássicas de troca 2-opt [18] e 3-opt [12, 46], com pequenas modificações para aceitarem soluções

Algoritmo 9: Procedimento VND

```
1: VND (s, N)
2: k ← 1
3: Enquanto k < |N| faça
4:    s' ← BuscaLocal(N_k(s))
5:    Se f(s') < f(s) então
6:       s ← s'
7:       k ← 1
8:    senão
9:       k ← k + 1
10:   Fim-se
11: Fim-enquanto
12: Retorne s
```

inviáveis de entrada. Além disso, seguindo-se as ideias de Lin e Kernighan [47], para cada cliente i, são armazenados os vizinhos j mais próximos de i, ordenados de maneira crescente pelo custo c_{ij}. Durante o processo de busca, a fim de diminuir o esforço computacional, somente os clientes presentes nessas listas são considerados e o tamanho de cada lista está relacionado com a quantidade de clientes n, definido por $4\sqrt{n}$.

A Figura 4.1 ilustra como acontecem as trocas na vizinhança 2-opt. Dois arcos não adjacentes são escolhidos e removidos da rota, gerando duas subrotas desconexas. Em seguida, essas duas rotas são reconectadas por dois novos arcos, da única maneira factível. É possível observar que uma das subrotas é completamente invertida durante a reconexão.

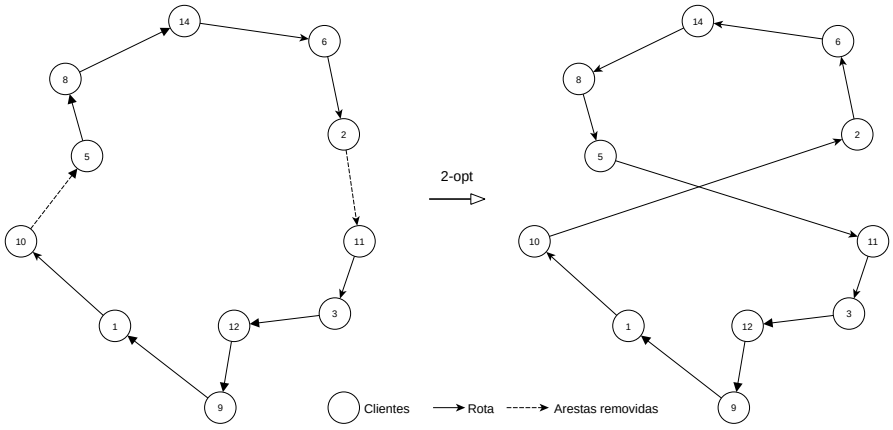


Figura 4.1: Troca 2-opt

A Figura 4.2 ilustra como acontecem as trocas na vizinhança 3-opt. Três arcos não adjacentes são escolhidos e removidos da rota, gerando três subrotas desconexas. Em seguida, existem oito possibilidades de reconectar essas três rotas, das quais a melhor é escolhida. A Figura 4.2 mostra somente uma das oito opções para reconectar as subrotas

e, ainda, é a única que mantém a direção original da rota.

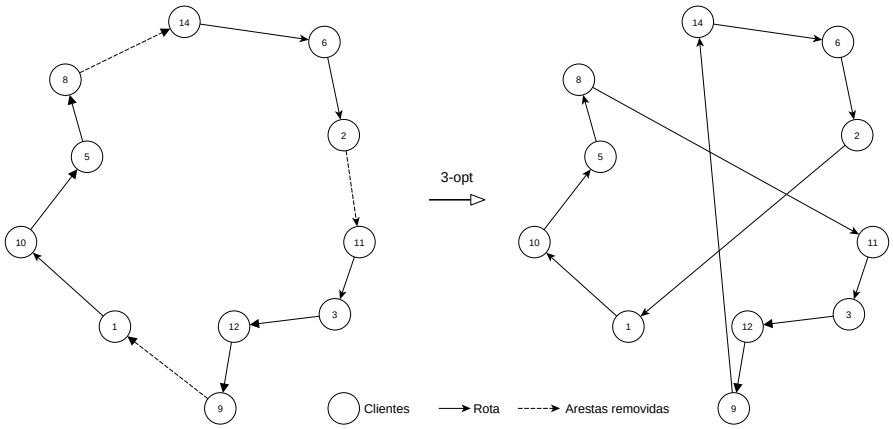


Figura 4.2: Troca 3-opt sem alterar direção da rota

Assim, o VND_1 utiliza essas duas vizinhanças em ordem crescente de esforço computacional. Primeiro a heurística 2-opt e, em seguida, aplica-se a heurística 3-opt. Como mencionado anteriormente, essas duas vizinhanças foram adaptadas para aceitarem soluções inviáveis de entrada. Assim, as vizinhanças devem ser capazes de tentar conduzir essas soluções à viabilidade.

Para gerenciar as soluções inviáveis geradas, Hernández-Pérez *et al.* [40] definiram uma função para avaliar a inviabilidade da solução, denominada $infeas(x)$, e um limiar de inviabilidade, denominado $threshold$. Com isso, cada um dos movimentos (do 2-opt e do 3-opt) que conduza a uma solução s' de menor custo será aceito se: i) s' é viável ou ii) se s' é inviável e $infeas(s') < threshold$.

No primeiro caso, s' é aceita e o valor de $threshold$ é atualizado com um valor muito pequeno, ϵ, próximo de zero, e, portanto, não permitindo mais soluções inviáveis na busca. No segundo caso, $threshold$ é atualizado com o valor de $infeas(s)$ e a busca prossegue com o novo valor de $threshold$. Segundo Hernández-Pérez *et al.* [40], com essa ideia foi possível variar a região do espaço de busca, evitando que ótimos locais ocorressem prematuramente.

Para calcular a inviabilidade das soluções foi utilizada a função $infeas(s)$, definida em [37] e apresentada na Equação 4.2. Considera-se l_i a carga do veículo após visitar o cliente i.

$$infeas(s) = \max_{i=0}^{n}\{l_i(s)\} - \min_{i=0}^{n}\{l_i(s)\} - Q \tag{4.2}$$

A solução s é viável sempre que $infeas(s) \leq 0$. O valor de $threshold$ inicial é calculado de acordo com a Equação 4.3.

$$threshold = 3 \max\{\sum_{i \in V: q_i > 0} q_i, -\sum_{i \in V: q_i < 0} q_i\}/n \tag{4.3}$$

Assim, usando as ideias anteriores, foi possível reconduzir soluções inviáveis à viabilidade. Além disso, o valor inicial de $threshold$, como definido na Equação 4.3, permite que soluções inicialmente viáveis possam aceitar movimentos que a tornem inviável (mas com custo menor) inicialmente, mas, com a atualização do $threshold$, ela poderá ser reconduzida à viabilidade posteriormente.

Além da busca local VND_1, como já foi mencionado, o GRASP/VND possui outra busca local, que é aplicada após o fim do laço principal. Essa etapa, denominada "otimização final", é composta por outra heurística VND, representada por VND_2, que é aplicada à melhor solução encontrada até o momento. Diferentemente do VND_1, o VND_2 só permite movimentos viáveis (com menor custo).

O procedimento VND_2 é composto pela estrutura de vizinhança denominada *Reinsertion* [56][1], dividida em dois operadores, aplicados na seguinte ordem: *Reinsertion Forward*, *Reinsertion Backward*.

A Figura 4.3 ilustra como acontecem as trocas na vizinhança *Reinsertion Forward*. Um cliente é removido da solução e reinserido em outra posição posterior à posição da qual foi retirado.

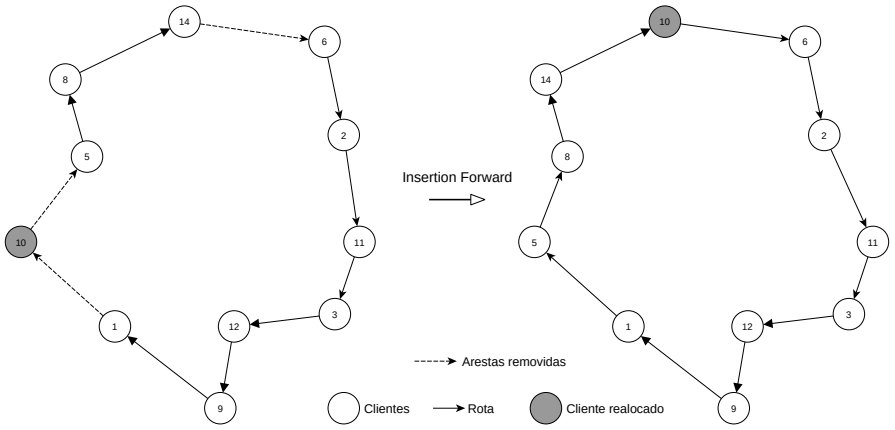


Figura 4.3: *Reinsertion Forward*

[1] A vizinhança *Reinsertion*, originalmente desenvolvida para o PCV, é também conhecida como *Or-Opt*, quando apenas uma cidade é selecionada para ser movida.

A Figura 4.4 mostra como acontecem as trocas na vizinhança *Reinsertion Backward*. Um cliente é removido da solução e, ao contrário da *Forward*, é reinserido em outra posição anterior à posição da qual foi retirado.

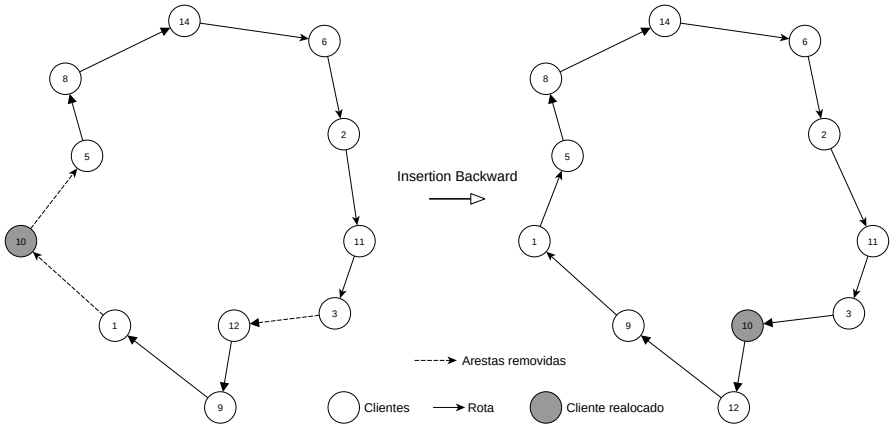


Figura 4.4: *Reinsertion Backward*

Nas seções a seguir, serão descritas duas abordagem que combinam mineração de dados com a heurística GRASP/VND, dando origem às heurísticas DM-GRASP/VND e MDM-GRASP/VND.

4.2 Heurística DM-GRASP/VND

Na área de mineração de dados (MD), a técnica de mineração de conjuntos frequentes (MCF) consiste em extrair padrões de uma base de dados de transações, onde cada transação é formada por um conjunto de elementos do domínio de aplicação. Por exemplo, cada transação da base pode ser composta por produtos comprados por um consumidor. Na aplicação de MD em problemas de otimização combinatória (OC), cada transação é, na verdade, uma possível solução para o problema em questão, e a base de dados consiste em um conjunto de soluções.

Nos trabalhos anteriores que combinam MD com metaheurísticas, as soluções dos problemas de OC são representadas por conjuntos de elementos. Dessa forma, a utilização da MCF foi natural, diretamente aplicável, e bem sucedida.

Para problemas cuja solução é representada por uma sequência de elementos, este trabalho propõe mapear cada par de elementos (arcos) consecutivos em um conjunto de identificadores únicos, de modo que uma solução possa ser representada por um conjunto de itens não ordenados. O mapeamento acontece gerando um identificador a_{ij} para cada

dois elementos i e j consecutivos da solução, conservando a ordem em que eles estão dispostos.

Para exemplificar essa ideia, considera-se a solução $s = \{5, 2, 7, 4, 1, 6, 3\}$ de um problema que envolve ordem, representada por uma sequência entre seus elementos. Essa solução pode ser mapeada em $s_{map} = \{a_{52}, a_{27}, a_{74}, a_{41}, a_{16}, a_{63}\}$, que representa a mesma solução s. O identificador a_{52}, por exemplo, representa o arco formado pelos elementos 5 e 2, nesta ordem.

Dessa maneira, a solução na forma de s_{map} é composta por um conjunto de itens cuja ordem não é mais importante. Qualquer que seja a ordem em que os elementos $a_{52}, a_{27}, a_{74}, a_{41}, a_{16}$ e a_{63} estejam dispostos, representará a mesma solução original $s = \{5, 2, 7, 4, 1, 6, 3\}$.

Torna-se possível então a utilização de técnicas de MCF em problemas que envolvem sequência, sem que essa informação seja perdida, pois o mapeamento inverso é trivial e direto, o que caracteriza uma das contribuições desta dissertação.

Assim, a heurística híbrida com MD proposta neste trabalho combina a heurística GRASP/VND descrita anteriormente com a técnica de mineração de conjuntos frequentes maximais *FPmax**, utilizando o mapeamento detalhado anteriormente para viabilizar essa combinação. A partir desse ponto, essa estratégia híbrida será denominada apenas DM-GRASP/VND.

O DM-GRASP/VND segue as características originais propostas por Ribeiro *et al.* [65, 66], sendo dividido em duas etapas, descritas nas subseções seguintes.

4.2.1 Primeira Fase: Geração do CE e Mineração

A primeira fase do algoritmo proposto, denominada fase de geração do conjunto elite, consiste em executar um número fixo k de iterações do GRASP/VND, armazenando as d melhores soluções encontradas no conjunto elite de soluções (CE).

Em seguida, é aplicado o processo de mineração no CE para extrair os padrões. Entretanto, como as soluções para o 1-PDTSP envolvem ordem, deve-se primeiro mapear essas soluções em identificadores únicos, de maneira que a mineração possa ser executada.

O mapeamento das soluções é feito da seguinte maneira. Dado um par de clientes sequenciais, representado pelo arco (i, j), o identificador único que representa esse arco é calculado por $a_{ij} = (i * n) + j$, onde n é o número de clientes envolvidos. Dessa maneira,

também é possível realizar o processo inverso, recuperando o arco (i, j) a partir de a_{ij} com os seguintes cálculos: $i = (a_{ij}\ div\ n)$ e $j = (a_{ij}\ mod\ n)$, sendo div e mod os operadores matemáticos que informam o quociente e o resto da divisão, respectivamente.

Após o mapeamento, aplica-se a mineração, que fornecerá os $numP$ maiores padrões (em relação à quantidade de arcos presentes nos padrões) que serão utilizados na próxima etapa. Antes disso, os identificadores contidos nos padrões são mapeados de volta em forma de arcos, como explicado anteriormente.

4.2.2 Segunda Fase: Iterações Híbridas

Em seguida, inicia-se a segunda fase do DM-GRASP/VND, denominada fase híbrida. Nessa etapa, a construção original é substituída por uma construção adaptada, que utiliza os padrões minerados como base para construir novas soluções, executando o mesmo número fixo k de iterações da primeira fase.

Cada padrão minerado é composto por um conjunto de arcos que se repetiram juntos em sup_{min} soluções do CE, parâmetro conhecido como suporte mínimo. Ser frequente em sup_{min} soluções distintas indica que os clientes i e j foram visitados consecutivamente nessas soluções e, por isso, devem ser considerados na construção de novas soluções. A quantidade e o tamanho dos padrões minerados variam conforme o valor de sup_{min}.

Além disso, dentro de cada padrão, pode ocorrer de dois (ou mais) arcos serem consecutivos e, se conectados, fornecem segmentos de rota maiores. Esses segmentos, denominados componentes conexas (CC), são identificados e ordenados de maneira decrescente de tamanho para serem utilizados na construção adaptada. Por exemplo, o padrão $p_1 = \{a_{12}, a_{58}, a_{34}, a_{75}, a_{83}, a_{26}, a_{60}\}$ possui duas componentes conexas: $cc_1 = \{a_{75}, a_{58}, a_{83}, a_{34}\}$ e $cc_2 = \{a_{12}, a_{26}, a_{60}\}$.

O Algoritmo 10 apresenta os passos da construção adaptada. Em cada construção, um dos $numP$ padrões é selecionado de maneira *round-robin* (linha 2). Em seguida, uma CC do padrão escolhido é selecionada de acordo com a quantidade de vezes que aquele padrão foi utilizado, ou seja, na primeira vez que o padrão é usado, escolhe-se a maior CC, na segunda vez a segunda maior e assim por diante (linha 3).

Uma vez selecionada a CC, a construção de uma nova solução s é guiada da seguinte maneira: identificam-se todas as soluções do CE que contêm a CC em sua rota e escolhe-se uma aleatoriamente, identificada como s_{esc} (linha 4). É importante ressaltar que o número de soluções que contém CC é sempre diferente de zero, sendo pelo menos igual a

Algoritmo 10: Construção adaptada para utilizar padrões

```
1: ConstruçãoAdaptada(α, listaPadrões, CE, E)
2: p_esc ← SelecionaPadrão(listaPadrões)
3: cc_esc ← SelecionaCC(p_esc)
4: s_esc ← SelecionaSoluçãoQueContémCC(cc_esc, CE)
5: s ← ExtraiSubrota(cc_esc, s_esc)
6: Enquanto ¬ SoluçãoCompleta(s) faça
7:    LRC ← ViáveisMaisPróximos(s, α, E)
8:    Se Vazia(LRC) então
9:       LRC ← MaisPróximos(s, α, E)
10:   Fim-se
11:   i ← EscolhaAleatória(LRC)
12:   s ← s ∪ i
13: Fim-enquanto
14: s ← s ∪ depósito
15: Retorne s
```

sup_{min}.

A solução em construção s recebe inicialmente uma parte da rota de s_{esc}, que começa no depósito e vai até o fim da CC em s_{esc} (linha 5). A partir desse ponto, constrói-se uma rota distinta, inserindo os clientes sempre no fim da solução, aplicando a mesma ideia da heurística construtiva original (linha 6).

Finalizada a construção, aplica-se o mesmo procedimento de busca local que é usado no GRASP/VND, o VND_1. Finalmente, após as iterações da etapa híbrida, acontece também a fase de "otimização final", com o procedimento VND_2.

O Algoritmo 11 apresenta o pseucódigo do DM-GRASP/VND. É composto por dois laços idênticos ao Algoritmo 7, das linhas 4 a 11 e da 13 a 19. Cada laço utiliza metade do número máximo de iterações e representa cada etapa do DM-GRASP/VND. Suas principais modificações em relação ao Algoritmo 7 estão representadas nas linhas 10, 12 e 14. A construção do CE é realizada na primeira etapa, linha 10, e o processo de mineração é aplicado entre os dois laços, na linha 12. A nova construção é inserida na segunda etapa, substituindo a construção original na linha 14.

A próxima seção fornecerá um exemplo de execução da heurística DM-GRASP/VND, desde a geração do conjunto elite de soluções até a fase de construção adaptada, a fim de facilitar o entendimento da proposta.

Algoritmo 11: Heurística híbrida com mineração de dados

1: **DM-GRASP/VND** (α, E, k, sup_{min}, d, $numP$)
2: $f(s^*) \leftarrow \infty$
3: $CE \leftarrow \emptyset$
4: **Para** $iter = 1$ **até** k **faça**
5: $s \leftarrow$ ConstruçãoGulosaAleatória(α, E)
6: $s \leftarrow VND_1(s)$
7: **Se** s é viável e $f(s) < f(s^*)$ **então**
8: $s^* \leftarrow s$
9: **Fim-se**
10: AtualizaCE(s, CE, d)
11: **Fim-para**
12: *listaPadrões* $\leftarrow$ ExecutaMineração(CE, sup_{min}, $numP$)
13: **Para** $iter = 1$ **até** k **faça**
14: $s \leftarrow$ ConstruçãoAdaptada(α, *listaPadrões*, CE, E)
15: $s \leftarrow VND_1(s)$
16: **Se** s é viável e $f(s) < f(s^*)$ **então**
17: $s^* \leftarrow s$
18: **Fim-se**
19: **Fim-para**
20: $s^* \leftarrow VND_2(s^*)$
21: **Retorne** s^*

4.2.3 Ilustração da Heurística DM-GRASP/VND

Esta seção tem como objetivo ilustrar a execução do DM-GRASP/VND, desde a construção do conjunto elite e extração dos padrões (e suas componentes conexas) até a fase de construção adaptada. Para esse exemplo, escolheu-se uma instância do 1-PDTSP com número de clientes $n = 20$ e capacidade do veículo $Q = 10$, denominada n20q10A, cujos clientes estão representados na Figura 4.5(a). Essa instância foi selecionada por ser pequena e facilitar a visualização das suas soluções.

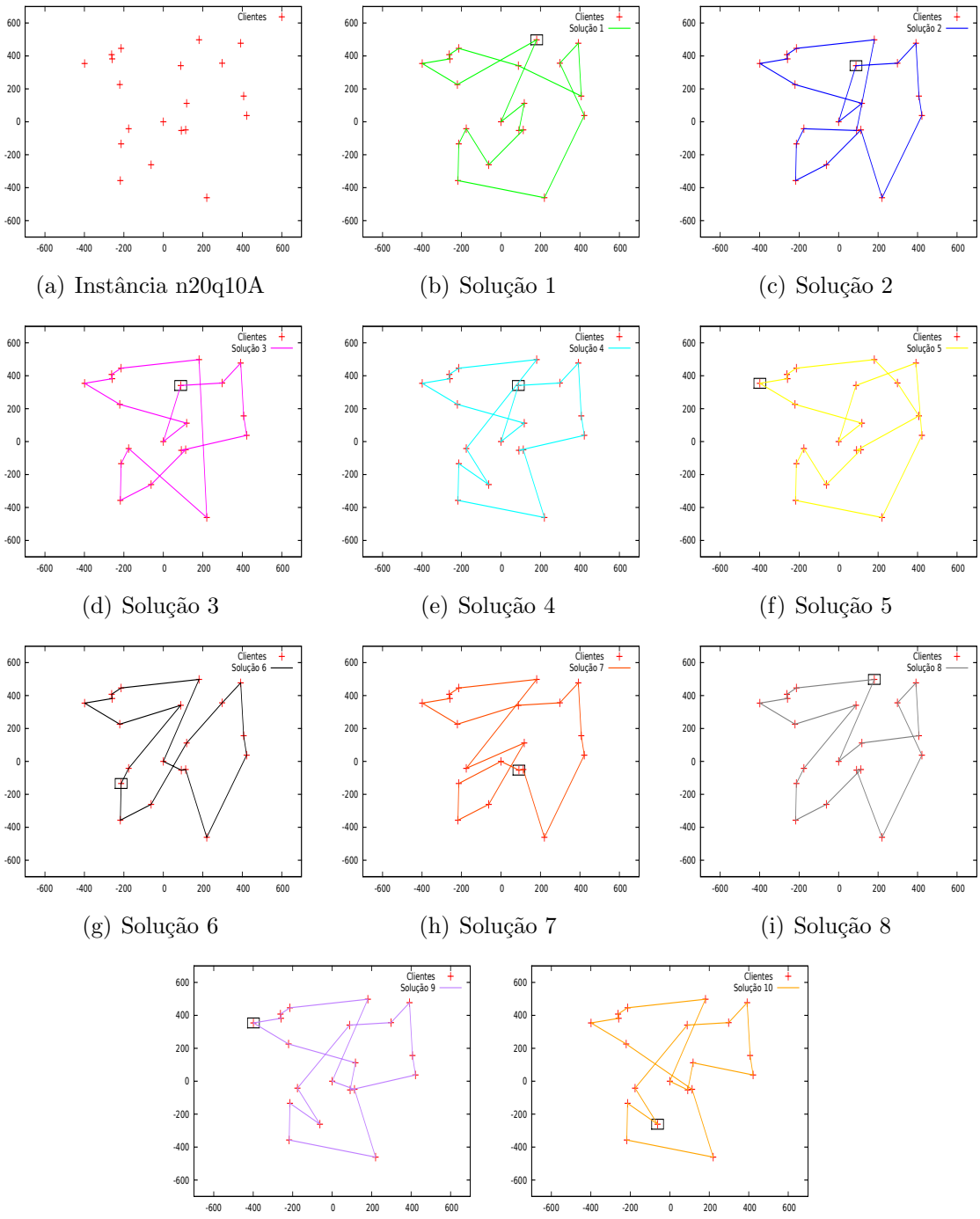
Após a execução de k iterações reservadas para a construção do CE, obtêm-se dez soluções do CE representadas pelas Figuras 4.5(b) até 4.5(k). Os clientes são ilustrados por uma cruz, a ligação entre clientes por uma aresta e o depósito por um retângulo.

Em seguida, os conjuntos frequentes de arcos são extraídos através do processo de mineração, que é aplicado sobre as soluções do CE. Conforme visto anteriormente, esses conjuntos possuem arcos consecutivos que podem ser conectados em componentes conexas. Para esse exemplo, foram minerados dez padrões, que estão ilustrados nas Figuras 4.6(a) até 4.6(j), separados por suas CC.

A construção é iniciada com a escolha de um padrão de maneira *round robin* e, assim, escolhe-se o *Padrão 1*, representado pela Figura 4.6(a). Nessa figura, também é possível visualizar suas CCs. Para esse padrão, as suas componentes conexas são ordenadas por tamanho de maneira decrescente e, como é a primeira vez que o *Padrão 1* é escolhido, deve-se utilizar a maior CC.

Após a escolha da CC, o próximo passo é procurar no CE quais soluções possuem essa CC e, então, sortear uma para iniciar a construção. No caso, para o *Padrão 1*, a maior CC é a CC 1, representada na Figura 4.7(c), e as soluções que a contém são as de número 2 e 6. Após o sorteio, inicia-se a construção fixando os clientes que ocorrem na solução sorteada desde o depósito até o final da CC 1. As Figuras 4.7(d) e 4.7(e) ilustram as duas possíveis soluções construídas a partir da CC 1, a primeira utilizando a solução 2 e a segunda utilizando a solução 6. Por fim, deve-se continuar a construção seguindo a mesma ideia do construtivo original do GRASP/VND, explicado anteriormente.

A próxima seção apresenta outra proposta híbrida com mineração de dados que tem como base o GRASP/VND, denominada MDM-GRASP/VND, e que aplica o processo de mineração mais de uma vez.



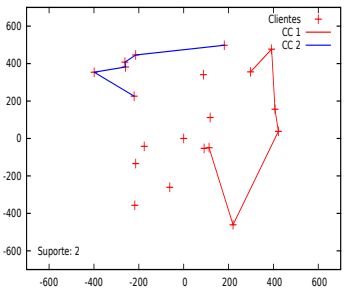
(a) Instância n20q10A (b) Solução 1 (c) Solução 2

(d) Solução 3 (e) Solução 4 (f) Solução 5

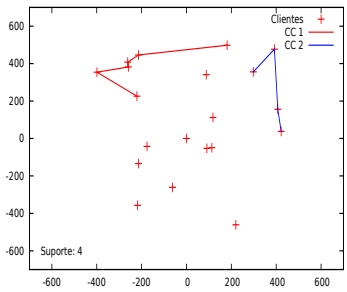
(g) Solução 6 (h) Solução 7 (i) Solução 8

(j) Solução 9 (k) Solução 10

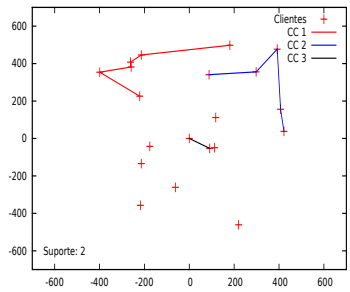
Figura 4.5: Soluções do Conjunto Elite



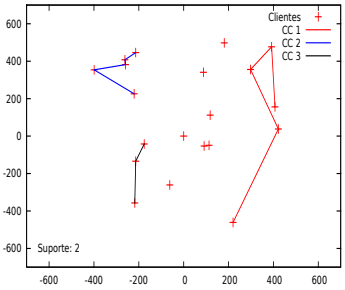
(a) Padrão 1



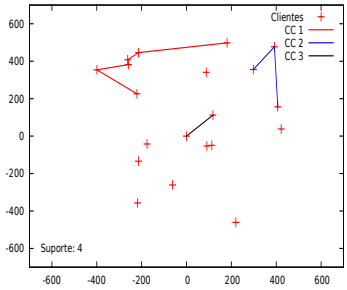
(b) Padrão 2



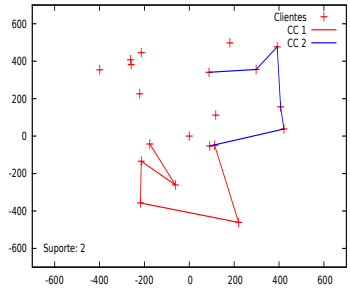
(c) Padrão 3



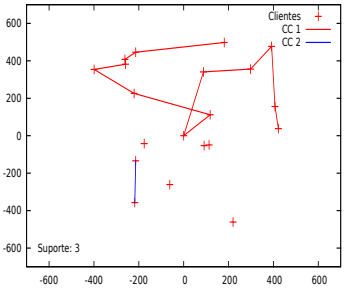
(d) Padrão 4



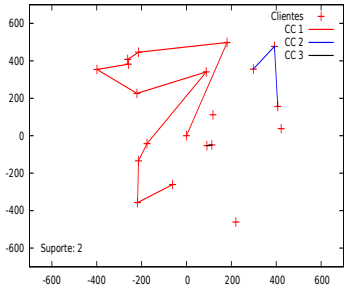
(e) Padrão 5



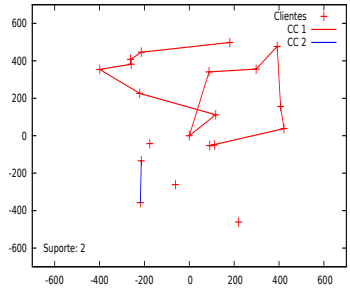
(f) Padrão 6



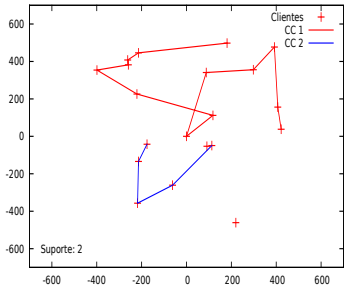
(g) Padrão 7



(h) Padrão 8

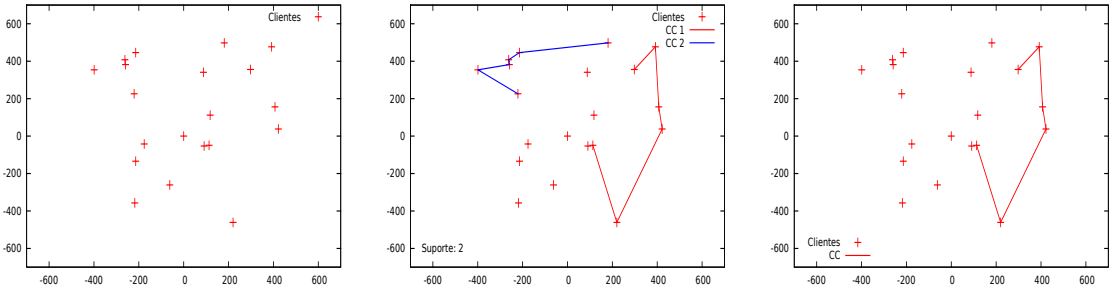


(i) Padrão 9



(j) Padrão 10

Figura 4.6: Padrões Minerados



(a) Instância n20q10A (b) Padrão 1 (c) Padrão 1 - CC 1

(d) Construção 1 (e) Construção 2

Figura 4.7: Ilustração do início da construção com padrões

4.3 Heurística MDM-GRASP/VND

Na heurística híbrida com mineração da dados DM-GRASP/VND, apresentada anteriormente, o processo de mineração é aplicado somente uma vez exatamente na metade do número total de iterações. Além dessa estratégia, este trabalho também propõe e examina uma estratégia que aplica a mineração mais de uma vez durante a execução do algoritmo.

Essa abordagem, denominada MDM-GRASP/VND, baseia-se na ideia de que aplicando o processo de mineração múltiplas vezes, os padrões minerados se tornam cada vez mais refinados. No MDM-GRASP/VND, a mineração é executada sempre que o CE se torna estável, ou seja, sempre que o CE permanece inalterado por um número fixo γ de iterações consecutivas, informado como parâmetro de entrada.

O Algoritmo 12 mostra os passos do MDM-GRASP/VND. O primeiro laço (linhas 4 até 16) é similar à primeira fase do DM-GRASP/VND e corresponde a executar algumas iterações do GRASP/VND original a fim de construir o CE.

Assim que o CE se torna estável, o algoritmo passa para o segundo laço (linhas 17 até 33), onde o processo de mineração é aplicado (linha 19). A seguir, inicia-se a etapa de construção que utiliza padrões, similar a segunda etapa do DM-GRASP/VND. Entretanto, a mineração é novamente aplicada sempre que o CE permanecer γ iterações consecutivas sem alterações.

O Capítulo 5 apresenta os resultados computacionais obtidos pelas três estratégias descritas neste capítulo: GRASP/VND, DM-GRASP/VND e MDM-GRASP/VND.

Algoritmo 12: Heurística híbrida com múltipla mineração de dados

1: **Função** MDM-GRASP/VND (α, E, *maxIter*, sup_{min}, γ)
2: $f(s^*) \leftarrow f(s) \leftarrow \infty$
3: $CE \leftarrow \emptyset$; $iter_{sm} \leftarrow 0$; $iter \leftarrow 0$
4: **Enquanto** $iter_{sm} < \gamma$ **faça**
5: $s \leftarrow$ ConstruçãoGulosaAleatória(α, E)
6: $s \leftarrow VND_1(s)$
7: **Se** s é viável e $f(s) < f(s^*)$ **então**
8: $s^* \leftarrow s$
9: **Fim-se**
10: **Se** AtualizaCE(s, CE) **então**
11: $iter_{sm} \leftarrow 0$
12: **senão**
13: $iter_{sm} \leftarrow iter_{sm} + 1$
14: **Fim-se**
15: $iter \leftarrow iter + 1$
16: **Fim-enquanto**
17: **Enquanto** $iter < maxIter$ **faça**
18: **Se** $iter_{sm} \geq \gamma$ **então**
19: *listaPadrões* $\leftarrow$ ExecutaMineração(CE, sup_{min})
20: $iter_{sm} \leftarrow 0$
21: **Fim-se**
22: $s \leftarrow$ ConstruçãoAdaptada(α, *listaPadrões*, CE)
23: $s \leftarrow VND_1(s)$
24: **Se** s é viável e $f(s) < f(s^*)$ **então**
25: $s^* \leftarrow s$
26: **Fim-se**
27: **Se** AtualizaCE(s, CE) **então**
28: $iter_{sm} \leftarrow 0$
29: **senão**
30: $iter_{sm} \leftarrow iter_{sm} + 1$
31: **Fim-se**
32: $iter \leftarrow iter + 1$
33: **Fim-enquanto**
34: $s \leftarrow VND_2(s)$
35: **Retorne** s^*

Capítulo 5

Resultados Computacionais

Neste capítulo, são apresentados os experimentos computacionais realizados com as três estratégias descritas no Capítulo 4, denominadas GRASP/VND, DM-GRASP/VND e MDM-GRASP/VND, com o objetivo de avaliar a inserção de mineração de dados na heurística base GRASP/VND, proposta em [40].

Este capítulo está organizado da seguinte maneira. Inicialmente, são detalhados o ambiente de execução dos algoritmos e as instâncias utilizadas nos testes. Em seguida, são apresentados os valores dos parâmetros utilizados por cada um dos algoritmos. A seguir, os resultados das estratégias híbridas com MD são comparados com os resultados da versão original do algoritmo. Finalmente, é realizada uma análise do comportamento dos três algoritmos a fim de comprovar o benefício da introdução da MD.

5.1 Ambiente de Execução

A heurística GRASP/VND de Hernández-Pérez *et al.* [40] e as duas heurísticas propostas neste trabalho foram implementadas na linguagem C++ e compiladas com $g++$ versão 4.6.3. Uma vez que os autores do GRASP/VND não disponibilizaram a sua implementação, foi necessária, para uma comparação justa, reimplementá-la e inserir o módulo de mineração de dados nesta implementação. Os testes computacionais foram executados em um computador equipado com processador Intel®Core™ i5 CPU 650 @ 3.20GHz, com 4GB de memória RAM e Sistema Operacional Linux Fedora versão 15. Somente um *core* foi utilizado nos testes.

5.2 Instâncias Utilizadas

O conjunto de instâncias utilizadas neste trabalho foram propostas[1] em [36] e [40], e consistem em problemas que possuem de 20 até 500 clientes, com o valor de capacidade do veículo Q variando no conjunto $\{10, 20, 30, 40\}$. A localização dos clientes nas instâncias foram geradas aleatoriamente no espaço $[-500, 500]$ x $[-500, 500]$, onde cada cliente possui uma demanda escolhida aleatoriamente no intervalo $[-10, 10]$. O custo de roteamento entre os clientes i e j, representado por $c(i, j)$, está relacionado com a distância euclidiana $d(i, j)$ calculada entre eles, onde $c(i, j) = \lfloor d(i, j) \rfloor$.

A identificação de cada instância é composta pela quantidade de clientes envolvidos, seguida do valor da capacidade do veículo, por exemplo, n100q10. Para cada n e Q, foram geradas dez instâncias aleatoriamente, que são diferenciadas por uma letra no conjunto {A, B, ..., I, J}, adicionada ao final do nome da instância. Assim, as instâncias com $n = 100$ e $Q = 10$ são chamadas de n100q10A a n100q10J.

As instâncias podem ser divididas em dois conjuntos: instâncias pequenas, com n variando no conjunto $\{20, 30, 40, 50, 60\}$, e grandes instâncias, com n variando no conjunto $\{100, 200, 300, 400, 500\}$. Vale ressaltar que quanto menor o valor de Q e quanto maior o valor de n, maior é a dificuldade do problema.

A próxima seção apresenta, inicialmente, os valores dos parâmetros adotados em cada uma das três estratégias e, em seguida, compara os resultados computacionais obtidos nos dois conjuntos de instâncias.

5.3 Comparação Entre as Estratégias

Para realizar a comparação entre as estratégias, cada uma foi executada dez vezes, com dez sementes distintas, sendo reportado, em cada caso, o custo da melhor solução obtida, a média de custo de solução e o tempo computacional médio despendido.

Para os algoritmos DM-GRASP/VND e MDM-GRASP/VND, o número máximo de iterações *maxIter*, o tamanho do conjunto elite d, o valor de suporte mínimo sup_{min} e o número de padrões $numP$ foram, respectivamente, 200, 10, 20% e 10. O valor de k é fixado em $maxIter/2$ para o DM-GRASP/VND. Para a versão MDM, o valor de γ foi definido como $maxIter * 0.05$. Com exceção do número de iterações, que foi definido

[1] No site http://hhperez.webs.ull.es/PDsite/#XM94, é possível obter o conjunto completo com todas as instâncias. Acessado em 1 de agosto de 2013.

de acordo com o critério de parada do GRASP/VND de Hernández-Pérez *et al.* [40], os valores dos demais parâmetros foram definidos a partir das recomendações de Plastino *et al.* em [61].

As Tabelas 5.1 e 5.2 apresentam, respectivamente, os resultados computacionais obtidos para o conjunto de instâncias menores, com Q variando em $\{10, 20, 30, 40\}$, e para o conjunto de instâncias maiores, com Q variando em $\{20, 30, 40\}$. Como esses conjuntos são compostos por instâncias de baixo de nível de dificuldade, optou-se por apresentar um resumo dos testes, reportando, para cada heurística híbrida com MD, a diferença percentual média dos três valores obtidos (custo da melhor solução, média do custo de solução e tempo computacional médio) em relação aos valores obtidos pelo GRASP/VND, agrupando as instâncias com mesmos valores de n e Q. Assim, cada linha das Tabelas 5.1 e 5.2 representa a diferença percentual média obtida por dez instâncias de cada combinação n e Q. Para o cálculo dessa diferença percentual, considera-se a Equação 5.1.

$$Dif\% = \frac{Valor\ do\ Algoritmo\ proposto - Valor\ do\ GRASP/VND}{Valor\ do\ GRASP/VND} \tag{5.1}$$

Como o 1-PDTSP é um problema de minimização, se a diferença percentual de uma determinada comparação é negativa, caracteriza-se um ganho obtido pelo algoritmo proposto (no caso, DM-GRASP/VND ou MDM-GRASP/VND) sobre o GRASP/VND. Caso contrário, indica-se um melhor desempenho do GRASP/VND sobre o algoritmo proposto.

No geral, os resultados da Tabela 5.1 indicam que, quanto mais fácil é a instância (menor n e maior Q), menor é o ganho obtido pelas heurísticas híbridas com mineração de dados em termos de melhor solução e qualidade média de solução. Por outro lado, quanto mais difícil se torna a instância (maior n e menor Q), maior se torna a diferença percentual.

Percebe-se também que, em termos de tempo computacional, as versões com MD apresentam um melhor desempenho em todo o conjunto de instâncias pequenas, com destaque para o MDM-GRASP/VND, que obteve na média geral uma redução de 35.25% do tempo em relação ao GRASP/VND. Além disso, a média geral para todas as instâncias em termos de melhor solução e de qualidade média de solução também foram negativas sendo, respectivamente, -0.05 e -0.09 para o DM-GRASP/VND, e -0.06 e -0.03 para o MDM-GRASP/VND.

Observando a Tabela 5.2, que considera as instâncias maiores com $Q \in \{20, 30, 40\}$, é possível perceber que, ambas as estratégias DM-GRASP/VND e MDM-GRASP/VND

Tabela 5.1: Diferença percentual média em relação ao GRASP/VND para as instâncias pequenas, com $Q \in \{10, 20, 30, 40\}$

Instâncias		DM-GRASP/VND			MDM-GRASP/VND		
n	Q	Dif % média Melhor solução	Dif % média Média solução	Dif % média Tempo médio	Dif % média Melhor solução	Dif % média Média solução	Dif % média Tempo médio
20	10	0.00	0.13	-18.37	0.00	0.31	-24.92
30	10	0.00	0.03	-23.88	0.00	0.13	-35.71
40	10	-0.06	-0.32	-26.17	0.16	0.09	-40.35
50	10	-0.59	-0.55	-28.46	-0.85	-0.44	-42.56
60	10	-0.28	-0.70	-26.93	-0.54	-0.89	-42.83
20	20	0.00	0.02	-19.01	0.00	0.02	-23.35
30	20	0.00	0.06	-24.83	0.00	0.06	-36.24
40	20	0.00	0.03	-22.66	0.00	0.09	-36.15
50	20	0.07	-0.04	-22.61	-0.09	-0.10	-38.44
60	20	-0.16	-0.52	-23.42	-0.12	-0.36	-40.49
20	30	0.00	0.00	-17.39	0.00	0.00	-18.56
30	30	0.00	0.01	-25.48	0.00	0.01	-37.18
40	30	0.00	0.03	-23.64	0.00	0.09	-38.62
50	30	0.00	-0.04	-23.81	0.00	0.04	-37.15
60	30	0.06	-0.08	-25.21	0.17	0.06	-37.22
20	40	0.00	0.00	-25.85	0.00	0.00	-22.53
30	40	0.00	0.00	-29.42	0.00	0.07	-37.09
40	40	0.00	0.03	-30.90	0.00	0.04	-39.24
50	40	0.00	0.03	-27.41	0.01	0.16	-38.05
60	40	0.01	0.06	-27.25	0.05	-0.03	-38.41
Média Geral		**-0.05**	**-0.09**	**-24.63**	**-0.06**	**-0.03**	**-35.25**

apresentam desempenhos superiores ao GRASP/VND original nos três critérios de comparação: melhor custo de solução obtido, médias de custo de solução e tempo computacional. Em todos os grupos de instâncias observam-se médias negativas, o que indica o ganho dos algoritmos híbridos com MD em relação ao algoritmo original. A diferença percentual média geral, para melhor custo de solução solução, média de custo de solução e tempo computacional foram, respectivamente, -0.61, -0.61 e -24.17 para o DM-GRASP/VND. Para o MDM-GRASP/VND, esses valores foram, respectivamente, -1.77, -1.67 e -35.81.

Tabela 5.2: Diferença percentual média em relação ao GRASP/VND para as instâncias maiores, com $Q \in \{20, 30, 40\}$

Instâncias		DM-GRASP/VND			MDM-GRASP/VND		
n	Q	Dif % média Melhor solução	Dif % média Média solução	Dif % média Tempo médio	Dif % média Melhor solução	Dif % média Média solução	Dif % média Tempo médio
100	20	-0.96	-0.89	-23.81	-2.15	-1.80	-36.02
200	20	-1.03	-1.12	-23.37	-2.49	-2.21	-36.29
300	20	-0.88	-0.85	-23.61	-2.55	-2.33	-35.64
400	20	-0.55	-0.83	-23.15	-2.14	-2.27	-35.55
500	20	-0.87	-0.96	-23.95	-2.45	-2.30	-31.68
100	30	-0.56	-0.49	-23.84	-1.32	-1.32	-34.18
200	30	-0.58	-0.46	-24.71	-2.14	-1.79	-34.30
300	30	-0.46	-0.67	-23.65	-1.69	-1.68	-34.68
400	30	-0.59	-0.65	-23.92	-2.20	-1.88	-36.69
500	30	-0.86	-0.70	-23.62	-1.88	-1.79	-35.60
100	40	-0.46	-0.43	-24.57	-0.71	-1.05	-37.12
200	40	-0.39	-0.35	-24.77	-1.19	-1.32	-37.57
300	40	-0.27	-0.25	-24.11	-1.16	-1.12	-37.36
400	40	-0.27	-0.27	-24.06	-1.16	-1.10	-37.82
500	40	-0.38	-0.28	-27.37	-1.28	-1.08	-36.66
Média Geral		**-0.61**	**-0.61**	**-24.17**	**-1.77**	**-1.67**	**-35.81**

As Tabelas 5.3, 5.4 e 5.5 apresentam os resultados obtidos por cada uma das três heurísticas nas instâncias maiores, com $Q = 10$, que são consideradas as mais difíceis. Essas tabelas reportam, para cada instância, o custo da melhor solução obtida, o valor médio do custo de solução e o tempo computacional médio das heurísticas, relativos às dez execuções de cada estratégia. A Tabela 5.5 reporta também o número médio de vezes que o processo de mineração foi executado, representado por $\#Miner$.

Além disso, as Tabelas 5.3 e 5.4 apresentam, respectivamente, a diferença percentual (Dif %) das heurísticas DM-GRASP/VND e MDM-GRASP/VND em relação ao GRASP/VND. Para as médias de custo de solução e tempo, são reportados também os valores do desvio padrão. Por fim, para cada conjunto de instâncias com mesmo tamanho, calculou-se a média parcial das diferenças percentuais e, ao final da tabela, calculou-se a média geral.

Tabela 5.3: Comparação entre as heurísticas GRASP/VND e DM-GRASP/VND

Instâncias	GRASP/VND			DM-GRASP/VND							
	Melhor Solução	Média Solução	Tempo Médio	Melhor Solução	Dif % Melhor	Média Solução	Dif % Média	Desvio Padrão	Tempo Médio	Dif % Tempo	Desvio Padrão
n100q10A	12369	12514.4	4.01	**11915**	-3.67	**12375.5**	-1.11	178.29	**2.97**	-25.98	0.14
n100q10B	13668	13885.7	3.86	**13596**	-0.53	**13823.1**	-0.45	116.00	**2.77**	-28.07	0.07
n100q10C	14619	14810.8	4.01	**14310**	-2.11	**14603.0**	-1.40	139.97	**2.85**	-28.92	0.12
n100q10D	14806	14993.4	4.15	**14666**	-0.95	**14772.7**	-1.47	99.59	**3.12**	-24.76	0.08
n100q10E	12594	12819.7	3.94	**12018**	-4.57	**12587.1**	-1.81	272.30	**2.63**	-33.27	0.09
n100q10F	12082	12297.2	3.57	**11891**	-1.58	**12125.1**	-1.40	183.01	**2.67**	-25.24	0.11
n100q10G	12344	12623.4	3.84	**12176**	-1.36	**12481.5**	-1.12	176.63	**2.71**	-29.56	0.12
n100q10H	13405	13590.7	3.72	**13362**	-0.32	**13459.8**	-0.96	80.20	**2.68**	-27.93	0.08
n100q10I	**14512**	14715.9	3.74	14514	0.01	**14698.0**	-0.12	171.33	**2.60**	-30.58	0.10
n100q10J	**13700**	13992.0	4.00	13713	0.09	**13905.9**	-0.62	129.06	**2.99**	-25.28	0.09
Média Parcial					-1.50		-1.05	154.64		-27.96	0.10
n200q10A	18707	19053.1	34.34	**18319**	-2.07	**18725.7**	-1.72	234.91	**24.00**	-30.10	0.75
n200q10B	19046	19406.7	33.27	**18689**	-1.87	**19273.4**	-0.69	314.77	**21.90**	-34.18	0.50
n200q10C	17445	17740.2	37.19	**17430**	-0.09	**17630.7**	-0.62	127.86	**27.45**	-26.17	1.40
n200q10D	22428	22772.4	33.65	**22047**	-1.70	**22524.4**	-1.09	327.11	**22.69**	-32.58	1.02
n200q10E	20409	20738.2	36.77	**20323**	-0.42	**20639.7**	-0.47	204.88	**24.63**	-33.02	0.48
n200q10F	22483	22709.4	37.10	**22295**	-0.84	**22615.9**	-0.41	199.50	**27.22**	-26.63	1.11
n200q10G	18585	18855.3	34.72	**18147**	-2.36	**18735.5**	-0.64	265.92	**21.81**	-37.16	1.14
n200q10H	22165	22588.2	39.85	**21907**	-1.16	**22348.4**	-1.06	217.83	**26.65**	-33.12	1.31
n200q10I	19533	19859.3	34.22	**19362**	-0.88	**19504.1**	-1.79	75.28	**22.76**	-33.47	0.94
n200q10J	20179	20471.6	32.80	**20011**	-0.83	**20244.1**	-1.11	184.81	**23.15**	-29.42	0.55
Média Parcial					-1.22		-0.96	215.29		-31.59	0.92
n300q10A	24942	25148.1	136.01	**24392**	-2.21	**24738.4**	-1.63	159.43	**92.17**	-32.23	2.91
n300q10B	24413	24802.3	133.15	**24347**	-0.27	**24595.0**	-0.84	177.65	**89.63**	-32.68	3.61
n300q10C	23212	23418.2	142.24	**22838**	-1.61	**23170.2**	-1.06	166.65	**92.90**	-34.69	2.26
n300q10D	27080	27614.3	147.46	**26325**	-2.79	**27113.1**	-1.82	399.42	**99.18**	-32.74	2.29
n300q10E	28643	28914.2	147.16	**27980**	-2.31	**28425.1**	-1.69	255.94	**99.90**	-32.11	3.69
n300q10F	25843	26213.9	143.07	**25592**	-0.97	**25895.3**	-1.22	215.77	**108.49**	-24.17	4.38
n300q10G	25631	25814.5	144.66	**25105**	-2.05	**25413.8**	-1.55	194.70	**108.70**	-24.86	2.88
n300q10H	23590	23795.3	138.41	**23143**	-1.89	**23512.1**	-1.19	225.31	**93.02**	-32.79	2.22
n300q10I	26018	26358.4	136.85	**25444**	-2.21	**25965.2**	-1.49	235.24	**94.40**	-31.02	2.75
n300q10J	24050	24466.0	140.90	**23806**	-1.01	**24139.1**	-1.34	211.37	**98.85**	-29.84	2.75
Média Parcial					-1.73		-1.38	224.15		-30.71	2.97
n400q10A	33087	33266.8	393.04	**32170**	-2.77	**32620.1**	-1.94	190.60	**282.19**	-28.20	5.43
n400q10B	26677	26797.2	347.47	**26107**	-2.14	**26395.1**	-1.50	189.54	**246.68**	-29.01	9.14
n400q10C	30394	30682.2	399.14	**29838**	-1.83	**30235.7**	-1.46	301.90	**269.07**	-32.59	10.74
n400q10D	25814	26267.5	400.79	**25291**	-2.03	**25750.1**	-1.97	324.48	**264.62**	-33.98	7.17
n400q10E	26795	27313.9	355.53	**26393**	-1.50	**26824.5**	-1.79	251.92	**260.04**	-26.86	9.42
n400q10F	**28107**	28910.0	361.85	28188	0.29	**28539.2**	-1.28	227.36	**256.23**	-29.19	12.21
n400q10G	25697	26220.6	398.57	**25113**	-2.27	**25492.7**	-2.78	241.05	**279.50**	-29.88	11.70
n400q10H	27158	27773.1	393.40	**26813**	-1.27	**27238.1**	-1.93	230.00	**278.53**	-29.20	8.46
n400q10I	**30115**	30898.7	387.77	30208	0.31	**30549.5**	-1.13	209.11	**263.87**	-31.95	6.89
n400q10J	27655	28059.0	383.00	**26921**	-2.65	**27536.1**	-1.86	300.73	**268.10**	-30.00	12.73
Média Parcial					-1.59		-1.76	246.67		-30.08	9.39
n500q10A	29874	30661.4	825.944	**29558**	-1.06	**30246.4**	-1.35	338.46	**579.69**	-29.82	21.64
n500q10B	28559	29042.9	846.077	**28253**	-1.07	**28583.9**	-1.58	237.47	**573.82**	-32.18	20.44
n500q10C	32360	33162.5	867.237	**32065**	-0.91	**32569.1**	-1.79	287.52	**577.93**	-33.36	25.49
n500q10D	32750	33074.3	863.707	**32117**	-1.93	**32484.5**	-1.78	205.99	**593.99**	-31.23	32.04
n500q10E	32298	32667.1	881.043	**31704**	-1.84	**32263.6**	-1.24	319.04	**598.28**	-32.09	33.37
n500q10F	30856	31354.6	813.255	**30432**	-1.37	**30991.2**	-1.16	289.39	**511.36**	-37.12	14.57
n500q10G	28879	29123.4	885.222	**28357**	-1.81	**28642.5**	-1.65	227.30	**597.69**	-32.48	32.32
n500q10H	38579	39023.5	849.812	**37926**	-1.69	**38350.5**	-1.72	303.54	**596.57**	-29.80	21.47
n500q10I	32718	33217.7	858.721	**32330**	-1.19	**32624.5**	-1.79	187.99	**547.15**	-36.28	20.10
n500q10J	**32407**	33131.7	873.12	32530	0.38	**32720.7**	-1.24	170.27	**576.76**	-33.94	26.92
Média Parcial					-1.25		-1.53	256.70		-32.83	24.84
Média Geral					**-1.46**		**-1.34**			**-30.63**	

Observando a Tabela 5.3, é possível notar que, para todas as instâncias, a heurística DM-GRASP/VND apresenta melhores médias de custo de solução, em menores tempos

computacionais, quando comparados com os resultados obtidos pelo GRASP/VND. Apenas em cinco instâncias, de um total de 50, o DM-GRASP/VND não conseguiu superar o GRASP/VND em relação à melhor solução obtida. O ganho percentual geral do DM-GRASP/VND foi, em média, igual a 1,34%, sendo, em média, 30,63% mais rápido em relação ao GRASP/VND original.

Tabela 5.4: Comparação entre as heurísticas GRASP/VND e MDM-GRASP/VND

Instâncias	GRASP/VND			MDM-GRASP/VND							
	Melhor Solução	Média Solução	Tempo Médio	Melhor Solução	Dif % Melhor	Média Solução	Dif % Média	Desvio Padrão	Tempo Médio	Dif % Tempo	Desvio Padrão
n100q10A	12369	12514.4	4.01	**12238**	-1.06	**12368.8**	-1.16	81.47	**2.36**	-41.14	0.23
n100q10B	13668	13885.7	3.86	**13400**	-1.96	**13691.8**	-1.40	188.79	**2.24**	-42.00	0.25
n100q10C	14619	14810.8	4.01	**14223**	-2.71	**14537.5**	-1.85	128.05	**2.10**	-47.75	0.24
n100q10D	14806	14993.4	4.15	**14608**	-1.34	**14801.6**	-1.28	139.76	**2.54**	-38.66	0.26
n100q10E	12594	12819.7	3.94	**12150**	-3.53	**12651.8**	-1.31	253.19	**2.16**	-45.11	0.34
n100q10F	12082	12297.2	3.57	**11895**	-1.55	**12128.8**	-1.37	202.56	**2.20**	-38.23	0.29
n100q10G	12344	12623.4	3.84	**12084**	-2.11	**12376.9**	-1.95	229.11	**2.12**	-44.85	0.25
n100q10H	13405	13590.7	3.72	**12995**	-3.06	**13360.2**	-1.70	205.11	**2.05**	-44.86	0.16
n100q10I	14512	14715.9	3.74	**14272**	-1.65	**14606.7**	-0.74	219.69	**1.97**	-47.34	0.26
n100q10J	13700	13992.0	4.00	**13285**	-3.03	**13804.3**	-1.34	191.80	**2.40**	-39.88	0.24
Média Parcial					-2.20		-1.41	183.95		-42.98	0.25
n200q10A	18707	19053.1	34.34	**18378**	-1.76	**18733.0**	-1.68	235.07	**21.36**	-37.79	2.99
n200q10B	19046	19406.7	33.27	**18605**	-2.32	**19249.6**	-0.81	301.50	**17.41**	-47.68	2.96
n200q10C	17445	17740.2	37.19	**17106**	-1.94	**17489.6**	-1.41	220.24	**23.56**	-36.64	3.52
n200q10D	22428	22772.4	33.65	**22130**	-1.33	**22446.7**	-1.43	226.81	**16.77**	-50.17	1.19
n200q10E	20409	20738.2	36.77	**20052**	-1.75	**20373.6**	-1.76	210.57	**18.11**	-50.76	2.46
n200q10F	22483	22709.4	37.10	**22190**	-1.30	**22520.7**	-0.83	179.55	**23.38**	-36.98	2.79
n200q10G	18585	18855.3	34.72	**18341**	-1.31	**18748.1**	-0.57	269.67	**17.21**	-50.43	3.76
n200q10H	22165	22588.2	39.85	**22097**	-0.31	**22399.3**	-0.84	176.42	**21.93**	-44.98	2.13
n200q10I	19533	19859.3	34.22	**18834**	-3.58	**19259.1**	-3.02	260.57	**18.20**	-46.80	2.68
n200q10J	20179	20471.6	32.80	**19922**	-1.27	**20157.0**	-1.54	138.10	**18.28**	-44.26	2.37
Média Parcial					-1.69		-1.39	221.85		-44.65	2.69
n300q10A	24942	25148.1	136.01	**24080**	-3.46	**24469.5**	-2.70	271.87	**71.83**	-47.19	9.02
n300q10B	24413	24802.3	133.15	**23894**	-2.13	**24330.4**	-1.90	317.17	**70.44**	-47.10	11.12
n300q10C	23212	23418.2	142.24	**22687**	-2.26	**22978.1**	-1.88	197.62	**74.89**	-47.35	13.76
n300q10D	27080	27614.3	147.46	**26493**	-2.17	**26885.9**	-2.64	211.29	**79.43**	-46.14	7.60
n300q10E	28643	28914.2	147.16	**27947**	-2.43	**28249.5**	-2.30	163.96	**79.45**	-46.01	9.05
n300q10F	25843	26213.9	143.07	**25012**	-3.22	**25631.9**	-2.22	340.97	**80.73**	-43.57	7.59
n300q10G	25631	25814.5	144.66	**24719**	-3.56	**25085.8**	-2.82	169.33	**86.35**	-40.31	11.87
n300q10H	23590	23795.3	138.41	**22982**	-2.58	**23367.8**	-1.80	299.96	**73.32**	-47.02	7.06
n300q10I	26018	26358.4	136.85	**25371**	-2.49	**25853.0**	-1.92	349.18	**70.41**	-48.55	12.30
n300q10J	24050	24466.0	140.90	**23664**	-1.60	**24078.0**	-1.59	352.84	**67.07**	-52.40	10.51
Média Parcial					-2.59		-2.18	267.42		-46.56	9.99
n400q10A	33087	33266.8	393.04	**32062**	-3.10	**32441.0**	-2.48	262.24	**214.12**	-45.52	25.84
n400q10B	26677	26797.2	347.47	**25436**	-4.65	**26041.3**	-2.82	332.32	**210.20**	-39.50	23.41
n400q10C	30394	30682.2	399.14	**29693**	-2.31	**30047.1**	-2.07	281.30	**205.43**	-48.53	46.21
n400q10D	25814	26267.5	400.79	**24999**	-3.16	**25690.9**	-2.20	333.91	**190.88**	-52.38	20.81
n400q10E	26795	27313.9	355.53	**25946**	-3.17	**26575.4**	-2.70	274.21	**209.04**	-41.20	27.39
n400q10F	28107	28910.0	361.85	**27887**	-0.78	**28280.0**	-2.18	226.86	**200.40**	-44.62	32.03
n400q10G	25697	26220.6	398.57	**25045**	-2.54	**25434.2**	-3.00	279.82	**224.96**	-43.56	24.98
n400q10H	27158	27773.1	393.40	**26443**	-2.63	**26882.7**	-3.21	335.61	**201.77**	-48.71	18.98
n400q10I	30115	30898.7	387.77	**29844**	-0.90	**30450.5**	-1.45	292.63	**191.96**	-50.50	38.13
n400q10J	27655	28059.0	383.00	**26756**	-3.25	**27367.8**	-2.46	255.57	**210.04**	-45.16	28.99
Média Parcial					-2.65		-2.46	287.45		-45.97	28.68
n500q10A	29874	30661.4	825.944	**29583**	-0.97	**29941.4**	-2.35	177.68	**460.20**	-44.28	65.31
n500q10B	28559	29042.9	846.077	**27511**	-3.67	**28013.1**	-3.55	240.95	**432.43**	-48.89	51.54
n500q10C	32360	33162.5	867.237	**32019**	-1.05	**32325.7**	-2.52	149.93	**432.81**	-50.09	58.90
n500q10D	32750	33074.3	863.707	**31516**	-3.77	**32119.7**	-2.89	322.58	**443.95**	-48.60	47.27
n500q10E	32298	32667.1	881.043	**31452**	-2.62	**31934.8**	-2.24	375.42	**493.50**	-43.99	62.01
n500q10F	30856	31354.6	813.255	**30254**	-1.95	**30917.4**	-1.39	408.35	**318.96**	-60.78	82.66
n500q10G	28879	29123.4	885.222	**27372**	-5.22	**28231.7**	-3.06	367.64	**392.66**	-55.64	50.90
n500q10H	38579	39023.5	849.812	**37427**	-2.99	**38011.6**	-2.59	269.58	**404.26**	-52.43	39.90
n500q10I	32718	33217.7	858.721	**32035**	-2.09	**32212.9**	-3.02	138.76	**374.09**	-56.44	76.40
n500q10J	32407	33131.7	873.12	**31844**	-1.74	**32421.1**	-2.14	411.42	**415.56**	-52.41	59.83
Média Parcial					-2.61		-2.58	286.23		-51.35	59.47
Média Geral					**-2.35**		**-2.00**			**-46.30**	

A Tabela 5.4 indica que a heurística MDM-GRASP/VND, que aplica a mineração mais de uma vez, obteve resultados ainda melhores do que o DM-GRASP/VND. É possível perceber que, para todas as instâncias, a versão MDM é melhor em todos os três critérios

(melhor solução, média de solução e tempo médio). O ganho percentual geral do MDM-GRASP/VND em relação ao GRASP/VND foi, em média, igual a 2,35% para melhor solução e, para média de custo de solução, igual a 2,00%, superando o ganho já apresentado pelo DM-GRASP/VND. Além disso, o MDM-GRASP/VND ainda conseguiu ser a estratégia mais eficiente em termos de tempo, sendo 46,30% mais rápido em relação ao GRASP/VND original.

A Tabela 5.5 confronta os resultados das três heurísticas. Os valores em negrito representam os melhores resultados obtidos. Observa-se que o desempenho do MDM-GRASP/VND foi superior em relação aos demais, obtendo 44 melhores médias de solução dentre 50 possíveis, enquanto a estratégia DM-GRASP/VND obteve as seis melhores médias de solução restantes.

Tabela 5.5: Comparação entre as heurísticas GRASP/VND, DM-GRASP/VND e MDM-GRASP/VND

Instâncias	GRASP/VND			DM-GRASP/VND			MDM-GRASP/VND			#Miner
	Melhor Solução	Média Solução	Tempo Médio	Melhor Solução	Média Solução	Tempo Médio	Melhor Solução	Média Solução	Tempo Médio	
n100q10A	12369	12514.4	4.01	**11915**	12375.5	2.97	12238	**12368.8**	**2.36**	2.6
n100q10B	13668	13885.7	3.86	13596	13823.1	2.77	**13400**	**13691.8**	**2.24**	2.1
n100q10C	14619	14810.8	4.01	14310	14603.0	2.85	**14223**	**14537.5**	**2.10**	2.3
n100q10D	14806	14993.4	4.15	14666	**14772.7**	3.12	**14608**	14801.6	**2.54**	2.2
n100q10E	12594	12819.7	3.94	**12018**	**12587.1**	2.63	12150	12651.8	**2.16**	2.1
n100q10F	12082	12297.2	3.57	**11891**	**12125.1**	2.67	11895	12128.8	**2.20**	2
n100q10G	12344	12623.4	3.84	12176	12481.5	2.71	**12084**	**12376.9**	**2.12**	2.3
n100q10H	13405	13590.7	3.72	13362	13459.8	2.68	**12995**	**13360.2**	**2.05**	2.3
n100q10I	14512	14715.9	3.74	14514	14698.0	2.60	**14272**	**14606.7**	**1.97**	2.3
n100q10J	13700	13992.0	4.00	13713	13905.9	2.99	**13285**	**13804.3**	**2.40**	2.7
n200q10A	18707	19053.1	34.34	**18319**	**18725.7**	24.00	18378	18733	**21.36**	2.3
n200q10B	19046	19406.7	33.27	18689	19273.4	21.90	**18605**	**19249.6**	**17.41**	3.2
n200q10C	17445	17740.2	37.19	17430	17630.7	27.45	**17106**	**17489.6**	**23.56**	2.5
n200q10D	22428	22772.4	33.65	**22047**	22524.4	22.69	22130	**22446.7**	**16.77**	2.5
n200q10E	20409	20738.2	36.77	20323	20639.7	24.63	**20052**	**20373.6**	**18.11**	2.2
n200q10F	22483	22709.4	37.10	22295	22615.9	27.22	**22190**	**22520.7**	**23.38**	2.5
n200q10G	18585	18855.3	34.72	**18147**	**18735.5**	21.81	18341	18748.1	**17.21**	3.4
n200q10H	22165	22588.2	39.85	**21907**	**22348.4**	26.65	22097	22399.3	**21.93**	2.7
n200q10I	19533	19859.3	34.22	19362	19504.1	22.76	**18834**	**19259.1**	**18.20**	2.6
n200q10J	20179	20471.6	32.80	20011	20244.1	23.15	**19922**	**20157**	**18.28**	2.4
n300q10A	24942	25148.1	136.01	24392	24738.4	92.17	**24080**	**24469.5**	**71.83**	3
n300q10B	24413	24802.3	133.15	24347	24595.0	89.63	**23894**	**24330.4**	**70.44**	3.1
n300q10C	23212	23418.2	142.24	22838	23170.2	92.90	**22687**	**22978.1**	**74.89**	3.1
n300q10D	27080	27614.3	147.46	**26325**	27113.1	99.18	26493	**26885.9**	**79.43**	2.9
n300q10E	28643	28914.2	147.16	27980	28425.1	99.90	**27947**	**28249.5**	**79.45**	2.7
n300q10F	25843	26213.9	143.07	25592	25895.3	108.49	**25012**	**25631.9**	**80.73**	2.9
n300q10G	25631	25814.5	144.66	25105	25413.8	108.70	**24719**	**25085.8**	**86.35**	1.9
n300q10H	23590	23795.3	138.41	23143	23512.1	93.02	**22982**	**23367.8**	**73.32**	2.9
n300q10I	26018	26358.4	136.85	25444	25965.2	94.40	**25371**	**25853**	**70.41**	2.4
n300q10J	24050	24466.0	140.90	23806	24139.1	98.85	**23664**	**24078**	**67.07**	3.3
n400q10A	33087	33266.8	393.04	32170	32620.1	282.19	**32062**	**32441**	**214.12**	2.5
n400q10B	26677	26797.2	347.47	26107	26395.1	246.68	**25436**	**26041.3**	**210.20**	2.3
n400q10C	30394	30682.2	399.14	29838	30235.7	269.07	**29693**	**30047.1**	**205.43**	3.4
n400q10D	25814	26267.5	400.79	25291	25750.1	264.62	**24999**	**25690.9**	**190.88**	3.6
n400q10E	26795	27313.9	355.53	26393	26824.5	260.04	**25946**	**26575.4**	**209.04**	2.7
n400q10F	28107	28910.0	361.85	28188	28539.2	256.23	**27887**	**28280**	**200.40**	2.5
n400q10G	25697	26220.6	398.57	25113	25492.7	279.50	**25045**	**25434.2**	**224.96**	2.5
n400q10H	27158	27773.1	393.40	26813	27238.1	278.53	**26443**	**26882.7**	**201.77**	2.5
n400q10I	30115	30898.7	387.77	30208	30549.5	263.87	**29844**	**30450.5**	**191.96**	3.5
n400q10J	27655	28059.0	383.00	26921	27536.1	268.10	**26756**	**27367.8**	**210.04**	3.2
n500q10A	29874	30661.4	825.944	**29558**	30246.4	579.69	29583	**29941.4**	**460.20**	2.7
n500q10B	28559	29042.9	846.077	28253	28583.9	573.82	**27511**	**28013.1**	**432.43**	3
n500q10C	32360	33162.5	867.237	32065	32569.1	577.93	**32019**	**32325.7**	**432.81**	2.4
n500q10D	32750	33074.3	863.707	32117	32484.5	593.99	**31516**	**32119.7**	**443.95**	2.8
n500q10E	32298	32667.1	881.043	31704	32263.6	598.28	**31452**	**31934.8**	**493.50**	2.8
n500q10F	30856	31354.6	813.255	30432	30991.2	511.36	**30254**	**30917.4**	**318.96**	3.9
n500q10G	28879	29123.4	885.222	28357	28642.5	597.69	**27372**	**28231.7**	**392.66**	2.6
n500q10H	38579	39023.5	849.812	37926	38350.5	596.57	**37427**	**38011.6**	**404.26**	4.1
n500q10I	32718	33217.7	858.721	32330	32624.5	547.15	**32035**	**32212.9**	**374.09**	3.1
n500q10J	32407	33131.7	873.12	32530	32720.7	576.76	**31844**	**32421.1**	**415.56**	3
#Melhores	0	0	0	9	6	0	41	44	50	

Para melhor solução, o MDM-GRASP/VND obteve 41 melhores soluções dentre 50 possíveis, enquanto o DM-GRASP/VND obteve as nove melhores soluções restantes. Entretanto, vale ressaltar que o tempo computacional despendido pelo MDM-GRASP/VND foi menor em relação aos demais, sendo, em média, 22,67% mais rápido que o DM-GRASP/VND.

A redução do tempo está diretamente relacionada a dois fatores e poderá ser verificada na Seção 5.5. Primeiro, a construção com padrões é mais rápida, pois usa parte de soluções do conjunto elite como base e, segundo, a qualidade da solução construída com padrões tende a ser melhor que a solução obtida na construção original e, por isso, garante que a busca local convirja mais rapidamente a um ótimo local.

A seguir, será apresentado um resumo de informações acerca dos padrões e componentes conexas (CC) extraídos a partir de uma execução do DM-GRASP/VND. Na Tabela 5.6, destacam-se: o número médio de CCs dos $numP$ padrões minerados, a maior (e a menor) CC extraída de todos os padrões e o tamanho médio dos padrões e das CCs. Com exceção do número médio de CCs, todas essas informações estão expressas em quantidade de arcos.

É possível perceber que o número médio de CCs e, consequentemente, o tamanho médio dos padrões, aumenta conforme o tamanho da instância. Já o tamanho da maior (e da menor) CC não varia consideravelmente. O tamanho da menor CC é sempre unitário, o que representa apenas um arco como CC. Observa-se ainda que o tamanho médio das CC é baixo, o que indica o alto número de CCs com tamanho pequeno.

Tabela 5.6: Resumo de informações sobre os padrões e componentes conexas, expressas em quantidades de arcos

Instância	# médio CCs	Maior CC	Menor CC	Tam médio padrões	Tam médio CC
n100q10A	15.37	12.7	1	29.61	1.99
n100q10B	14.11	14.1	1	28.91	2.14
n100q10C	15.14	10.3	1	29.6	2.01
n200q10A	28.68	10.5	1	49.15	1.73
n200q10B	28.78	10.6	1	48.71	1.71
n200q10C	29.17	12.5	1	52.31	1.81
n300q10A	40.43	11.6	1	65.13	1.63
n300q10B	40.01	11.1	1	65.38	1.65
n300q10C	41.67	11.9	1	70.41	1.70
n400q10B	54.29	13.1	1	88.08	1.63
n400q10E	54.93	12.7	1	85.43	1.57
n400q10H	54.29	12.2	1	85.74	1.59
n500q10A	67.59	9.5	1	107.14	1.59
n500q10D	64.67	10.4	1	98.11	1.52
n500q10E	65.64	13.8	1	101.02	1.54
Média	40.98	11.80	1	66.98	1.72

Nas seções a seguir, são apresentados alguns experimentos adicionais a fim de ilustrar e comparar o comportamento dos algoritmos analisados neste trabalho.

5.4 Significância Estatística

Nesta seção, será feita uma verificação se os ganhos referentes à média de solução reportados nas Tabelas 5.3 e 5.4 têm significância estatística, realizando o teste estatístico não paramétrico de *Friedman* [73]. Esse teste é geralmente aplicado para avaliar dois algoritmos que possuem componentes aleatórias, e identificar se a diferença entre as médias de resultados obtidos por esses algoritmos foi, de fato, devido à superioridade de algum deles ou simplesmente devido à aleatoriedade dos métodos. Para realizar o teste, foram considerados p-valor igual a 0.05 e duas hipóteses:

- Hipótese nula (H_0): Não há diferença entre as médias encontradas pelos algoritmos comparados; e
- Hipótese alternativa (H_1): Há diferença entre as médias encontradas pelos algoritmos comparados;

Dessa forma, H_0 poderá ser rejeitada com 95% de certeza se, para cada instância, o valor retornado pelo teste de *Friedman* para a comparação entre os algoritmos for menor ou igual ao p-valor definido. Caso H_0 seja rejeitada, a hipótese alternativa H_1 é considerada.

A Tabela 5.7 mostra uma comparação, dois a dois, entre os algoritmos GRASP/VND, DM-GRASP/VND e MDM-GRASP/VND, separados por grupos de instâncias com mesmo número de clientes. O número fora dos parênteses indica em quantas instâncias aquela estratégia foi melhor que a outra em relação à média de solução, enquanto o valor entre parênteses indica o número de vezes em que o p-valor foi menor que $0,05$, indicando que a probabilidade de a diferença de desempenho dos algoritmos ser devido à aleatoriedade é menor que 5%.

Tabela 5.7: Análise de significância estatística

Par de Algoritmos	Tamanhos das instâncias				
	n100	n200	n300	n400	n500
GRASP/VND	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)
DM-GRASP/VND	10(6)	10(4)	10(7)	10(9)	10(8)
GRASP/VND	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)
MDM-GRASP/VND	10(4)	10(7)	10(10)	10(10)	10(9)
DM-GRASP/VND	3 (0)	3 (0)	0 (0)	0 (0)	0 (0)
MDM-GRASP/VND	7 (0)	7 (1)	10(2)	10(2)	10(3)

Na comparação entre os algoritmos GRASP/VND e DM-GRASP/VND, nota-se que grande parte das diferenças de desempenho têm significância estatística, destacando-

se o grupo com $n = 400$ clientes, onde foi possível obter significância estatística em nove das dez instâncias. No geral, as diferenças de desempenho têm significância estatística em 34 de 50 instâncias. Quando comparados os algoritmos GRASP/VND e MDM-GRASP/VND, as diferenças de desempenho possuem significância estatística em 40 instâncias de um total de 50, destacando-se os grupos de instância com $n = 300$ e $n = 400$, nos quais houve significância estatística em todas as instâncias.

As últimas duas linhas dessa tabela mostram a comparação entre o DM-GRASP/VND e o MDM-GRASP/VND. Nessa comparação é possível observar que a versão MDM, na maioria das instâncias, obtém os melhores resultados (representados pelo número fora dos parênteses). No entanto, a diferença de desempenho entre as versões tem significância estatística somente em oito de um total de 50 instâncias.

No geral, esses resultados indicam que as estratégias com mineração de dados, quando comparadas entre si, apresentam desempenho semelhante, ou seja, nem sempre é possível rejeitar a hipótese nula. Todavia, quando são comparadas ambas as versões com MD com o GRASP/VND, nota-se a superioridade delas sobre o GRASP/VND, sendo possível, na maioria das vezes, considerar a hipótese alternativa.

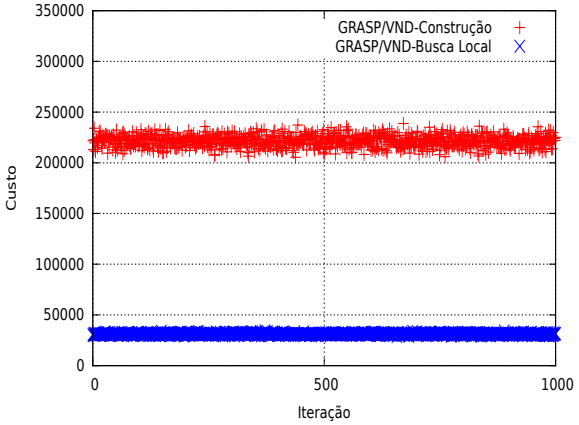
5.5 Análise do Comportamento das Estratégias

Nesta seção será analisado, a partir de execuções utilizando-se a instância n500q10G, o comportamento das estratégias, a fim de ilustrar o que acontece após a inserção da mineração de dados na heurística original.

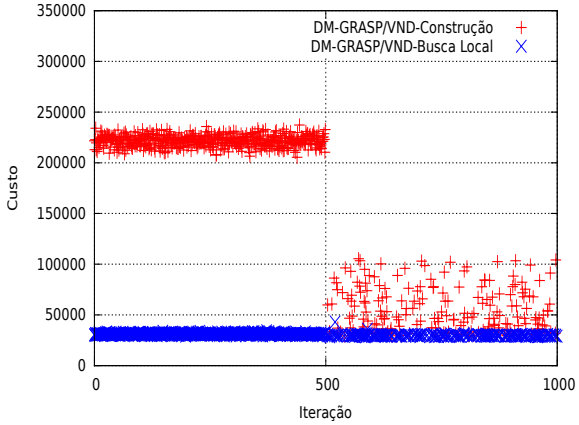
As Figuras 5.1(a), 5.1(b) e 5.1(c) mostram como se comportam as fases de construção e busca local nos três algoritmos, reportando, por iteração, os valores de solução obtidos em cada uma das fases. Esse teste foi realizado para a instância n500q10G, executando 1000 iterações de cada estratégia, com uma mesma semente aleatória.

Nota-se que os algoritmos GRASP/VND e DM-GRASP/VND tem exatamente o mesmo comportamento até a iteração 500. No entanto, na iteração 500, a Figura 5.1(b) mostra o momento exato em que ocorre a mineração, revelando uma grande redução no custo das soluções construídas a partir de então.

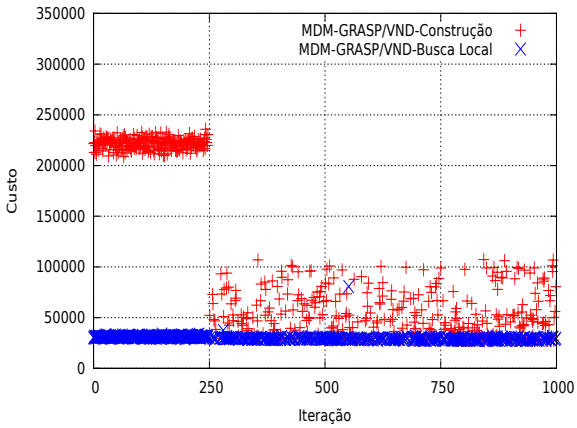
Na Figura 5.1(c), o comportamento é o mesmo somente até a iteração 250. Nesse ponto, ocorre a primeira mineração da versão MDM-GRASP/VND e, a partir daí, é possível reparar que o custo das soluções construídas também reduz consideravelmente.



(a) GRASP/VND



(b) DM-GRASP/VND



(c) MDM-GRASP/VND

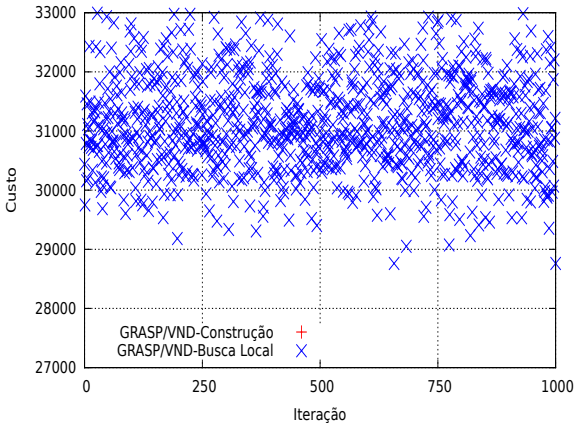
Figura 5.1: Mapa de soluções para a instância n500q10G

Após a segunda mineração, na iteração 600, a redução do custo é ainda maior.

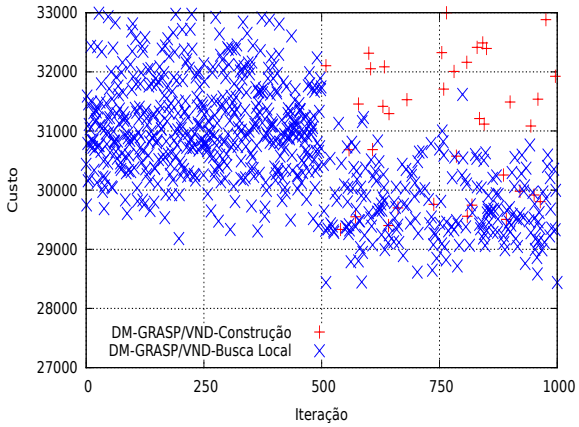
Em ambos os algoritmos com MD, após os pontos de mineração, o custo das soluções após a busca local também reduz e será melhor visualizado na Figura 5.2.

Para que seja percebida a tendência de melhoria da busca local após as minerações no DM-GRASP/VND e no MDM-GRASP/VND, os gráficos da Figura 5.1 foram ampliados, originando a Figura 5.2. Nessa última, o intervalo do eixo de custo foi ampliado entre os valores de 27000 e 33000.

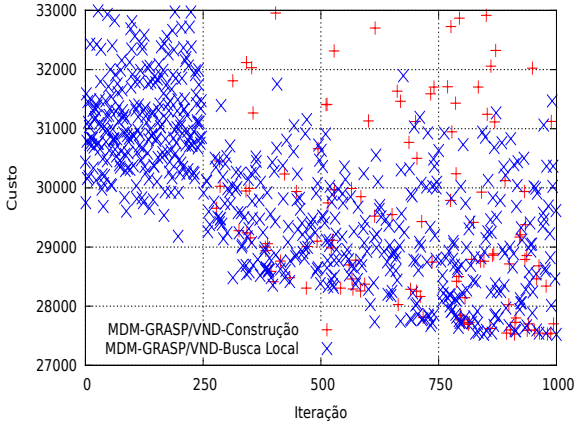
Na Figura 5.2, é possível perceber que o algoritmo GRASP/VND encontrou apenas uma solução inferior a 29000 durante todas as 1000 iterações. Para o DM-GRASP/VND, após a iteração 500, esse valor foi alcançado em diversas vezes. Já o MDM-GRASP/VND, na Figura 5.2(c), as soluções com custo inferior a 29000 foram obtidas logo após a primeira mineração, na iteração 250, bem antes das demais abordagens. Além disso, após a segunda mineração, na iteração 600, foi possível obter soluções ainda melhores, com custo menor



(a) GRASP/VND



(b) DM-GRASP/VND



(c) MDM-GRASP/VND

Figura 5.2: Mapa de soluções ampliado para a instância n500q10G

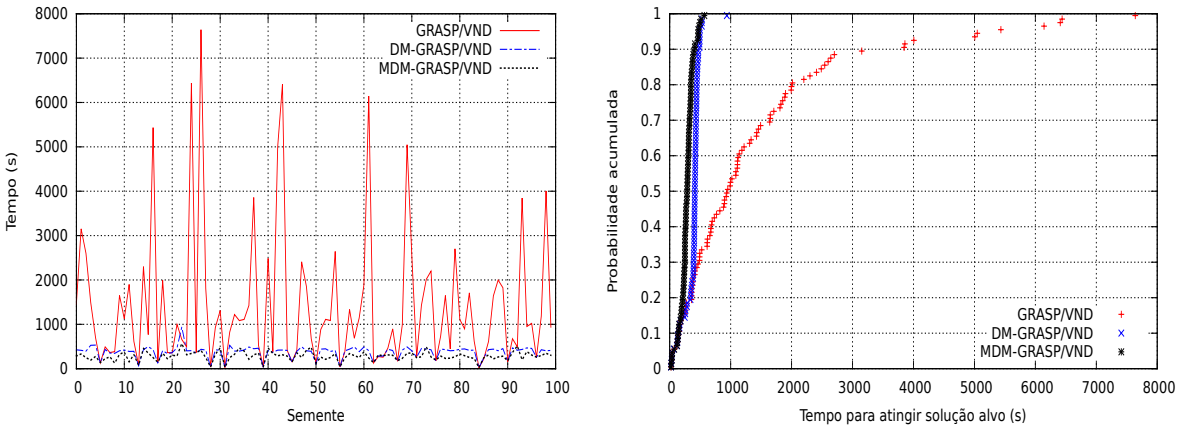
que 28000, fato que não aconteceu nas duas estratégias anteriores.

Outro fator importante é que, nas duas versões com mineração de dados, representados nas Figuras 5.2(b) e 5.2(c), existem soluções construídas após as minerações que possuem custos de solução tão bons quanto as soluções já exploradas pela busca local. Para o MDM-GRASP/VND, algumas soluções construídas após a segunda mineração são ainda melhores do que as construídas após a primeira.

Em outra análise, ambos os algoritmos foram executados com 100 sementes aleatórias e foram coletados os tempos necessários para encontrar uma solução alvo. A instância escolhida foi a n500q10G e o valor do custo adotado como alvo foi 29123.

A Figura 5.3(a) mostra o tempo de execução de cada algoritmo para atingir a solução alvo em cada semente escolhida. Pode-se observar que os algoritmos DM-GRASP/VND e MDM-GRASP/VND se mostram bem mais eficientes, alcançando a solução alvo sempre antes que o GRASP/VND. Entre as versões com MD, destaca-se o MDM, que obteve

tempos computacionais quase sempre inferiores aos da versão DM-GRASP/VND.



(a) Análise de convergência

(b) Gráfico TTT-Plot

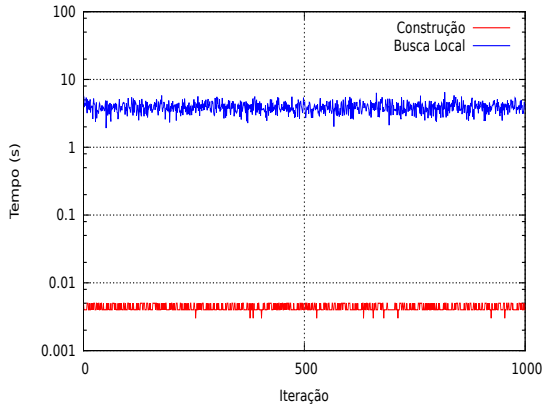
Figura 5.3: Análise com solução alvo para a instância n500q10G

A Figura 5.3(b) apresenta outra comparação entre os três algoritmos abordados, baseada nos gráficos *Time-to-Target Plots* (TTT-plots) [3], que são usados para analisar o comportamento de algoritmos com componentes aleatórias. Esses gráficos mostram a probabilidade acumulada, no eixo das ordenadas, de um algoritmo encontrar uma solução melhor ou igual a uma solução alvo prefixada, em um tempo de execução definido no eixo das abscissas.

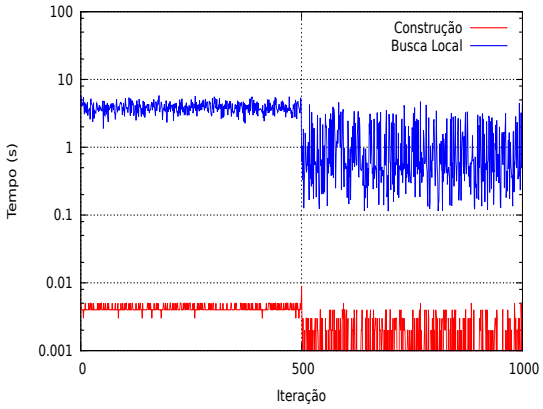
É possível notar que o comportamento das versões híbridas com mineração de dados superam o do GRASP/VND. Por exemplo, a probabilidade de ambas as versões encontrarem a solução alvo em 500 segundos é de quase 100% enquanto que para o GRASP/VND essa probabilidade está em torno de 35%. Observando as versões com MD, percebe-se que na versão MDM a probabilidade de encontrar a solução alvo aumenta mais intensamente.

A fim de justificar a redução do tempo computacional obtido pelos algoritmos DM-GRASP/VND e MDM-GRASP/VND, foi feita a comparação apresentada na Figura 5.4. Essa análise exibe o tempo computacional despendido por cada uma das estratégias, GRASP/VND, DM-GRASP/VND e MDM-GRASP/VND, respectivamente nas Figuras 5.4(a), 5.4(b) e 5.4(c), em cada uma das etapas de construção e busca local.

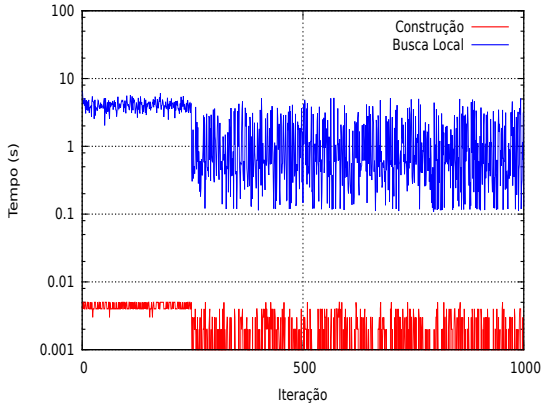
Verifica-se que o DM-GRASP/VND obteve uma redução significativa de tempo após a iteração 500 (onde ocorre a mineração de dados), tanto na construção, quanto na busca local, ilustrando que a construção adaptada é mais rápida e também que a busca local converge mais rapidamente. Para o MDM-GRASP/VND, as reduções ocorreram bem antes, na iteração 250.



(a) GRASP/VND



(b) DM-GRASP/VND



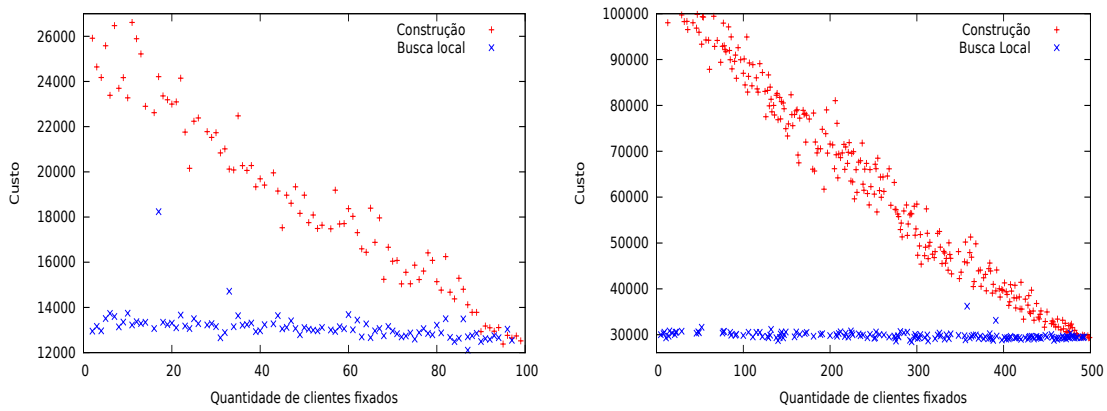
(c) MDM-GRASP/VND

Figura 5.4: Análise de tempo para a instância n500q10G

Na Figura 5.5, verifica-se como a escolha de componentes conexas dos padrões influencia no desempenho das estratégias com MD para as instâncias n100q10A e n500q10G. A cada iteração do algoritmo DM-GRASP/VND, são reportados o número de clientes fixados durante a construção (devido à CC escolhida na iteração corrente) e o custo de solução após as etapas de construção e busca local.

Nessa figura, observa-se que o custo da solução inicial construída diminui à medida que aumenta o número de clientes nela fixados. Esse comportamento, que se mantém após a aplicação da busca local, está relacionado com a escolha das CCs e com a qualidade das soluções presentes no CE. Como o CE, em geral, armazena boas soluções, quanto mais clientes são fixados na construção da solução, mais próxima ela estará da solução que é usada como guia (que faz parte do CE) e, pelo o que ficou evidenciado, menor tende a ser o custo.

Finalmente, para examinar o comportamento das estratégias com o aumento do número de iterações, variou-se o valor de $maxIter$ no conjunto $\{100, 200, 400, 600, 800,$-



(a) Instância n100q10A (b) Instância n500q10G

Figura 5.5: Análise do custo de solução em função da fixação de componentes conexas

$1000, 1200, 1600, 2000\}$. O resultado dessa análise pode ser observado nas Figuras 5.6 e 5.7.

Na Figura 5.6, cada ponto no gráfico representa a diferença percentual que a estratégia em questão obteve sobre o GRASP/VND, para o respectivo critério (indicado na legenda). Nessa figura, é possível observar que, para as médias de custo de solução, a diferença percentual tanto do DM quanto do MDM-GRASP/VND estão em constante melhoria em relação ao GRASP/VND, com destaque maior para a versão MDM, cuja diferença percentual aumenta mais rapidamente.

No intervalo de valores entre 1200 e 2000 iterações, o aumento da diferença percentual foi menor, se comparado com o intervalo entre 100 e 1200 iterações. Observa-se também que a versão MDM-GRASP/VND, mesmo aumentando consideravelmente o número de iterações, ainda tende a melhorar os resultados, enquanto que para o DM-GRASP/VND essa melhoria tende a se tornar estável.

Analisando a diferença percentual para as melhores soluções, também foi possível perceber que, no geral, a diferença percentual de ambas estratégias DM e MDM aumentou em relação ao GRASP/VND, apesar de não ter sido sempre constante. Novamente, o aumento da diferença percentual da estratégia MDM-GRASP/VND em relação à heurística original foi superior, quando comparado com o aumento ocorrido para o DM-GRASP/VND.

Em relação ao tempo computacional, a Figura 5.7 mostra que a diferença percentual das estratégias DM e MDM-GRASP/VND em relação ao GRASP/VND sofreram modificações ao longo da mudança do número de iterações. Para a versão DM, essa diferença foi sempre superior a 30%, enquanto que para o MDM, essa diferença não foi menor que

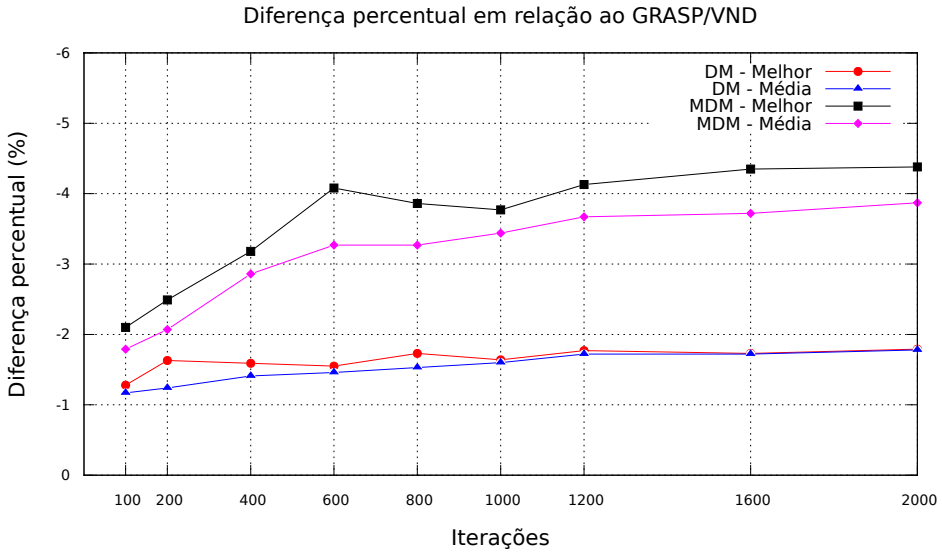


Figura 5.6: Variação do número de iterações - Melhores soluções e média de solução

40%.

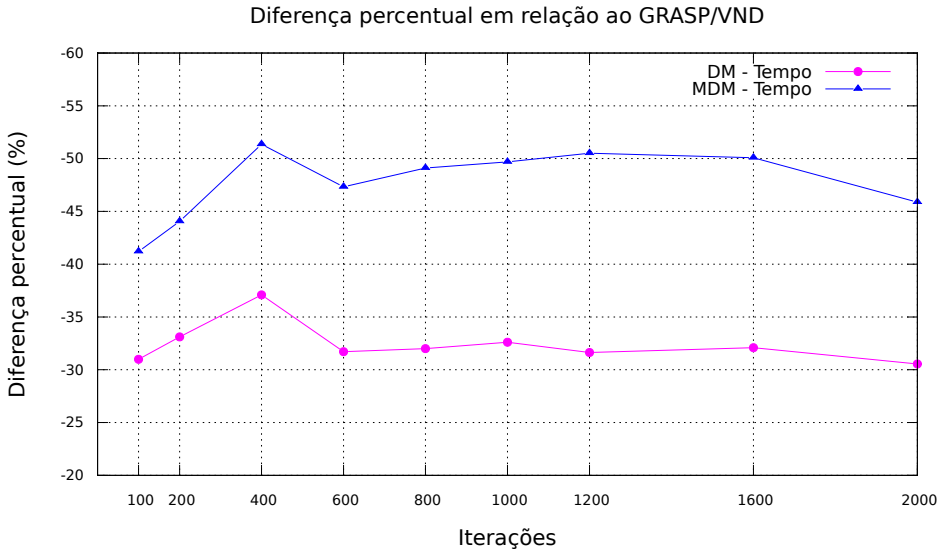


Figura 5.7: Variação do número de iterações - Tempo computacional

Neste capítulo, foram apresentados os resultados experimentais obtidos pelas três estratégias híbridas, com o objetivo de avaliar o impacto da inserção de mineração de dados na heurística GRASP/VND na qualidade das soluções obtidas, e também no tempo computacional despendido. Dessa forma, as conclusões finais obtidas a partir das análises descritas anteriormente são apresentadas no Capítulo 6, assim como sugestões de trabalhos futuros.

Capítulo 6

Conclusão

A hibridização de metaheurísticas baseadas em GRASP com técnicas de mineração de dados caracteriza-se por extrair padrões de um conjunto elite de soluções construído ao longo de iterações iniciais. A ideia principal é utilizar, na fase de construção, os padrões minerados para construir soluções de melhor qualidade, guiando assim, a busca no espaço de soluções.

Neste trabalho, foram apresentadas duas heurísticas híbridas que incorporam a técnica de mineração de conjuntos frequentes a uma heurística GRASP/VND previamente proposta e de desempenho competitivo para o 1-PDTSP. A primeira, denominada DM-GRASP/VND, aplica o processo de mineração somente uma vez, exatamente na metade das iterações. A segunda, denominada MDM-GRASP/VND, aplica a mineração sempre que o conjunto elite de soluções se estabiliza após sofrer alterações. O objetivo foi apresentar uma estratégia híbrida para um problema cuja solução é representada pela ordem dos elementos, característica que diferencia essa abordagem dos trabalhos em que essa combinação já fora aplicada.

Foram realizados experimentos computacionais em 400 instâncias da literatura, com o número de clientes variando entre 20 e 500, e os resultados indicaram o benefício da introdução de mineração de dados, conseguindo soluções melhores em um menor tempo de execução. Considerando somente as instâncias mais difíceis, o DM-GRASP/VND obteve um ganho médio em relação às melhores soluções de 1.46% e um ganho médio em relação às médias de solução de 1.34%, com uma redução média de 30.63% em termos de tempo de execução, quando comparado com os resultados obtidos pela versão original GRASP/VND. Já o MDM-GRASP/VND, quando comparado com o GRASP/VND, apresentou um ganho médio em relação às melhores soluções de 2.35%, um ganho médio de 2.00% em relação à média de qualidade de solução, reduzindo também, em média, os

tempos computacionais em 46.30% sobre versão a original. Esses resultados demonstram o bom desempenho das hibridizações propostas neste trabalho. Parte desse estudo foi aceito para publicação em [33].

Além disso, experimentos complementares analisaram o comportamento das três estratégias. Tais experimentos comprovaram a superioridade das versões híbridas com mineração de dados, no que diz respeito à convergência do método, com destaque maior para a versão MDM-GRASP/VND. Vale ressaltar ainda que a maioria dos resultados reportados têm significância estatística, quando comparadas ambas as versões DM-GRASP/VND e MDM-GRASP/VND sobre a versão original GRASP/VND.

Como trabalho futuro, pretende-se avaliar a utilização de padrões sequenciais no 1-PDTSP, como forma de representar padrões que envolvem ordem. Para tanto, os algoritmos de mineração de padrões sequenciais poderiam ser adaptados para extrair apenas sequências consecutivas de elementos, ou ainda apresentando uma construção adaptada que consiga tirar benefícios da sequência de elementos não consecutivos.

A mineração de conjuntos frequentes (MCF), como já mencionado anteriormente no Capítulo 2, foi originalmente proposta para encontrar conjuntos de elementos frequentes em transações (conjunto de itens) de uma base de dados. Por ter essa característica, a combinação da MCF com metaheurísticas foi avaliada inicialmente em problemas de otimização combinatória cuja representação da solução não considera a sequência dos elementos.

Na área de mineração de dados, quando os elementos das transações são organizados de forma que a ordem seja importante, a tarefa indicada é a mineração de padrões sequenciais, que tem como objetivo identificar sequências de eventos que ocorrem com frequência no banco de dados. Para o problema 1-PDTSP estudado neste trabalho, a representação da solução considera a sequência dos elementos e, portanto, uma possibilidade seria minerar padrões sequenciais em vez de conjuntos frequentes.

No entanto, os padrões sequenciais tratam apenas a sequência em que os elementos ocorrem, sem considerar sua consecutividade. Clientes consecutivos são extremamente importantes no problema 1-PDTSP, pois expressam quais clientes serão os próximos a serem atendidos a partir de um outro determinado cliente, levando em conta a distância e suas demandas. Considerando padrões sequenciais, seria possível identificar quais os clientes são atendidos após um determinado cliente, sem que essa ocorrência seja consecutiva.

Por esse motivo, optou-se por utilizar a mineração de conjuntos frequentes da maneira

como foi explicado no Capítulo 4, extraindo arcos frequentes como forma de preservar a característica da consecutividade. Além disso, o histórico apresentado pela combinação GRASP e mineração de conjuntos frequentes (sempre bem sucedida) contribuiu para essa escolha.

Outro trabalho futuro foi motivado pela observação das Figuras 5.5 (a) e 5.5 (b). Tais figuras ilustram que, à medida que aumenta a quantidade de clientes fixados na solução em construção, menor é o custo da solução construída e, ainda, menor será o custo de solução ao final da busca local. Assim, para os próximos trabalhos, pretende-se direcionar ainda mais a construção adaptada, dando prioridade às componentes conexas que fixam o maior número possível de clientes.

Outra possível vertente de estudo é avaliar o impacto da inserção de mineração de dados em outras metaheurísticas, como por exemplo os Algoritmos Genéticos e Busca Tabu.

Finalmente, pretende-se avaliar as estratégias apresentadas neste trabalho em outros problemas de otimização combinatória, tais como as variantes do problema de roteamento de veículos, citados no Capítulo 3, além de outros problemas de escalonamento de tarefas, a fim de comprovar a eficácia da hibridização com mineração de dados em mais problemas que envolvem ordem.

Referências

[1] Agrawal, R., Imieliński, T., Swami, A. Mining Association Rules Between Sets of Items in Large Databases. *SIGMOD Record 22* (1993), 207–216.

[2] Agrawal, R., Srikant, R. Fast Algorithms for Mining Association Rules in Large Databases. Em *Proceedings of the* 20^{th} *International Conference on Very Large Data Bases (VLDB'94)* (San Francisco, CA, USA, 1994), Morgan Kaufmann Publishers Inc., p. 487–499.

[3] Aiex, R. M., Resende, M. G. C., Ribeiro, C. C. TTT plots: A Perl Program to Create Time-to-Target Plots. *Optimization Letters 1* (2006), 10–1007.

[4] Aloise, D., Ribeiro, C. Adaptive Memory in Multistart Heuristics for Multicommodity Network Design. *Journal of Heuristics 17* (2011), 153–179.

[5] Anily, S., Bramel, J. Approximation Algorithms for the Capacitated Traveling Salesman Problem with Pickups and Deliveries. *Naval Research Logistics 46* (1999), 654–670.

[6] Anily, S., Hassin, R. The Swapping Problem. *Networks 22* (1992), 419–433.

[7] Barbalho, H., Rosseti, I., Martins, S. L., Plastino, A. A Hybrid Data Mining GRASP with Path-Relinking. *Computers & Operations Research* (2013). (Aceito para publicação, DOI: http://dx.doi.org/10.1016/j.cor.2012.02.022).

[8] Barra, T. W., Fuchshuber, R., Santos, L. F., Martins, S. L., Plastino, A. Explorando a Heurística DM-GRASP para o Problema das p-medianas. Em *Anais do XLI Simpósio Brasileiro de Pesquisa Operacional (XLI SBPO)* (Porto Seguro, BA, Brasil, 2009), p. 189–198.

[9] Baxter, J. Local Optima Avoidance in Depot Location. *The Journal of the Operational Research Society 32* (1981), 815–819.

[10] Benders, J. F. Partitioning Procedures for Solving Mixed-variables Programming Problems. *Numerische Mathematik 4* (1962), 238–252.

[11] Berger, D., Gendron, B., yves Potvin, J., Raghavan, S., Soriano, P. Tabu Search for a Network Loading Problem with Multiple Facilities. *Journal of Heuristics 6* (2000), 253–267.

[12] Bock, F. An Algorithm for Solving Traveling-Salesman and Related Network Optimization Problems. *Artigo não publicado, associado a apresentação no* 14^{th} *ORSA National Meeting* (1958).

[13] Cerný, V. Thermodynamical Approach to the Traveling Salesman Problem: An Efficient Simulation Algorithm. *Journal of Optimization Theory and Applications 45* (1985), 41–51.

[14] Chalasani, P., Motwani, R. Approximating Capacitated Routing and Delivery Problems. *SIAM Journal on Computing 28* (1999), 2133–2149.

[15] Cook, W. J., Cunningham, W. H., Pulleyblank, W. R., Schrijver, A. *Combinatorial Optimization*. John Wiley & Sons, Inc., New York, NY, USA, 1997.

[16] Cordeau, J.-F., Gendreau, M., Laporte, G. A Tabu Search Heuristic for Periodic and Multi-depot Vehicle Routing Problems. *Networks 30* (1997), 105–119.

[17] Cordeau, J.-F., Iori, M., Laporte, G., Salazar-González, J. J. A Branch-and-Cut Algorithm for the Pickup and Delivery Traveling Salesman Problem with LIFO Loading. *Networks 55* (2010), 46–59.

[18] Croes, G. A. A Method for Solving Traveling Salesman Problems. *Operations Research 6* (1958), 791–812.

[19] Feo, T. A., Resende, M. G. C. Greedy Randomized Adaptive Search Procedures. *Journal of Global Optimization 6* (1995), 109–133.

[20] Festa, P., Resende, M. G. C. An Annotated Bibliography of GRASP Part I: Algorithms. *International Transactions in Operational Research 16* (2009), 1–24.

[21] Festa, P., Resende, M. G. C. An Annotated Bibliography of GRASP Part II: Applications. *International Transactions in Operational Research 16* (2009), 131–172.

[22] Fleurent, C., Glover, F. Improved Constructive Multistart Strategies for the Quadratic Assignment Problem Using Adaptive Memory. *INFORMS Journal on Computing 11* (1999), 198–204.

[23] Gendreau, M., Hertz, A., Laporte, G. The Traveling Salesman Problem with Backhauls. *Computers & Operations Research 23* (1996), 501–508.

[24] Gendreau, M., Potvin, J.-Y. *Handbook of Metaheuristics*, 2^a ed., vol. 146 de *International Series in Operations Research & Management Science*. Morgan Kaufmann Publishers, 2010.

[25] Glover, F. Heuristics for Integer Programming using Surrogate Constraints. *Decision Sciences 8* (1977), 156–166.

[26] Glover, F. Future Paths For Integer Programming and Links to Artificial Intelligence. *Computers & Operations Research 13* (1986), 533 – 549.

[27] Glover, F. Tabu Search and Adaptive Memory Programing Advances, Applications and Challenges. Em *Interfaces in Computer Science and Operations Research* (1996), Kluwer, p. 1–75.

[28] Glover, F., Laguna, M., Martí, R. Fundamentals of Scatter Search and Path Relinking. *Control and Cybernetics 39* (2000), 653–684.

[29] Goethals, B., Zaki, M. J., Eds. *FIMI'03, Frequent Itemset Mining Implementations, Proceedings of the ICDM 2003 Workshop on Frequent Itemset Mining Implementations* (Melbourne, Florida, USA, 2003), vol. 90 de *CEUR Workshop Proceedings*, CEUR-WS.org.

[30] Goethals, B., Zaki, M. J. Advances in Frequent Itemset Mining Implementations: report on FIMI'03. *SIGKDD Exploration Newsletter 6* (2004), 109–117.

[31] Goldberg, D. E. *Genetic Algorithms in Search, Optimization and Machine Learning*, 1ª ed. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1989.

[32] Grahne, G., Zhu, J. Efficiently Using Prefix-trees in Mining Frequent Itemsets, 2003. Em Goethals e Zaki [29].

[33] Guerine, M., Rosseti, I., Plastino, A. Incorporando Mineração de Dados a uma Heurística GRASP/VND para o Problema do Caixeiro Viajante com Coleta e Entrega Envolvendo um Único Tipo de Produto. Em *Anais do XLV Simpósio Brasileiro de Pesquisa Operacional (XLV SBPO)* (Natal, RN, Brasil, 2013). (Aceito para publicação).

[34] Han, J., Kamber, M. *Data Mining: Concepts and Techniques*, 3ª ed. Morgan Kaufmann Publishers, 2011.

[35] Han, J., Pei, J., Yin, Y. Mining Frequent Patterns without Candidate Generation. *SIGMOD Record 29* (2000), 1–12.

[36] Hernández-Pérez, H., Salazar-González, J. A Branch-and-Cut Algorithm for a Traveling Salesman Problem with Pickup and Delivery. *Discrete Applied Mathematics 145* (2004), 453–459.

[37] Hernández-Pérez, H., Salazar-González, J. Heuristics for the One-Commodity Pickup-and-Delivery Traveling Salesman Problem. *Transportation Science 38* (2004), 245–255.

[38] Hernández-Pérez, H., Salazar-González, J. The One-Commodity Pickup and Delivery Traveling Salesman Problem: Inequalities and Algorithms. *Networks 50* (2007), 258–272.

[39] Hernández-Pérez, H., Salazar-González, J. The Multi-Commodity One-to-One Pickup-and-Delivery Traveling Salesman Problem. *European Journal of Operational Research 196* (2009), 987–995.

[40] Hernández-Pérez, H., Salazar-González, J., Rodríguez-Martín, I. A Hybrid GRASP/VND Heuristic for the One-Commodity Pickup-and-Delivery Traveling Salesman Problem. *Computers & Operations Research 36* (2009), 1639–1645.

[41] Holland, J. H. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, MI, USA, 1975.

[42] Hosny, M. I., Mumford, C. L. Solving the One-Commodity Pickup and Delivery Problem using an Adaptive Hybrid VNS/SA Approach. Em *Proceedings of the* 11^{th} *International Conference on Parallel Problem Solving From Nature (PPSN2010)* (2010), p. 189–198.

[43] Kirkpatrick, S., Gelatt, C. D., Vecchi, M. P. Optimization by Simulated Annealing. *Science 220* (1983), 671–680.

[44] Laguna, M., Martí, R. GRASP and Path Relinking for 2-layer Straight Line Crossing Minimization. *INFORMS Journal on Computing 11* (1999), 44–52.

[45] Lian-Ming, M., Xi-Li, D. A novel Ant Colony System for Solving the One-Commodity Traveling Salesman Problem with Selective Pickup and Delivery. Em *Proceedings of the* 8^{th} *International Conference on Natural Computation (ICNC)* (2012), p. 1096–1101.

[46] Lin, S. Computer Solutions of the Traveling Salesman Problem. *Bell System Technical Journal 44* (1965), 2245–2269.

[47] Lin, S., Kernighan, B. W. An Effective Heuristic Algorithm for the Traveling-Salesman Problem. *Operations Research 21* (1973), 498–516.

[48] Lodi, A., Allemand, K., Liebling, T. M. An Evolutionary Heuristic for Quadratic 0–1 Programming. *European Journal of Operational Research 119* (1999), 662–670.

[49] Lourenço, H. R., Martin, O. C., Stützle, T. Iterated Local Search. Em *Handbook of Metaheuristics*, F. Glover and G. Kochenberger, Eds. Kluwer Academic Publishers, 2003, p. 321–354.

[50] Louveaux, F., Salazar-González, J.-J. On the One-Commodity Pickup-and-Delivery Traveling Salesman Problem with Stochastic Demands. *Mathematical Programming 119* (2009), 169–194.

[51] Martinovic, G., Aleksi, I., Baumgartner, A. Single-Commodity Vehicle Routing Problem with Pickup and Delivery Service. *Mathematical Problems in Engineering* (2008), 1–18.

[52] Martins, D., Rosseti, I., Martins, S. L., Plastino, A. Making Heuristics Faster with Data Mining. Em *Anais do XLIV Simpósio Brasileiro de Pesquisa Operacional (XLIV SBPO)* (Rio de Janeiro, Brasil, 2012), p. 4379–4386.

[53] Min, H. The Multiple Vehicle Routing Problem with Simultaneous Delivery and Pick-up Points. *Transportation Research Part A: General 23* (1989), 377–386.

[54] Mladenović, N., Hansen, P. Variable Neighborhood Search. *Computers & Operations Research 24* (1997), 1097–1100.

[55] Mladenović, N., Urosevic, D., Hanafi, S., Ilic, A. A General Variable Neighborhood Search for the One-Commodity Pickup-and-Delivery Travelling Salesman Problem. *European Journal of Operational Research 220* (2012), 270–285.

[56] OR, I. *Traveling Salesman-Type Combinatorial Problems and their Relation to the Logistics of Regional Blood Banking*. Tese de Doutorado, Department of Industrial Engineering and Management Sciences, Northwestern University, Evanston, IL, EUA, 1976.

[57] Osman, I., Laporte, G. Metaheuristics: A Bibliography. *Annals of Operations Research 63* (1996), 511–623.

[58] Paes, B. C., Subramanian, A., Ochi, L. S. Uma Heurística Híbrida para o Problema do Caixeiro Viajante com Coleta e Entrega Envolvendo um Único Tipo de Produto. Em *Anais do XLII Simpósio Brasileiro de Pesquisa Operacional (XLII SBPO)* (Bento Gonçalves, RS, Brasil, 2010), p. 1513–1524.

[59] Pietracaprina, A., Zandolin, D. Mining Frequent Itemsets using Patricia Tries, 2003. Em Goethals e Zaki [29].

[60] Plastino, A., Fonseca, E., Fuchshuber, R., Martins, S. L., Freitas, A. A., Martino, L., Salhi, S. A Hybrid Data Mining Metaheuristic for the p-median Problem. Em *Proceedings of the SIAM International Conference on Data Mining* (Sparks, NV, EUA, 2009), p. 189–198.

[61] Plastino, A., Fuchshuber, R., Martins, S. L., Freitas, A. A., Salhi, S. A Hybrid Data Mining Metaheuristic for the p-median Problem. *Statistical Analysis and Data Mining 4* (2011), 313–335.

[62] Psaraftis, H. A Dinamic Programming Solution to the Single Vehicle Many-to-many immediate request dial-a-ride problem. *Transportation Science 14* (1980), 130–154.

[63] Resende, M. G. C., Ribeiro, C. C. Greedy Randomized Adaptive Search Procedures. Em *Handbook of Metaheuristics*, F. Glover and G. Kochenberger, Eds. Kluwer Academic Publishers, 2003, p. 219–249.

[64] Ribeiro, M. H. Incorporando Técnicas de Mineração de Dados à Metaheurística GRASP. Dissertação de Mestrado, Instituto de Computação, Universidade Federal Fluminense, Niterói, RJ, Brasil, 2005.

[65] Ribeiro, M. H., de Aragão Trindade, V., Plastino, A., Martins, S. L. Hybridization of GRASP Metaheuristics with Data Mining Techniques. Em *Proceedings of the Workshop on Hybrid Metaheuristcs in Conjunction with the* 16^{th} *European Conference on Artificial Intelligence* (2004), p. 69–78.

[66] Ribeiro, M. H., Plastino, A., Martins, S. L. Hybridization of GRASP Metaheuristic with Data Mining Techniques. *Journal of Mathematical Modelling Algorithms 5* (2006), 23–41.

[67] Rochat, Y., Taillard, R. D. Probabilistic Diversification And Intensification In Local Search For Vehicle Routing. *Journal of Heuristics 1* (1995), 147–167.

[68] Santos, L. F. Metaheurística Híbrida GRASP-MD: Novas aplicações e Paralelização. Dissertação de Mestrado, Instituto de Computação, Universidade Federal Fluminense, Niterói, RJ, Brasil, 2006.

[69] Santos, L. F., Martins, S. L., Plastino, A. Applications of the DM-GRASP Heuristic: a Survey. *International Transactions in Operational Research 15* (2008), 387–416.

[70] Santos, L. F., Milagres, R., Albuquerque, C. V., Martins, S. L., Plastino, A. A Hybrid GRASP with Data Mining for Efficient Server Replication for Reliable Multicast. Em *Proceedings of the* 49^{th} *Annual IEEE GLOBECOM Technical Conference (GLOBECOM'06)* (2006), p. 1–6.

[71] Santos, L. F., Ribeiro, M. H., Plastino, A., Martins, S. L. A Hybrid GRASP with Data Mining for the Maximum Diversity Problem. Em *Proceedings of the* 2^{nd} *International Workshop on Hybrid Metaheuristics* (Barcelona, Espanha, 2005), vol. 3636 de *Lecture Notes in Computer Science*, p. 116–127.

[72] Schrijver, A. On the History of Combinatorial Optimization (till 1960). Em *Handbook of Discrete Optimization*, K. Aardal, G. Nemhauser, and R. Weismantel, Eds. Elsevier, 2005, p. 1–68.

[73] Siegel, S., Castellan Jr, N. J. *Nonparametric Statistics for the Behavioral Sciences*, 2^a ed. McGraw-Hill, 1988.

[74] Solomon, M. M. Algorithms for the Vehicle Routing Problem and Scheduling Problem with Time Window Constraints. *Operations Research 35* (1987), 254–265.

[75] Wang, F., Lim, A., Xu, Z. The One-Commodity Pickup and Delivery Travelling Salesman Problem on a Path or a Tree. *Networks 48* (2006), 24–35.

[76] Zhao, F., Li, S., Sun, J., Mei, D. Genetic Algorithm for the One-Commodity Pickup-and-Delivery Traveling Salesman Problem. *Computers & Industrial Engineering 56* (2009), 1642–1648.