Universidade Federal Fluminense

TIAGO ARAÚJO NEVES

Heurísticas com Memória Adaptativa Aplicadas ao Problema de Roteamento e Scheduling de Sondas de Manutenção

NITERÓI

2007

Universidade Federal Fluminense

TIAGO ARAÚJO NEVES

Heurísticas com Memória Adaptativa Aplicadas ao Problema de Roteamento e Scheduling de Sondas de Manutenção

Dissertação de Mestrado submetida ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense como requisito parcial para a obtenção do tíulo de Mestre em Computação. Area de concentração: Otimização Combinatória e Inteligência Artificial.

Orientador:
Luiz Satoru Ochi

NITERÓI

2007

Heurísticas com Memória Adaptativa Aplicadas ao Problema de Roteamento e *Scheduling* de Sondas de Manutenção

Tiago Araújo Neves

Dissertação de Mestrado submetida ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense como requisito parcial para a obtenção do tíulo de Mestre em Computação.

Aprovada por:

Prof. Luiz Satoru Ochi / IC-UFF (Presidente)

Prof. Eduardo Uchoa Barboza / EP-UFF

Profa. Lúcia M. A. Drummond / IC-UFF

Prof. Marcone J. F. Souza / DECOM-UFOP

Profa. Nair Maria Maia de Abreu / COPPE-UFRJ

Niterói, 17 de Maio de 2007.

Para meus pais, minha irmã e meus avós.

Agradecimentos

A Deus, pela saúde e pela ajuda que mais ninguém pode dar.

À minha família, em especial à minha avó Tereza, aos meus pais Willes e Consolação, à minha irmã e a meu tio Afonso pelo apoio e incentivo.

Ao professor e orientador Luiz Satoru Ochi, pelo apoio, orientação e amizade.

Aos professores Eduardo Uchoa, Cristina Boeres e Lúcia Drummond pelo apoio dado.

Ao amigo Marcone Jamilson Freitas Souza, pelo companheirismo e amizade.

Aos grandes amigos que fiz em Ouro Preto e reencontrei em Niterói.

Aos bons amigos que fiz aqui em Niterói.

Aos irmãos de Mestrado, Doutorado e laboratório.

Aos irmãos de república.

Aos amigos de outras cidades, que quando precisei, sempre estenderam as mãos para me ajudar.

A todos os demais, que de alguma forma contribuíram para a realização deste trabalho.

Resumo

O Problema de Roteamento e *Scheduling* de Sondas de Manutenção é uma generalização do Problema de Roteamento de Veículos clássico, que consiste em criar rotas para o conjunto de sondas responsáveis por fazer manutenção nos poços de petróleo. O objetivo do problema é reduzir o tempo em que os poços ficam ociosos, aumentando assim a quantidade de óleo coletado. Para resolver o problema foram utilizadas técnicas de Memória Adaptativa, que fazem uso estratégico de memória, associadas à heurísticas. Quatro heurísticas foram propostas: um GRASP tradicional, um GRASP que faz uso de técnicas de memória adaptativa, uma Busca Tabu e uma Busca Local Iterada. Os algoritmos propostos foram testados com instâncias de tamanho variado e, para algumas dessas instâncias, comparados com métodos exatos. Resultados computacionais mostram que o uso de técnicas de memória adaptativa pode melhorar o desempenho dessas metaheurísticas, tornando-as mais robustas.

Abstract

The Workover Rig Routing and Scheduling Problem is a generalization of the classic Vehicle Routing Problem and consists on creating routes to the workover rig set responsable for doing mantenance on oil wells. The goal is reduce the wells' idle time, increasing the amount of oil colected. Adaptive Memory techniques, that are techniques that use memory in a strategic way, associated with heuristics were used to solve the problem. Four heuristics were proposed: a traditional GRASP, a GRASP that uses adaptive memory techniques, a Tabu Search and a Iterated Local Search. The proposed algorithms were tested with variable size instances, and, for some of these instances, compared with exact methods. Computacional results show that the use of adaptive memory techniques can improve metaheuristics performance, making them more robust.

Palavras-chave

1. Metaheurística
2. Memória Adaptativa
3. Busca Tabu
4. GRASP
5. Busca Local Iterada
6. Problema de Roteamento e *scheduling* de Sondas de Manutenção

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

O petróleo é uma das riquezas minerais mais importantes da atualidade. Seu uso como fonte de energia encontra-se amplamente difundido em nossa sociedade, bem como os seus derivados diretos e indiretos, gerando empregos e riquezas em todos os estágios de sua cadeia produtiva.

O mercado de petróleo é muito competitivo, movimentado milhões de dólares anualmente. Por ter um mercado tão competitivo e também por se tratar de um recurso não renovável, sua exploração deve ser feita de maneira eficiente, de modo que os custos sejam reduzidos e as jazidas melhor aproveitadas. Por isso, os problemas relacionados à extração do petróleo vêm se tornando um grande desafio para as empresas.

Um destes problemas, abordado neste trabalho, é o Problema de Roteamento e *Scheduling* de Sondas de Manutenção (PRSSM), que consiste em construir um conjunto de boas rotas para um conjunto de sondas responsáveis pela manutenção dos poços. O objetivo do problema é minimizar o tempo que os poços em manutenção permanecem ociosos, aumentando assim a quantidade de óleo coletado.

Para resolver este problema foram feitos, neste trabalho, estudos em técnicas de memória adaptativa. Tais técnicas fazem uso de estruturas de memória com o objetivo de tentar melhorar a qualidade das soluções encontradas. Com base nestes estudos foram propostas quatro heurísticas para resolver o problema: um GRASP tradicional; uma versão do GRASP que faz uso de memória adaptativa, chamada GRASP+MA; uma Busca Local Iterada (*Iterated Local Search*) e uma Busca Tabu, que também fazem uso de estruturas de memória.

Inicialmente, para avaliar o desempenho das técnicas de memória adaptativa, foram feitos testes comparativos entre os algoritmos GRASP tradicional, que não usa memória, e

a versão adaptativa GRASP+MA. Os resultados empíricos indicam que o uso das técnicas de memória adaptativa pode melhorar consideravelmente a qualidade final das soluções encontradas por estas heurísticas.

Em uma segunda fase, foram analisados os comportamentos das heurísticas Busca Local Iterada e a Busca Tabu, que são procedimentos que tem se mostrado eficientes na solução de diferentes problemas de otimização combinatória. Em simulações feitas com diferentes instâncias do PRSSM, foi observado que estas heurísticas, que também fazem uso de memória adaptativa, produzem soluções altamente competitivas para o problema analisado.

Para comprovar a qualidade das heurísticas propostas, foram feitos testes comparativos entre estas e um modelo matemático, também proposto neste trabalho. Os resultados mostram que as heurísticas propostas são capazes de atingir o valo ótimo para as instâncias testadas.

O restante deste trabalho encontra-se dividido da seguinte maneira: no capítulo 2, é feita uma breve revisão da literatura; no capítulo 3, é feita uma descrição detalhada do problema abordado e proposta uma formulação matemática para o problema; no capítulo 4, é feita uma discussão sobre memória adaptativa e são apresentadas as heurísticas propostas para o problema; no capítulo 5, são apresentados os resultados computacionais e no capítulo 6, as conclusões e propostas para trabalhos futuros.

Capítulo 2

Revisão de literatura

O PRSSM é uma generalização do problema clássico de roteamento de veículos (PRV). Nas últimas décadas, o problema de roteamento de veículos e suas variações têm sido massivamente estudados devido ao grande número de aplicações reais e a elevada complexidade destes problemas.

O modelo clássico de roteamento de veículos e uma variante com múltiplos depósitos (origens) são descritos a seguir.

2.1 O Problema de Roteamento de Veículos - PRV

O Problema de Roteamento de veículos (PRV) foi proposto nos anos 50 por Dantzig e Ramser [10] e desde então vem sendo estudado massivamente a ponto de ser um dos mais estudados dentro da área de otimização combinatória [8].O conceito deste problema é bastante comum no cotidiano das pessoas, entre suas aplicações se incluem a distribuição de gás, a coleta de lixo, a entrega e coleta de encomendas, a construção de itinerários de ônibus, caminhões, helicópteros, navios, aeronaves, trens, etc.

A literatura apresentou, nas últimas décadas, diferentes aplicações do PRV bem como inúmeros métodos exatos e heurísticos para a sua resolução. Contudo, Grande destaque tem sido dado às técnicas heurísticas devido ao fato de os modelos de PRV serem, em sua maioria, classificados como NP-Difíceis.

O problema típico de roteamento de veículos (PRV) pode ser descrito, conforme Bräysy e Gendreau [4], como o problema de definição do menor custo das rotas de um depósito a um conjunto de clientes distribuídos em um plano geográfico. As rotas devem ser definidas para que cada cliente seja visitado somente uma única vez por exatamente um

único veículo, começando e terminando em um mesmo depósito, de forma a não exceder a capacidade do veículo. Todas as demandas dos clientes devem ser satisfeitas.

Uma revisão sobre diversos métodos de solução para o PRV, bem como a apresentação de diferentes variantes do problema é encontrada em Cordeau *et al.* [8]. Nesse trabalho, os procedimentos de solução existentes na literatura são classificados em quatro categorias, a saber: precisão, velocidade, simplicidade e flexibilidade.

Golden *et al.* [18] fazem uma revisão histórica do PRV, apresentando uma comparação entre diversas técnicas utilizadas na resolução deste problema. Os autores sugerem maneiras de apresentar resultados e parâmetros. Estes ainda recomendam que os pesquisadores construam heurísticas capazes de auto-ajustar alguns de seus parâmetros, porque, segundo eles, algumas heurísticas recentes possuem um número de parâmetros maior que o número de problemas teste existentes.

Laporte *et al.*, em [23], fazem uma revisão sobre as heurísticas clássicas apresentadas na literatura para o PRV. Estes afirmam que pouco espaço ainda resta para novos melhoramentos na área de heurísticas clássicas para o problema, uma vez que, as metaheurísticas são capazes de encontrar soluções de melhor qualidade e, algumas vezes, exigindo menos tempo computacional.

Gendreau *et al.*, em [14], fazem uma comparação entre diversas metaheurísticas aplicadas ao PRV, chegando a conclusão de que as versões de Busca Tabu são as que apresentam os melhores resultados, enquanto as versões de *Simulated Annealing* e Colônia de Formigas usualmente não se mostram competitivas neste problema.

No que tange aos métodos exatos, Toth *et al.* [40] apresentam uma revisão dos mais importantes algoritmos de *branch-and-bound* propostos, durante a última década, para as versões simétrica e assimétrica do PRV. Estes afirmam que ainda há muito para ser feito, uma vez que ainda existem instâncias do problema em aberto.

Uma possível formulação matemática para o PRV é apresentada [19], e aqui transcrita com o objetivo de ilustrar o uso de métodos exatos para o PRV. Nesta formulação considera-se: um conjunto de vértices (clientes) N enumerado por $\{1, ..., n\}$, representando os clientes e 0 a origem das rotas; um conjunto de arcos ligando estes vértices, com c_{ij} representando o custo de viagem para se ir do cliente i ao cliente j; q_j a demanda do cliente j; Q a capacidade dos veículos e, finalmente, k a quantidade disponível de veículos homogêneos.

Assim, definimos as seguintes variáveis:

- y_{ij} representa a quantidade de fluxo que passa de um cliente i para um cliente j.
- $x_{ij} = \begin{cases} 1 & \text{se o cliente } j \text{ é atendido logo após o cliente } i \\ 0 & \text{caso contrário} \end{cases}$

$$Min \quad \sum_{i=0}^{n}\sum_{j=0}^{n} c_{ij}x_{ij} \tag{2.1}$$

$s.a.$

$$\sum_{i=0}^{n} x_{ij} = 1 \quad (j = 1, ..., n) \tag{2.2}$$

$$\sum_{j=0}^{n} x_{ij} = 1 \quad (i = 1, ..., n) \tag{2.3}$$

$$\sum_{i=0}^{n} x_{i0} = \sum_{j=0}^{n} x_{0j} = k \tag{2.4}$$

$$\sum_{i=0}^{n} y_{ij} - \sum_{i=0}^{n} y_{ji} = q_j \ (j = 1, ..., n) \tag{2.5}$$

$$q_j x_{ij} \leq y_{ij} \leq (Q - q_i)x_{ij} \quad (i, j = 1, ..., n) \tag{2.6}$$

$$x_{ij} \in \{0, 1\}, \quad (i, j = 1, ..., n) \tag{2.7}$$

Na formulação descrita por (2.1)-(2.7), a função objetivo descrita em 2.1 requer que se minimize o custo total de viagem. As restrições 2.2 exigem que todo cliente só seja visitado uma única vez. As restrições 2.3 exigem que só se possa partir uma única vez de um cliente. As restrições 2.4 e 2.7 especificam que existem exatamente k rotas, fazendo assim, com que todos os veículos sejam utilizados. As restrições 2.5 e 2.6 evitam a criação de rotas desconexas da origem e as restrições 2.6 estabelecem a ligação entre as variáveis y_{ij} e x_{ij}, assegurando que haverá transporte de carga em um arco se algum veículo utilizar este arco.

Diversas são as variações do PRV básico, na maioria das vezes apenas compostas por um maior número de restrições, seja de tempo, de distância, de composição de frota, de limitação de frota, etc. Mais informações sobre o PRV e suas variantes podem ser encontradas em trabalhos relacionados como em [39].

2.2 O Problema de Roteamento de Veículos com Múltiplos Depósitos - PRVMD

O PRVMD difere do PRV pelo fato de que existem vários depósitos de onde as rotas podem se originar. Dentre os modelos mais populares da literatura, este é o que mais se assemelha ao PRSSM, abordado neste trabalho. Contudo, é um problema ainda pouco explorado quando comparado com o PRV.

Na maioria dos casos, para resolvê-lo é feita uma associação dos clientes aos depósitos. Esta associação faz com que o problema seja reduzido a vários PRV tradicionais, que são então resolvidos separadamente ou em conjuntos.

Uma das primeiras heurísticas para resolver o PRVMD foi proposta em 1969 por Tillman [38] e usa o critério de economia de Clarke e Wright [7]. Primeiramente, o algoritmo associa cada cliente ao depósito que se encontra mais próximo a este, e depois, constrói rotas para cada depósito. Essas rotas são gradualmente combinadas usando um critério de economia que leva em conta a presença de depósitos diferentes, resultando assim em rotas maiores. Essa heurística é apenas construtiva e não possui uma fase posterior de refinamento.

Taillard [33] descreve métodos para resolver problemas de clusterização com centróides através da decomposição do problema em problemas menores. Apesar de os métodos terem sido propostos para a resolução do problema de p-medianas, o autor menciona que alguns destes métodos podem ser utilizados para o PRVMD. Através de especializações, os métodos podem ser usados para fazer a associação de clientes aos depósitos, bem como para otimizar os sub-problemas produzidos por esta associação. Porém, o autor não apresenta resultados para estas especializações.

Tansini *et al.* [37] apresentam uma comparação entre vários métodos de associação entre clientes e depósitos. Resultados apresentados apontam, segundo os autores, que usar métodos exatos para resolver o problema de associação é uma boa opção, uma vez que estes apresentam soluções de boa qualidade em um tempo computacional viável.

Renaud *et al.* [29] propõem uma Busca Tabu composta de uma fase de construção inicial, seguido de uma busca local, uma fase de intensificação e uma de diversificação. O algoritmo foi testado em diversas instâncias já conhecidas na literatura, retiradas de [15, 6, 5] , que possuem entre 50 e 360 clientes e entre 2 e 9 depósitos. O algoritmo foi comparado com os propostos em [15] e [5], conseguindo melhores resultados para quase todas as instâncias.

Nagy e Salhi [25] abordam o PRV com entrega e coleta de mercadorias, apresentando uma breve discussão sobre os modelos existentes para este tipo de problema. Os autores apresentam métodos para modificar rotas e mostram como estender suas propostas para o PRVMD. Foram apresentados resultados de testes comparativos com outras heurísticas da literatura, utilizando tanto um único depósito quanto múltiplos depósitos, para instâncias bem conhecidas da literatura.

Recentemente, em 2007, Crevier, Cordeau e Laporte [9] apresentaram um estudo sobre uma versão do PRVMD na qual os veículos podem parar em depósitos intermediários para reabastecimento de carga durante a execução das rotas. Uma formulação matemática para o problema é apresentada, e um algoritmo de Busca Tabu é proposto baseado nas idéias de Cordeau [8] e Taillard [34]. Testes foram feitos para instâncias clássicas e para instâncias criadas especificamente para este problema. Resultados mostram uma heurística robusta e que exige um tempo computacional aceitável.

No que diz respeito aos algoritmos exatos, podemos mencionar o trabalho de Laporte [22], que apresenta um algoritmo baseado em uma formulação com relaxações combinatórias. O problema é reduzido a um problema de associação através de uma transformação em grafos. Um algoritmo *Branch-and-Bound* modificado é então utilizado para resolver as inviabilidades geradas para o problema original.

Uma possível formulação para o problema, aqui proposta com a finalidade de ilustrar o uso de métodos exatos para o PRVMD, pode ser definida da seguinte maneira: considere um grafo $G = (V, D, A)$, onde $V = \{v_1, ..., v_n\}$ é o conjunto de vértices que representam os clientes. Considere ainda o conjunto $D = \{d_1, ..., d_k\}$ de vértices que representam depósitos e A o conjunto de arestas interligando estes vértices, com um custo c_{ij} associado, representando o custo de ir do vértice i ao vértice j. Cada cliente j possui uma demanda q_j associada e cada depósito k possui um número de veículos associado r_k. A carga máxima comportada pelos veículos homogêneos é denotada por Q.

Assim definem-se as seguintes variáveis:

- y_{ijk} representa a quantidade de fluxo que passa do cliente i para o cliente j vindo do depósito k.
- $x_{ijk} = \begin{cases} 1 & \text{se o cliente } j \text{ é atendido logo após o cliente } i \text{ por um veículo vindo} \\ & \text{do depósito } k \\ 0 & \text{caso contrário} \end{cases}$

$$Min \quad \sum_{i \in V \cup D} \sum_{j \in V \cup D} \sum_{k \in D} c_{ij} x_{ijk} \tag{2.8}$$

S.a.

$$\sum_{j \in V \cup D} \sum_{k \in D} x_{ijk} = 1 \ \ \forall i \in V \tag{2.9}$$

$$\sum_{i \in V \cup D} \sum_{k \in D} x_{ijk} = 1 \ \ \forall j \in V \tag{2.10}$$

$$\sum_{i \in V \cup D} \sum_{j \in V} x_{ijk} = r_k \ \ \forall k \in D \tag{2.11}$$

$$\sum_{i \in V} \sum_{j \in V \cup D} x_{ijk} = r_k \ \ \forall k \in D \tag{2.12}$$

$$\sum_{i \in V \cup D} \sum_{k \in D} y_{ijk} - \sum_{i \in V \cup D} \sum_{k \in D} y_{jik} = q_j \ \ \forall j \in V \tag{2.13}$$

$$\sum_{i \in V} x_{ikk} = r_k \ \ \forall k \in D \tag{2.14}$$

$$\sum_{j \in V} x_{kjk} = r_k \ \ \forall k \in D \tag{2.15}$$

$$\sum_{i \in V \cup D} x_{ijk} = \sum_{i \in V \cup D} x_{jik} \ \ \forall j \in V, \ \forall k \in D \tag{2.16}$$

$$y_{ijk} \leq x_{ijk} Q \ \ \forall i, j \in V \cup D, \ \ \forall k \in D \tag{2.17}$$

$$x_{ijk} \in \{0, 1\}, \ \ \forall i, j \in V \cup D, \ \ \forall k \in D \tag{2.18}$$

Na formulação descrita por (2.8)-(2.18), a função objetivo é descrita em 2.8 e requer a solução de menor custo para o problema. As restrições 2.9 e 2.10 exigem que para cada cliente haja exatamente uma entrada e uma saída. As restrições 2.11 e 2.12 tratam do número de veículos utilizados por cada depósito, visando que todos os veículos sejam utilizados. As restrições 2.13 tratam da proibição da formação de subrotas. As restrições 2.14, 2.15 e 2.16 exigem que as rotas terminem no mesmo depósito que começaram. As restrições 2.17 estabelecem a ligação entre as variáveis y_{ijk} e x_{ijk} exigindo que só haja fluxo passando por uma aresta se esta estiver ativa na solução e assegurando a viabilidade da rota no que tange à capacidade dos veículos.

Capítulo 3

O Problema de Roteamento e Scheduling de Sondas de Manutenção (PRSSM)

A retirada de petróleo e gás do subsolo, seja em terra ou mar, é uma atividade de elevado custo, pois requer mão-de-obra especializada, alta tecnologia e equipamentos sofisticados. E por se tratarem de recursos não renováveis, as jazidas necessitam ser exploradas por métodos comprovadamente eficientes, que garantam o melhor aproveitamento das mesmas.

Como dito em [26], [41] e [2], em muitos casos, a elevação dos fluidos é feita de forma artificial com o uso de equipamentos especiais devido ao fato de que os poços nem sempre possuem pressão suficiente para que os fluidos atinjam a superfície. Por isso, serviços como os de limpeza e manutenção dos equipamentos de elevação de fluidos são essenciais para se evitar paradas no processo de extração.

Esses serviços de manutenção, limpeza e até mesmo reparo para poços terrestres (*onshore*) são realizados por equipamentos móveis, denominados sondas, que estão disponíveis em um número limitado e muito pequeno se comparado à quantidade de poços que demandam serviços. Seria ideal que houvesse uma sonda de manutenção fixa para cada poço; assim, um poço que precisasse de manutenção seria prontamente atendido. Porém, devido ao seu alto custo de aquisição, não é lucrativo manter um número tão grande desse tipo de equipamento. Sendo assim, a manutenção imediata de um poço nem sempre é possível, o que provoca a ocorrência de uma fila de poços à espera de atendimento.

Quando é detectada uma falha nos equipamentos de algum poço, para que este volte à funcionar o mais breve possível, é feito um pedido de manutenção. Em seguida, é feito um planejamento de atendimento para cada sonda disponível em um dado período de tempo pré-estabelecido e, posteriormente, essas sondas são designadas para a correção dos poços que aguardam manutenção na ordem estabelecida pelo seu escalonamento. Todo este

processo, da detecção de problemas nos poços, até a execução dos escalonamentos pelas sondas, é chamado de horizonte de trabalho. Trindade [41] apresenta uma descrição detalhada do protocolo de agendamento de manutenção para os poços.

É importante lembrar que cada poço possui sua respectiva vazão, que é a capacidade de produção de óleo bruto por unidade de tempo. Quando algum problema nos equipamentos de extração de um poço é detectado, por questões de segurança, os equipamentos são desligados e a vazão desse poço deixa de ser coletada até que o problema seja corrigido.

Desta forma, podemos colocar como objetivo do Problema de Roteamento e *Scheduling* de Sondas de Manutenção (PRSSM) encontrar o melhor escalonamento para as sondas disponíveis, minimizando a perda total de produção de petróleo associada aos poços que estão aguardando por atendimento. Entende-se por perda de produção de petróleo de um poço como sendo a vazão desse poço multiplicada pela quantidade de unidades de tempo que esse poço teve sua produção interrompida e, por perda total de produção, a soma da perda de produção de todos os poços que aguardam pela manutenção. Além disso, para gerar uma boa solução do problema, deve-se fazer um escalonamento de sondas para o conjunto de poços, considerando informações como: vazão do poço, sua posição em relação aos demais poços que necessitam ser reparados e o prazo limite para o seu atendimento.

Assim, cada poço i que solicita a manutenção possui: uma vazão, que indica a capacidade de produção deste poço; um tempo de serviço, que indica o tempo estimado para realizar a manutenção e um limite de tempo, que representa o tempo máximo que este poço pode ficar ocioso (sem produzir). Cada poço deve ser atendido uma única vez e por uma única sonda.

Cada sonda parte de um ponto inicial diferente no começo do planejamento. Esse fato torna o PRSSM integrante da classe dos Problemas de Roteamento de Veículos com Múltiplos Depósitos.

A Figura 3.1 mostra uma representação gráfica para o PRSSM. Nela são vistos, um conjunto de poços, em cinza claro e marcados pela letra P; um conjunto de sondas, cinza escuro e marcadas pela letra S; e as rotas para cada sonda. Note que, as rotas não precisam retornar ao ponto de origem. Este fato ocorre devido à inexistência de garagens para as sondas. Os pontos iniciais das sondas, em um horizonte de trabalho, são os últimos poços atendidos por cada uma delas no horizonte anterior. Assim, um poço que se encontra no final da rota de uma sonda, será o ponto de origem desta mesma sonda no próximo horizonte de trabalho.

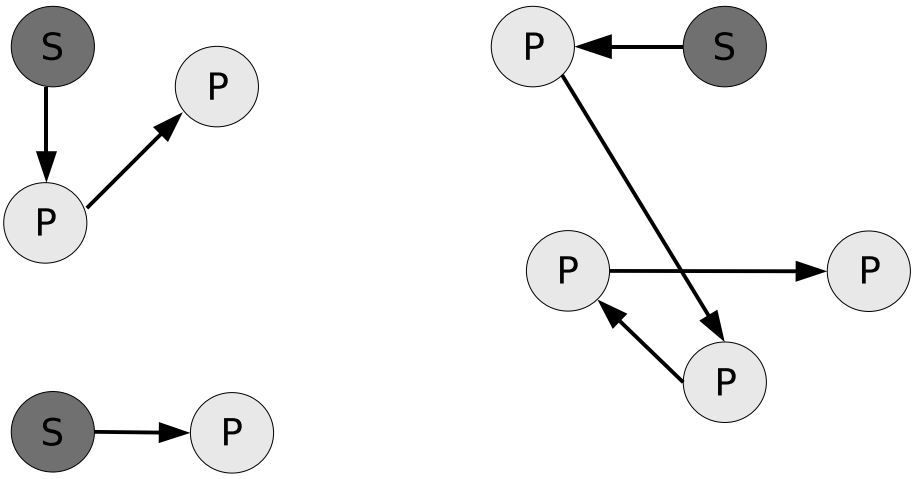


Figura 3.1: Representação Gráfica para o PRSSM

O PRSSM é na verdade, uma generalização dos problemas PRV e PRVMD. Como o PRV pertence à classe NP-difícil [24] e por se tratar de uma generalização deste último, o PRSSM também é classificado como pertencente à classe NP-difícil, limitando com isso o uso exclusivo de métodos exatos para a solução do problema de maneira viável.

Entretanto, mesmo com estas limitações ao uso de métodos exatos, é importante desenvolver formulações matemáticas para estes problemas, tanto para avaliar os resultados de métodos aproximados em instâncias de pequeno porte, como para tentar obter limites inferiores de boa qualidade para o valor ótimo (valor ou função objetivo associado à solução ótima). Desta forma, propomos neste capítulo uma formulação matemática do PRSSM, modelando-o como um problema de programação linear inteira mista, com o objetivo de possibilitar o uso de métodos exatos, via *software* CPLEX, na resolução de instâncias do PRSSM. Para isto, considere a seguinte notação:

Seja N o conjunto de poços, $N = \{p_1, p_2, ..., p_n\}$, $S = \{s_1, s_2, ..., s_r\}$ o conjunto de sondas e t_{ij} o tempo gasto para percorrer a distância entre os poços p_i e p_j, com $i \neq j$ e $p_i, p_j \in N$. Cada poço p_i possui associado uma vazão v_i por unidade de tempo, um tempo de serviço ts_i e um tempo limite de atendimento l_i associado. No início da rota, cada sonda s_k está localizada em algum ponto o_k que pertence ao conjunto de origens de sondas, dado como entrada e indexado por $O = \{o_1, o_2, ..., o_r\}$. Cada poço deve ser visitado uma única vez e por uma única sonda, sendo que nenhum poço pode ficar sem atendimento no período. Na formulação descrita por (3.1)-(3.11), a função objetivo descrita em (3.1) requer a minimização da quantidade de óleo que deixa de ser coletado no período. As restrições (3.2) exigem que todos os poços sejam atendidos dentro de seus respectivos

limites de tempo de atendimento.

Definimos também, as seguintes variáveis:

- $y_i \geq 0$; tempo que o poço i ficou parado.
- $x_{ijk} = \begin{cases} 1 & \text{se o poço } j \text{ é atendido pela sonda } k \text{ logo após o poço } i \\ 0 & \text{caso contrário} \end{cases}$

$$Min \quad \sum_{i \in N} y_i v_i \tag{3.1}$$

S.a.

$$y_i \leq l_i, \quad \forall i \in N \tag{3.2}$$

$$\sum_{\substack{i \in N \cup O \\ i \neq j}} \sum_{k \in S} x_{ijk} = 1, \forall j \in N \tag{3.3}$$

$$\sum_{\substack{j \in N \\ i \neq j}} \sum_{k \in S} x_{ijk} \leq 1, \quad \forall i \in N \cup O \tag{3.4}$$

$$\sum_{\substack{i \in N \cup O \\ i \neq j}} x_{ijk} \quad - \quad \sum_{\substack{i \in N \cup O \\ i \neq j}} x_{jik} \geq 0, \forall j \in N, \forall k \in O \tag{3.5}$$

$$\sum_{j \in N} x_{ijk} \leq 1, onde\ i = o_k, \forall k \in S \tag{3.6}$$

$$\sum_{i \in O} \sum_{j \in N} x_{ijk} \leq 1, \forall k \in S \tag{3.7}$$

$$\sum_{\substack{i \in O \\ i \neq o_k}} \sum_{j \in N} x_{ijk} = 0, \ \forall k \in S \tag{3.8}$$

$$(y_i + t_{ij} x_{ijk} + ts_j x_{ijk}) - y_j \leq (1 - x_{ijk}) M, \quad \forall i \in N \cup O, \forall j \in N (i \neq j), \forall k \in S \ e\ onde\ M >>> 0\ (constante) \tag{3.9}$$

$$y_i \geq 0, \forall i \in N \tag{3.10}$$

$$x_{ijk} \in \{0, 1\}, \forall i \in N \cup O, \forall j \in N, \forall k \in S \tag{3.11}$$

Na formulação descrita por (3.1)-(3.11), a função objetivo descrita em (3.1) requer a minimização da quantidade de óleo que deixa de ser coletado no período. As restrições (3.2) exigem que todos os poços sejam atendidos dentro de seus respectivos limites de tempo de atendimento.

As restrições (3.3) exigem que cada poço j seja atendido uma única vez e por uma única sonda. As restrições (3.4) impõem que, para cada poço, exista no máximo uma

sonda partindo deste. As restrições (3.5) exigem que, se uma sonda chega a um poço p_i, somente ela sairá deste mesmo poço. As restrições (3.6) obrigam que cada sonda saia no máximo uma vez de sua origem. As restrições (3.7) exigem que cada sonda saia da origem no máximo uma vez para atender um poço. As igualdades (3.8) exigem que uma sonda só possa sair de sua origem correspondente. As inequações (3.9) indicam que se um poço p_j é atendido após um poço p_i, pela mesma sonda, o tempo y_j deve ser maior ou igual ao tempo y_i somado ao tempo de viagem t_{ij} e ao tempo de atendimento ts_j, sendo M é uma constante com valor arbitrariamente alto. Finalmente, (3.10) e (3.11) são as restrições de integralidade e não negatividade referentes às variáveis do problema.

Esta formulação foi utilizada neste trabalho na resolução de pequenas instâncias do problema através do software CPLEX. Esta formulação foi utilizada neste trabalho na resolução de pequenas instâncias do problema através do software CPLEX.

Capítulo 4

Algoritmos Propostos Para o PRSSM

Pelo fato de o PRSSM ser um problema da classe NP-difícil, a tendência é que métodos exatos não consigam resolvê-lo para a maioria das instâncias de maior porte ou, em alguns casos, não consigam resolvê-lo em tempo satisfatório. Devido a este fato, foram propostas algumas heurísticas para serem usadas isoladamente ou para serem incorporadas a diferentes metaheurísticas na a resolução do problema. Metaheurísticas não são métodos exatos, ou seja, não dão garantia de encontrar uma solução ótima para o problema, mas freqüentemente apresentam resultados de boa qualidade em um tempo computacional muito inferior ao dos métodos exatos.

Neste contexto, para alcançar estas metas, uma tendência que tem trazido bons resultados é o uso de algum tipo de memória nos algoritmos heurísticos, como já ocorre, por exemplo, nas heurísticas de Busca Tabu [16, 34] e em algumas versões de metaheurísticas híbridas [35, 36].

De uma forma geral, as técnicas de memória adaptativa consistem em fazer uso de memória para armazenar informações relevantes sobre soluções ou regiões já pesquisadas no espaço de busca e utilizar essas informações para direcionar uma estratégia de busca nos passos seguintes, objetivando por exemplo: evitar percorrer novamente as soluções pesquisadas anteriormente, retomar a busca nas proximidades de pontos já pesquisados, combinar características de diferentes soluções já analisadas, ou ainda uma combinação e/ou permutação destas estratégias.

O objetivo maior em incorporar memória numa metaheurística é obter soluções de melhor qualidade a partir de informações dos pontos do espaço de busca já visitados. Assim, escolhida uma estratégia para uso da memória, deve-se selecionar estratégias para gerar novas soluções fazendo uso desta memória. As técnicas de geração de soluções variam

de acordo com a estratégia adotada. Por exemplo, se a técnica escolhida for combinar elementos de boas soluções anteriores, podemos utilizar a construção de vocabulário[16] como técnica de geração de novas soluções. Caso o objetivo seja intensificar a busca ao redor de determinadas soluções, podemos usar o conceito de Reconexão de Caminhos [16]. Agora, se a estratégia adotada for de evitar repetir a busca em espaços já vasculhados, uma técnica de construção gulosa aleatória, penalizando os elementos comuns entre as soluções encontradas, pode ser utilizada.

Por fim, estratégias para atualizar a estrutura de memória devem ser escolhidas. Para isso, deve-se levar em consideração o uso de informações das novas soluções geradas, e também o fato de que as informações contidas na estrutura de memória podem estar desatualizadas. Às vezes é conveniente desconsiderar informações menos recentes. Em outros casos, manter informação sobre todo o espaço já verificado é extremamente necessário. Todas estas questões devem ser levadas em conta no momento de atualizar a estrutura de memória utilizada.

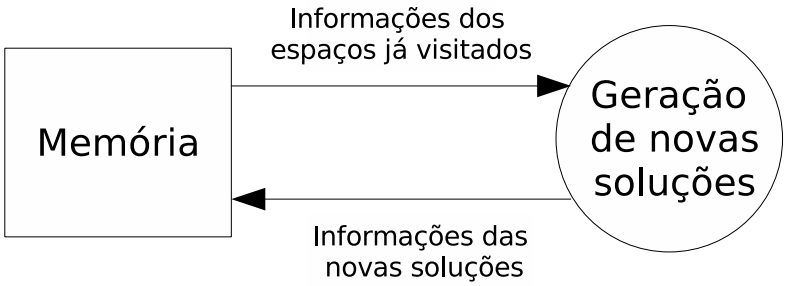


Figura 4.1: Memória Adaptativa

A Figura 4.1 mostra, resumidamente, o esquema de funcionamento das técnicas de memória adaptativa. Nela podemos ver informações contidas na memória sendo usadas na construção de novas soluções, bem como informações das soluções geradas sendo usadas para atualizar a memória.

As técnicas de memória adaptativa normalmente possuem uma descrição muito sucinta, porém, seu uso ideal na prática não é trivial. Questões como: que atributos das soluções deve-se armazenar, que técnicas utilizar para combiná-los, por quanto tempo um atributo deve ser levado em consideração ainda são temas polêmicos e normalmente tratadas caso a caso [16, 35, 17, 27, 36].

Existem duas grandes vertentes sobre uso de memória adaptativa em metaheurísticas: A proposta por Glover [16], e a proposta por Taillard [35]. A primeira é mais específica

para Busca Tabu e encontra-se a mais tempo em discussão estando, portanto, mais amadurecida. Já a segunda, é mais genérica e pode ser aplicada a diversas metaheurísticas, porém, por ser mais recente, ainda é pouco aplicada.

4.1 Memória Adaptativa segundo Glover

Para Fred Glover, a técnica de memória adaptativa é parte do *framework* do algoritmo de Busca Tabu (BT) [16]. Segundo ele, a BT popularizada não é mais do que uma reduzida porção do método que possui originalmente, além da BT popular, uma memória de longo prazo e processos de diversificação e intensificação associados.

Ainda segundo Glover, quatro fatores são essenciais para a técnica de memória adaptativa: o quão recente[1] o atributo é, freqüência, qualidade e influência.

O *quão recente* um elemento é refere-se à memória de curto prazo da BT. Relaciona-se com atributos modificados recentemente e que passam a ser considerados tabu por um período de tempo.

A *freqüência*, por sua vez, está associada à memória de longo prazo. Normalmente vistas como frações, onde o numerador costuma ser o número de vezes que um evento ocorre ou o número de vezes que um atributo aparece nas soluções já visitadas. Os denominadores, geralmente, são o número total de eventos ocorridos representados pelo numerador ou um valor máximo assumido pelo numerador.

Estes dois primeiros fatores se complementam, fazendo com que a BT tenha uma visão dos espaços de busca já visitado a curto e a longo prazo.

A *qualidade* se refere à habilidade de diferenciar o mérito de soluções visitadas durante a busca. Na prática, a qualidade se torna um fundamento para o aprendizado baseado em incentivo. Prêmios são dados para estimular ações que levem a boas soluções e penalidades são dadas para desencorajar ações que levem a soluções ruins. A qualidade também pode ser um guia para restringir o espaço de busca.

A *influência* considera o impacto das escolhas feitas durante a busca, não apenas na qualidade, mas também na estrutura. Em um algoritmo *branch-and-bound* por exemplo, as regras de separação são pré-especificadas e as direções de divisão fixas. Porém, é óbvio que algumas decisões têm mais influência do que outras, por exemplo, que tipos de movimento de vizinhança utilizar e em que ordem utilizá-los (em um *branch-and-bound*

[1] o termo original descrito pelo autor é *recency*

que usa uma estratégia de busca em profundidade, o ramo da árvore a ser vasculhado primeiro tem uma grande influência).

4.1.1 Fazendo uso da memória adaptativa

Trabalhos relacionados às idéias de Glover seguem quase sempre os mesmos padrões de uso para memória adaptativa. A técnica utilizada depende muito do tipo de estrutura de memória adotada. Neste trabalho serão mostradas as técnicas para duas estruturas de memória: *conjunto elite* e *matriz de freqüências*.

O *conjunto elite* é nada mais do que um conjunto de boas soluções distintas encontradas ao longo do processo de busca. Fatores como o tamanho deste conjunto e a semelhança entre as soluções presentes nele são importantes e não podem ser deixados de lado durante a implementação.

Uma *matriz de freqüências* é simplesmente uma matriz que contém as freqüências de aspectos relacionados ao problema. Tais aspectos podem ser desde o número de vezes que um evento ocorre em uma solução até o número de vezes que um determinado atributo aparece nas soluções encontradas durante a busca.

Para o *conjunto elite*, duas técnicas têm sido mais exploradas: i) a descoberta de padrões (feita por exemplo com Mineração de Dados (*Data Mining* [31, 32]) ou Construção de Vocabulário (*Vocabulary Building*) [16]) e ii) a Reconexão de Caminhos (*Path Relinking*)[16].

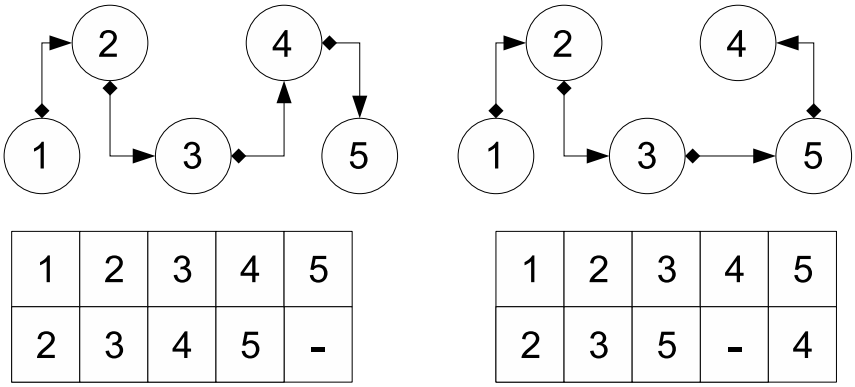
Já para matriz de freqüências, as duas técnicas mais utilizadas são a mineração de dados e a Construção de Vocabulário.

A *Mineração de Dados*, no contexto de módulo de uma metaheurística [31, 32], visa a obtenção de tendências, padrões e correlações em uma base de dados, no caso, um conjunto elite. Vários tipos de conhecimento podem ser descobertos, dentre eles podemos citar regras de associação (elas representam um padrão de relacionamento entre os dados), regras de classificação (uma hierarquia entre as várias regras presentes em um sistema), padrões de seqüências (seqüências de ações, eventos ou atributos), padrões em séries temporais (padrões que se repetem em determinadas posições de uma série temporal) e agrupamento (um conjunto de itens ou eventos pode ser dividido em subconjuntos de elementos similares).

A *Construção de Vocabulário* (*Vocabulary Building*) foi proposta por Glover [16] e

consiste em unir fragmentos (palavras) contidos em várias soluções de boa qualidade, tentando construir soluções melhores a partir destes fragmentos.

A Figura 4.2 ilustra um exemplo de duas soluções que serão submetidas à construção de vocabulário. As soluções, neste exemplo, são grafos acíclicos direcionados que são representados pelo vetor de adjacência logo abaixo delas.



1	2	3	4	5
2	3	4	5	-

1	2	3	4	5
2	3	5	-	4

Figura 4.2: Soluções a serem submetidas à Construção de Vocabulário

Já a Figura 4.3 ilustra o processo de busca por palavras. Neste caso, foi ilustrada a busca por fragmentos comuns entre as soluções. De uma forma geral, para que boas palavras sejam extraídas, é necessário que as soluções utilizadas no processo sejam de boa qualidade, uma vez que, caso uma das soluções não seja de boa qualidade, um número muito pequeno de palavras (elementos comuns) pode ser encontrado, dada a diferença entre as soluções, e, caso ambas sejam de má qualidade, as palavras extraídas serão ruins porque representarão semelhanças entre soluções ruins. Estas palavras ruins não são interessantes para a técnica de Construção de Vocabulário, uma vez que, a metodologia desta técnica é construir soluções melhores, a partir de palavras encontradas nas boas soluções. As palavras extraídas durante o processo de busca por palavras são armazenadas para uso em uma fase posterior da técnica de Construção de Vocabulário que é a construção de frases.

A Figura 4.4 mostra a construção de frases. Nesta etapa, é feita uma junção das palavras encontradas na etapa de busca por palavras. O objetivo desta fase é juntar fragmentos de boas soluções para construir novas soluções. No exemplo dado, a junção de palavras resultou em uma solução completa e viável, mas isto nem sempre acontece, e, freqüentemente, é necessário usar um procedimento de correção para transformar a solução produzida (frase) em uma solução viável.

A Reconexão de Caminhos (RC) [16] pode ser vista como uma estratégia de busca in-

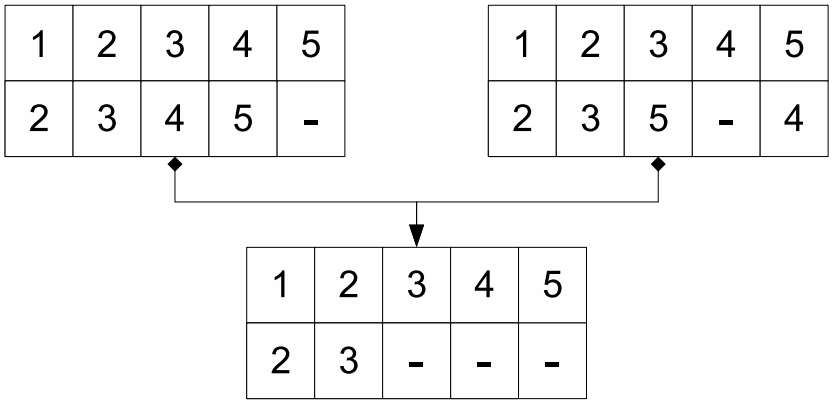


Figura 4.3: Busca por Palavras

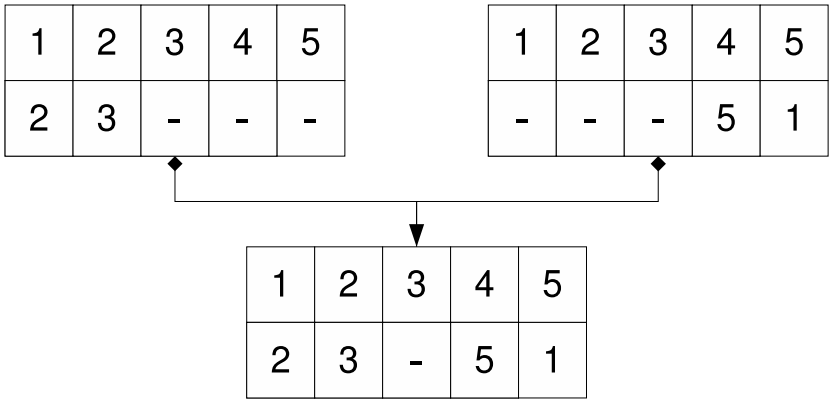


Figura 4.4: Construção de Frases

tensiva. Ela explora a vizinhança entre duas soluções extremas de boa qualidade. Temos uma solução base e uma solução guia. O objetivo é tentar encontrar uma solução intermediária de melhor qualidade percorrendo o caminho entre as duas soluções extremas. A idéia é usar duas boas soluções, pois, a justificativa neste caso, é que no caminho entre duas boas soluções pode haver uma outra (ou outras) de melhor qualidade.

A técnica de RC funciona da seguinte maneira: inicialmente calculam-se as diferenças entre a solução base e a solução guia. Para cada uma destas diferenças, será gerada uma solução candidata. Suponha o conjunto $D = \{d_1, d_2, ..., d_n\}$ o conjunto de diferenças entre as soluções base e guia. Para cada $d_i \in D$, será gerada uma solução candidata s_i idêntica à solução base, porém, s_i ão conterá a diferença d_i, fato este que torna s_i um pouco mais semelhante à solução guia. Com isso, podemos definir o conjunto de diferenças entre a solução candidata s_i e a solução guia, denotado por D_i, como $D_i = D - \{d_i\}$. Todas as soluções candidatas são avaliadas, e aquela que possui o melhor custo é escolhida como nova solução intermediária, e será usada como se fosse a solução base na próxima iteração.

Este processo é repetido até que não haja diferenças entre a solução base e a solução guia.

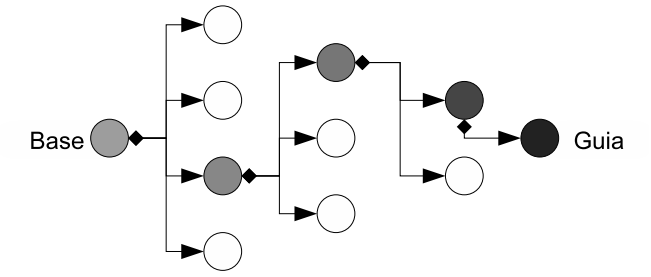


Figura 4.5: Reconexão de Caminhos

A RC executa uma busca mais intensa na escolha das primeiras soluções intermediárias. Como as soluções intermediárias vão se tornando mais semelhantes à solução guia ao longo do procedimento, no inicio da RC há um número maior de diferenças entre as soluções intermediárias e a solução guia, o que faz com que uma busca mais intensa seja feita na escolha das primeiras soluções intermediárias.

Pode ser feito o uso de busca local nas soluções intermediárias para melhorar a qualidade da solução retornada pela RC, contudo as soluções resultantes da aplicação de busca local devem ser tratadas separadamente, pois, usá-las como solução intermediária da RC pode fazer com que a RC não atinja a solução guia.

Outro fator a ser considerado é o sentido em que a RC é aplicada. Esta técnica pode ser aplicada no sentido da pior solução para a melhor, no sentido da melhor para pior, ou ainda nos dois sentidos simultaneamente. É evidente que, quando aplicada nos dois sentidos, a RC tende a conseguir resultados melhores do que quando aplicada em um único sentido. Contudo, aplicar esta técnica nos dois sentidos pode ser muito oneroso em termos computacionais. Quando a execução da RC nos dois sentidos é inviável, a RC deve, preferencialmente, ser aplicada no sentido da melhor solução para a pior, visto que ela faz uma busca mais intensiva na escolha das primeiras soluções intermediárias.

A Figura 4.5 ilustra graficamente como o caminho de soluções percorrido pelo processo de reconexão por caminhos funciona.

4.2 Memória Adaptativa segundo Taillard

Taillard *et al.* [35] propõem uma versão mais genérica para a técnica de memória adaptativa. A princípio, a versão de Taillard pode ser aplicada a qualquer metaheurística.

Para os autores, a técnica de memória adaptativa é composta de quatro elementos: (1) um conjunto de soluções ou uma estrutura de dados especial que agrega as particularidades das soluções produzidas pela busca. (2) Uma solução provisória construída usando os dados na memória. (3) Um método para tentar melhorar a solução provisória fazendo uso de um algoritmo de busca local ou uma metaheurística mais sofisticada. (4) Um modo de fazer com que a nova solução seja incluída na memória ou usada para atualizar a estrutura de dados que armazena o histórico da busca.

Ainda, segundo os autores, várias metaheurísticas incorporam partes dessa técnica, quando não, ela toda, como apresentadas a seguir:

Algoritmos Genéticos (*Genetic Algorithms*)[20]

- Memória: população de soluções;
- Memória na construção de novos indivíduos: operador *crossover*;
- Busca local: algumas versões de Algoritmos Genéticos (chamados de Algoritmos Meméticos) fazem uso de um procedimento de busca local nos novos indivíduos gerados;
- Uso das soluções geradas para atualizar a memória: reposição da população;

Busca Tabu (*Tabu Search*)[16]

- Memória: Lista Tabu, Conjunto Elite, matriz de freqüências, etc.
- Memória na construção de novas soluções: uso da lista tabu para evitar ciclagem (busca em áreas já visitadas), construção de novas soluções semelhantes (intensificação) ou não (diversificação) à(s) melhor(res) solução(ões).
- Busca local: busca na vizinhança.
- Uso das soluções geradas para atualizar a memória: reposição do(s) melhor(es) indivíduo(s) no conjunto elite; atributos das soluções geradas armazenados na lista tabu, etc.

Colônia de formigas (*Ant Colony Systems*)[11]

- Memória: armazenamento das trilhas de feromônio.

- Memória na construção de novos indivíduos: constrói uma nova solução utilizando as trilhas armazenadas.
- Busca local: qualquer método.
- Uso das soluções geradas para atualizar a memória: atualização das trilhas de feromônio.

Como mostrado, a técnica de memória adaptativa proposta por Taillard é aplicável a diversas metaheurísticas e este é, sem dúvida, seu principal mérito.

Existem ainda poucos trabalhos relacionados que abordam a versão de memória adaptativa proposta por Taillard *et al.* [35]. Um dos que mais merece destaque é o trabalho de Kadluczka *et al.* [21], que constrói um *framework* usando esta idéia. Esta linha de pesquisa, a de construção de *frameworks* que incorporam a versão de Taillard para memória adaptativa, parece ser promissora e poderá render bons trabalhos no futuro.

Algumas metaheurísticas, como as versões iniciais do GRASP [30], não fazem uso de nenhum tipo de memória entre suas iterações. Atualmente, já existem propostas para tornar as iterações GRASP adaptativas como ocorre nas iterações de sua fase de construção [12, 27, 28].Algumas destas idéias incluem os conceitos de: i) atualização dinâmica do parâmetro α na etapa de construção (GRASP Reativo) [12, 27]; ii) a inclusão de um módulo de Reconexão de Caminhos fazendo uso de um conjunto elite [41].

Neste contexto, colocamos como objetivo deste trabalho propor diferentes heurísticas adaptativas para a solução do PRSSM e efetuar comparações entre elas e metaheurísticas que não fazem uso de memória, como é o caso do GRASP tradicional.

4.3 Propostas

Para resolver o PRSSM foram propostas quatro heurísticas. Um GRASP tradicional e um GRASP com memória adaptativa (GRASP+MA), baseados nas propostas apresentadas por Trindade em [41], uma Busca Tabu e uma Busca Local Iterada (BLI).

4.3.1 GRASP e GRASP+MA

Para avaliar a eficiência das técnicas de memória adaptativa em metaheurísticas foram construídas inicialmente duas heurísticas, sendo um GRASP tradicional (GRASP Puro)

e uma versão que faz uso de memória chamado de GRASP+MA (GRASP + Memória Adaptativa). Ambos utilizam o mesmo método construtivo e a mesma busca local.

Trindade, em [41], abordou um problema similar ao PRSSM e propôs dois algoritmos construtivos e 3 buscas locais para o problema abordado. Com os algoritmos construtivos e buscas locais propostas, foram construídas seis versões GRASP (G1, G2, G3, G4, G5, G6), uma para cada combinação de algoritmo construtivo e busca local.

Nesse trabalho, foram implementadas e testadas todas as combinações propostas por Trindade, devidamente adaptadas para o PRSSM. Dentre todas as combinações, a que apresentou os melhores resultados empíricos foi a combinação G5. Esta combinação é constituída pelo algoritmo construtivo chamado C1 e pela busca local chamada BL3, ambos serão descritos a seguir.

O construtivo C1 [41], descrito no Algoritmo 1, é baseado no conceito de inserção do vizinho mais próximo. Para isso, inicialmente é definida uma fórmula para determinar o nível de prioridade de cada poço no tocante ao seu atendimento. Esta prioridade é calculada da seguinte maneira: suponha uma rota r em construção, uma sonda s_k para atendê-la e o último poço p_i pertencente à rota em construção. A prioridade de um poço p_j é dada pela equação $prioridade(p_j) = v_j/(t_{ij} + ts_j)$, onde v_j é a vazão do poço p_j, t_{ij} o tempo de viagem do último poço p_i pertencente à rota até p_j e ts_j o tempo de serviço de p_j. Este critério de prioridade é interessante, pois leva em consideração, tanto a vazão do poço candidato a ser inserido como o tempo de percurso entre o poço a ser inserido e último poço pertencente à rota parcial. Este algoritmo construtivo se mostrou mais adequado para o PRSSM do que o construtivo C2, também apresentado em [41]. O construtivo C2 leva em conta somente a vazão dos poços no momento de definir a prioridade, desprezando o tempo de viagem entre os poços.

O algoritmo C1, a cada passo, insere alternadamente um poço em cada sonda, construindo assim as rotas de maneira gradativa, procurando gerar rotas balanceadas no tocante à quantidade de poços alocados em cada rota (sonda).

A busca local utilizada em ambas as versões do GRASP (puro e com memória adaptativa) é uma variação da busca local BL3, também apresentada em [41]. Esta busca local é composta de duas buscas locais, uma de realocação de poços dentro da mesma sonda (Intra-Rota) e uma realocação de poços para sondas diferentes (Inter-Rota). A BL3 é uma busca gulosa e exaustiva, logo, a cada iteração, os poços sempre são colocados na melhor posição possível.

Algoritmo 1 Procedimento C1 ()

```
1: Solução saída = nova Solução
2: índiceSonda = 0
3: enquanto houver poços não atendidos faça
4:    Lista lis = construirLRC(α, saída, índiceSonda)
5:    Poço p = sorteio(lis);
6:    inserirPoçoNoFinalDaRota(p, índiceSonda, saída)
7:    índiceSonda = (índiceSonda+1) mod numTotalDeSondas
8: fim enquanto
9: retorne saída
```

Primeiramente, a BL3 tenta realocar os poços dentro das próprias sondas, tentando encontrar um melhor arranjo dos poços para cada sonda. O procedimento que implementa esta busca local é ilustrado pelo Algoritmo 2. Feito isso, tenta-se uma realocação de poços para sondas diferentes da sonda original. Testa-se a inserção de todos os poços em todas as posições das rotas de todas as sondas e a melhor posição é escolhida para inserção do poço. O pseudocódigo que implementa esta busca local é apresentado pelo Algoritmo 3. A busca local BL3 executa sucessivamente estas duas buscas locais a cada iteração, e pára quando a busca local Inter-Rota não consegue melhorar a solução incumbente.

Algoritmo 2 Procedimento Intra-Rota (Solução entrada)

```
 1: para todo rota pertencente ao conjunto de rotas de entrada faça
 2:    rotaBase = copia(rota)
 3:    rotaMelhor = copia(rota)
 4:    para  i variando de 1 até tamanho de rota faça
 5:       poçoAtual = rota[i]
 6:       para j variando de 1 até tamanho de rota faça
 7:          rotaAux = inserePoçoPosição(rotaBase, poçatual, j)
 8:          se rotaAux é melhor que rotaMelhor então
 9:             rotaMelhor = rotaAux
10:          fim se
11:       fim para
12:       rotaBase = rotaMelhor
13:    fim para
14:    substituirRota(rota, rotaBase, entrada)
15: fim para
```

A busca local utilizada neste trabalho, chamada BL4, utiliza também as duas buscas locais utilizadas pela BL3 (Intra-Rota e Inter-Rota). A BL4 difere da BL3 somente no critério de parada. Enquanto na busca local BL3 o algoritmo pára quando a busca local Inter-Rota não consegue melhorar a solução incumbente, a BL4 só pára quando nenhuma das duas buscas locais consegue melhorar a solução incumbente. Ou seja, A busca local BL4 só pára quando, nem a busca Intra-Rota nem a busca Inter-Rota conseguem encontrar

Algoritmo 3 Procedimento Inter-Rota (Solução entrada)

```
1: soluçãoBase = copia(entrada)
2: soluçãoMelhor = copia(entrada)
3: para i variando de 1 até o número de sondas faça
4:    para j variando de 1 até o número de poços da sonda i faça
5:       poçoAtual = sol.sonda(i).poço(j)
6:       para k variando de 1 até o número de sondas faça
7:          se  i ≠ k então
8:             para l variando de 1 até a o número de poços da sonda k +1 faça
9:                soluçãoAux = realocaPoço(entrada, i, j, k, l)
10:               se soluçãoAux é melhor que soluçãoMelhor então
11:                  soluçãoMelhor = soluçãoAux
12:               fim se
13:            fim para
14:         fim se
15:      fim para
16:      soluçãoBase = soluçãoMelhor
17:   fim para
18: fim para
19: entrada = soluçãoBase
```

uma solução melhor. Testes experimentais revelaram que esta simples diferença pode acarretar melhora em cerca de 40% dos casos. O pseudocódigo da busca local BL4 é apresentado no Algoritmo 4.

Algoritmo 4 Procedimento BL4 (Solução entrada)

```
1: valor = 0
2: enquanto valor ≠ custo(entrada) faça
3:    valor = custo(entrada)
4:    V1(entrada)
5:    V2(entrada)
6: fim enquanto
```

Duas versões da metaheurística GRASP foram implementadas utilizando o algoritmo construtivo C1 e a busca local BL4, um GRASP tradicional [30](denotado a partir de agora simplesmente por GRASP)e um GRASP que faz uso de memória (denotado a partir de agora por GRASP+MA), proposto neste trabalho.

A heurística GRASP, descrita pelo Algoritmo 5, funciona da seguinte maneira: primeiramente, uma solução é criada pelo algoritmo construtivo C1. Em seguida, é aplicada a busca local BL4 na solução gerada pelo construtivo. O GRASP repete estes dois passos até que um critério de parada estabelecido seja atingido. Ao final de sua execução, o GRASP retorna a melhor solução encontrada.

O GRASP+MA, que é mostrado no Algoritmo 9, faz uso de uma série de técnicas bem conhecidas de memória adaptativa, que serão citadas e explicadas a seguir.

Algoritmo 5 Procedimento GRASP()

```
1: Solução melhor = C1()
2: enquanto critério de parada não atingido faça
3:    Solução aux = C1()
4:    BL4(aux)
5:    se custo(aux) melhor que custo(melhor) então
6:       melhor = aux
7:    fim se
8: fim enquanto
9: retorne melhor
```

Auto configuração do parâmetro α (coeficiente de aleatoriedade do algoritmo construtivo do GRASP): um dos problemas fundamentais de sintonia fina do GRASP é a calibração do α [27, 30, 28]. Para cada instância é preciso encontrar um valor de α que produza boas soluções mas que também possibilite ao algoritmo não ficar preso em ótimos locais ainda distantes de um ótimo global. Para resolver este problema, foi proposto um modelo de GRASP reativo, que configura automaticamente o valor deste parâmetro. Esta configuração é feita de acordo com a instância trabalhada a partir de uma fase de treinamento com diferentes valores para α. Ao final deste treinamento, fixa-se o valor que retornou a melhor solução, e, no restante do tempo (ou nas iterações remanescentes), o GRASP será executado com este valor α, agora fixado. Neste trabalho, o valor de α escolhido é aquele que gerou a melhor solução durante a fase de treinamento. Esta configuração é feita pelo procedimento encontrarMelhorAlpha(), mostrado na linha 1 do algoritmo GRASP+MA.

Conjunto Elite e Reconexão de Caminhos: A Reconexão de Caminhos (RC), foi proposta inicialmente para a Busca Tabu e *Scatter Search* por Glover e Laguna [16, 17]. Como dito anteriormente, ela utiliza duas soluções extremas de boa qualidade (base e guia), para tentar encontrar uma solução intermediária que seja melhor que as duas soluções da extremidade. A idéia é fazer com que, passo a passo, a solução base fique mais parecida com a solução guia ou solução alvo. O conjunto elite é usado como repositório de soluções base e/ou guia, às quais a RC será aplicada. Esta técnica pode ser aplicada no sentido da solução de melhor qualidade para a de pior qualidade, no sentido da pior para a melhor, ou ainda nos dois sentidos simultaneamente.

Trindade, em [41], também propõe um modelo de RC para uma versão do PRSSM. Nesse modelo, a solução base é modificada de forma a tornar na iteração i da RC, a i-

ésima posição de cada rota da solução base igual à i-ésima posição de cada rota da solução guia. Desta forma, na iteração 1 todas as rotas da solução base têm sua primeira posição modificada, de forma que elas fiquem iguais à primeira posição das rotas da solução guia. Esta solução com a primeira posição de cada rota alterada é tomada como nova solução base. Na iteração 2 todas as rotas da solução base têm sua segunda posição modificada e assim sucessivamente. A solução é modificada coluna por coluna, fazendo uma caminhada linear entre a solução base e a solução guia. A cada passo da RC a busca local BL3 é utilizada para tentar melhorar a solução.

A RC apresentada por Trindade [41] difere da RC clássica. O modelo clássico executa uma busca mais intensa na escolha das primeiras soluções intermediárias. A retirada de cada uma das diferenças entre as soluções base e guia é avaliada e aquela que resulta em uma melhor solução é implementada. A solução resultante dessa implementação é tomada como nova solução intermediária. Ao longo do procedimento, como as soluções base e guia vão se tornando mais semelhantes, há, no inicio da RC, um número maior de diferenças entre as soluções base e guia, o que faz com que uma busca mais intensa seja feita na escolha das primeiras soluções intermediárias. Por outro lado, no modelo apresentado por Trindade a mesma quantidade de trabalho é feita a cada iteração da RC.

Estes dois modelos, o clássico, proposto por Glover [16] e o proposto por Trindade [41], foram implementados e em testes preliminares constatou-se que a versão clássica apresenta melhor desempenho médio. Este fato fez com que somente o modelo clássico de Reconexão por Caminhos fosse utilizado durante os testes com as instâncias.

Associada à RC, está a busca local BL4 que é aplicada somente na melhor solução intermediária.

Também foram feitos testes experimentais com a RC implementada para o PRSSM, no sentido da pior para melhor (1), no sentido da melhor para pior (2) e nos dois sentidos simultaneamente (3). Entre os sentidos 1 e 2 da RC, o sentido 2 obteve melhores resultados na grande maioria os testes, como já era esperado [16]. Na comparação do sentido 2 com o sentido 3, os resultados foram muito parecidos, com ligeira vantagem para o sentido 3. Porém, o tempo de execução da opção 3 é muito maior que o do sentido 2. Por isso, optou-se por usar somente o sentido 2 durante os demais experimentos.

Neste trabalho, a RC é utilizada de maneira hierárquica. Como o tempo de execução foi escolhido como critério de parada para as heurísticas aqui propostas, depois de transcorrido 90% do tempo de execução, as soluções produzidas pelas iterações GRASP são submetidas à RC com cada um dos membros presentes no conjunto elite no sentido 2(a

solução base será a de melhor qualidade e a guia a solução de pior qualidade). Durante esse procedimento, o conjunto elite pode ser modificado caso uma solução melhor que a pior solução elite for encontrada. Caso isto ocorra, ao final das iterações GRASP, a RC é novamente ativada, agora entre cada par de soluções deste conjunto, no sentido da melhor para a pior.

Fatores como a dimensão do conjunto elite e a diversidade das soluções contidas nele são importantes para um bom desempenho destas técnicas. Um número grande de soluções elites pode fazer com que a execução da RC se torne muito demorada e um número pequeno pode fazer com que a RC não encontre soluções de melhor qualidade. No que tange a diversidade das soluções, é importante lembrar que se duas soluções semelhantes são passadas à RC, poucos passos (poucas soluções intermediárias) serão feitos para que uma se torne igual à outra. Isto pode fazer com que a RC tenha menos chance de melhorar as soluções extremas. O ideal é que ambas as soluções extremas dadas como entrada sejam de boa qualidade, mas que tenham um certo grau de diversidade entre si. Neste trabalho, o tamanho do conjunto elite utilizado após testes preliminares foi fixado em 10 (dez) soluções e a diversidade mínima entre soluções de 6 (seis) componentes. A diferença calculada em números de componentes (variáveis de uma solução) foi preferida à opção de calcular a diferença somente a partir do valor da função objetivo, uma vez que soluções com custos bem diferentes podem ter estruturas muito semelhantes e, reciprocamente, soluções com custos semelhantes podem ter estruturas bem diferentes. Para calcular esta diferença faz-se uma varredura na solução e conta-se o número de vezes em que poços aparecem em locais diferentes.

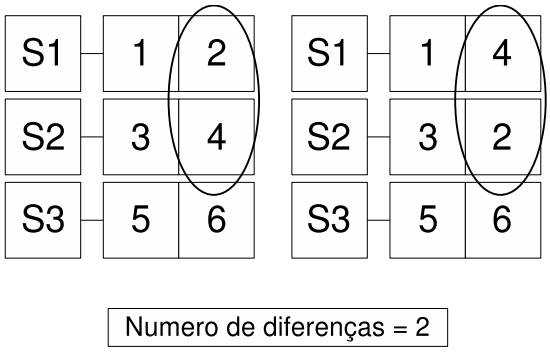


Figura 4.6: Contagem da diferença entre duas soluções

A Figura 4.6 ilustra duas soluções que possuem uma diferença de duas unidades entre si. Estas diferenças se encontram na primeira e na segunda rota (sondas S1 e S2).

Algoritmo 6 Procedimento executaRC(Solução sol)

```
1: para todo solução s no conjunto elite faça
2:    Solução aux
3:    se s melhor que sol então
4:       aux = reconexãoCaminhos(s, sol)
5:    se não
6:       aux = reconexãoCaminhos(sol, s)
7:    fim se
8:    tentarInserirConjuntoElite(aux)
9: fim para
10: se conjunto elite foi modificado então
11:    conjuntoEliteRC()
12: fim se
```

Algoritmo 7 Procedimento reconexãoCaminhos(Solução base, Solução guia)

```
1: Solução baseInterna = copia(base)
2: Solução retorno = copia(base)
3: enquanto baseInterna for diferente de guia faça
4:    Lista lis = computarDiferenças(baseInterna, guia)
5:    Solução melhorIntermediaria
6:    para i variando de 1 até tamanho(lis) faça
7:       Solução aux = desfazDiferença(baseInterna, guia, lista[i])
8:       se aux melhor que melhorIntermediaria então
9:          melhorIntemediaria = aux
10:      fim se
11:   fim para
12:   BL4(melhorIntermediaria)
13:   se melhorIntermediaria melhor que retorno então
14:      retorno = melhorIntermediaria
15:   fim se
16: fim enquanto
17: retorne retorno
```

Algoritmo 8 Procedimento conjuntoEliteRC()

```
1: continuar = verdadeiro
2: enquanto continuar for verdadeiro faça
3:   para todo Solução s1 no conjunto elite faça
4:     para todo Solução s2 no conjunto elite tal que s1 ≠ s2 faça
5:       Solução aux
6:       se s1 for melhor que s2 então
7:         aux = reconexãoCaminhos(s1, s2)
8:       se não
9:         aux = reconexãoCaminhos(s2, s1)
10:      fim se
11:      tentarInserirConjuntoElite(aux)
12:    fim para
13:  fim para
14:  se conjunto elite nao foi modificado então
15:    continuar = falso
16:  fim se
17: fim enquanto
```

Diversificação Reativo+Conjunto Elite: Também foi implementada uma estratégia simples de diversificação que é a variação do parâmetro α em conjunto com a atualização do conjunto elite durante a execução. Esta variação é feita da seguinte maneira: durante a busca, é feita uma medição de quanto tempo o conjunto elite ficou estático (sem modificação). Toda vez que este tempo atinge um valor t (parâmetro de entrada), o valor de α é aumentado em 0.1, tornando o método construtivo menos guloso. Quando o Conjunto elite é atualizado, α retorna ao valor original que possuía antes da primeira modificação. Esta estratégia tem como objetivo fazer com que o GRASP+MA tente escapar com mais facilidade de ótimos locais ainda distantes de um ótimo global, uma vez que, usando por muito tempo o mesmo valor para o parâmetro α, o algoritmo construtivo tende a produzir mais soluções similares.

4.3.2 Busca Tabu

Além de heurísticas GRASP, são propostos outros algoritmos heurísticos que fazem uso de memória do tipo Busca Tabu e Busca Local Iterada para o PRSSM.

A heurística Busca Tabu implementada para a resolução do PRSSM é baseada na proposta por Taillard *et al.*, em [34], para o PRV. Nesta Busca Tabu, uma estrutura de dados contendo rotas de soluções geradas anteriormente é explorada na construção de novas soluções que são melhoradas com a Busca Tabu e então utilizadas para atualizar a estrutura de dados.

Algoritmo 9 procedimento GRASP+MA(tempo t)

```
1: encontrarMelhorAlpha()
2: enquanto critério de parada não atingido faça
3:    Solução aux = C1()
4:    BL4(aux)
5:    tentarInserirConjuntoElite(aux)
6:    se tempoSemMelhora > t então
7:       aumentar o valor de α
8:    fim se
9:    se condição da Reconexão por Caminhos foi atingida então
10:      executaRC(aux)
11:   fim se
12: fim enquanto
13: retorne melhor solução do conjunto elite
```

Primeiramente, um conjunto inicial de soluções é gerado utilizando o método construtivo C1, também utilizado no GRASP e no GRASP+MA. Como as soluções produzidas por este construtivo podem ser melhoradas, uma Busca Tabu básica é aplicada a cada uma das soluções iniciais para aprimorar este conjunto. Estas soluções melhoradas são armazenadas em uma estrutura de memória especial e, ao longo do procedimento, serão utilizadas como base para construção de novas soluções.

A estrutura de memória mencionada armazena o conjunto de rotas extraídas das melhores soluções encontradas durante a busca. Todas as rotas de uma solução são armazenadas contiguamente e as soluções são ordenadas pelo valor da função objetivo. Conseqüentemente, as rotas associadas à melhor solução se encontram nas primeiras posições da memória. Este conjunto é utilizado para construir soluções iniciais para a Busca Tabu.

Para gerar novas soluções, são escolhidas e agregadas algumas rotas presentes na memória gerando assim uma solução provisória. Como as rotas escolhidas têm uma grande possibilidade de não contemplarem todos os clientes, um procedimento de correção é utilizado para completar a solução provisória que, agora, será utilizada como nova solução para a Busca Tabu adaptativa.

Para escolher uma rota, primeiramente, é preciso escolher, entre as soluções presentes na memória, uma solução doadora, da qual será escolhida um rota para ser incorporada à solução em construção. Para escolher uma solução para ser doadora, foi utilizado o processo estocástico descrito em [34] que privilegia soluções de melhor qualidade. Para executar este processo primeiramente, ordenamos as soluções presentes no conjunto em ordem crescente da função objetivo, assim, as melhores soluções encontram-se nas pri-

meiras posições do conjunto. Em seguida, é atribuída à cada solução uma probabilidade de escolha, baseada na posição que a solução ocupa no conjunto ordenado. Esta probabilidade de escolha é dada da seguinte maneira: sendo T o conjunto de soluções, i a posição que uma solução ocupa neste conjunto, a probabilidade de escolha é dada pela seguinte equação: $probEscolha(i) = (2 \times (|T| - i + 1))/(|T| \times (|T| + 1))$, onde $|T|$ denota o tamanho do conjunto T. Uma vez escolhida a solução doadora, seguindo este processo, uma rota pertencente à esta solução é escolhida, aleatoriamente, e inserida na solução em construção.

Uma vez escolhida uma rota r, as demais rotas na memória (conjunto) que possuem um ou mais poços em comum com a rota escolhida são descartadas do processo de seleção. Recorde que no PRSSM as rotas possuem sondas associadas, uma vez que, uma mesma rota (seqüência de poços) possui vazões perdidas diferentes quando associada à sondas diferentes. Assim, todas as rotas atribuídas à sonda associada com a rota escolhida também são descartadas do processo de seleção. Em seguida, uma segunda rota é escolhida entre as rotas restantes na memória. Este processo é repetido até que o conjunto de rotas selecionadas cubra todos os clientes ou até não haver mais rotas disponíveis para seleção. Neste último caso, o procedimento para corrigir a solução (uma vez que, por não atender todos os poços, ela é inviável) é utilizado. Este procedimento de correção é simplesmente a mesma heurística de inserção que escolhe a posição de menor custo para cada poço. A ordem em que os poços são inseridos no procedimento de correção é aleatória.

Uma vez concluída a fase de geração de uma nova solução, uma Busca Tabu básica é aplicada à solução gerada. No final, a solução retornada por esta Busca Tabu básica é utilizada para atualizar a estrutura de memória.

Este ciclo de geração de uma nova solução, otimização através de uma Busca Tabu básica e atualização de memória se repete até que um critério de parada seja atingido.

Em [34], os autores ainda particionam o conjunto de clientes presentes na solução gerada a partir da memória em três sub-conjuntos, de acordo com sua disposição espacial em relação ao ponto de origem dos veículos. Com isso, os autores visam reduzir o tamanho do problema original. Após esta divisão, a Busca Tabu básica é aplicada à cada um dos subconjuntos, e, ao final deste processo, as soluções encontradas para os sub-conjuntos são justapostas, formando uma solução para o problema original.

Como existem vários pontos de origem de veículos (sondas) no PRSSM, optou-se por não utilizar o mecanismo de particionamento de clientes (poços) utilizado em [34].

Algoritmo 10 Procedimento BuscaTabuBásico(Solucao entrada)

```
Solução solucaoMelhor = copia(entrada)
Solucao atual = copia(melhor)
enquanto Critério de para não for atingido faça
  Solução aux = melhorVizinho(atual)
  se aux for melhor que solucaoMelhor então
    solucaoMelhor = aux
    atual = aux
  se não se aux não for tabu então
    atual = aux
  se não
    atual = melhorVizinhoNãoTabu(atual)
  fim se
  atualizarListaTabu(atual)
fim enquanto
retorne solucaoMelhor
```

Algoritmo 11 Procedimento Busca Tabu

```
Construir um conjunto de soluções usando o construtivo C1
Otimizar o conjunto utilizando BuscaTabuBásico
enquanto  Critério de parada não for atingido faça
  Solução aux = construirNovaSolucaoUsandoMemoria()
  corrigirSolucao(aux)
  aux = BuscaTabuBásico(aux)
  atualizarMemoria(aux)
fim enquanto
retorne a melhor solução do conjunto
```

4.3.3 Busca Local Iterada (*Iterated Local Serach*)

A idéia por trás de uma Busca Local Iterada (BLI) é construir um caminho aleatório no espaço de ótimos locais, definido como resultado da aplicação de um algoritmo de busca local [3]. Quatro componentes são necessários para a produção de um algoritmo BLI: Um procedimento para gerar uma solução inicial (1), uma busca local (2), um esquema de perturbação (3), e um critério de aceitação (4).

A eficiência dos quatro componentes é importante no desempenho final da BLI, porém, a eficiência da busca local é decisiva uma vez que ela influencia fortemente na qualidade das soluções e no tempo computacional. O esquema de perturbações deve permitir que a BLI escape de ótimos locais, mas, ao mesmo tempo, evitar as desvantagens de uma reinicialização do algoritmo. Percebe-se que a combinação do esquema de perturbação e do critério de aceitação influencia fortemente no tipo de caminho que se irá construir no espaço de busca de uma solução.

O problema de configuração da BLI aparece na forma de como encontrar as melhores estratégias para os quatro componentes. Devido à interação entre os componentes, este problema se torna difícil e deve também ser resolvido heuristicamente.

Algoritmo 12 Procedimento BLI()

```
1: Solução s_0 = gerarSoluçãoInicial()
2: solução s* = buscaLocal(s_0)
3: enquanto critério de parada não for atingido faça
4:    Solução s' = perturbação(s*, históricoDeBusca)
5:    Solução s*' = buscaLocal(s')
6:    s* = critérioDeAceitação(s*, s*', históricoDeBusca)
7: fim enquanto
8: retorne s*
```

O Algoritmo 12 mostra o pseudocódigo de um método BLI geral. Apesar de a BLI prever a execução de perturbações baseadas no histórico da busca, muitas vezes elas são feitas de maneira aleatória, dada a grande dificuldade de decisão sobre quais atributos do histórico de busca utilizar para fazer as perturbações ou mesmo que tipo de perturbações usar.

Como procedimento para gerar uma solução inicial (linha 1), utilizou-se o método construtivo C1, utilizado nas heurísticas anteriores, e, como busca local (linhas 2 e 5), o algoritmo BL4, utilizado pelo GRASP e pelo GRASP+MA. Como critério de aceitação (linha 6), foi estabelecido que uma solução seria aceita sempre que melhorasse o custo da função objetivo.

As perturbações do BLI (linha 4) foram feitas da seguinte maneira: A cada perturbação, um número fixo (doze) de movimentos será feito para perturbar uma solução, nove deles feitos aleatoriamente e três baseados em memória (estes valores foram definidos a partir de testes preliminares, que indicaram que esta configuração de movimentos é suficiente para que a BLI escape de ótimos locais.).

Os movimentos aleatórios podem ser de dois tipos: *realocação* e *troca*. Para cada movimento a ser executado, uma série de decisões precisa ser tomada. A primeira decisão é a respeito de qual dos dois tipos de movimentos (*realocação* ou *troca*) executar no momento. Esta decisão é aleatória com cinqüenta por cento de chance para cada tipo.

Em seguida, é decidida a quantidade de rotas que serão utilizadas no movimento. Para os movimentos aleatórios podem ser usadas duas rotas ou também pode ser utilizada uma única rota. A escolha entre usar uma ou duas rotas também é feita de forma aleatória e com cinqüenta por cento de chance para cada um destes valores.

Uma vez decidido o número de rotas, o terceira passo a ser feito é escolher quais as rotas serão utilizadas. Esta escolha é aleatória e todas as rotas possuem a mesma chance, tomando cuidado para que, no caso de duas rotas serem utilizadas, sejam escolhidas rotas distintas.

Por último, é necessário escolher as posições a serem utilizadas no movimento. Esta escolha é feita também aleatoriamente e com a mesma chance para cada posição (a chance de cada posição depende do tamanho das rotas envolvidas no movimento, isto é, para instâncias maiores cada posição possui uma chance menor e vice e versa). Quando uma única rota é utilizada, deve-se tomar cuidado para que as posições escolhidas não sejam as mesmas. Tal preocupação desaparece quando são utilizadas duas rotas, uma vez que, a mesma posição em rotas diferentes armazenam poços diferentes e geram movimentos válidos.

Os movimentos baseados em memória são apenas de *realocação* e sempre envolvem duas sondas. Para gerar estes movimentos foi construída uma matriz de seqüências de tamanho dois, que armazena a freqüência com que dois poços aparecem de forma consecutiva, ao longo da busca. As seqüências de maior freqüência são então escolhidas e, caso estejam presentes na solução a ser perturbada, terão suas partes iniciais utilizadas como base para os movimentos de realocação.

Para realizar um destes movimentos, o primeiro passo a ser feito é verificar quais das seqüências mais freqüentes da matriz estão presentes na solução a ser perturbada. Feito

isso, para cada seqüência freqüente presente na solução, deve-se localizar o seu poço inicial, determinando sua posição dentro da solução. Em seguida, uma segunda posição de uma rota destino qualquer (diferente da rota em que o poço inicial da seqüência se encontra) é escolhida e o movimento de realocação entre estas duas posições é executado. Apenas os poços iniciais das seqüências são utilizados na construção dos movimentos baseados em memória. As rotas de destino e as posições que os poços iniciais das seqüências freqüentes ocuparão nestas rotas, para este tipo de movimento, são escolhidas aleatoriamente.

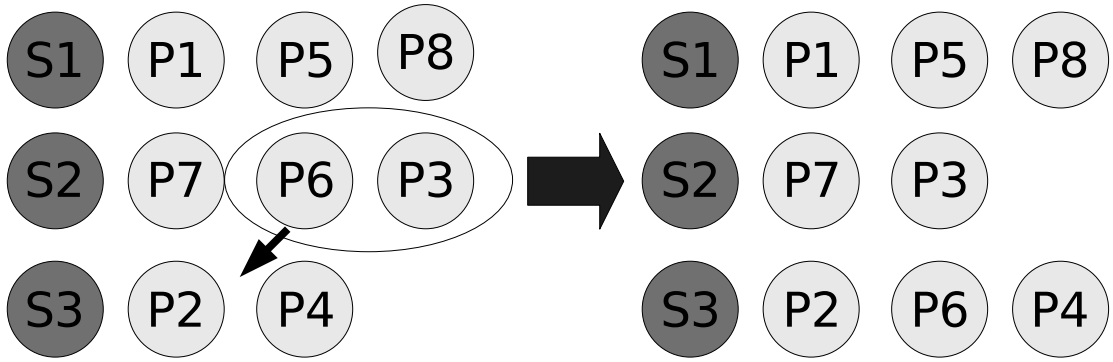


Figura 4.7: Exemplo de Perturbação Baseada em Memória

A Figura 4.7, mostra um exemplo de perturbação baseada em memória utilizado na BLI. Nela podemos observar uma seqüência freqüente, $P6 - P3$, que tem o poço inicial ($P6$), movido para uma posição aleatória de uma rota qualquer, também escolhida aleatoriamente. Neste exemplo, a posição escolhida foi entre os poços $P2$ e $P4$, pertencentes à sonda $S3$.

Capítulo 5

Resultados computacionais

Todos os componentes de *software* implementados neste trabalho foram feitos na linguagem C++, usando o compilador G++, versão 4.0.2, com as opções de compilação "-O3" e "-mtune=pentium4".

Os algoritmos GRASP, GRASP+MA e Busca Tabu foram executados em um computador com processador Pentium 4 HT, 3.2 GHz, com 1 GB de memóra RAM, rodando um sistema operacional Linux (Fedora Core), kernel versão 2.6.11.

Os testes com a Busca Local Iterada (BLI) foram feitos em um computador com processador Pentium 4 HT, 2.8 GHz, com 512 MB de Memória RAM, rodando um sistema operacional linux (Ubuntu), kernel versão 2.6.17.

Os testes com a formulação matemática foram feitos utilizando o solver CPLEX 10, com a tecnologia Concert 22, em um computador com processador Intel Pentium D, de 3.0 GHz, com 2 GBytes de memória RAM, rodando um sistema operacional Linux (Ubuntu), kernel versão 2.6.15.

5.1 Instâncias

Devido ao desconhecimento, da parte dos autores deste trabalho, da existência de instâncias em bibliotecas públicas para o PRSSM, um gerador de instâncias foi implementado neste trabalho.

As instâncias são combinações de dois arquivos, um contendo as informações dos poços e outro contendo as informações das sondas.

Os arquivos dos poços foram criados seguindo a distribuição normal utilizando os se-

guintes critérios: coordenadas cartesianas x e y variando entre 0 (zero) e 200 (duzentos), sendo que não há mais que um poço no mesmo ponto cartesiano dentro da mesma instância, vazão dos poços variando entre 1 (um) e 10 (dez), tempo de atendimento dos poços variando de 1 (um) a 5 (cinco) e tempo limite de atendimento dos poços variando entre 1500 (mil e quinhentos) e 2000 (dois mil).

Foram gerados arquivos de três dimensões diferentes em relação ao número de poços, a saber, 50, 100 e 500 poços, sendo que, para cada dimensão, 10 instâncias foram geradas.

Os dados das sondas ficam em um arquivo separado dos poços. Assim, um mesmo arquivo de poços pode ser utilizado em testes com vários conjuntos de sondas. Estes arquivos foram gerados seguindo a distribuição normal com as coordenadas cartesianas x e y variando de 0 (zero) a 100 (cem), sendo que, em uma mesma instância, não há mais do que uma sonda em um ponto cartesiano. Foram gerados dois conjuntos de sondas, um contendo 5 sondas e um outro contendo 10 sondas.

Para os testes com o *software* CPLEX foi gerado um outro conjunto de instâncias devido ao fato de que as instâncias mencionadas acima, mesmo as de 50 poços, serem muito difíceis para resolução de forma exata, usando a formulação apresentada no Capítulo 3. Para gerar este novo conjunto de instâncias, foi utilizado como base o conjunto de tamanho 50. Para fazer com que as instâncias pudessem ser resolvidas de forma exata, os últimos poços foram removidos das instâncias de tamanho 50 (por exemplo, se a instância utilizada possui 8 poços, os últimos 42 poços foram removidos).

Para a utilização no CPLEX apenas um conjunto de sondas (o de tamanho 5) foi utilizado. Ainda sim, uma sonda teve que ser removida dele para tornar possível a resolução das instâncias de forma exata. Assim um novo conjunto de 4 sondas foi gerado removendo uma sonda do conjunto de 5 sondas.

5.2 Resultados Computacionais

As médias mostradas nas tabelas e nos gráficos deste capítulo são as médias aritméticas dos valores obtidos pelos algoritmos após cinco execuções, cada uma com uma semente diferente. As sementes usadas neste trabalho foram 3, 5, 7, 11 e 13.

O critério de parada para o GRASP e para o GRASP+MA foi de tempo fixo. Para as instâncias de 50 poços, foram dados 30 minutos; para as de 100 poços, uma hora e para as de 500 poços, quatro horas (Estes valores foram definidos através de simulações

prévias).

Para o GRASP, o valor de α foi fixado em 0.1 (zero ponto um) e, para o GRASP+MA, foram usados os seguintes parâmetros: tamanho do conjunto elite igual a 10 soluções; diferença mínima entre as soluções do conjunto elite igual a 6 unidades (4); o tamanho do intervalo de avaliação para o incremento de α foi fixado em 300 segundos e limite máximo para o valor de α foi fixado em 0.8 (zero ponto oito).

Estes valores foram todos definidos a partir de testes preliminares.

Nas Tabelas de 5.1 à 5.4, a coluna "Instância" mostra o nome da instância para qual os valores foram obtidos, as colunas referentes à função objetivo, "Melhor" e "Média", mostram, respectivamente, o valor da melhor solução encontrada e a média aritmética de todos os valores encontrados. A coluna "Tempo Médio" mostra o tempo médio, em segundos, gasto para encontrar a melhor solução e a coluna "Iterações" mostra o número médio de iterações feito ao longo da busca. Os valores em negrito indicam os melhores valores encontrados quando comparados o GRASP e o GRASP+MA.

	Função Objetivo			
Instância	Melhor	Média	Tempo Médio	Iterações
50/5-1	**42682**	42682	420.7	168410.8
50/5-2	**45516**	45516	834.1	162278.2
50/5-3	**41622**	41622	494.0	168198.0
50/5-4	**40524**	40524	835.3	163327.8
50/5-5	**42785**	42785	767.5	165279.6
50/5-6	44757	44783.8	907.6	175246.2
50/5-7	**51554**	51554	658.6	171517.6
50/5-8	**37665**	37665	478.3	169542.6
50/5-9	**43498**	43498	630.3	172663.6
50/5-10	**42970**	42970	1087.0	174185.8
100/5-1	135217	135976	2769.3	47759.2
100/5-2	112599	112953	1480.6	47611.2
100/5-3	124852	125539	1565.2	48209.0
100/5-4	123297	124211	1020.5	43253.6
100/5-5	129779	130248	1455.0	46053.4
100/5-6	115140	115448	1097.0	47193.6

Tabela 5.1: GRASP: resultados para instâncias de 5 sondas

	Função Objetivo			
Instância	Melhor	Média	Tempo Médio	Iterações
100/5-7	133741	134522	2408.3	46528.0
100/5-8	129387	129786	1734.7	47870.6
100/5-9	117506	117789	1322.8	46261.0
100/5-10	123947	124112	1104.1	46436.0
500/5-1	1862305	1905405	11056.0	191.0
500/5-2	1856058	1877151	10533.2	181.2
500/5-3	1891220	1905701	7660.1	184.0
500/5-4	1939533	1955036	6255.4	179.0
500/5-5	1894201	1907483	5382.9	173.3
500/5-6	1799649	1817957	6279.6	172.0
500/5-7	1898304	1917840	8477.7	193.5
500/5-8	1850295	1887152	7762.4	192.0
500/5-9	**1872865**	1908202	9513.8	174.4
500/5-10	1925922	1962192	5440.6	222.0

Tabela 5.1: GRASP: resultados para instâncias de 5 sondas

A Tabela 5.1 mostra os resultados obtidos pela heurística GRASP para as instâncias de cinco sondas.

	Função Objetivo			
Instância	Melhor	Média	Tempo Médio	Iterações
50/10-1	**30486**	30486	703.0	139830.6
50/10-2	30952	30971.2	522.4	105156.4
50/10-3	**29352**	29352	946.6	113146.2
50/10-4	28869	28884.2	998.4	104723.4
50/10-5	29370	29400.8	937.7	109251.2
50/10-6	30484	30502.6	1058.4	110091.0
50/10-7	37506	37509.2	505.2	109647.0

Tabela 5.2: GRASP: resultados para instâncias de 10 sondas

	Função Objetivo			
Instância	Melhor	Média	Tempo Médio	Iterações
50/10-8	**26451**	26490.4	1039.8	113089.2
50/10-9	**31350**	31376.6	1120.4	107831.4
50/10-10	29907	29935.4	1008.9	107486.4
100/10-1	82993	83590.2	1705.4	88652.1
100/10-2	71723	72109.1	2394.3	72654.3
100/10-3	75995	78579.8	1781.5	65851.1
100/10-4	71546	71768.4	2541.4	74645.2
100/10-5	78284	79657.8	1954.3	68972.3
100/10-6	71130	74341.9	2014.1	62347.4
100/10-7	82968	83337.2	1754.6	71268.3
100/10-8	77312	79588.2	1522.1	80162.4
100/10-9	73188	74976.7	1601.2	65413.7
100/10-10	75586	75843.1	1683.8	77632.8
500/10-1	966966	973286	7708.4	961.8
500/10-2	940852	943352	7490.5	974.6
500/10-3	952534	959979	7765.3	959.4
500/10-4	**957145**	984200	3550.2	956.6
500/10-5	991807	996290	7824.8	933.4
500/10-6	928871	938368	5534.1	946.6
500/10-7	952842	964152	10128.5	961.4
500/10-8	953220	960610	8757.6	971.0
500/10-9	963855	972176	2980.2	955.4
500/10-10	1000136	1005396	6317.2	929.6

Tabela 5.2: GRASP: resultados para instâncias de 10 sondas

A Tabela 5.2 mostra os resultados obtidos pelo GRASP para instâncias de 10 sondas.

	Função Objetivo			
Instância	Melhor	Média	Tempo Médio	Iterações
50/5-1	**42682**	42682	751.9	90217.8
50/5-2	**45516**	45516	1183.7	90303.2
50/5-3	**41622**	41622	677.1	90652.8
50/5-4	**40524**	40524	376.7	90637.6
50/5-5	**42785**	42785	1181.4	90809.0
50/5-6	**44682**	44682	1623.8	94926.0
50/5-7	**51554**	51554	538.1	90147.2
50/5-8	**37665**	37665	585.3	90035.2
50/5-9	**43498**	43498	1246.3	90866.6
50/5-10	**42970**	42970	730	91723.0
100/5-1	**134885**	135250	3519.9	25818.0
100/5-2	**111949**	111999	3495.9	32877.0
100/5-3	**124160**	124847	3559.7	25953.6
100/5-4	**122246**	122827	3585.9	25191.4
100/5-5	**128874**	129248	3571.5	24937.8
100/5-6	**114685**	114794	3590.7	25661.0
100/5-7	**132911**	132917	3518.2	25328.0
100/5-8	**128612**	128805	3384.5	25977.6
100/5-9	**117343**	117603	3566.2	25063.0
100/5-10	**123144**	123301	3583.9	25566.3
500/5-1	**1853045**	1878979	14602.4	164.0
500/5-2	**1818763**	1838369	14614.5	146.8
500/5-3	**1802280**	1849696	14644.3	144.5
500/5-4	**1872395**	1921074	14634.3	150.4
500/5-5	**1872539**	1877458	14636.7	147.2
500/5-6	**1785956**	1805513	14765.7	141.0
500/5-7	**1860601**	1872882	14745.5	153.0
500/5-8	**1841788**	1872877	14728.4	148.2
500/5-9	1883346	1900247	14728.4	149.0
500/5-10	**1865705**	1916187	14762.8	143.3

Tabela 5.3: GRASP+MA: resultados para instâncias de 5 sondas

	Função Objetivo			
Instância	Melhor	Média	Tempo Médio	Iterações

Tabela 5.3: GRASP+MA: resultados para instâncias de 5 sondas

A Tabela 5.3 mostra os resultados obtidos pela heurística GRASP+MA para as instâncias de cinco sondas. As colunas são as mesmas da tabela anterior. Destaque deve ser dado aos fatos de que este algoritmo conseguiu as melhores médias para todas as instâncias quando comparado ao algoritmo GRASP e que só não alcançou o melhor resultado para uma única instância, a saber, a instância 500/5-9.

	Função Objetivo			
Instância	Melhor	Média	Tempo Médio	Iterações
50/10-1	**30486**	30486	933.5	91369.6
50/10-2	**30933**	30933	1466.5	102394.2
50/10-3	**29352**	29352	689.1	108820.2
50/10-4	**28826**	28826	1630.9	100648.0
50/10-5	**29348**	29355.6	1681.7	93563.6
50/10-6	**30462**	30462.4	1627.6	90848.8
50/10-7	**37502**	37502	1625.5	106040.8
50/10-8	**26451**	26451	1640.5	106207.0
50/10-9	**31350**	31350	1594.8	102901.0
50/10-10	**29825**	29825	1630.2	102597.2
100/10-1	**82542**	82596.4	3414.2	53251.8
100/10-2	**71063**	71118.6	3493.2	52274.8
100/10-3	**75297**	75542.8	3517.1	54514.2
100/10-4	**70815**	70922.6	3453.2	52202.0
100/10-5	**77547**	77567.8	3447.8	52217.4
100/10-6	**70330**	70354.8	3485.8	53255.2
100/10-7	**82261**	82303.2	3487.6	51849.2
100/10-8	**76642**	76678	3438.8	52593.4
100/10-9	**72488**	72497.6	3488.3	53812.2

100/10-10	**74868**	74874	3405.3	53100.4
500/10-1	**943335**	950639	14498.8	958.6
500/10-2	**925540**	934259	14498.5	879.4
500/10-3	**941958**	947480	14492.2	825.8
500/10-4	969484	978300	14490.8	784.4
500/10-5	**970355**	979093	14498.8	737.0
500/10-6	**914980**	923819	14495.3	781.4
500/10-7	**944301**	952262	14507.3	876.6
500/10-8	**945750**	952073	14494.4	800.6
500/10-9	**958511**	964221	14495.3	803.8
500/10-10	**979184**	984955	14496.2	921.8

Tabela 5.4: GRASP+MA: resultados para instâncias de 10 sondas

A Tabela 5.4 mostra os resultados obtidos pelo GRASP+MA para instâncias de dez sondas. Destaque deve ser dado ao fato de que esta heurística conseguiu os melhores resultados para todas as instâncias exceto uma, 500/10-4.

	GRASP		GRASP+MA	
Instância	Melhor	Média	Melhor	Média
50/5-1	**42682**	42682	**42682**	42682
50/5-2	**45516**	45516	**45516**	45516
50/5-3	**41622**	41622	**41622**	41622
50/5-4	**40524**	40524	**40524**	40524
50/5-5	**42785**	42785	**42785**	42785
50/5-6	44757	44783.8	**44682**	44682
50/5-7	**51554**	51554	**51554**	51554
50/5-8	**37665**	37665	**37665**	37665
50/5-9	**43498**	43498	**43498**	43498
50/5-10	**42970**	42970	**42970**	42970

Tabela 5.5: GRASP X GRASP+MA para instâncias de 50 poços e 5 sondas

A Tabela 5.5 mostra um comparativo entre os algoritmos GRASP e GRASP+MA para as instâncias de 50 poços e 5 sondas. Os valores em negrito indicam o melhor resultado encontrado por uma destas heurísticas. É importante notar que o GRASP alcança o

mesmo resultado em todas as execuções para todas as instâncias exceto a instância 50-6. O GRASP+MA alcança o mesmo resultado em todas as execuções para todas as instâncias.

	GRASP		GRASP+MA	
Instância	Melhor	Média	Melhor	Média
100/5-1	135217	135976	**134885**	135250
100/5-2	112599	112953	**111949**	111999
100/5-3	124852	125539	**124160**	124847
100/5-4	123297	124211	**122246**	122827
100/5-5	129779	130248	**128874**	129248
100/5-6	115140	115448	**114685**	114794
100/5-7	133741	134522	**132911**	132917
100/5-8	129387	129786	**128612**	128805
100/5-9	117506	117789	**117343**	117603
100/5-10	123947	124112	**123144**	123301

Tabela 5.6: GRASP X GRASP+MA para instâncias de 100 poços e 5 sondas

A Tabela 5.6 mostra um resumo dos resultados obtidos pelo GRASP e pelo GRASP +MA para as instâncias de 100 poços e 5 sondas. Destaque deve ser dado ao fato de que a média obtida pelo GRASP+MA em muitas instâncias é melhor que o melhor resultado obtido pelo GRASP.

	GRASP		GRASP+MA	
Instância	Melhor	Média	Melhor	Média
500/5-1	1862305	1905405	**1853045**	1878979
500/5-2	1856058	1877151	**1818763**	1838369
500/5-3	1891220	1905701	**1802280**	1849696
500/5-4	1939533	1955036	**1872395**	1921074
500/5-5	1894201	1907483	**1872539**	1877458
500/5-6	1799649	1817957	**1785956**	1805513
500/5-7	1898304	1917840	**1860601**	1872882
500/5-8	1850295	1887152	**1841788**	1872877
500/5-9	**1872865**	1908202	1883346	1900247
500/5-10	1925922	1962192	**1865705**	1916187

Tabela 5.7: GRASP X GRASP+MA para instâncias de 500 poços e 5 sondas

A Tabela 5.7 mostra o comparativo entre o GRASP e o GRASP+MA para as instâncias de 500 poços e 5 sondas. Novamente o GRASP+MA mostrou-se superior em todas as instâncias, exceto a instância 500-9 onde, apesar de conseguir a melhor média, ele não conseguiu o melhor resultado.

A Tabela 5.8 mostra o comparativo entre o GRASP e GRASP+MA para as instâncias de 50 poços e 10 sondas. O GRASP+MA obteve os melhores resultados.

	GRASP		GRASP+MA	
Instância	Melhor	Média	Melhor	Média
50/10-1	**30486**	30486	**30486**	30486
50/10-2	30952	30971.2	**30933**	30933
50/10-3	**29352**	29352	**29352**	29352
50/10-4	28869	28884.2	**28826**	28826
50/10-5	29370	29400.8	**29348**	29355.6
50/10-6	30484	30502.6	**30462**	30462.4
50/10-7	37506	37509.2	**37502**	37502
50/10-8	**26451**	26490.4	**26451**	26451
50/10-9	**31350**	31376.6	**31350**	31350
50/10-10	29907	29935.4	**29825**	29825

Tabela 5.8: GRASP X GRASP+MA para instâncias de 50 poços e 10 sondas

	GRASP		GRASP+MA	
Instância	Melhor	Média	Melhor	Média
100/10-1	82993	83590.2	**82542**	82596.4
100/10-2	71723	72109.1	**71063**	71118.6
100/10-3	75995	78579.8	**75297**	75542.8
100/10-4	71546	71768.4	**70815**	70922.6
100/10-5	78284	79657.8	**77547**	77567.8
100/10-6	71130	74341.9	**70330**	70354.8
100/10-7	82968	83337.2	**82261**	82303.2
100/10-8	77312	79588.2	**76642**	76678
100/10-9	73188	74976.7	**72488**	72497.6
100/10-10	75586	75843.1	**74868**	74874

Tabela 5.9: GRASP X GRASP+MA para instâncias de 100 poços e 10 sondas

A Tabela 5.9 mostra um comparativo entre o GRASP e o GRASP+MA para as instâncias de 100 poços e 10 sondas. O GRASP+MA mostrou-se superior em todas as instâncias.

A Tabela 5.10 mostra um comparativo entre o GRASP e o GRASP+MA para as instâncias de 500 poços e 10 sondas. O GRASP+MA obteve melhores resultados e melhores médias em todas as instâncias exceto a instancia 50/10-4, onde apesar de ter conseguido a melhor média, não obteve o melhor resultado.

Deve-se ressaltar a importância do fato de que aumentar o número de sondas para um mesmo grupo de poços, faz com que a vazão perdida seja reduzida. Para este problema isto ocorre indefinidamente devido ao fato de que o custo fixo para usar sondas foi desprezado. Estudos para levar em consideração o custo operacional das sondas serão alvo de trabalhos futuros.

	GRASP		GRASP+MA	
Instância	Melhor	Média	Melhor	Média
500/10-1	966966	973286	**943335**	950639
500/10-2	940852	943352	**925540**	934259
500/10-3	952534	959979	**941958**	947480
500/10-4	**957145**	984200	969484	978300
500/10-5	991807	996290	**970355**	979093
500/10-6	928871	938368	**914980**	923819
500/10-7	952842	964152	**944301**	952262
500/10-8	953220	960610	**945750**	952073
500/10-9	963855	972176	**958511**	964221
500/10-10	1000136	1005396	**979184**	984955

Tabela 5.10: GRASP X GRASP+MA para instâncias de 500 poços e 10 sondas

Com base nos resultados apresentados até aqui, pode-se concluir que as técnicas utilizadas no GRASP+MA realmente são capazes de melhorar as soluções encontradas pelo GRASP tradicional. Contudo, pode-se constatar também, que estas técnicas demandam um tempo computacional consideravelmente alto, visto que, o número médio de iterações do GRASP+MA é menor do que o número médio de iterações do GRASP.

Os testes com a Busca Tabu foram feitos nas mesmas condições dos testes realizados com o algoritmo GRASP e com o GRASP+MA, porém além do tempo como critério de parada, foram utilizados valores alvo. Assim, a Busca Tabu pára quando encontrar uma solução igual, ou melhor, que o alvo estabelecido ou quando atingir o limite de tempo. Os alvos utilizados foram os valores das melhores soluções encontradas para cada instância, mostrado em negrito nas tabelas anteriores. Os tempos utilizados são os mesmos utilizados para o GRASP e o GRASP+MA.

O conjunto usado para criar novas soluções, comporta seis soluções, com a diferença mínima entre elas sendo de duas unidades (Capítulo 4) e, o número de soluções que serão geradas a partir da memória, foi fixado em três soluções.

Na Busca Tabu básica, a cada iteração, o tamanho da lista tabu é modificado [16]. Este tamanho é escolhido, aleatoriamente, em um intervalo que varia entre a metade do número de poços da instância em execução (mínimo) e número de poços da instância (máximo).

Estes valores foram fixados após testes preliminares.

A Tabela 5.11 mostra os resultados encontrados pela heurística Busca Tabu. As colunas são as mesmas da tabela 5.1, com a exceção de que nesta tabela não há a coluna com

o número médio de iterações. Os valores em negrito são os melhores valores encontrados para as respectivas instâncias.

A Tabela 5.12 Mostra os resultados alcançados pela heurística Busca Tabu para as instâncias de dez sondas. As colunas são as mesmas da tabela 5.11. Em negrito estão os melhores valores encontrados para as respectivas instâncias.

A Busca Tabu apresentou resultados melhores que os do GRASP tradicional, contudo, ao contrário do esperado, segundo os resultados de Gendreau *et al.* [14], o GRASP+MA atingiu os melhores resultados quando comparamos apenas estes três algoritmos. Uma possível causa para o baixo desempenho da Busca Tabu é a limitação de tempo imposta. Talvez a Busca Tabu necessite de mais tempo do que o GRASP+MA para atingir melhores resultados.

Em seguida, mostraremos os resultados obtidos pela Busca Local Iterada. As condições de parada para esta heurística são as mesmas da Busca Tabu, ou seja, ela para quando encontra uma solução melhor ou igual a um alvo ou quando atinge o limite de tempo.

	Função Objetivo		
Instância	Melhor	Média	Tempo Médio
50/5-1	**42682**	42682	20.9
50/5-2	**45516**	45516	84.0
50/5-3	**41622**	41622	21.8
50/5-4	**40524**	40524	36.9
50/5-5	**42785**	42785	53.5
50/5-6	**44682**	44682	13.7
50/5-7	**51554**	51554	9.0
50/5-8	**37665**	37665	6.2
50/5-9	**43498**	43498	22.3
50/5-10	**42970**	42970	106.0
100/5-1	**134885**	135695.2	2158.9
100/5-2	**111949**	112024.2	1065.7
100/5-3	**124160**	125082	2544.3
100/5-4	**122246**	123294.8	2090.1
100/5-5	**128874**	130642.2	2169.8
100/5-6	**114685**	115791	1947.8

Instância	Melhor	Média	Tempo Médio
100/5-7	133583	134464.8	2156.3
100/5-8	**128612**	129415.4	1871.6
100/5-9	**117343**	118119.6	1874.1
100/5-10	**123144**	124144	2730.5
500/5-1	**1842876**	1850095.8	3073.8
500/5-2	**1807606**	1814424.6	1075.1
500/5-3	**1796122**	1798805.4	3558.8
500/5-4	**1858770**	1867327	4503.8
500/5-5	**1861193**	1868589.4	1028.5
500/5-6	**1782885**	1784137	1388.1
500/5-7	**1855872**	1858298.8	1137.3
500/5-8	**1828625**	1835835.6	1429.9
500/5-9	**1866013**	1874685.6	1042.1
500/5-10	**1859803**	1870431.4	8137.4

Tabela 5.13: BLI: resultados para instâncias de 5 sondas

A Tabela 5.13 mostra os resultados obtidos pela BLI para as instâncias de cinco sondas. As colunas são as mesmas das tabelas 5.1 e 5.3. **é importante ressaltar o fato de que ela conseguiu os melhores resultados para todas as instâncias exceto uma: 100/5-7**. Os resultados para as instâncias de 500 poços foram em sua maioria os melhores encontrados até então mesmo com o uso do valor da função objetivo como critério de parada.

	Função Objetivo		
Instância	Melhor	Média	Tempo Médio
50/10-1	**30486**	30486	8.4
50/10-2	**30933**	30933	7.3
50/10-3	**29352**	29352	38.5
50/10-4	**28826**	28826	4.2
50/10-5	**29348**	29348	240.9
50/10-6	**30462**	30462	110.6

Instância	Melhor	Média	Tempo Médio
50/10-7	**37502**	37502	30.3
50/10-8	**26451**	26451	81.9
50/10-9	**31350**	31350	1353.4
50/10-10	**29825**	29825	4.1
100/10-1	**82542**	82766.8	2452.6
100/10-2	**71063**	71188	1409.4
100/10-3	**75227**	75504.6	2048.8
100/10-4	70860	71158	1751.4
100/10-5	**77547**	77866	2449.7
100/10-6	**70330**	70498	1716.8
100/10-7	**82261**	82519.2	1991.4
100/10-8	**76642**	76967.4	929.2
100/10-9	**72470**	72549.8	1304.8
100/10-10	**74868**	75350.8	2712.3
500/10-1	**942991**	947589.4	7618.6
500/10-2	**924830**	929466.2	8712.0
500/10-3	**940585**	941464.2	2128.5
500/10-4	**955416**	958940	3747.1
500/10-5	**968999**	969933.2	2686.8
500/10-6	**913407**	921551.8	5547.0
500/10-7	**923592**	940114.2	12866.5
500/10-8	944513	945017.2	2173.4
500/10-9	**954567**	958498.6	3266.4
500/10-10	**978283**	982183	6585.6

Tabela 5.14: BLI: resultados para instâncias de 10 sondas

A Tabela 5.14 mostra os resultados obtidos pela BLI para as instâncias de dez sondas. Destaque deve ser dado ao fato de que esta metaheurística obteve os melhores resultados para todas as instâncias exceto a 100/10-4 e a 500/10-8.

A seguir serão apresentados comparativos entre todas as heurísticas propostas para o problema.

A Tabela 5.15 mostra um comparativo das heurísticas propostas para as instâncias com cinco sondas. A coluna "Melhor" mostra o melhor valor de função objetivo encontrado para a instância tratada. As demais colunas mostram a diferença percentual entre o valor da função objetivo da melhor solução encontrada por cada heurística e o valor mostrado na coluna "Melhor". O objetivo desta tabela é ilustrar o quão distante cada heurística está do melhor valor encontrado. Para calcular esta diferença percentual (dp) considere vh o valor encontrado pela heurística e mv o melhor valor encontrado. Assim a diferença Percentual pode ser calculada por $dp = (vh - mv)/mv \times 100$. Quanto menor é a diferença percentual, mais próximo o valor atingido se encontra do melhor valor encontrado. Quando esta diferença é igual a zero, significa que o melhor valor conhecido foi atingido.

A Tabela 5.16 mostra um comparativo das médias alcançadas por cada heurística em relação ao melhor valor encontrado. O objetivo é ilustrar o quão distante a média alcançada por cada heurística está do melhor valor encontrado.

A Tabela 5.17 Mostra um comparativo das heurísticas para instâncias de 10 sondas. Para este grupo de instâncias ficou mais evidente que a heurística que obteve o melhor desempenho foi a heurística BLI, que não conseguiu atingir o melhor valor da função objetivo somente em duas instâncias.

A Tabela 5.18 mostra um comparativo das médias para as instâncias de dez sondas. Neste quesito, a heurística que obteve o melhor desempenho para as instâncias de cem poços foi a heurística GRASP+MA, que em muitos casos, conseguiu manter a média das execuções exatamente igual ao melhor valor encontrado, mostrando assim, ser um algoritmo robusto. Para as instâncias de quinhentos poços a heurística BLI apresentou melhores resultados, mesmo sendo um algoritmo menos regular que o GRASP+MA. Os resultados atingidos pela BLI podem ser explicados pelo fato de que esta heurística conseguiu os melhores valores para as instâncias com quinhentos poços, fato este que ocasionou uma melhora na média total dos resultados obtidos pelo algoritmo.

Pelos resultados ilustrados nas tabelas 5.15 a 5.18 percebemos que os melhores resultados foram obtidos pelas heurísticas BLI e GRASP+MA. O pior desempenho foi do GRASP sem memória.

Quando confrontamos as heurísticas GRASP e GRASP+MA, ficou evidente a superioridade do GRASP+MA. Isso já justifica o uso de memória em heurísticas do tipo GRASP.

A Busca Tabu, utilizando os conceitos de reconhecimento de padrões proposto por

Taillard [34] *et al.* para um PRV, embora seja idealizada por conceitos bastante interessantes não apresentou os resultados esperados. Isso em parte ocorreu provavelmente pela forma de escolha das rotas do conjunto de soluções e pela limitação de tempo imposta. Acreditamos que o critério de seleção de rotas deva ser melhor analisado.

Já o desempenho da BLI foi bastante promissor principalmente para instâncias maiores (500 poços) onde este apresentou os melhores resultados.

Testes adicionais são propostos a seguir para avaliar a regularidade das heurísticas propostas.

5.3 Analise Probabilística

Nesta seção, é colocada como meta uma avaliação empírica da regularidade das heurísticas propostas. Isso se torna importante na medida em que heurísticas com componentes aleatórias, como é o caso do GRASP, podem apresentar soluções bem distintas a cada execução de uma dada instância. Desta forma, foi utilizada uma outra abordagem para análise de desempenho empírico de metaheurísticas, proposta em [1]. Este tipo de análise consiste em tomar os tempos de processamento que cada algoritmo requer para atingir um valor alvo na função objetivo. Para tanto, no instante em que cada algoritmo encontra uma solução melhor ou igual ao alvo fixado, o tempo é registrado e a execução é interrompida.

Cada algoritmo foi executado 25 vezes para cada instância avaliada. Após as execuções, os tempos foram tomados e dispostos em ordem crescente numa lista L. A cada tempo de CPU t obtido, foi associada a probabilidade $p_i = (i - \frac{1}{2})/25$, onde i é a posição em que t aparece na lista ordenada L. A plotagem foi feita então tomando-se cada ponto (t_i, p_i). Caso o algoritmo não tenha sucesso em encontrar o alvo durante o tempo dado, considera-se que, nesta execução, ele não foi capaz de atingir o alvo.

As instâncias foram escolhidas com base nos valores obtidos pela heurística GRASP+MA. Foram escolhidas as instâncias, uma fácil (50/10-3) e uma difícil (100/10-4). A instância 50/10-3 foi considerada fácil pelo fato de que todas as heurísticas foram capazes de atingir o mesmo valor em todas as execuções. A Instância 100/10-4 foi considerada difícil pelo fato de que o algoritmo GRASP+MA, que apresenta as melhores médias para as instâncias de cem poços, encontrou soluções diferentes para cada execução nos experimentos realizados com esta instância.

Para demonstrar a importância da Reconexão de Caminhos na convergência do al-

goritmo GRASP +MA, durante a fase de analise probabilística, esta técnica foi ativada com maior freqüência durante a busca. Para isto, o tempo total de execução foi dividido em quatro partes. Ao fim de cada quarto, a Reconexão de Caminhos é executada entre os membros do conjunto elite.

Todos os demais parâmetros para o GRASP+MA e para as outras heurísticas foram mantidos idênticos à fase de experimentos com as instâncias.

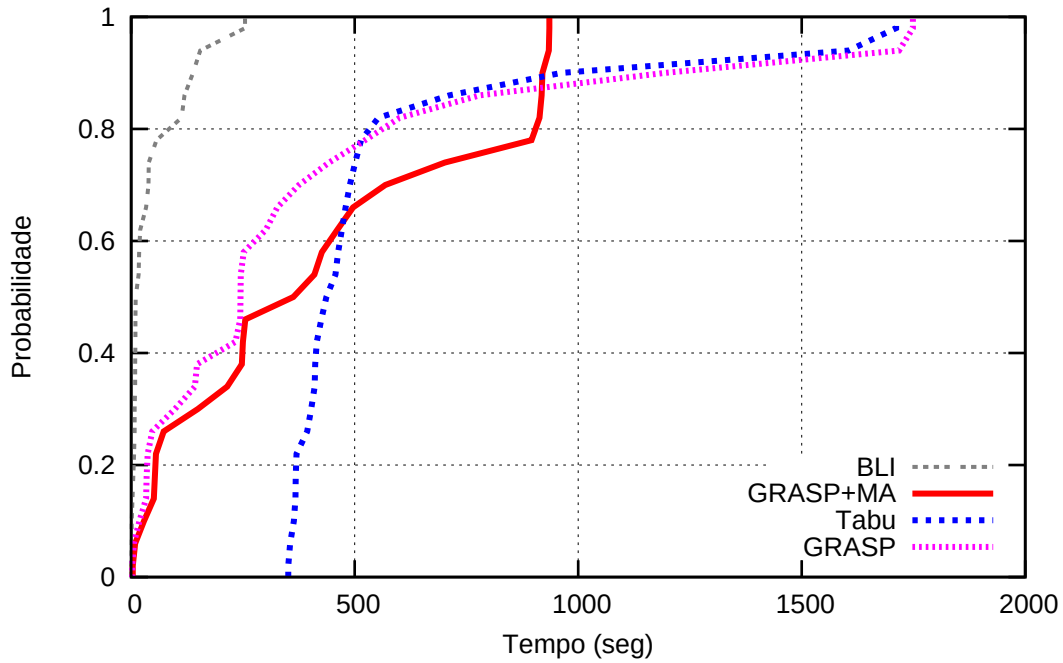


Figura 5.1: Análise Probabilística para instância 50/10-3

A Figura 5.1 mostra a analise probabilística para a instância 50/10-3. O alvo escolhido foi 29352, que é o valor atingido por todas as heurísticas em todas as execuções. Por este gráfico pode-se inferir que a heurística que converge mais rapidamente para o alvo é a heurística BLI, seguida da heurística GRASP+MA. A Busca Tabu, apesar de mostrar uma convergência mais lenta nos primeiros instantes, logo ultrapassa as heurísticas GRASP e GRASP+MA, porém, ao se aproximar de 800 segundos, tempo no qual a Reconexão de caminhos é ativada no GRASP+MA, a Busca Tabu é ultrapassada e fica em terceiro lugar na convergência.

A Figura 5.2 mostra a analise probabilística para a instância 100/10-4. O alvo escolhido foi 71922, que é o pior valor atingido pelo GRASP durante as execuções. Este gráfico apresenta um comportamento muito semelhante ao gráfico anterior. Novamente, a BLI apresentou a melhor convergência, seguida do GRASP+MA, Busca Tabu, e por último, a heurística GRASP. Pode-se notar, novamente, que a heurística GRASP+MA consegue

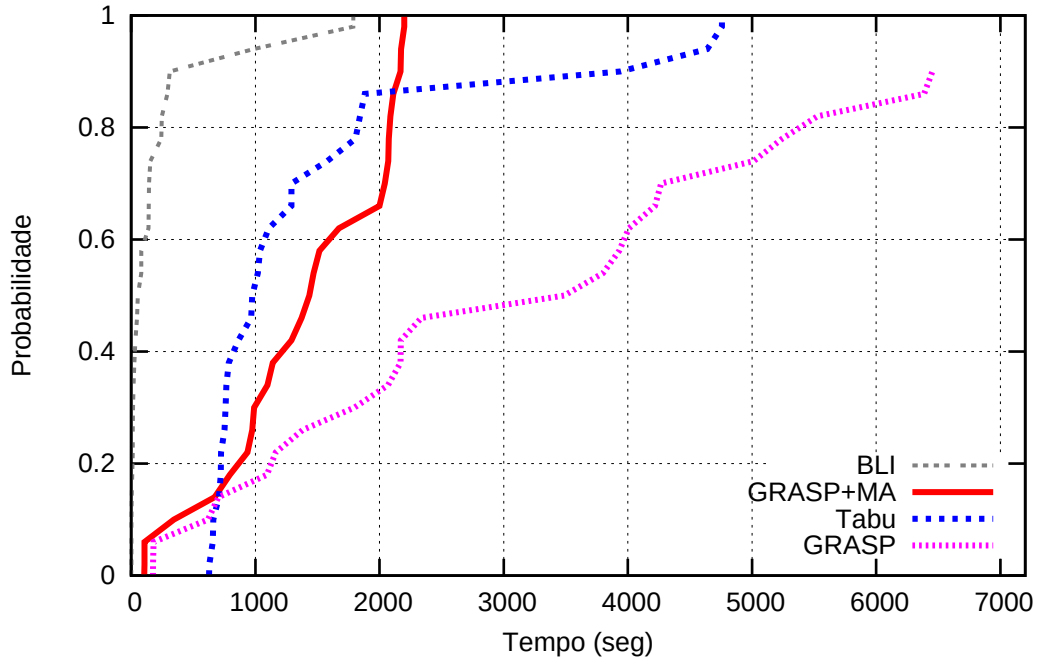


Figura 5.2: Análise Probabilística para instância 100/10-4 (alvo fácil)

ultrapassar a heurística Busca Tabu somente depois da ativação da Reconexão de Caminhos. Contudo, pode-se observar que a heurística GRASP não foi capaz de convergir em todas as execuções e também que houve uma aproximação das curvas das heurísticas BLI e GRASP+MA.

Esta aproximação entre as curvas das heurísticas BLI e GRASP+MA indica que a medida que a dificuldade do alvo aumenta, a eficiência da BLI em atingir o alvo diminui. Este fato pôde ser comprovado ao se escolher um alvo mais difícil para esta mesma instância.

A Figura 5.3 mostra a analise probabilística para a instância 100/10-4. O alvo escolhido foi 70912, que é o valor mediano atingido pelo GRASP+MA durante as execuções. Para este alvo a heurística GRASP não convergiu em nenhuma execução e heurística Busca Tabu apresentou uma convergência menor que 10%.

Este gráfico comprova a hipótese de que a eficiência da BLI decresce à medida em que a dificuldade do alvo aumenta. Pode-se notar também que a convergência da heurística GRASP+MA é praticamente nula antes de 1800 segundos, quando a Reconexão de Caminhos é ativada pela primeira vez.

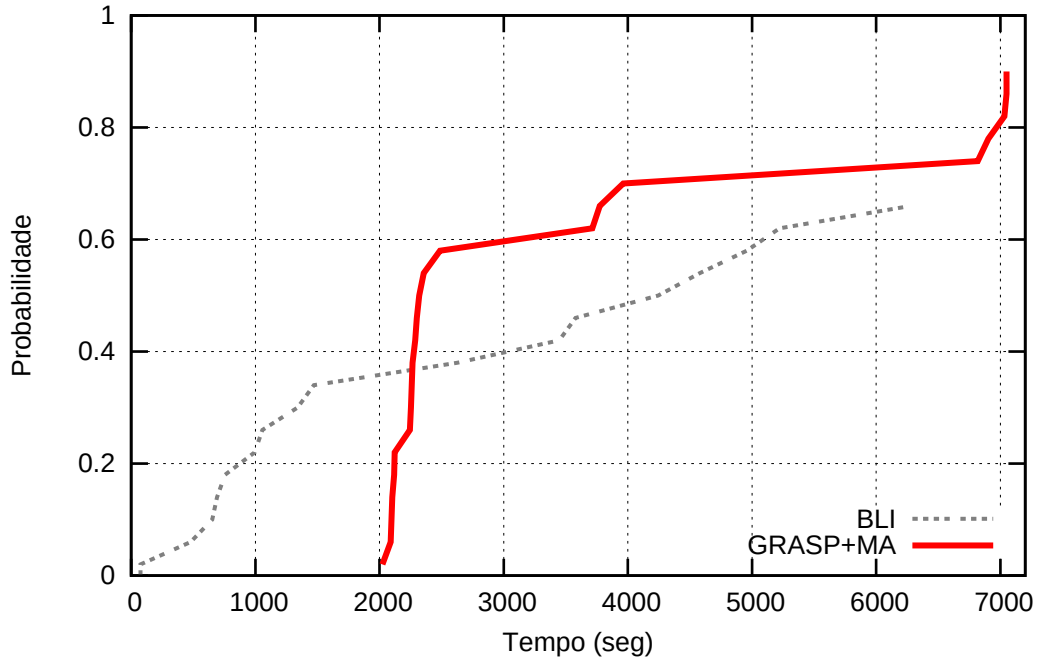


Figura 5.3: Análise Probabilística para instância 100/10-4 (alvo difícil)

5.4 Testes com Instâncias exatas

Nesta seção apresentaremos os resultados obtidos pelo software CPLEX para um grupo de instâncias pequenas, contendo no máximo 9 (nove) poços e 4 (quatro) sondas. O modelo matemático utilizado foi proposto no capítulo 3 e implementado utilizando-se a tecnologia Concert 22 e executados no software CPLEX 10.

A Tabela 5.19 mostra as características das instâncias que foram resolvidas pelo software CPLEX. A coluna "Instância" mostra o nome das instâncias tratadas, a coluna "Poços" mostra o número de poços de cada instância e a coluna "Sondas" mostra o número de sondas presentes em cada uma das instâncias.

Para todas as instâncias, o limite de tempo das heurísticas foi fixado em mil e oitocentos (1800) segundos. Todos os demais parâmetros dos testes anteriores foram mantidos com exceção do tamanho do Conjunto Elite para o GRASP+MA e do tamanho do Conjunto de Rotas para a Busca Tabu. Estes dois parâmetros foram configurados de maneira que o número máximo de soluções armazenadas fosse de uma solução.

Finalmente, para avaliar a qualidade das soluções aproximadas geradas pelas heurísticas propostas, foram efetuados testes utilizando a formulação matemática proposta com o *software* CPLEX.

A Tabela 5.20 mostra os resultados obtidos pelo CPLEX e pelas 4 heurísticas para as instâncias analisadas. A coluna "Instância" indica qual a instância esta sendo tratada. A coluna "Resultado" mostra o valor da função objetivo da solução encontrada pelo CPLEX. A coluna "Tempo" mostra o tempo de execução em segundos. A coluna "Convergência" (presente apenas nas heurísticas) mostra a porcentagem de vezes que cada heurística conseguiu atingir o valor ótimo para a instância tratada; Quando a convergência de uma heurística apresenta o valor 100 (cem), significa que ela foi capaz de encontrar o valor ótimo em todos os testes; Quando a convergência de uma heurística apresenta o valor 0 (zero), significa que esta heurística não conseguiu atingir o valor ótimo em nenhum dos testes. A coluna "T. Médio" (presente apenas nas heurísticas) mostra o tempo médio, em segundos, que cada heurística levou para atingir o valor ótimo, ou o tempo máximo permitido. Deve-se ressaltar o fato de que o algoritmo GRASP nem sempre conseguiu atingir o valor ótimo, nas instâncias em que este fato ocorre, a coluna "T. Médio" da heurística GRASP mostra o valor 1800, que foi o limite de tempo dado para as execuções das heurísticas.

Pelos resultados ilustrados na tabela 5.20, observa-se que as heurísticas GRASP +MA, Busca Tabu e BLI sempre conseguem chegar à solução ótima exigindo tempos bem menores que os do CPLEX.

Em relação aos tempos, nota-se que a Busca Tabu e a BLI foram os mais rápidos. O GRASP+MA exigiu tempos bem maiores devido ao uso da Reconexão de Caminhos.

E finalmente, o GRASP, sem o uso de memória adaptativa, apresentou o pior desempenho, conseguindo chegar à solução ótima, somente em duas instâncias.

	Função Objetivo		
Instância	Melhor	Média	Tempo Médio
50/5-1	**42682**	42682.0	187.0
50/5-2	**45516**	45550.4	873.9
50/5-3	**41622**	41622.0	199.5
50/5-4	**40524**	40524.0	354.1
50/5-5	**42785**	42785.0	246.9
50/5-6	44757	44695.2	1392.4
50/5-7	**51554**	51554.0	300.2
50/5-8	**37665**	37665.0	296.3
50/5-9	**43498**	43498.0	225.2
50/5-10	**42970**	42970	512.1
100/5-1	134937	135568.0	3454.0
100/5-2	112005	113728.8	3410
100/5-3	124437	125435.8	3468.7
100/5-4	122786	123390.6	3512.4
100/5-5	128998	130113.8	3451.2
100/5-6	114827	115260.8	3432.4
100/5-7	**132911**	134343.6	3473.0
100/5-8	128566	129780.2	3448.1
100/5-9	**117343**	117686.0	3161.4
100/5-10	**123144**	123887.4	3480.8
500/5-1	1886511	1911843.3	14103.6
500/5-2	1844266	1865604.3	14186.6
500/5-3	1865799	1884180	14321.5
500/5-4	1915528	1945795.5	13960
500/5-5	1894727	1914186.2	14273.7
500/5-6	1806710	1833883.8	14338.0
500/5-7	1848649	1894806.5	12321.3
500/5-8	1875004	1889833.5	14323.0
500/5-9	1906575	1908854.3	14299.3
500/5-10	1927175	1961793.0	14297.0

Tabela 5.11: Busca Tabu: resultados para instâncias de 5 sondas

	Função Objetivo		
Instância	Melhor	Média	Tempo Médio
50/10-1	**30486**	30486.0	422.7
50/10-2	**30933**	30952.5	1473.5
50/10-3	**29352**	29352.0	547.5
50/10-4	**28826**	28848.0	1586.2
50/10-5	**29348**	29369.8	1397.8
50/10-6	30464	30475.4	1703.2
50/10-7	**37502**	37502.0	1028.7
50/10-8	**26451**	26462.4	1282.6
50/10-9	**31350**	31354.7	894.7
50/10-10	**29825**	29825.0	247.7
100/10-1	82542	82587.2	3363.8
100/10-2	71162	71361.4	3384.2
100/10-3	75939	75982.6	3456.4
100/10-4	71596	71691.5	3587.5
100/10-5	77672	78227.2	3541.5
100/10-6	**70330**	70595.7	3107.5
100/10-7	82319	82426.8	3480
100/10-8	76646	76999.3	2553.3
100/10-9	72796	73117.0	3466.5
100/10-10	74967	75131.0	3308.0
500/10-1	955686	960535.6	14291.0
500/10-2	933371	941072.5	14314.0
500/10-3	950369	953495.6	14294.3
500/10-4	958181	965085.3	14302.0
500/10-5	969205	969499.0	6246.6
500/10-6	919087	924778.6	14283.6
500/10-7	947530	949054.6	14295.3
500/10-8	**939949**	954409.0	10283.1
500/10-9	958319	967514.3	12376.6
500/10-10	986039	991040.5	14339.5

Tabela 5.12: Busca Tabu: resultados para instâncias de 10 sondas

Instância	Melhor	GRASP	Busca Tabu	GRASP+MA	BLI
50/5-1	42682	0	0	0	0
50/5-2	45516	0	0	0	0
50/5-3	41622	0	0	0	0
50/5-4	40524	0	0	0	0
50/5-5	42785	0	0	0	0
50/5-6	44682	0.17	0.17	0	0
50/5-7	51554	0	0	0	0
50/5-8	37665	0	0	0	0
50/5-9	43498	0	0	0	0
50/5-10	42970	0	0	0	0
100/5-1	134885	0.24	0.4	0	0
100/5-2	111949	0.58	0.5	0	0
100/5-3	124160	0.56	0.22	0	0
100/5-4	122246	0.86	0.44	0	0
100/5-5	128874	0.7	0.1	0	0
100/5-6	114685	0.39	0.12	0	0
100/5-7	132911	0.62	0	0	0.5
100/5-8	128612	0.6	03	0	0
100/5-9	117343	0.14	0	0	0
100/5-10	123144	0.65	0	0	0
500/5-1	1842876	1.05	2.37	0.55	0
500/5-2	1807606	2.68	2.03	0.62	0
500/5-3	1796122	5.29	3.88	0.34	0
500/5-4	1858770	4.34	3.05	0.73	0
500/5-5	1861193	1.78	1.8	0.61	0
500/5-6	1782885	0.94	1.34	0.17	0
500/5-7	1855872	2.29	0.39	0.25	0
500/5-8	1828625	1.85	2.54	0.72	0
500/5-9	1866013	0.37	2.17	0.93	0
500/5-10	1859803	3.55	3.62	0.32	0

Tabela 5.15: Comparativo para instâncias de 5 sondas

Instância	Melhor	GRASP	Busca Tabu	GRASP+MA	BLI
50/5-1	42682	0	0	0	0
50/5-2	45516	0	0.1	0	0
50/5-3	41622	0	0	0	0
50/5-4	40524	0	0	0	0
50/5-5	42785	0	0	0	0
50/5-6	44682	0.2	0	0	0
50/5-7	51554	0	0	0	0
50/5-8	37665	0	0	0	0
50/5-9	43498	0	0	0	0
50/5-10	42970	0	0	0	0
100/5-1	134885	0.8	0.5	0.3	0.6
100/5-2	111949	0.9	1.6	0.1	0.1
100/5-3	124160	1.1	1.0	0.6	0.7
100/5-4	122246	1.6	0.9	0.5	0.9
100/5-5	128874	1.1	1.0	0.3	1.4
100/5-6	114685	0.7	0.5	0.1	1
100/5-7	132911	1.2	1.1	0.1	1.2
100/5-8	128612	0.9	0.9	0.2	0.6
100/5-9	117343	0.4	0.3	0.2	0.7
100/5-10	123144	0.8	0.6	0.1	0.8
500/5-1	1842876	3.4	3.7	2	0.4
500/5-2	1807606	3.8	3.2	1.7	0.4
500/5-3	1796122	6.1	4.9	3	0.1
500/5-4	1858770	5.2	4.7	3.4	0.5
500/5-5	1861193	2.5	2.8	0.9	0.4
500/5-6	1782885	2	2.9	1.3	0.1
500/5-7	1855872	3.3	2.1	0.9	0.1
500/5-8	1828625	3.2	3.3	2.4	0.4
500/5-9	1866013	2.3	2.3	1.8	0.5
500/5-10	1859803	5.5	5.5	3	0.6

Tabela 5.16: Comparativo das médias para instâncias de 5 sondas

Instância	Melhor	GRASP	Busca Tabu	GRASP+AM	BLI
50/10-1	30486	0	0	0	0
50/10-2	30933	0.6	0	0	0
50/10-3	29352	0	0	0	0
50/10-4	28826	0.15	0	0	0
50/10-5	29348	0.7	0	0	0
50/10-6	30462	0.7	0.1	0	0
50/10-7	37502	0.1	0	0	0
50/10-8	26451	0	0	0	0
50/10-9	31350	0	0	0	0
50/10-10	29825	0.27	0	0	0
100/10-1	82542	0.85	0	0	0
100/10-2	71063	0.92	0.14	0	0
100/10-3	75227	1.02	0.94	0.9	0
100/10-4	70815	1	1.1	0	0.6
100/10-5	77547	0.95	0.16	0	0
100/10-6	70330	1.13	0	0	0
100/10-7	82261	0.86	0.7	0	0
100/10-8	76642	0.87	0.1	0	0
100/10-9	72470	0.99	0.45	0.2	0
100/10-10	74868	0.95	0.13	0	0
500/10-1	942991	2.54	1.34	0.3	0
500/10-2	924830	1.73	1.04	0.7	0
500/10-3	940585	1.27	1.04	0.14	0
500/10-4	955416	0.18	0.29	1.47	0
500/10-5	968999	2.35	0.2	0.14	0
500/10-6	913407	1.69	0.62	0.17	0
500/10-7	923592	3.16	2.59	2.24	0
500/10-8	939949	1.41	0	0.62	0.48
500/10-9	954567	0.97	0.39	0.41	0
500/10-10	978283	2.23	0.79	0.9	0

Tabela 5.17: Comparativo para instâncias de 10 sondas

Instância	Melhor	GRASP	Busca Tabu	GRASP+AM	BLI
50/10-1	30486	0	0	0	0
50/10-2	30933	0.12	0.1	0	0
50/10-3	29352	0	0	0	0
50/10-4	28826	0.2	0.1	0	0
50/10-5	29348	0.18	0	0	0
50/10-6	30462	0.13	0	0	0
50/10-7	37502	0.2	0	0	0
50/10-8	26451	0.15	0	0	0
50/10-9	31350	0.8	0	0	0
50/10-10	29825	0.37	0	0	0
100/10-1	82542	0.28	0.1	0.1	0.3
100/10-2	71063	0.47	0.4	0.1	0.2
100/10-3	75227	4.45	1.0	0.4	0.4
100/10-4	70815	4.41	1.2	0.2	0.5
100/10-5	77547	2.72	0.9	0.1	0.4
100/10-6	70330	5.7	0.4	0.1	0.2
100/10-7	82261	1.31	0.2	0.1	0.3
100/10-8	76642	3.84	0.5	0.1	0.4
100/10-9	72470	3.46	0.9	0.1	0.1
100/10-10	74868	0.1	0.4	0.1	0.6
500/10-1	942991	3.21	1.9	0.8	0.5
500/10-2	924830	2	1.8	1	0.5
500/10-3	940585	2.06	1.4	0.7	0.1
500/10-4	955416	3.01	1.0	2.4	0.4
500/10-5	968999	2.81	0.1	1	0.1
500/10-6	913407	2.73	1.2	1.1	0.9
500/10-7	923592	4.39	2.8	3.1	1.8
500/10-8	939949	2.2	1.5	1.3	0.5
500/10-9	954567	1.84	1.4	1	0.4
500/10-10	978283	2.77	1.3	0.7	0.4

Tabela 5.18: Comparativo das médias para instâncias de 10 sondas

Instância	Poços	Sondas
E-1	8	4
E-2	8	4
E-3	9	4
E-4	9	4
E-5	8	4
E-6	8	4
E-7	8	4
E-8	9	4
E-9	9	4
E-10	9	4

Tabela 5.19: Instâncias exatas

	CPLEX		GRASP		GRASP+MA		Busca Tabu		BLI	
Instância	Resultado	Tempo	Convergência	T. Médio	Convergência	T. Médio	Convergência	T. Médio	Convergência	T. Médio
E-1	4327	3707	0	1800	100	240	100	0.4	100	0.3
E-2	5097	3047	0	1800	100	330	100	0.4	100	0.5
E-3	3268	14104	0	1800	100	108	100	0.1	100	0.2
E-4	4779	21787	100	1	100	61	100	0.2	100	0.2
E-5	3807	2513	0	1800	100	642	100	0.1	100	0.1
E-6	5882	2665	0	1800	100	60	100	0.1	100	0.1
E-7	6869	5485	0	1800	100	120	100	0.1	100	0.1
E-8	4192	26945	0	1800	100	385	100	0.2	100	0.1
E-9	4141	11119	100	1	100	60	100	0.8	100	0.1
E-10	3977	9440	0	1800	100	342	100	0.1	100	0.1

Tabela 5.20: Resultados obtidos para as instâncias exatas

Capítulo 6

Conclusões e Trabalhos Futuros

Neste trabalho foram propostas diferentes heurísticas para a solução do PRSSM. Inicialmente é apresentada uma formulação matemática descrevendo este problema como um problema de programação linear inteira. O objetivo maior desta formulação, é possibilitar o uso de métodos exatos via *software* CPLEX, obtendo-se com isso, soluções ótimas para instâncias de pequeno porte do PRSSM.

A principal contribuição está associada à proposta de heurísticas GRASP, Busca Tabu e Busca Local Iterada (BLI). Neste contexto, além do objetivo de simplesmente resolver aproximadamente o PRSSM, foi feito um estudo empírico sobre o desempenho de heurísticas que fazem uso de memória adaptativa.

As conclusões obtidas na comparação de um GRASP tradicional sem memória e o GRASP+MA, mostraram que a inclusão de memória em um GRASP tende a torná-lo mais competitivo frente a metaheurísticas que tradicionalmente já fazem uso destas estratégias, como é o caso da Busca Tabu e do BLI.

A Busca Tabu aqui proposta, faz uso de conceitos de descoberta de padrões originalmente proposta por Eric Taillard *et al.* [34] para um PRV.

Apesar das idéias propostas por Taillard serem interessantes, os resultados mostraram que alguns aspectos devem ainda ser melhor investigados, como por exemplo: dado um conjunto de rotas de diferentes soluções, como combinar estas rotas para produzir soluções de melhor qualidade.

A BLI obteve desempenho melhor do que o esperado. Esta técnica, que possui semelhanças com a Busca Tabu, se colocou ao lado do GRASP+MA como as heurísticas de melhor desempenho médio nas diferentes simulações efetuadas.

Em relação à trabalhos futuros, pode-se sugerir o desenvolvimento de formulações matemáticas alternativas e outras técnicas exatas, com o intuito de resolver, de forma ótima, um conjunto maior de instâncias do problema.

No tocante a heurísticas, algumas sugestões seriam, tentar fazer melhor uso dos padrões armazenados pela Busca Tabu proposta, e testar outras variantes da BLI, com novas formas de diversificação na etapa de busca.

Outra alternativa que parece promissora, é investir em técnicas híbridas, conjugando métodos exatos com metaheurísticas, usando, por exemplo, a noção de *Local Branching* [13].

Referências

[1] Aiex, R., Resende, M., Ribeiro, C. Probability distribution of solution time in GRASP: An experimentak investigations. *Journal of Heuristics 8* (2002), 343–373.

[2] Aloise, D. J., Aloise, D., Rocha, C. T. M., Ribeiro, C. Scheduling workover rigs for onshore oil production. *Discrete Applied Mathematics 154*, 5 (2006), 695–702.

[3] Besten, M. D., Stützle, T., Dorigo, M. Design of Iterated Local Search Algorithms. In *Applications of Evolutionary Computing. EvoWorkshops2001: EvoCOP, EvoFlight, EvoIASP, EvoLearn, and EvoSTIM. Proceedings* (Como, Italy, 2001), E. J. W. Boers, S. Cagnoni, J. Gottlieb, E. Hart, P. L. Lanzi, G. Raidl, R. E. Smith, and H. Tijink, Eds., vol. 2037, Springer-Verlag, p. 441–451.

[4] Bräysy, O., Gendreau, M. Genetic Algorithms for the Vehicle Routing Problem with Time Windows. Relatório Técnico, Department of Mathematics and Statistcs, University of Vaasa, Vaasa, Finlândia, 2001.

[5] Chao, I. M., Golden, B., Wasil, E. A new heuristic for the multi-depot vehicle routing problem that improves upon best-known solutions. *American Journal of Mathematical and Management Sciences 13* (1993), 371–406.

[6] Christofides, N., Eilon, S. An algorithm for the vehicle-dispatching problem. *Operational Research 20*, 3 (1969), 309–318.

[7] Clarke, G., Wright, J. W. Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research 46* (1964), 93–100.

[8] Cordeau, J.-F., Gendreau, M., Laporte, G., Potvin, J.-Y., Semet, F. A Guide to Vehicle Routing Heuristics. *Journal of the Operational Research Society 53* (2002), 512–522.

[9] Crevier, B., Cordeau, J.-F., Laporte, G. The multi-depot vehicle routing problem with inter-depot routes. *European Journal of Operational Research 176* (2007), 756–773.

[10] Dantzig, G. B., Ramser, J. The truck dispatching problem. *Management Science 6* (1959), 80–91.

[11] Dorigo, M., Stützle, T. *Ant Colony Optimization.* The MIT Press, 2004.

[12] Festa, P., Resende, M. GRASP: An annotated bibliography. In *Essays and surveys in metaheuristics*, C. Ribeiro and P. Hansen, Eds. Kluwer Academic Publishers, 2002, p. 325–367.

[13] Fischetti, M., Lodi, A. Local Branching. *Mathematical Programmiing Series B 98* (2003), 23–47.

[14] Gendreau, M., Laporte, G., Potvin, J.-Y. *The Vehicle Routing Problem.* Society for Industrial and Applied Mathematics, 2001, cap. Metaheuristics for the capacitated VRP, p. 129–154.

[15] Gillett, B. E., Johnson, J. G. Multi-terminal vehicle-dispatch algorithm. *Omega 4* (1976), 711–718.

[16] Glover, F., Laguna, M. *Tabu Search.* Kluwer Academic Publishers, 1997.

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

[18] Golden, B., Wasil, E. A., Kelly, J. P., Chao, I. M. *Fleet Management and Logistics.* Kluwer Academic Publishers, 1998, cap. Metaheuristics in Vehicle routing, p. 33–56.

[19] Hjorring, C. *The Vehicle Routing Problem and Local Search Metaheuristics.* PhD thesis, The University of Auckland, Departament of Engeneering Science, Auckland, Nova Zelandia, 1995.

[20] Holland, J. *Adaptation in Natural and Artificial Systems.* The University of Michigan Press, Ann Arbor, 1975.

[21] Kadluczka, M., Nelson, P., Tirpak, T. Adaptive memory programming: A parameterized framework for meta-seach. Relatório Técnico, Motorola, 2004.

[22] Laporte, G., Nobert, Y., Taillefer, S. Solving a family of multi-depot vehicle routing and location-routing problems. *Transportation Science 22*, 3 (1988), 161–172.

[23] Laporte, G., Semet, F. *The vehicle Routing Problem.* Society for Industrial and Applied Mathematics, 2001, cap. Classical heuristics for the capacitated VRP, p. 109–128.

[24] Lenstra, J., Kan, A. R. Complexity of vehicle routing and scheduling problems. *Networks 11* (1981), 221–228.

[25] Nagy, G., Salhi, S. Heuristic algorithms for single and multiple depot vehicle routing problems with pickups and deliveries. *European Journal of Operation Research 162* (2005), 126–141.

[26] Noronha, T., Aloise, D. J. Algoritmos e estratégias de solução para o problema do gerenciamento de sondas de produção terrestre na bacia petrolífera potiguar. *Revista eletrônica de iniciação científica 1*, 2 (2001), 1–11.

[27] Prais, M., Ribeiro, C. Reactive GRASP: an application to a matrix decomposition problem in TDMA traffic assignment. *INFORMS Journal on Computing 12* (2000), 64–176.

[28] Prais, M., Ribeiro, C. Variação de parâmetros em procedimentos GRASP. *Investigación Operativa 9* (2000), 1–20.

[29] Renaud, J., Laporte, G., Boctor, F. F. A Tabu Search heuristic for the multi-depot vehicle routing problem. *Computers Operational Research 23*, 3 (1996), 229–235.

[30] Resende, M., Ribeiro, C. *Handbook of Metaheuristics*. Kluwer Academic Publishers, 2002, cap. Greedy randomized adaptive search procedures (GRASP), p. 219–249.

[31] Santos, H. G., Merschmann, L. H., Ochi, L. S., Dalboni, F. L., Drummond, L. M. A., Marinho, E. H. An improved evolutionary algorithm with Data Mining for a Vehicle Routing Problem. In *Proc of the VIII SBRN* (2004), vol. 1, p. 1–6.

[32] Santos, H. G., Ochi, L. S., Marinho, E. H., Drummond, L. M. A. Combining an Evolutionary Algorithm with Data Mining to solve a Vehicle Routing Problem. *NEUROCOMPUTING - ELSEVIER 70* (2006), 70–77.

[33] Taillard, E. Heuristic Methods for Large Centroid Clustering Problems. Relatório Técnico, IDSIA, 1998.

[34] Taillard, E., Badeau, P., Gendreau, M., Guertin, F., Potvin, J.-Y. A Tabu Search heuristic for the Vehicle Routing Problem with Soft Time Windows. *Transportation science 31*, 2 (1997), 170–186.

[35] Taillard, E., Gambardella, L., Gendreau, M., Potvin, J.-Y. Adaptive memory programming: A unified view of metaheuristics. *European Journal of Operation Research 135* (2001), 1–16.

[36] Talbi, E.-G. A Taxinomy of Hybrid Metaheuristics. *Journal of Heuristics 8* (2002), 541–564.

[37] Tansini, L., Urquhart, M., Viera, O. Comparing assignment algorithms for the Multi-Depot VRP. Relatório Técnico, Universidade de Montevideo, 2001.

[38] Tillman, F. A. The multiple terminal deliver problem with probabilistic demands. *Transportation Science 3* (1969), 192–204.

[39] Toth, P., Vigo, D., Eds. *The vehicle Routing Problem*. Society for Industrial and Applied Mathematics, 2001.

[40] Toth, P., Vigo, D. Models, relaxations and exact approaches for the capacitated vehicle routing problem. *Discrete Applaid Mathematics 123* (2002), 487–512.

[41] Trindade, V. Desenvolvimento e análise experimental da metaheurística GRASP para um problema de planejamento de sondas de manutenção. Dissertação de Mestrado, Universidade Federal Fluminense - UFF, 2005.

Livros Grátis

(http://www.livrosgratis.com.br)

Milhares de Livros para Download:

Baixar livros de Literatura
Baixar livros de Literatura de Cordel
Baixar livros de Literatura Infantil
Baixar livros de Matemática
Baixar livros de Medicina
Baixar livros de Medicina Veterinária
Baixar livros de Meio Ambiente
Baixar livros de Meteorologia
Baixar Monografias e TCC
Baixar livros Multidisciplinar
Baixar livros de Música
Baixar livros de Psicologia
Baixar livros de Química
Baixar livros de Saúde Coletiva
Baixar livros de Serviço Social
Baixar livros de Sociologia
Baixar livros de Teologia
Baixar livros de Trabalho
Baixar livros de Turismo