UNIVERSIDADE FEDERAL FLUMINENSE

LUCIANA BRUGIOLO GONÇALVES

Heurísticas GRASP para um Problema de Roteamento Periódico de Veículos

NITERÓI

2005

UNIVERSIDADE FEDERAL FLUMINENSE

LUCIANA BRUGIOLO GONÇALVES

Heurísticas GRASP para um Problema de Roteamento Periódico de Veículos

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

Orientadores:

Luiz Satoru Ochi

Simone de Lima Martins

NITERÓI

2005

Heurísticas GRASP para um Problema de Roteamento Periódico de Veículos

Luciana Brugiolo Gonçalves

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

Aprovada por:

Luis Satoru Ochi / IC-UFF (Presidente)

Simone de Lima Martins / IC-UFF

Celso da Cruz Carneiro Ribeiro / IC-UFF

Nair Maria Maia de Abreu / UFRJ

Dario José Aloise / UFRN

Niterói, 14 de outubro de 2005.

À Mamãe e ao Papai.

Agradecimentos

Tenho muito a agradecer e a muitos. Primeiramente quero agradecer a Papai do Céu, por me mostrar o caminho, me dar forças e me acompanhar a cada passo. Agradeço aos meus pais que sempre acreditam que iria dar certo e apoiam-me incondicionalmente. Aos maninhos e à maninha, unidos e torcendo uns pelos outros vamos longe! À toda a família que fez muita falta e que tenho certeza que está torcendo sempre por mim. Ao Vicente que esteve sempre perto e que me deu muita força, principalmente nos passos iniciais, obrigado.

Aos meus grandes orientadores, Satoru e Simone, agradeço pela amizade, por cada palavra de incentivo, por cada agradável reunião, cada dica e pela paciência. Vocês são especiais!

Agradeço também ao prof Dario pela ajuda na definição do problema abordado neste trabalho e pelas valiosas contribuições. Agradeço ainda aos integrantes da banca, profa Nair e prof Celso pelas sugestões dadas no texto final.

A amiga de todas as horas, especialmente naquelas mais difíceis, July, sem essa pessoinha tenho certeza que teria sido muito mais difícil. À galera da República dos UFFanistas pseudo-mineiros, Bruno, Lê, Marcos, mais recentemente, Ary e Stênio obrigado pela companhia e carinho. Aos grandes amigos que fiz, não me referindo apenas ao Ary e ao Rodrigo, mas também a grande Renathinha, Dany, Vivi, Cris, Jacques, Jonivam, Jonny, Ivairton, Idalminha, Lu, Luiz, Glauco, Rafael, Kennedy, Sandô, Haroldo, Geiza, Adriana, André, Adria, enfim, a todos que contribuiram para fazer mais feliz cada dia que passei por aqui. Obrigado pelas mãos sempre dispostas a ajudar, pelo carinho, pela companhia, pelos passeios! Em especial, agradeço ao Stênio pelo carinho, amizade, grande ajuda e pelo empurraõzinho na reta final, valeu Sã.

Não poderia deixar de expressar gratidão àquelas pessoas que incentivaram-me desde os tempos da graduação, em especial ao amigo e professor Raul Fonseca, que com sua dedicação e exemplo de profissional serviu-me de motivação, e sem o qual não teria sequer começado esta empreitada, obrigado!

Resumo

O Problema de Roteamento de Veículos (PRV) consiste em minimizar o custo de atender um conjunto de clientes à partir de uma frota de veículos localizados num depósito central. Várias versões deste problema tem sido estudadas na literatura modelando diferentes aplicações práticas que apresentam objetivos e restrições específicas. Uma destas versões, não obriga a presença de todos os clientes numa solução, ou seja, neste caso, apenas um subconjunto dos clientes devem fazer parte de uma solução viável. Uma outra classe do PRV denominado Problema de Roteamento Periódico de Veículos (PRPV) trata do problema de alocar visitas de clientes para cada dia de um horizonte de planejamento. Esta dissertação apresenta um modelo que reúne restrições destes dois problemas para a solução de uma aplicação real encontrada na região nordeste do Brasil associada à explotação de óleo em poços petrolíferos. O modelo proposto denominado Problema de Roteamento Periódico de Unidades Móveis de Pistoneio (PRP-UMP) difere dos demais PRPV em diversos aspectos. Uma diferença fundamental, é que no PRP-UMP, o número de visitas a cada cliente no período não é previamente definido, aumentando com isso a complexidade do problema abordado. Neste trabalho, são propostos: uma formulação matemática descrevendo o PRP-UMP como um problema de programação linear e versões puras e hibridas da heurística GRASP.

Abstract

The Vehicle Routing Problem (VRP) consists of minimizing the cost of supplying a set of customers by a fleet of vehicles operating from a central depot. Several versions of the VRP have been studied in the literature modeling practical applications that present specific objectives and constraints. One of these versions does not oblige that all customers are visited, i.e., only a subset of them may be supplied. Another class of VRP, called Period Vehicle Routing Problem (PVRP) deals with the problem of designing the visits to the customers for each day of a given period. This work presents a model for a real application found in the Northeastern part of Brazil concerning the exploitation of oil in onshore oil wells by joining the constraints found in these two problems. The proposed model called The Period Bump Mobile Units Routing Problem (PBMURP) differs from the well-known PVRP in several aspects. One major difference between them, responsible for increasing the complexity of the problem, is that, in the PBMURP, the number of visits required by a customer during the period is not previously determined. In this work, are proposed a mathematical formulation describing the PBMURP as an linear programming problem and GRASP heuristics including pure and hybrid versions.

Palavras-chave

1. Roteamento Periódico de Veículos
2. Roteamento de Unidades Móveis de Pistoneio
3. Heurísticas GRASP

Glossário

GRASP	:	*Greedy Randomized Adaptive Search Procedures*
UMP	:	Unidade Móvel de Pistoneio
PRP-UMP	:	Problema de Roteamento Periódico de UMP
IMP	:	Inserção Mais Próxima
BLD	:	Busca Local Diária
BLP	:	Busca Local Periódica

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

O Problema de encontrar rotas eficientes para uma frota de veículos é um dos temas mais abordados na literatura nas áreas de Pesquisa Operacional e Otimização. O modelo básico de Problema de Roteamento de Veículos, aqui denotado por PRV, tem como objetivo minimizar os custos de transporte relacionados ao atendimento de um conjunto de clientes a partir de uma frota de veículos homogêneos localizados num depósito origem. Este modelo básico consiste em gerar um conjunto de rotas (uma para cada veículo) com início e término no depósito de modo que cada cliente seja atendido por um único veículo numa única visita respeitando as restrições de capacidade dos veículos.

A literatura apresenta um razoável número de versões do PRV modelando diferentes aplicações. Em algumas destas versões não é necessário estabelecer rotas que contemplem todos os clientes envolvidos, apenas um subconjunto destes fazem parte da solução viável do problema, como pode ser visto em [Golden et al. 1987], [Keller 1989], [Pearn e Chiens 1998] e [Teeninga e Volgenant 2004]. Uma outra variante do PRV muito abordada pela literatura afim é conhecida como Problema de Roteamento Periódico de Veículos (PRPV) ([Christofides e Beasley 1984]), pela qual o objetivo é planejar visitas para os clientes num dado horizonte de planejamento.

Neste trabalho é abordado um modelo real de roteamento de veículos englobando características destes dois modelos citados anteriormente, ou seja, onde uma solução apresenta, para um subconjunto dos clientes, um planejamento de visitas em um determinado período. O modelo real enfocado está associado a um problema de extração de derivados de petróleo de poços terrestres da região nordeste do Brasil [Aloise et al. 2001].

O Problema de Roteamento Periódico de Unidades Móveis de Pistoneio (PRP-UMP) é um PRPV modelado para o problema de extração de óleo de campos petrolíferos ter-

restres onde é necessário o uso de mecanismos artificiais, denominados Unidades Móveis de Pistoneio (UMP), para a elevação do óleo até a superfície. Neste caso o objetivo é determinar rotas para todas as UMP em cada dia de forma que seja extraída a maior quantidade possível de óleo.

Para resolver este problema, são propostas uma formulação matemática que descreve o PRP-UMP como um problema de programação linear e heurísticas utilizando o conceito de GRASP.

O restante deste trabalho está assim dividido: no Capítulo 2 são apresentados o PRVP e variações abordadas neste trabalho. No capítulo 3 são descritas as heurísticas e a formulação propostas. No capítulo 4 são apresentados os resultados computacionais e, por fim, no capítulo 5 as considerações finais e sugestões de trabalhos futuros.

Capítulo 2

O Problema de Roteamento Periódico de Veículos - PRPV

O modelo básico do Problema de Roteamento de Veículos (PRV) tem como objetivo minimizar os custos de transporte relacionados ao atendimento de um conjunto de clientes usando uma frota de veículos homogêneos a partir de um depósito origem. Este modelo básico consiste em gerar uma rota para cada veículo, com início e término no depósito, de modo que cada cliente seja atendido por um único veículo numa única visita atendendo as restrições de capacidade dos veículos. O PRV é um problema NP-difícil como pode ser visto em [Laporte 1992].

Uma variante muito abordada na literatura é o Problema de Roteamento Periódico de Veículos (PRPV). Este modelo pode ser visto como uma generalização do PRV, dado que o período de planejamento é estendido de um para t dias, onde busca-se determinar os dias em que cada cliente será atendido e as rotas que devem ser executadas por cada veículo durante o período. Neste caso, para cada dia é resolvido um PRV e o objetivo consiste em minimizar o custo de todas as rotas executadas durante todo o período. Na próxima seção é apresentado o PRPV e também duas variações deste modelo.

2.1 Descrição do PRPV

O Problema de Roteamento Periódico de Veículos é uma generalização do clássico PRV onde o horizonte de planejamento é de t dias e, para cada dia, deve-se resolver um PRV com os clientes selecionados para serem visitados neste dia. Para este problema, considera-se $N = \{1, 2, \ldots, n\}$ o conjunto dos clientes, e r_i $(1 \leq r_i \leq t)$ o número de visitas que devem ser realizadas a cada cliente $i \in N$ durante o horizonte de planejamento. Além

disso, cada cliente possui um conjunto de possíveis combinações de dias de visita $V(i)$, onde cada combinação é composta por r_i dias em que a cidade i deve ser visitada. Por exemplo, se uma cidade j requer três visitas durante o período de $t = 5$ dias e existindo duas possíveis combinações, $V(j) = \{\{1, 3, 4\}, \{2, 3, 5\}\}$, então, deve-se visitar a cidade j nos dias 1, 3 e 4 se a combinação $\{1, 3, 4\}$ for selecionada ou nos dias 2, 3 e 5, caso a segunda combinação seja escolhida. Desta forma, no PRPV deve-se simultaneamente selecionar uma combinação de visitas para cada cliente e determinar rotas para os veículos em cada dia de modo que a distância total percorrida no horizonte de planejamento seja minimizada.

Alguns trabalhos relevantes disponíveis na literatura sobre este problema merecem destaque. As heurísticas propostas em [Christofides e Beasley 1984] são baseadas na escolha inicial de combinações de dias de visitação que atendam aos clientes, seguido por um procedimento de troca de combinações onde busca-se minimizar o custo associado às rotas. A dificuldade desta abordagem para o PRPV está relacionada ao alto custo da avaliação do efeito de uma troca de combinação, visto que torna-se necessário resolver um PRV para cada dia envolvido na alteração. Por isto, os autores consideram duas relaxações. Na primeira destas, aproxima-se o PRV de cada dia a um Problema de Medianas e na segunda a um Problema do Caixeiro Viajante.

Em [Tan e Beasley 1984] foi apresentado um procedimento para solucionar o PRPV baseado na heurística de alocação proposta por [Fisher e Jaikumar 1981] aplicada ao problema de roteamento de veículos. Inicialmente, pontos sementes são determinados, em seguida um problema de programação inteira relaxado é solucionado para associar combinações aos clientes, considerando para isto a distância dos clientes aos pontos sementes. Finalmente, a heurística de alocação é aplicada para solucionar o PRV em cada dia do período de planejamento.

Uma abordagem multi-fase foi proposta em [Russel e Gribbin 1991] para solucionar o PRPV. Na primeira fase, combinações são associadas a clientes utilizando uma aproximação que encontra um ponto semente em cada dia do período de planejamento. A combinação alocada é determinada de acordo com a soma das distâncias dos clientes aos pontos sementes de cada dia da combinação. Na segunda fase, o procedimento de troca proposto em [Christofides e Beasley 1984] é utilizado buscando refinar a solução corrente pela troca das combinações às quais os clientes estão associados. Por fim, na última fase, um método de programação inteira é aplicado para tentar melhorar a solução.

Em [Golden et al. 1995] é apresentada uma abordagem baseada em um procedimento

de melhoria denotado *record-to-record* de [Dueck 1990]. Esta heurística produz uma solução inicial e utiliza um processo de refinamento. Para obter a solução inicial é utilizado um modelo *zero-um* relaxado com o intuito de obter soluções onde a demanda total dos clientes de cada dia seja balanceada. Após a obtenção da solução inicial, é aplicado um processo de refinamento para tentar produzir melhores soluções, processo este que consiste num passo de atualização, no qual move-se um ou mais clientes de suas rotas atuais para novas rotas, mudando ou não os dias de visitação.

Uma heurística Busca Tabu (BT) foi proposta em [Cordeau et al. 1997] para solucionar três tipos de problema de roteamento: Problema de Roteamento Periódico de Veículos, Problema do Caixeiro Viajante Periódico e Problema de Roteamento Periódico Multi-depósito. O método proposto tem como principais características: a utilização da heurística GENIUS [Gendreau et al. 1992], usada para efetuar a inserção de clientes nas rotas ou para remover clientes de suas rotas atuais e reinserí-los em outras rotas; a permissão de gerar soluções intermediárias inviáveis; e o emprego de um esquema de diversificação baseado em uma função de penalização.

Em [Angelelli e Speranza 2002] é proposta uma heurística BT para uma extensão do PRPV denominado *Periodic Vehicle Routing Problem with Intermediate Facilities* - PVRP-IF, onde os veículos podem renovar suas capacidades em pontos intermedários. Desta forma, os veículos necessitam retornar aos ponto de partida somente ao final da jornada, visto que quando o limite de sua capacidade é alcançado o veículo renova sua capacidade descarregando (ou carregando) nos pontos intermediários. Da mesma forma que no PRPV, o PVRP-IF consiste em associar a cada cliente uma combinação de dias de visitação e, para cada dia do horizonte de planejamento, encontrar rotas para os veículos de forma que a distância total percorrida por estes seja minimizada. A BT proposta teve como base o algoritmo apresentado em [Cordeau et al. 1997]. Duas variações do procedimento de construção também são apresentados. Além dos movimentos na vizinhança aplicados em [Cordeau et al. 1997], dois novos movimentos foram propostos. Para avaliação do algoritmo, foram feitos testes com as instâncias do PRPV e geradas instâncias específicas para o PVRP-IF.

Na próxima seção são apresentadas duas variações do PRPV abordadas neste trabalho.

2.2 Duas Variações do Problema de Roteamento Periódico de Veículos

Nesta seção são apresentadas duas variações do PRPV abordadas de modo a atender características particulares de algumas aplicações. A primeira delas é uma simplificação do PRPV, denominado o Problema do Caixeiro Viajante Periódico, e a segunda trata-se de uma generalização do mesmo.

2.2.1 O Problema do Caixeiro Viajante Periódico

Considerando uma frota composta por um único veículo de capacidade ilimitada, o PRPV é reduzido ao Problema do Caixeiro Viajante Periódico (PCVP). Nesta variante do PRPV, para cada dia do período de planejamento deve-se buscar um ciclo Hamiltoniano de custo mínimo que envolva todos os clientes marcados para visitação neste dia.

O objetivo deste problema é atender a demanda de todas as cidades no horizonte de planejamento minimizando a distância percorrida. O PCVP é um problema NP-difícil [Bodin e A. Assad 1983] presente em várias aplicações, incluindo problemas de distribuição de alimentos, entrega/coleta postal e coleta de lixo urbano [Bertazzi et al. 2004], [Chao et al. 1995], [Christofides e Beasley 1984], [Cordeau et al. 1997], [Ochi et al. 2001], [Paletta 2002].

Algoritmos aproximados já foram propostos para solucionar o PCVP, dentre os quais pode-se destacar o trabalho de [Chao et al. 1995], que propôs um procedimento heurístico que constrói uma solução inicial e, na etapa de refinamento, faz uso de uma rotina baseada na heurística *record-to-record* [Dueck 1990].

Em [Paletta 2002] é proposta uma heurística que combina um algoritmo construtivo com um procedimento de refinamento da solução parcial, e em [Bertazzi et al. 2004] é apresentada uma versão aperfeiçoada deste algoritmo que, segundo seus autores, apresentam os melhores resultados heurísticos para as instâncias do PCVP analisadas.

2.2.2 O Problema de Roteamento Periódico de Unidade Móvel de Pistoneio

Nos campos petrolíferos terrestres onde há incidência de poços que não têm capacidade própria de elevação de fluido até a superfície, referenciado como poços não surgentes, há necessidade de utilização de mecanismos artificiais para a extração do petróleo. Tais

mecanismos podem ser divididos em fixos e móveis. Quando um poço não apresenta vazão suficiente que justifique o custo da utilização de métodos fixos de elevação artificial (bombeio mecânico, bombeio por cavidade progressivas, etc), a utilização dos mecanismos móveis (Unidade Móvel de Pistoneio, por exemplo) apresenta-se como uma boa estratégia para a solução do problema [Aloise et al. 2001].

Uma Unidade Móvel de Pistoneio (UMP) é um mecanismo móvel composto por acessórios de elevação de fluido montados sobre um caminhão, como mostra a Figura 2.1, ou acoplado a um trator. No uso da UMP, o fluido coletado é inicialmente armazenado na própria unidade e posteriormente transferido para a estação de tratamento de óleo - ETO. Tal estação é o local de onde toda UMP parte e para onde deve retornar após concluir o pistoneio[1] de um conjunto de poços, sempre respeitando um tempo máximo determinado pela jornada de trabalho. Como a capacidade do tanque da UMP é limitada, utiliza-se um caminhão tanque como apoio para realizar o transporte do fluido, deixando assim a UMP com a tarefa exclusiva de pistonear os poços.

Após a retirada de fluido, o processo de reenchimento do poço inicia-se e prossegue até atingir o nível estático. Desta forma, é necessário aguardar o tempo de reenchimento para que uma nova coleta seja feita.

No Problema de Roteamento Periódico de Unidades Móveis de Pistoneio (PRP-UMP), dentro de um horizonte de planejamento deve-se definir, para cada dia, os poços a serem pistoneados por cada UMP, bem como a ordem de visitação, o que determina uma rota. Tal rota é definida no subconjunto de poços que já tenham alcançado seu nível estável até aquele dia. Tendo em vista que se utilizam caminhões tanque como apoio às UMP, a restrição relativa à capacidade da unidades pode ser relaxada. Além disso, o intervalo entre as visitas a um dado poço não é fixo, devendo-se apenas respeitar o tempo mínimo para o reenchimento do mesmo. Por exemplo, um poço que é visitado no primeiro dia do período de planejamento e que tem tempo de reenchimento de três dias, somente poderá ser novamente pistoneado a partir do quarto dia. Desta forma é possível que alguns poços não sejam atendidos durante o período ou que sejam visitados numa menor freqüência que a permitida. Isto pode ocorrer na prática devido ao elevado número de poços e ao número reduzido de UMP's disponíveis. O objetivo do PRP-UMP é gerar rotas diárias para cada UMP de modo a maximizar a quantidade de fluido coletado durante todo o período de planejamento respeitando as restrições envolvidas, como jornada de trabalho e intervalo de reenchimento dos poços.

[1] Ato ou efeito de coletar fluido do poço.

Figura 2.1: UMP acoplada a um caminhão tanque [Aloise et al. 2001].

O PRP-UMP é um problema proposto neste trabalho. Na literatura existem abordagens referentes ao roteamento de UMP para um único dia. Em [Aloise et al. 2001] são apresentadas uma formulação matemática e uma heurística GRASP para o problema diário de roteamento de UMP. Em [Aloise et al. 2002] é proposto um algoritmo heurístico que utiliza o modelo de Colônia de Formigas para o problema de roteamento de UMP.

Capítulo 3

Propostas

Neste capítulo, são apresentadas propostas de heurísticas para os dois problemas de roteamento periódico de uma frota de veículos descritos no capítulo anterior. O primeiro problema é um caso particular do PRVP composto de um único veículo com capacidade ilimitada, conhecido na literatura como Problema do Caixeiro Viajante Periódico (PCVP). O segundo problema é o PRP-UMP descrito anteriormente. Para o PCVP é proposta uma heurística GRASP. Para o PRP-UMP apresenta-se uma formulação matemática e heurísticas de construção e busca local para serem utilizadas pelas heurísticas GRASP.

3.1 Considerações Básicas sobre GRASP

A metaheurística GRASP (*Greedy Randomized Adaptive Search Procedures*) é basicamente um processo iterativo do tipo multi-partida onde cada iteração consiste de duas fases: construção e busca local. A fase de construção consiste de um procedimento para determinar soluções viáveis. As vizinhanças destas soluções são investigadas na fase de busca local até que um ótimo local seja encontrado. Cada iteração GRASP trabalha de forma independente e o resultado final é então a melhor solução dentre todas as encontradas ao final da execução de todas as iterações ([Feo e Resende 1995]).

A cada execução da etapa de construção, uma solução viável é gerada de forma incremental, onde em cada passo um novo elemento é inserido na solução parcial. Para seleção do próximo elemento a ser incorporado na solução, todos os candidatos são avaliados segundo uma função gulosa e seleciona-se um subconjunto formado pelos melhores candidatos para serem inseridos em uma lista denominada lista restrita de candidatos LRC. O elemento a ser introduzido na solução parcial é selecionado aleatoriamente entre os pertencentes à LRC. Estas escolhas aleatórias permitem gerar soluções diferentes a

cada execução do algoritmo. Uma vez determinado o elemento, este é inserido na solução parcial, os custos da função de avaliação são recalculados e a LRC é atualizada.

A fase de busca local tenta aprimorar a solução S gerada na fase de construção efetuando uma varredura na vizinhança de S até que um ótimo local seja encontrado.

O modelo do GRASP descrito até aqui se refere a sua versão básica. Existem várias estratégias sugeridas para o aperfeiçoamento desta metaheurística, tais como: uso de tabela *hash* para armazenar as soluções construídas de forma a evitar o trabalho redundante pela execução de procedimento de busca local sobre uma mesma solução inicial e sobre soluções construídas de baixa qualidade [Martins et al. 1999]; mecanismos de memória de longo prazo [Fleurent e Glover 1999]; GRASP reativo [Prais e Ribeiro 2000]; e Reconexão de Caminhos [Resende e Ribeiro 2005].

Nas heurísticas propostas neste trabalho, faz-se o uso do GRASP reativo e da estratégia de Reconexão de Caminhos (RC).

3.2 Uma Heurística GRASP para o PCVP

O objetivo principal deste trabalho é o desenvolvimento de propostas para resolver o PRP-UMP. Para isto, inicialmente foram estudadas as principais contribuições existentes para diferentes modelos de PRPV.

Um caso particular do PRPV é o Problema do Caixeiro Viajante Periódico (PCVP). Para esta versão, foi verificada a existência de um algoritmo heurístico construtivo embutindo etapas de refinamento durante a construção da solução [Paletta 2002], cujas características poderiam, a princípio, ser aproveitadas para outros modelos de PRPV, incluindo a aplicação proposta neste trabalho (PRP-UMP).

Desta forma, numa etapa preliminar, foi implementado o algoritmo construtivo proposto em [Bertazzi et al. 2004], apresentado na próxima seção, que é uma versão aperfeiçoada do algoritmo proposto em [Paletta 2002]. Uma versão randomizada deste algoritmo acoplado a uma busca local é então proposta para ser utilizada no algoritmo GRASP que será apresentado na Seção 3.2.2.

3.2.1 Algorítmo Heurístico - H_BPS

A heurística proposta por [Bertazzi et al. 2004], aqui denominada H_BPS (Heurística proposta por Bertazzi, Paletta e Speranza), é uma combinação de um processo de cons-

trução iterativo com um procedimento de refinamento. O seu funcionamento pode ser visto de forma simplificada no Algoritmo 1, cujos os dois primeiros parâmetros são utilizados nas etapas de aperfeiçoamento e o terceiro indica a primeira cidade a ser inserida nas rotas. Neste algoritmo, N representa o conjunto das cidades a serem atendidas pelo caixeiro viajante e $V(f)$ o conjunto das possíveis combinações de visita da cidade f. Inicialmente, a solução S é inicializada contendo apenas a cidade de partida do caixeiro (linha 1). Em seguida, a primeira cidade f é inserida nas rotas após uma combinação v_f do conjunto $V(f)$, que contém as combinações disponíveis para f, ser selecionada aleatoriamente (linha 2). Assim, é possível inserir a cidade f nas rotas dos dias pertencentes a v_f (linha 3), conforme as regras de inserção propostas para esta abordagem, que serão descritas posteriormente. Um conjunto auxiliar Z é criado para armazenar o conjunto das cidades que já foram inseridas na solução parcial, que inicialmente contém apenas a primeira cidade f (linha 4).

Algoritmo 1 *H_ BPS* (p_1, p_2, f)

```
1: Inicialização(S);
2: v_f ← Combinação_A(V(f));
3: Inserção(S, f, v_f);
4: Z = {f}
5: enquanto N − Z ≠ ∅ faça
6:    Insere_ Cidade (S, N − Z);
7:    se Interrupção(p_1) então
8:       Aperfeiçoamento(S, p_2);
9:    fim se
10: fim enquanto
11: se Inviável(S) então
12:    Viabilização(S);
13: fim se
14: Aperfeiçoamento_Modif(S, p_2);
```

Após a etapa de inicialização (linhas 1-4), realiza-se a construção através da execução do procedimento *Insere_ Cidade*, acionado na linha 6, que consiste em selecionar uma cidade k ainda não selecionada e a esta associar uma das combinações do seu conjunto $V(k)$. Para cada dia da combinação associada a esta cidade, é feita a inserção da mesma nas rotas dos dias correspondentes. Assim, a partir do início do processo de construção, existem rotas parciais para cada dia, ou seja, as rotas são construídas em paralelo. Este processo é executado enquanto houver cidade ainda não atendida pelo caixeiro viajante (linha 5). Durante este processo, toda vez que o número de cidades processadas for múltiplo de um certo parâmetro p_1, ocorrem interrupções para que o aperfeiçoamento seja executado, almejando-se diminuir o custo total das rotas, como mostrado nas linhas

7 e 8. O parâmetro p_2 ajusta a granularidade dos procedimentos de aperfeiçoamento descritos adiante.

Nas linhas 11-13, o algoritmo realiza um procedimento para tornar a solução viável. Para que a solução seja viável, as rotas de cada dia do período de planejamento não podem estar vazias, ou seja, cada rota deve ter pelo menos uma cidade a ser visitada pelo caixeiro. Caso a solução construída seja inválida, é executado o procedimento denominado *Viabilização* para viabilizar a solução gerada. Na linha 14, aplica-se à solução viável um outro procedimento, denominado $Aperfeiçoamento_{Modif}$, muito similar ao procedimento apresentado na linha 8, mas que garante a manutenção da viabilidade da solução.

3.2.1.1 Procedimento *Insere_ Cidade*

Este procedimento, cujo pseudocódigo é apresentado no Algoritmo 2, é aplicado para selecionar a próxima cidade que será inserida na solução parcial, associar a esta uma combinação de dias de visita e adicioná-la nas rotas dos dias da combinação escolhida.

Algoritmo 2 *Insere_ Cidade* (S, $N - Z$)

1: $s \leftarrow$ *Custo_ Seleção_ Max* ($N - Z$);
2: $v_S \leftarrow$ *Combinação* ($V(f)$);
3: *Inserção* (S, s, v_S);
4: $Z = Z \cup \{s\}$

Na linha 1 do Algoritmo 2, o custo da seleção de cada cidade ainda não visitada, dado pela Equação 3.1, é calculado e aquela de maior custo de seleção é escolhida para fazer parte da solução. Na Equação 3.1 pode-se verificar que o custo de seleção $cs(i)$ de uma cidade i é dado pelo número de visitas exigidas pela cidade i (r_i), multiplicado pela distância euclidiana d_{ij} entre i e seu vizinho mais próximo já atendido ($min_{j \in Z} d_{ij}$). A cidade s com custo de seleção máximo é selecionada.

$$cs(i) = r_i \times min_{j \in Z}\ d_{ij} \tag{3.1}$$

Na linha 2, define-se a combinação de dias de visitação a ser associada à cidade s. Obtém-se o custo de inserir s em cada dia do período de planejamento. Depois disso, para cada combinação $v \in V(s)$, associa-se um custo calculado pela soma dos custos de inserção de s em cada dia de v. A notação v_S é atribuída àquela combinação de menor custo. Estabelecida v_S, na linha 3 do algoritmo a cidade é inserida em cada dia desta combinação e, por fim, o conjunto Z das cidades já presentes na solução parcial é atualizado na última linha do algoritmo.

3.2.1.2 Procedimento *Aperfeiçoamento*

O procedimento *Aperfeiçoamento*, que está descrito no Algoritmo 3, é executado quando ocorre uma interrupção durante a etapa de construção. Para este algoritmo, as cidades pertencentes ao conjunto Z são divididas em subconjuntos G_k contendo p_2 cidades. Na tentativa de melhorar a solução corrente S, remove-se temporariamente as cidades pertencentes a um subconjunto G_k da solução corrente, gerando a solução temporária S_{tmp} (linha 2). Para cada uma das cidades pertencentes a G_k, busca-se uma nova combinação de visitas de forma que o custo de inserção seja mínimo e então, atualiza-se S_{tmp} (linha 3). Se o custo da solução temporária S_{tmp} for menor que o custo da solução corrente, S é atualizada (linha 4 e 5).

Algoritmo 3 *Aperfeiçoamento*(S, p_2)

1: **para** *cada* G_k **faça**
2: $S_{tmp} \leftarrow Remoção_{tmp}(S, G_k)$;
3: $Seleciona\_Combinações(S_{tmp}, G_k)$;
4: **se** Custo(S_{tmp}) < Custo(S) **então**
5: $S \leftarrow S_{tmp}$;
6: **fim se**
7: **fim para**

3.2.1.3 Procedimento *Viabilização*

Após a fase de construção, pode ser necessário executar o procedimento *Viabilização* caso a rota de algum dia tenha permanecido vazia. Para tanto, uma das cidades pertencentes à rotas com pelo menos outra cidade é selecionada para ter sua combinação de dias de visita alterada. Para determinar a nova combinação a ser associada a cidade selecionada, verifica-se entre as possíveis combinações aquela que contém o maior número possível de rotas vazias. A cidade é então removida das rotas da combinação anterior e inserida nas rotas da nova combinação. Este procedimento é repetido enquanto houver rotas vazias.

3.2.1.4 Procedimento $Aperfeiçoamento_{Modif}$

O $Aperfeiçoamento_{Modif}$ é muito similar ao procedimento *Aperfeiçoamento*. A diferença é que, neste caso, há uma restrição que garante que as alterações não afetarão a viabilidade da solução. Cada cidade é examinada para trocar de combinação somente se esta pertence à rotas com pelo menos uma outra cidade, evitando, desta forma, que sejam geradas rotas vazias.

3.2.1.5 Regras de Inserção e Remoção

Nos procedimentos apresentados anteriormente, quando há necessidade de inserir ou remover uma cidade de uma rota, aplicam-se as regras apresentadas a seguir.

3.2.1.5.1 Regras para Inserção A regra de inserção é utilizada para incluir uma cidade i em uma determinada rota, um processo necessário aos procedimentos *Insere_ Cidade* e *Aperfeiçoamento* do H_BPS. Se a rota onde a cidade será inserida estiver vazia, então a cidade i deve ser conectada diretamente com a cidade origem, ou seja, cria-se um laço ligando a cidade origem à cidade i. Mas, se a rota já possuir alguma cidade, então é necessário verificar em que posição a nova cidade causa menor acréscimo ao custo da rota. Considere $su(j)$ o sucessor do no j na rota R já exixtente para o dia d. Para cada par de arcos $(x,\ su(x))$ e $(y,\ su(y))$, os métodos mostrados a seguir são aplicados.

- Método 1:
  Remover os arcos $(x,\ su(x))$ e $(y,\ su(y))$ da rota, onde $x \neq y$, reconectando-a com os arcos $(x,\ i)$, $(i,\ y)$ e $(su(x),\ su(y))$, invertendo o caminho entre as cidades y e $su(x)$, conforme ilustrado na Figura 3.1.

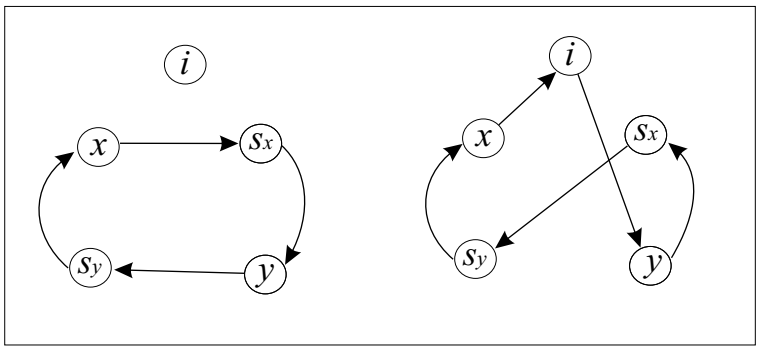


Figura 3.1: Método de Inserção 1

- Método 2:
  Remover da rota os arcos $(x,\ su(x))$ e $(y,\ su(y))$, onde $x \neq y$, e inserir os arcos $(su(x),\ i)$, $(i,\ su(y))$ e $(x,\ y)$, invertendo o caminho entre as cidades y e $su(x)$, conforme ilustrado na Figura 3.2.

Todos os pares de arcos são examinados e a rota corrente será aquela que obtiver o menor custo. O custo de inserção de uma cidade em uma rota é igual ao somatório dos custos dos arcos inseridos menos o somatório dos custos dos arcos removidos.

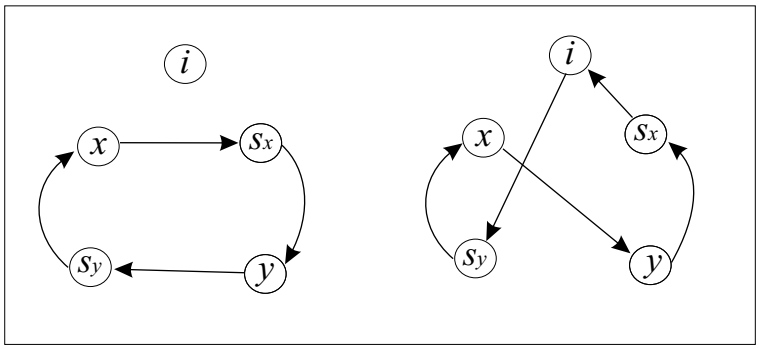


Figura 3.2: Método de Inserção 2

3.2.1.5.2 Regra para Remoção Para remover uma cidade r da rota de um determinado dia do período, os arcos ligados à cidade r são removidos, deixando a rota aberta. Então, para cada par de arcos remanescentes, é feita a remoção do arco de forma a gerar uma floresta com três árvores, que são então conectadas pelos arcos que impliquem em menor custo. Depois de examinar todas as possibilidades, a rota que apresentar menor custo entre todas as obtidas torna-se a rota corrente.

3.2.2 Heurística GRASP proposta para o PCVP

Como apresentado em [Bertazzi et al. 2004], os resultados desta heurística são os melhores conhecidos para um grupo de trinta e três instâncias da literatura. Analisando as instâncias sobre as quais os testes foram feitos, é possível verificar que o custo computacional para alcançar tais resultados pode ser demasiadamente elevado. Mesmo que cada execução da heurística demore em média alguns segundos, para se testar todas as possíveis combinações de parâmetros sugeridas pelos autores ($p_1 \in \{1, 2, 3, \ldots, 20\}$, $p_2 \in \{1, 2, 3, \ldots, 20\}$ e $f \in N$), o tempo computacional aumenta consideravelmente. Sabendo que o número médio de cidades nas instâncias testadas é 119, há em média $20 \times 20 \times 119 = 47.600$ possíveis combinações de parâmetros para cada instância. Deste modo, para alcançar os resultados apresentados em [Bertazzi et al. 2004], o H_BPS necessita de milhares de execuções, onerando significativamente o seu tempo computacional final.

A constatação de que o algoritmo H_BPS obtém as melhores soluções aproximadas para o PCVP, mas apresentando um alto custo computacional para determiná-las, motivou o desenvolvimento de uma heurística inspirada no H_BPS que pudesse alcançar resultados similares exigindo um menor esforço computacional. Desta forma, é apresentado um GRASP inspirado na heurística H_BSP onde a fase de construção é uma adaptação do procedimento *Insere_Cidade* e a busca local é realizada executando-se o

procedimento $Aperfeiçoamento_{Modif}$.

3.2.2.1 Fase de Construção

Para a etapa de construção proposta é necessário fazer uma adaptação no procedimento *Insere_ Cidade* para agregar a este as características da construção do GRASP (gulosa, aleatória e adaptativa).

Algoritmo 4 *Construção*(S, α , f, p_1, p_2, p)

```
1: Inicialização(S);
2: V_f ← Combinação_A (V(f));
3: Inserção (S, f, V_f);
4: LC ← Inicializa_ LC (N − {f});
5: enquanto LC ≠ ∅ faça
6:    LRC ← Atualiza_ LC (α, LC)
7:    Insere_ Cidade_G_GMO (LRC, S);
8:    se Interrupção (p_1, p) então
9:       Aperfeiçoamento (S, p_2);
10:   fim se
11: fim enquanto
12: se Inviável (S) então
13:    Viabilização (S);
14: fim se
```

O funcionamento da fase de construção pode ser visto no Algoritmo 4. Os parâmetros deste procedimento são os mesmos que aparecem na heurística já apresentada, havendo apenas a inclusão de p e α, onde p indica o número de iterações do GRASP em que o algoritmo de aperfeiçoamento é executado durante a fase de construção e α quantifica os elementos da LRC.

Como pode ser visto nas linhas de 1 a 3, a inicialização da solução é a mesma que aparece na heurística H_BSP. Após inicializada a solução, na linha 4 todas as cidades ainda não visitadas são colocadas na lista LC (Lista de Candidatas), em ordem não decrescente de acordo com a função de avaliação apresentada na Equação 3.1. Enquanto houver alguma cidade ainda não visitada ($LC \neq \emptyset$), é preciso efetuar as operações apresentadas entre as linhas 5 e 11 para a inserção destas cidades. Na linha 6 atualiza-se o valor da função de avaliação das cidades a serem inseridas e selecionam-se $\alpha \times |LC|$ elementos de LC para formarem a LRC (Lista Restrita de Candidatos), onde $0 < \alpha \leq 1$. Na linha 7, executa-se uma adaptação do procedimento de [Bertazzi et al. 2004]. Para determinar a próxima cidade a ser inserida na solução S, ao invés de se escolher sempre a cidade melhor classificada como em [Bertazzi et al. 2004], é feita uma seleção aleatória entre as cidades

pertencentes a LRC, sendo os demais passos deste procedimento comuns a ambos os algoritmos. A função de aperfeiçoamento da linha 9 somente é executada nas p primeiras iterações do GRASP. Na linha 8, o parâmetro p_1 define quando o aperfeiçoamento deve ser aplicado nas p primeiras iterações. A viabilização apresentada nas linhas 12-14 é a mesma da heurística H_BSP.

3.2.2.2 Fase de Busca Local

Após cada construção é executado o procedimento $Aperfeiçoamento_{Modif}$, buscando na vizinhança da solução atual uma solução com menor tempo total de percurso.

O Algoritmo 5, resume o funcionamento do GRASP proposto. O parâmetro α não tem um valor fixo durante toda a execução do algoritmo. O conjunto $Vetor\_\alpha$, possui alguns valores pré-definidos que são assumidos pela variável α durante o processamento. As Max_it iterações são divididas de forma que o algoritmo execute um determinado número de vezes com cada valor do conjunto.

Algoritmo 5 $G\_GMO(f, p_1, p_2, p, Max\_it)$

1: **para** iteração $\leftarrow 1, \ldots Max\_it$ **faça**
2: $\alpha \leftarrow Sorte\ (Vetor\_\alpha)$;
3: $Construção(S, \alpha, f, p_1, p_2, p)$;
4: $Aperfeiçoamento_{Modif}(S, p_2)$;
5: **se** *Custo(solução_atual)* $<$ *Custo(solução_best)* **então**
6: solução_*best* $\leftarrow$ solução_atual;
7: **fim se**
8: **fim para**

Na linha 1, executa-se o controle do número máximo de iterações a serem realizadas, Max_it. Antes de executar o procedimento de construção é escolhido um valor para α, que será utilizado durante a iteração atual, como mostrado na linha 2. O procedimento de construção é executado na linha 3 e, com uma solução válida construída, é executado o procedimento de busca local na linha 4. Se a solução obtida tiver um custo menor que a melhor das soluções encontradas até o momento, esta é atualizada com o valor da primeira (linhas 5-6).

Deve-se ressaltar que foi feita uma alteração na forma de tratamento dos parâmetros p_1 e p_2 para uma melhor adaptação ao tamanho de cada instância, enquanto na heurística H_BPS da literatura, um mesmo valor fixo é utilizado para instâncias de diferentes dimensões. Neste trabalho optou-se por operar com o percentual do número de cidades para uma melhor adequação dos parâmetros às dimensões das instâncias. Por exemplo,

uma instância com 50 cidades e outra com 200 reagem de forma diferente se suas cidades são retiradas em grupos de 15 cidades. Para a primeira instância este valor pode ser grande demais, já para a segunda pode ser um valor razoável. Mas, para as mesmas instâncias, se as cidades são retiradas em grupos de 15% do número de cidades, espera-se um aproveitamento semelhante em ambas às instâncias.

3.3 Propostas para o PRP-UMP

Nesta seção são apresentadas uma formulação matemática e as heurísticas propostas para o Problema de Roteamento Periódico de Unidades Móveis de Pistoneio (PRP-UMP).

3.3.1 Formulação Matemática

Considere um grafo não direcionado $G = (V, E)$, onde V representa o conjunto de poços acrescido da Estação de Tratamento de Óleo (ETO), com $|V| = n$, e E o conjunto de arestas (u, v) às quais encontram-se associados tempos t_{uv} que indicam o tempo de percurso entre os poços u e v. O PRP-UMP consiste em construir, para um período de p dias, rotas com duração máxima de $TMAX$ unidades de tempo para cada uma das r UMP's. A cada poço $v \in V$ é associado um tempo de reenchimento, um tempo de permanência[1] e o valor da vazão esperada. Desta forma, as constantes e varíaveis empregadas neste modelo são definidas como segue.

Constantes

$\widehat{V}$ = conjunto dos índices dos poços, onde p_0 = ETO. V = $\widehat{V} \cup \{0\}$.
D = período de planejamento, $|D| = p$
R = número de veículos (UMP), $|R| = r$
$T = (t_{ij})$; onde t_{ij} é o tempo de percurso entre i e j, $\forall i, j \in V$
v_i = vazão esperada do poço i
s_i = tempo de permanência no poço i
g_i = tempo de reenchimento do poço i
TMAX = tempo máximo para concluir uma jornada diária

[1] tempo necessário para instalação e desinstalação da UMP acrescido do tempo de coleta.

Variáveis

$$x_{it} = \begin{cases} 1, & \text{se poço } i \text{ for visitado no dia } t \\ 0, & \text{caso contrário} \end{cases}$$

$$z_{ijkt} = \begin{cases} 1, & \text{se poço } j \text{ é visitado logo após o poço } i \text{ pela UMP } k \text{ no dia } t \\ 0, & \text{caso contrário} \end{cases}$$

w_{ijkt} = quantidade de óleo transportado no percurso (i,j) pela UMP k no dia t

A seguir é apresentada uma formulação matemática que descreve o PRP-UMP como um problema de Programação Linear.

Função Objetivo:

$$Maximizar\ F = \sum_{t=1}^{p} \sum_{i=1}^{n} v_i \times x_{it} \tag{3.2}$$

Restrições:

$$\sum_{i=0}^{n} \sum_{j=0, j\neq i}^{n} (t_{ij} + s_i) \times z_{ijkt} \leq TMAX; \qquad \forall t \in D \quad \text{e} \quad \forall k \in R; \tag{3.3}$$

$$\sum_{k=1}^{r} \sum_{i=0; i\neq q}^{n} z_{iqkt} = x_{qt}; \quad \forall q \in \widehat{V} \quad \text{e} \quad \forall t \in D; \tag{3.4}$$

$$\sum_{k=1}^{r} \sum_{i=0; i\neq q}^{n} z_{qikt} = x_{qt}; \quad \forall q \in \widehat{V} \quad \text{e} \quad \forall t \in D; \tag{3.5}$$

$$\sum_{i=0; i\neq q}^{n} z_{iqkt} - \sum_{i=0; i\neq q}^{n} z_{qikt} = 0; \quad \forall q \in \widehat{V}, \quad \forall t \in D \quad \text{e} \quad \forall k \in R; \tag{3.6}$$

$$\sum_{k=1}^{r} z_{ijkt} \leq \frac{x_{it} + x_{jt}}{2}; \quad \forall i \in \widehat{V}, \quad j \in \widehat{V}, \quad \text{com} \quad i \neq j \quad \text{e} \quad \forall t \in D; \tag{3.7}$$

$$x_{0t} = 1; \forall t \in D; \tag{3.8}$$

$$\sum_{j=1}^{n} z_{0jkt} = 1; \qquad \forall t \in D \quad \text{e} \quad \forall k \in R; \tag{3.9}$$

$$\sum_{i=1}^{n} z_{i0kt} = 1; \qquad \forall t \in D \quad \text{e} \quad \forall k \in R; \tag{3.10}$$

$$\sum_{j=0;j\neq q}^{n} w_{qjkt} = \sum_{i=0;i\neq q}^{n} w_{iqkt} + v_q \times z_{iqkt}; \qquad \forall k \in R, \quad \forall t \in D \quad \text{e} \quad \forall q \in \widehat{V}; \tag{3.11}$$

$$w_{0jkt} = z_{0jkt}; \quad \forall j \in \widehat{V}, \quad \forall k \in R \quad \text{e} \quad \forall t \in D; \tag{3.12}$$

$$w_{ijkt} \geq z_{ijkt}; \quad \forall i \in V, \forall j \in V, \text{ com } i \neq j, \quad \forall k \in R \quad \text{e} \quad \forall t \in D; \tag{3.13}$$

$$z_{ijkt} \geq \frac{w_{ijkt}}{\sum_{j=1}^{n} v_j}; \quad \forall i \in V, \forall j \in V, \text{ com } i \neq j, \quad \forall k \in R \quad \text{e} \quad \forall t \in D; \tag{3.14}$$

$$\sum_{t=t'+1}^{t'+g_i} x_{it} \leq (1 - x_{it'}); \qquad \forall t' \in D \quad \text{e} \quad \forall i \in \widehat{V}; \tag{3.15}$$

$$x_{it} \in \{0,1\}, \forall i \in V, \forall t \in D; \tag{3.16}$$

$$z_{ijkt} \in \{0,1\}, \forall i \in V, \forall j \in V, \forall k \in R, \forall t \in D; \tag{3.17}$$

$$w_{ijkt} \geq 0, \forall i \in V, \forall j \in V, \forall k \in R, \forall t \in D. \tag{3.18}$$

Na função objetivo do problema (Equação 3.2) almeja-se maximizar a vazão coletada durante todo o período por todas as Unidades Móveis de Pistoneio.

As desigualdades (3.3) garantem que o somatório dos tempos gastos em cada rota não exceda o limite da jornada diária. As restrições (3.4), (3.5) e (3.6) impõem que a seleção de um poço i para pistoneio implica que uma, e somente uma UMP deve chegar, pistonear e partir desse poço a cada visita.

A restrições (3.7) garantem que, dados dois poços i e j marcados para pistoneio em um determinado dia t, caso a coleta seja feita na seqüência j logo após i, apenas um veículo poderá fazê-la.

Em (3.8), certifica-se que a ETO deve estar presente em cada rota em todos os dias. Nas restrições (3.9) e (3.10) se estabelece que não sejam geradas rotas vazias, ou seja, cada UMP deve coletar o óleo de pelo menos um poço por dia.

As restrições de (3.11) a (3.14) garantem que as rotas geradas não sejam desconexas da origem. Para isso, uma variável de fluxo (w_{ijkt}) é associada a cada aresta (i,j) para cada veículo em cada dia. Em (3.11) temos que a quantidade de fluxo que sai de um poço é igual a quantidade de fluxo que entra mais a vazão associada ao poço. Para garantir a corretude do modelo, foi associado um fluxo unitário a todos os arcos utilizados pelas UMP's para sair da ETO (restrição (3.12)) fluxo esse que não é computado pela função objetivo. Pelas restrições (3.13), pode ser observado que se um veículo faz uso de uma determinada aresta em um determinado dia ($z_{ijkt} = 1$), então haverá algum fluxo positivo associado a este uso ($w_{ijkt} \geq 1$). As restrições (3.14) confirmam que numa aresta, quando

há um fluxo positivo associado ($\frac{w_{ijkt}}{\sum_{j=1}^{n} v_j} > 0$), necessariamente um veículo deve passar pela aresta no dia do período correspondente ($z_{ijkt} = 1$).

As restrições (3.15) impedem que os poços sejam pistoneados em intervalos que não permitam o reenchimento dos mesmos. Finalmente, as demais restrições (3.16) a (3.18) determinam o domínio das variáveis do problema.

Devido a complexidade do problema, a solução pelo método exato só é possível na prática para instâncias de pequeno porte. Diante desta limitação, na próxima seção, são apresentadas as propostas heurísticas para a solução aproximada do PRP-UMP.

3.3.2 Heurísticas GRASP para o PRP-UMP

Nesta seção são apresentadas as heurísticas que vão compor os GRASP's propostos para o PRP-UMP.

3.3.2.1 Heurísticas de Construção

Foram desenvolvidas duas heurísticas de construção aplicadas nos GRASP's propostos. A primeira baseada no Método das Pétalas, e a segunda na heurística de Inserção Mais Próxima (IMP). Nas duas heurísticas, as rotas são geradas de forma seqüencial, ou seja, a cada dia do período de planejamento, uma rota é construída para cada UMP.

A forma de se determinar qual será o primeiro poço a fazer parte de uma rota é a mesma nas duas heurísticas de construção. Inicialmente, os poços $i \in V$ disponíveis para pistoneio são classificados segundo uma função e colocados em uma lista denominada Lista de Candidatos (LC). Dado que f_{min} e f_{max} são, respectivamente, o menor e o maior valor encontrado para $f(i)$, os poços que obtiverem avaliação superior ao lim_{LRC}, dado pela Equação 3.19, são selecionados para formarem a Lista Restrita de Candidatos (LRC). Desta lista LRC, seleciona-se aleatóriamente o poço que iniciará a rota, onde α é um parâmetro do algoritmo estabelecido no intervalo [0,1].

$$lim_{LRC} = \alpha \times f_{min} + (1 - \alpha) \times f_{max} \tag{3.19}$$

Duas funções foram testadas para avaliação inicial dos poços. A primeira delas, f_1, considerando a vazão (v_j) e o tempo de reenchimento dos poços (g_j), como apresentado na Equação 3.20. A segunda, f_2, considerando também o tempo de percurso do poço i em relação a origem ($t_{i,\mathrm{ETO}}$), Equação 3.21.

$$f_1(p_i) = v_i/g_i \tag{3.20}$$

$$f_2(p_i) = v_i/(t_{i\ \text{ETO}} \times g_i) \tag{3.21}$$

Para uma melhor compreensão da escolha do primeiro poço a compor uma rota, considere a Figura 3.3, onde tem-se um campo petrolífero fictício composto por dezessete poços não surgentes e a ETO. Pressupondo que todos os poços estejam disponíveis para pistoneio no primeiro dia do período, uma avaliação da Tabela 3.1 permite concluir que, pela função regida pela Equação 3.20, apresentada na quinta coluna da tabela, para um $\alpha = 0.30$, o valor de lim_{LRC} é igual a 3.58. Sendo assim, a lista restrita seria composta por aqueles poços com avaliação superior a 3.58, ou seja, $LRC = \{p_1, p_4, p_7\}$. Por outro lado, ao considerar, além da vazão e tempo de reenchimento, o tempo de deslocamento dos poços à ETO, que é o caso da função dada pela Equação 3.21 e apresentada na sexta coluna da tabela, a lista seria dada pelos poços com avaliação maior que 0.21, ou seja, $LRC = \{p_1, p_4, p_{10}\}$. Após a definição da LRC, o poço inicial da rota é escolhido aleatoriamente entre os poços pertencentes a esta lista. Esta escolha aleatória contribuiu para que diferentes soluções sejam construídas a cada execução deste algoritmo.

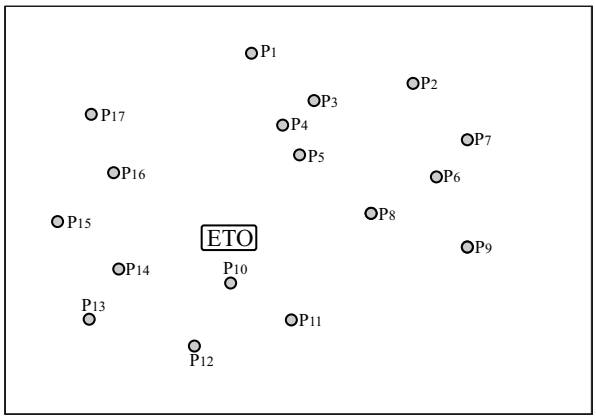


Figura 3.3: Exemplo fictício de um Campo Petrolífero

Nas heurísticas de construção propostas, deseja-se gerar, para cada dia do período, uma rota para cada uma das Unidades Móveis de Pistoneio, de forma que a duração não seja maior que a jornada de trabalho pré-estabelecida e a maior quantidade de óleo seja coletada. Uma solução do PRP-UMP é composta por um conjunto S de $|R| \times |D|$ rotas, onde R é o conjunto de UMP's e D o conjunto de dias, sendo associado a cada rota um custo igual ao tempo de duração das mesmas.

No Algoritmo 6 é apresentado o funcionamento básico da heurística de construção.

Dados dos Poços				Funções de Avaliação	
i	vazão(i)	g(i)	$t_{i,\text{ETO}}$	$\frac{\text{vazão}(i)}{g(i)}$	$\frac{\text{vazão}(i)}{t_{i,\text{ETO}} \times g(i)}$
p_1	5	1	20,07	5,00	0,249
p_2	1	4	26,55	0,25	0,009
p_3	3	3	17,47	1,00	0,057
p_4	8	2	13,28	4,00	0,301
p_5	2	3	11,75	0,67	0,057
p_6	1	3	24,13	0,33	0,014
p_7	5	1	28,86	5,00	0,173
p_8	4	3	16,31	1,33	0,082
p_9	6	3	26,97	2,00	0,074
p_{10}	5	3	5,91	1,67	0,282
p_{11}	10	4	12,57	2,50	0,199
p_{12}	3	3	13,52	1,00	0,074
p_{13}	7	2	18,53	3,50	0,189
p_{14}	4	4	13,10	1,00	0,076
p_{15}	1	2	19,22	0,50	0,026
p_{16}	5	4	14,45	1,25	0,087
p_{17}	2	1	20,22	2,00	0,099

Tabela 3.1: Avaliação inicial dos poços

Para cada dia, verificam-se os poços disponíveis que vão compor a lista de candidatos (LC) aptos a participarem das rotas naquele dia (linha 2). Depois disso, para cada UMP, uma rota é gerada. Para selecionar o primeiro poço a integrar a rota, a LC é ordenada segundo uma das funções já descritas (Equação 3.20 ou 3.21), sendo que aqueles poços com avaliação superior a lim_{LRC} são selecionados para compor a LRC (linha 4). Após determinar a LRC, um dos poços que a compõem é escolhido aleatoriamente (linha 5). A rota inicial ETO$\rightarrow$ p $\rightarrow$ETO é determinada na linha 6, sendo associado a esta um custo igual ao tempo de permanência da UMP para atender o poço, acrescido do tempo necessário para ir e voltar ao mesmo. Após a inicialização da rota, o processo de inserção de poços prossegue até que, na tentativa de uma nova inserção, a duração da rota ultrapasse a jornada (linha 7). Para selecionar os próximos poços que farão parte da rota, a LRC é atualizada utilizando um critério descrito posteriormente (linha 8) e, em seguida, um poço é sorteado da LRC (linha 9). Se for possível a inserção deste poço, a rota e seu custo são atualizados (linhas 10), caso contrário a rota é concluída.

Com uma rota montada, duas funções de aperfeiçoamento são executadas (linhas 17 e 18), denominadas 2-*optimal* e *Novas-Inserções*. A primeira delas busca determinar a melhor ordem para visitação dos poços para diminuir o tempo de percurso da rota, e a segunda visa inserir poços disponíveis ainda não alocados que estejam próximos à rota da UMP.

Algoritmo 6 *Heurística de Construção*

```
1: para dia=0 até |D| faça
2:    LC = Disponíveis(N)
3:    para ump=0 até |R| faça
4:       LRC ← Cria_LRC (LC, α)
5:       p ← Escolhe_Elemento_LRC(LRC)
6:       Insere_Poço_Rota(dia, ump, p)
7:       enquanto duração da rota não ultrapassar jornada faça
8:          Atualizar(LRC)
9:          p ← Escolhe_Elemento_LRC(LRC)
10:         Insere_Poço_Rota(dia, ump, p)
11:      fim enquanto
12:      2-optimal ( S,dia,ump)
13:      Novas-Inserções (S,dia,ump)
14:   fim para
15: fim para
```

Como o objetivo do PRP-UMP é maximizar a vazão, o que se busca é explorar todo o tempo da jornada de trabalho de forma a coletar a maior quantidade de óleo. A função 2-*optimal* não altera o total coletado em uma rota, ou seja, não afeta o valor da função de avaliação da rota, mas tenta encontrar uma ordem de visitação dos poços de forma a deixar uma folga que possibilite a inserção de novos poços pela função *Novas-Inserções*.

A função 2-*optimal* implementa o método das k-substituições ou k-*optimal*, que pode ser visto em [Campello e Maculan 1994], com k=2, visando obter um ciclo Hamiltoniano de menor duração. O procedimento das k-substituições pode ser visto no Algoritmo 7, pelo qual, dada uma rota h passada como parâmetro, obtida por qualquer tipo de heurística, deve-se obter, como mostrado na linha 1, o conjunto L com todas as possíveis combinações de k arestas não adjacentes pertencentes à rota h. Enquanto o conjunto L for diferente de $\emptyset$ (linha 2), removem-se a cada iteração, as arestas referentes a uma combinação $l^* \in L$ (linhas 3 e 4). Depois disso, todas as possíveis rotas contendo as arestas remanescentes em h são geradas, e a que apresentar menor tempo de percurso é atribuída a h', como pode ser visto na linha 5. Se o custo da nova solução gerada h' for menor ou igual ao custo de h, então h é substituída por h' e o procedimento recomeça; caso contrário, remove-se a combinação l^* de L.

Em geral, quanto maior o valor de k, maior a chance de se alcançar uma rota de menor duração. Entretanto, como o procedimento possui complexidade de $O\left(n^k\right)$, optou-se por empregar neste trabalho o procedimento com $k = 2$ (*2-optimal*).

Após aplicar o procedimento *2-optimal*, o resultado esperado é que a reorganização

Algoritmo 7 *Procedimento k-optimal* (h)

```
1: L ← Combinações(h, k)
2: enquanto L ≠ ∅ faça
3:    l* ← l ∈ L
4:    h' ← Remover(h, l*)
5:    h' ← Reconectar(h')
6:    se Custo(h') ≤ Custo(h) então
7:       h ← h'
8:       L ← Combinações(h, k)
9:    se não
10:      L ← L − {l*}
11:   fim se
12: fim enquanto
```

da ordem de visitação dos poços crie uma folga em relação ao tempo máximo da jornada de trabalho. Com isso, pode ser possível a inserção de novos poços através da função *Novas-inserções*. Esta função verifica quais poços disponíveis estão próximos da rota da UMP e avalia se a inserção de alguns deles não viola a restrição quanto à duração da rota. Caso seja possível, novos poços são inseridos.

As duas heurísticas de construção propostas (baseada no método da pétala e heurística de IMP) diferem entre si quanto à forma como a LRC é criada e ao modo como os poços são inseridos na rota. Na primeira proposta é utilizada a heurística da pétala com inserção dos poços sempre no final da rota. A segunda heurística baseia-se na avaliação da vizinhança dos elementos já pertencentes à rota parcial, com inserção feita no melhor local desta solução.

3.3.2.1.1 Heurística Construtiva por Pétala O método da pétala foi inicialmente proposto por [Foster e Ryan 1976] para tratar do problema de roteamento de veículos (PRV). Esta técnica aplicada ao PRV consiste em particionar o conjunto dos clientes em subconjuntos chamados pétalas, que são associados às rotas.

Para um melhor entendimento do método, considere um PRV para o qual o conjunto $N = \{p_0, p_1, p_2, p_3, \ldots, p_n\}$ onde p_0 representa o depósito e os demais elementos representam os clientes a serem atendidos. No depósito encontra-se uma frota de k veículos homogêneos com capacidade Q. Além disso, cada cliente p_i possui uma demanda determinística associada d_i. O PRV consiste em determinar um conjunto de rotas, uma para cada veículo, para atender todos os clientes, sem que a restrição de capacidade dos veículos seja violada. O objetivo é que a duração total das rotas seja minimizada.

Cada pétala define os clientes que serão atendidos por um mesmo veículo, sendo que a soma da demanda dos clientes vinculados a uma mesma pétala não pode superar a capacidade Q do veículo.

Na aplicação deste método é necessário definir inicialmente o ponto base (p_b) e o cliente de referência (p_r). A primeira pétala é composta inicialmente por p_r e sua demanda acumulada tem valor inicial igual a d_r. Para determinar qual o próximo cliente a ser inserido na pétala, é obtida a semi-reta $\overrightarrow{p_b p_r}$ que é rotacionada sobre o ponto p_b, até atingir um próximo cliente que será então inserido na pétala. Antes da inserção, é verificado se a pétala suporta acumular a sua demanda. Para determinar o próximo cliente a ser inserido basta continuar a rotação até que outro cliente seja alcançado.

Este procedimento é exemplificado utilizando-se o mesmo cenário apresentado na Seção 3.3.2.1 para o PRP-UMP. A ETO é o ponto base e p_1 o poço selecionado para ser o cliente de referência (p_r). Na Figura 3.4 pode-se verificar que o eixo $\overrightarrow{\text{ETO}p_1}$ quando rotacionado no sentido horário, alcança primeiramente o ponto p_4. Considerando as demandas apresentadas na Tabela 3.1, verifica-se que a demanda inicial associada a pétala é dada pela demanda de p_1 que é igual a cinco. Supondo que a capacidade do veículo seja de vinte unidades ($Q = 20$), verifica-se se a inclusão de p_4 irá violar a capacidade do veículo. Como a demanda associada a p_4 é igual a oito, tem-se um valor total de demanda igual a treze unidades, que não excede o valor de Q, possibilitando a inclusão de p_4 na pétala corrente. A adição de pontos na pétala ocorre até que, na tentativa de uma nova inserção, a restrição de capacidade do veículo seja violada. Neste momento a pétala atual é finalizada e inicia-se uma nova pétala.

A primeira pétala, como visto na Figura 3.5, é iniciada com p_1 sendo possível acumular a demanda dos poços p_4, p_3, p_5 e p_2, totalizando 19 unidades. A inserção do próximo poço, p_7, faria o total acumulado ultrapassar a capacidade Q do veículo. Desta forma a primeira pétala é finalizada e p_7 torna-se o primeiro poço na nova pétala. Este processo é executado até que as pétalas englobem todos os poços disponíveis.

Heurística da Pétala Randomizada para PRP-UMP

Nesta heurística de construção, um procedimento inspirado no método da pétala é utilizado para determinar quais poços farão parte da LRC, de onde são selecionados os poços que vão compor a rota. Sempre que for necessário selecionar um novo poço, uma única pétala é gerada considerando a ETO como ponto base e o último poço inserido na rota como ponto de referência (p_r), sendo o tamanho da pétala limitado pelo número de

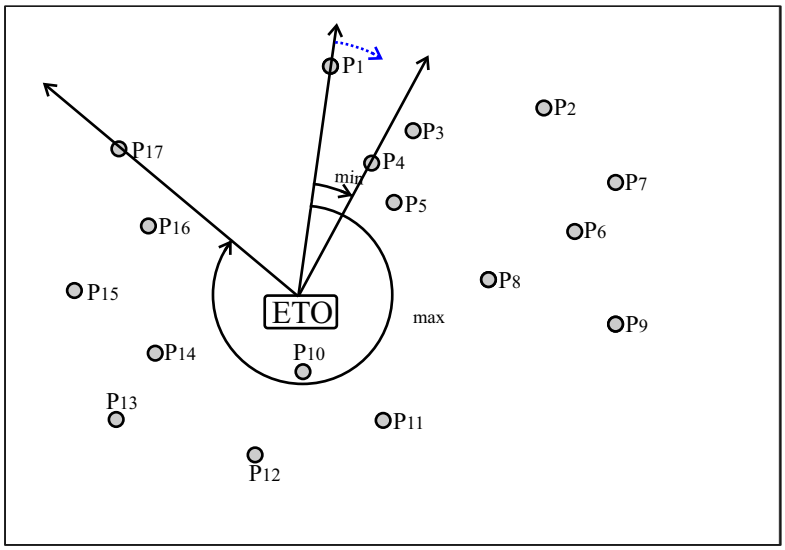


Figura 3.4: Ângulo mínimo (φ_{min}) e ângulo máximo (φ_{max}).

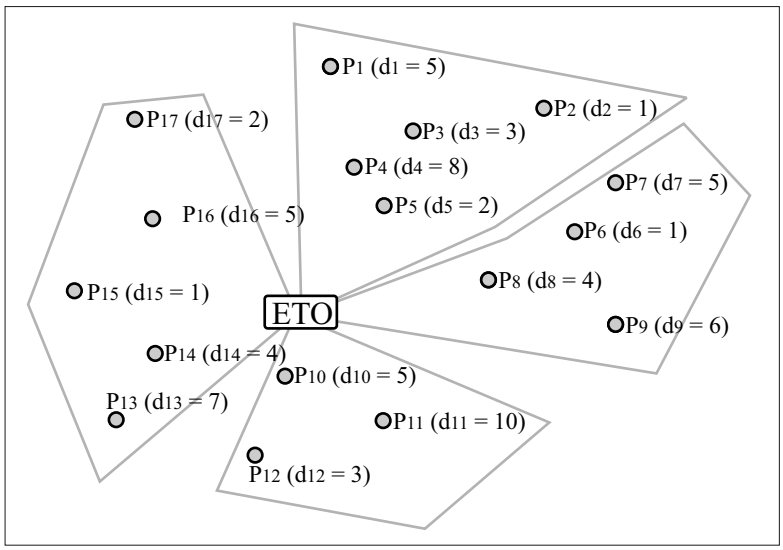


Figura 3.5: Exemplo de pétalas onde $Q = 20$.

poços que integrarão a LRC. Depois de gerada a pétala, os poços p que a compõem são avaliados segundo a função de custo $f(p)$ dada pela Equação 3.22.

$$f(p) = v_p / t_{p,p_r} \tag{3.22}$$

A escolha do poço é feita através da aplicação do método da roleta, pelo qual a probabilidade de um poço ser escolhido é proporcional ao custo $f(p)$. Caso seja possível realizar a inserção sem que a duração da rota ultrapasse a jornada de trabalho, a inserção é efetuada e o ponto de referência é atualizado ($p_r \leftarrow$ poço inserido) para geração da próxima pétala (atualização da LRC). Caso contrário, o poço não é inserido e a rota é

finalizada fazendo com que UMP retorne ao ponto de partida (ETO).

Considere o exemplo da Figura 3.6, onde existe a rota inicial ETO$\rightarrow p_1 \rightarrow$ETO. Supondo que $|LRC| = 4$, gera-se a primeira pétala, que inclui os poços p_2, p_3, p_4 e p_5. Supondo que p_3 tenha sido sorteado para compor a solução, gera-se uma nova rota ETO$\rightarrow p_1 \rightarrow p_3 \rightarrow$ETO e, caso o tempo de percurso desta rota não exceda o limite permitido, o poço é inserido logo após o último que tenha sido inserido anteriormente, no caso p_1. Na Figura 3.7 pode ser vista a solução parcial acrescida do poço selecionado, bem como a nova pétala gerada, tendo como ponto de referência p_3 (área sombreada).

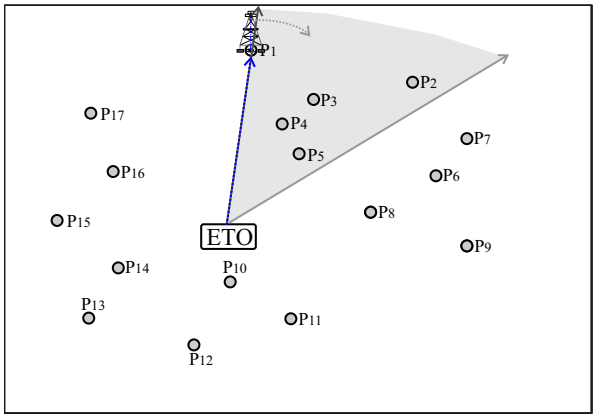


Figura 3.6: Poços pertencentes à primeira pétala onde $|LRC| = 4$.

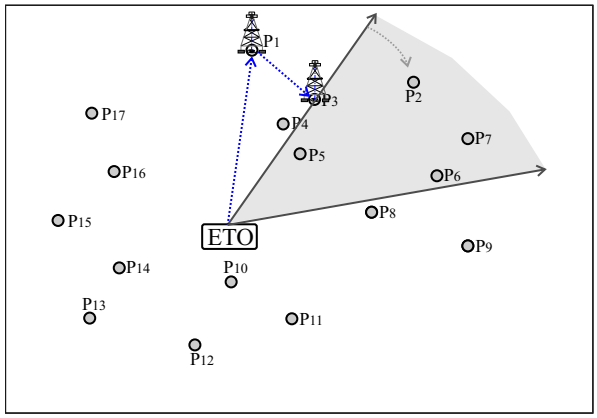


Figura 3.7: Segundo poço e próxima LRC.

O processo se repete até que não seja possível inserir o poço sorteado sem violar a restrição da jornada. Neste momento a rota é terminada, fazendo o veículo retornar ao ponto de partida, como mostrado na Figura 3.8.

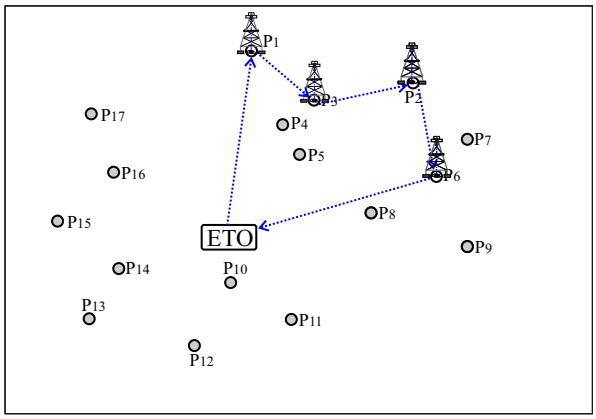


Figura 3.8: Rota Completa.

3.3.2.1.2 Heurística Construtiva por Inserção Mais Próxima Aleatorizada

A Inserção Mais Próxima - IMP, segundo [Campello e Maculan 1994], é uma heurística

utilizada para determinar o ciclo Hamiltoniano de menor custo de um grafo G(V,A) sendo aplicada sobre uma solução de partida contendo três vértices. A IMP pode ser dividida em duas fases: na primeira, determina-se, para a rota atual, o vértice $v \in V$, não pertencente à mesma, mais próximo de qualquer vértice pertencente a esta. Na segunda, determina-se a posição de inserção entre dois vértices consecutivos quaisquer pertencentes à rota que minimize o custo da rota remanescente.

Na construção proposta para o PRP-UMP, inspirada nesta heurística, foi acrescida uma característica aleatória necessária nos módulos de construção do GRASP. Inicialmente, define-se a rota ETO$\rightarrow p_1 \rightarrow$ETO, onde p_1 é o primeiro poço inserido como apresentado no início na Seção 3.3.2.1.

Para selecionar o próximo poço a ser inserido, os k poços fora da rota considerados mais atrativos para cada poço pertencente à mesma são avaliados segundo a função dada pela Equação 3.23. Estes poços são dispostos em ordem não crescente em relação a esta função e o próximo poço a fazer parte da rota é selecionado aleatóriamente entre aqueles com avaliação superior ao valor $lim_{LRC} = \alpha \times f_{min} + (1 - \alpha) \times f_{max}$.

$$f(p) = v_p / t_{p\ vizinho} \tag{3.23}$$

Apresenta-se na Figura 3.9, um exemplo da execução deste procedimento considerando-se o mesmo exemplo apresentado na seção anterior e utilizando-se o número de vizinhos k igual a três. Para a rota inicial ETO$\rightarrow p_1 \rightarrow$ETO, obtém-se os seis poços candidatos a participarem da LRC, onde p_3, p_4 e p_5 são mais próximos de p_1 e p_{10}, p_{11} e p_{14} estão mais próximos da ETO. Supondo que após a avaliação e ordenação o poço p_{14} tenha sido selecionado, como existem somente dois elementos na rota, a inserção é feita entre estes.

Este procedimento é ilustrado na Figura 3.10, que mostra a rota obtida após a inserção do poço p_{14}. Além disso, é possível visualizar que os poços candidatos a serem inseridos na nova rota são $p_3, p_4, p_5, p_{10}, p_{11}, p_{12}, p_{13}, p_{15}$ e p_{16}. Supondo que o poço p_{12} tenha sido escolhido para integrar a rota, deve-se inseri-lo na rota já construída. Diferente da idéia original da IMP, onde todas as possíveis posições de inserção são testadas, na heurística proposta somente dois locais de inserção são testados. No exemplo, verificam-se os custos para inserir p_{12} antes e depois de seu vizinho mais próximo, que neste caso é a ETO. Para tanto, calcula-se o custo da rota inserindo-se p_{12} entre p_{14} e a ETO e entre a ETO e p_1. A inserção que causar o menor acréscimo na duração da rota é efetivada.

Na Figura 3.11 pode-se verificar que a inserção foi realizada entre o poço p_{14} e a ETO.

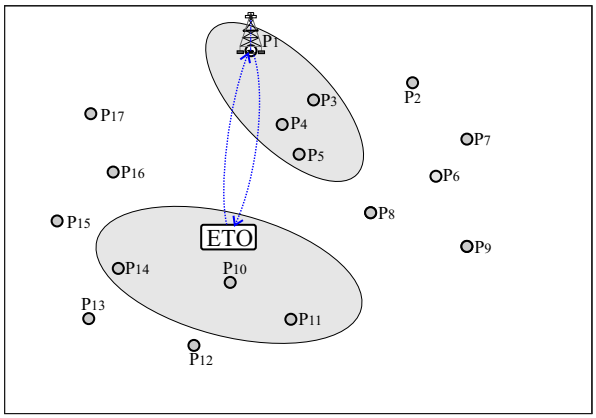


Figura 3.9: Rota inicial e vizinhança analisada.

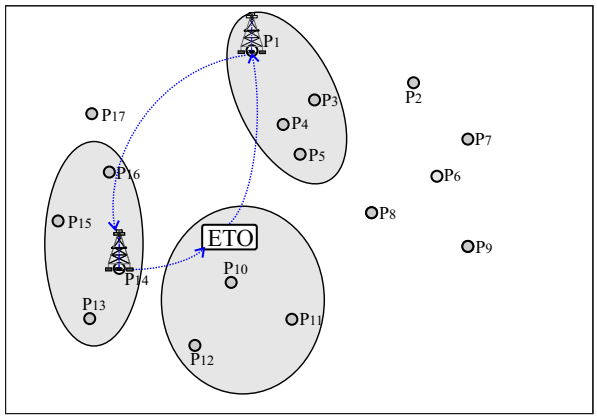


Figura 3.10: Rota após inserção do poço p_{14} e vizinhança.

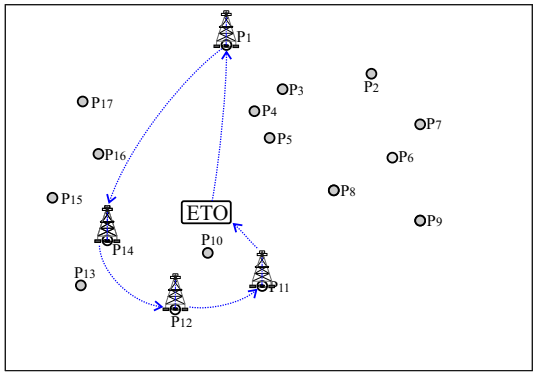


Figura 3.11: Rota após inserção do poço p_{12}.

Esse processo de análise e inserção é repetido até não ser mais possível adicionar poços na rota, ou seja, até que a tentativa de inserção de um poço faça o tempo da rota ultrapassar o limite da jornada diária.

Como na heurística de construção anterior, após a conclusão de cada rota as mesmas funções de aperfeiçoamento, 2-*optimal* e *Novas_Inserções*, são executadas.

3.3.2.2 Busca Local

Nesta seção são apresentados dois procedimentos desenvolvidos para busca local. O primeiro analisa cada dia do período de forma independente e o segundo trata o período de forma global.

3.3.2.2.1 Busca Local Diária - BLD O objetivo desta busca local é conseguir uma melhor distribuição, nas diferentes rotas, dos poços alocados para visitação em um determinado dia, bem como adicionar novos poços que possivelmente estejam disponíveis

neste momento. A BLD trata individualmente cada dia do período.

Caso exista entrelaçamento entre duas ou mais rotas, como pode ser visto na Figura 3.12, muitas vezes é possível conseguir uma nova distribuição dos poços onde o tempo total das rotas envolvidas seja reduzido. Na Figura 3.13 pode-se perceber que a nova distribuição dos mesmos poços presentes na Figura 3.12 causa um leve aumento no percurso da rota da UMP_1. Entretanto a redução no percurso na rota da UMP_2 é mais significativa e a soma dos tempos das rotas envolvidas é diminuído. Desta forma, a alteração é proveitosa e possivelmente esta redução possibilitará a inserção de outros poços na rota da UMP_2.

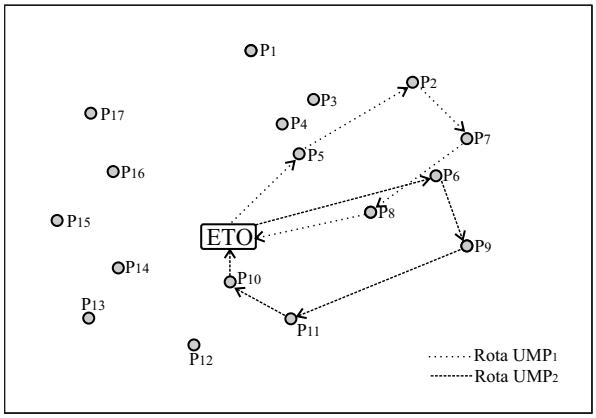


Figura 3.12: Rotas Entrelaçadas.

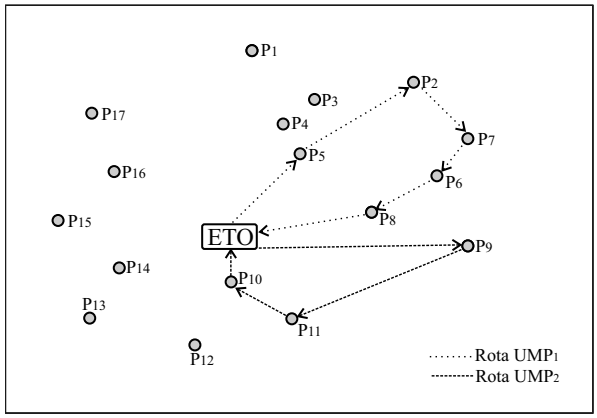


Figura 3.13: Rotas Otimizadas.

No Algoritmo 8, cujo parâmetro de entrada é uma solução S obtida na etapa de construção, é apresentada uma visão geral do funcionamento da BLD.

Pode-se observar nas linhas de 1 a 3 que, para cada dia do período, verificam-se os poços p pistoneados por cada uma das UMP's. Para cada poço p, na linha 4 são verificados os poços da sua vizinhança $N(p)$, que é definida pelo conjunto dos poços v mais próximos de p que estão sendo pistoneados por uma UMP diferente daquela que pistoneia p. Caso seja possível remover um poço v de sua rota atual, inserindo-o na rota da UMP que visita p, a soma dos tempos das rotas envolvidas é analisado. Havendo redução neste tempo, esta troca é efetuada na linha 6. Após verificar a vizinhança de todos os poços de uma rota, o procedimento *Novas_ Inserções* é executado na linha 10 na tentativa de inserir outros poços que estejam disponíveis, sempre com a precaução de não causar inviabilidade na solução.

Algoritmo 8 *Busca Local Diária* (S)

```
1: para dia=0 até |P| faça
2:   para ump=0 até |R| faça
3:     para cada poço p ∈ S[dia, ump] faça
4:       para v ∈ N(p) faça
5:         se Custo_Troca(S, p, v) então
6:           Atualizar(S)
7:         fim se
8:       fim para
9:     fim para
10:    Novas_Inserções(S)
11:  fim para
12: fim para
```

3.3.2.2.2 Busca Local Periódica - BLP Neste módulo, o que se propõe é alterar os dias de visitação daqueles poços que podem ser melhor explorados devido a sua boa vazão e ao reduzido número de visitas realizadas a este poço na solução atual. Para se determinar o desperdício em relação a vazão de cada poço, computa-se a quantidade máxima de óleo possível de ser extraída no período, subtraída da quantidade de óleo que está sendo coletada na solução atual. Na Equação 3.24 calcula-se o desperdício de um poço p, sendo $max\_visitas(p, |D|)$ o número máximo de visitas que podem ser feitas a cada poço no intervalo de $|D|$ dias, $vazão(p)$ a vazão do poço p e $visitas(p, S)$ o número de visitas realizadas ao poço p na solução atual S.

$$Desperdício(p) = \overbrace{(max\_visitas(p, |D|) \times vazão(p))}^{\textit{coleta máxima no período}} - \overbrace{(visitas(p, S) \times vazão(p))}^{\textit{coleta na solução atual}} \quad (3.24)$$

O Algoritmo 9 mostra cada etapa da busca local BLP. Os parâmetros de entrada são uma solução S e um fator δ que determina a abrangência da vizinhança de S a ser explorada. Nas linhas de 1 a 3 é calculado o desperdício para cada poço do conjunto $\widehat{V}$. Os poços são classificados em ordem não crescente segundo o desperdício e os δ primeiros poços que possuem maior valor têm seus dias de visitação alterados. Para isto, são determinados novos conjuntos de dias de visitação que explorem ao máximo a vazão dos poços selecionados, fazendo com que estes passem a ser pistoneados somente nos dias que compõem os novos conjuntos, reduzindo o desperdício total.

Algoritmo 9 *Busca Local Periódica*(S, δ)

1: **para** $i \leftarrow 1$ até $|\widehat{V}|$ **faça**
2: Desperdício$[i] = (max_visitas(i, |D|) \times vazão(i)) - (visitas(i, S) \times vazão(i))$;
3: **fim para**
4: *Ordena_Poços* (Desperdício, *Lista_Poços*);
5: **para** $i \leftarrow 1$ até δ **faça**
6: $maior \leftarrow Lista\_Poços[i]$;
7: $combinação \leftarrow Seleciona\_Melhor\_Combinação(maior)$;
8: $Atualiza\_Combinação(S, combinação)$;
9: **fim para**
10: $Viabilizar(S)$;
11: $Otimização(S)$;

A solução é atualizada entre as linhas 6 a 8. Este processo de alteração e atualização da solução se dá iterativamente até que toda a vizinhança determinada pelo parâmetro δ seja explorada. Nesta etapa, a restrição quanto ao tempo da rota é relaxada, permitindo que soluções inviáveis seja geradas, ou seja, soluções cuja duração da rota seja maior que a jornada diária de trabalho. Na linha 10 tem-se a chamada do procedimento que viabiliza o conjunto de rotas. Para tal, calcula-se para cada poço p sua economia, $economia(p)$, definida pela redução na duração da rota proporcionada pela remoção do poço p. Após este cômputo, obtém-se a razão $\eta = economia(p)/vazão(p)$. Os poços com maior valor de η são removidos até que a duração da rota seja menor que a jornada de trabalho. Devido às alterações feitas na troca do conjunto de dias de visitação dos poços e também devido à viabilização, há possibilidade da formação de folgas nas rotas. Para que a jornada seja explorada da melhor forma possível, uma função é executada para verificar, caso a rota tenha alguma folga, se é possível a inserção de outros poços, o que é assegurado pela execução na última linha do algoritmo do procedimento $Otimização(S)$.

3.3.2.3 Reconexão de Caminhos

Na metaheurística GRASP, cada iteração é realizada de forma independente. Este comportamento não possibilita que as melhores características obtidas a cada iteração sejam reaproveitadas nas iterações posteriores.

Uma abordagem que vem sendo bastante explorada com o propósito de obter soluções a partir da combinação de características promissoras, oriundas de diferentes soluções de boa qualidade, é a Reconexão de Caminhos (RC). Esta heurística, proposta originalmente por [Glover 1996], inicia com duas soluções, sendo o caminho entre estas explorado em busca de soluções ainda melhores. Para tanto, definida as soluções origem s_0 e destino

s_d, para gerar os caminhos, movimentos são selecionados de forma a introduzir na solução atual (inicialmente s_0) atributos presentes na solução destino (s_d).

Num primeiro passo, as diferenças entre as duas soluções são analisadas de forma a gerar um conjunto Δ, definindo os possíveis movimentos que podem ser executados. Dentre todos os movimentos pertencentes ao conjunto Δ, aquele que implica em maior ganho ao alterar a solução origem é aplicado, gerando uma nova solução intermediária s_{rc}, sendo então o conjunto Δ atualizado pela remoção do movimento efetuado. Iterativamente, são aplicados movimentos sobre a solução corrente s_{rc} até que a solução destino seja alcançada, isto é, $\Delta = \emptyset$. A melhor solução encontrada no caminho é tida como resultado da heurística.

Objetivando-se tirar proveito das melhores características das soluções obtidas em cada iteração do GRASP, foi desenvolvido um módulo de Reconexão de Caminhos para o PRP-UMP. Dentre os possíveis atributos que poderiam ser avaliados para o problema, optou-se por abordar as rotas de cada dia do período como um atributo da solução.

Na Figura 3.14 são apresentadas duas soluções s_0 e s_d para um problema que possui dezessete poços, duas UMP's e com período composto por seis dias. Pode-se observar que não há nenhum dia em que todas as rotas sejam iguais, assim, o conjunto de atributos diferentes para s_0 e s_d é composto por cada dia do período (tamanho máximo $\Delta = |D|$), ou seja, $\{d_1, d_2, d_3, d_4, d_5, d_6\}$.

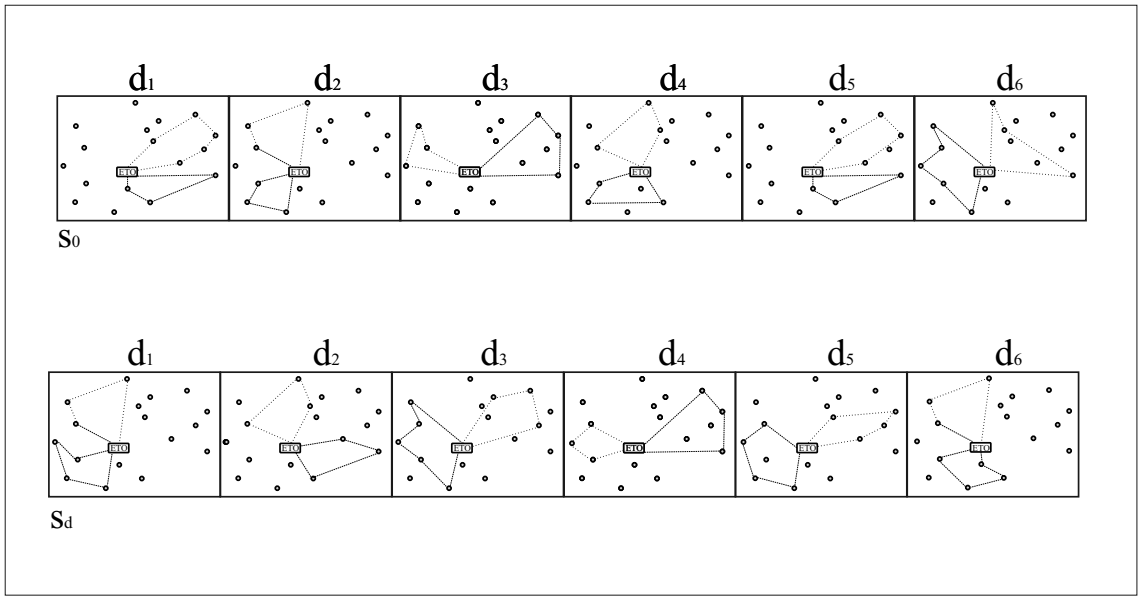


Figura 3.14: Reconexão de Caminhos

Para fazer com que s_0 seja alterada com a adição de características de s_d, a cada etapa as rotas referentes a um determinado dia da solução s_0 são substituídas pelas rotas asso-

ciadas da solução destino. Este tipo de movimento garante que não haverá inviabilidade entre as rotas de um mesmo dia, mas não garante que sejam respeitadas as restrições referentes ao tempo de reenchimento de cada poço. Por isso, após cada movimento é necessário verificar a viabilidade da solução.

Para cada solução gerada em uma iteração GRASP, realiza-se um procedimento de Reconexão de Caminhos entre esta solução e uma solução selecionada de um conjunto elite, S_{elite}. Neste conjunto S_{elite} estão armazenadas as melhores soluções geradas pela heurística até o momento.

No Algoritmo 10, cujos parâmetros são a solução S obtida após a busca local e o conjunto de soluções elite S_{elite}, pode ser observado o pseudocódigo do módulo de Reconexão de Caminhos proposto para o PRP-UMP. O primeiro passo do algoritmo é determinar qual das soluções do conjunto elite vai participar junto a solução S do processo RC. Para tanto escolhe-se a solução do conjunto elite que apresenta maior grau de diferença em relação a S (dias com rotas diferentes). Após esta etapa, é necessário estabelecer qual das soluções será a solução origem e qual será a solução destino (linhas 2 e 3). Na RC proposta, o caminho é percorrido sempre da solução de melhor qualidade para a de pior qualidade, tendo em vista que a vizinhança da solução origem é sempre investigada em maior amplitude [Resende e Ribeiro 2005]. Desta maneira, a solução de melhor qualidade é atribuída à s_0 e a outra à s_d.

Na linha 5 obtém-se o conjunto de diferenças entre as soluções origem e destino. No laço da linha 6 à 13, verifica-se a execução de cada movimento da solução atual, s_{rc}, em direção à solução destino. Na linha 7, verifica-se qual dos movimentos presentes em Δ gera uma solução intermediária de melhor qualidade, este movimento é executado e o conjunto Δ é atualizado na linha 8. Após o movimento, conforme apresentado na linha 9, é executada a verificação de viabilidade na solução corrente. Caso a restrição quanto ao intervalo de reenchimento de um poço seja violada, é necessário remover o poço de uma das rotas. Ocorrendo remoções, é realizada uma investigação para determinar se existe algum poço disponível que pode ser inserido nesta rota sem violar o tempo máximo da jornada. Se o custo da solução atual for melhor que o custo da melhor solução encontrada até o momento, a melhor solução (s^*) é atualizada, linhas 10 a 12.

Algoritmo 10 *Reconexão por Caminho* (S, S_{elite})

1: $s \leftarrow$ Maior_Diferença(S_{elite}, S);
2: $s_0, s_{rc} \leftarrow Melhor\_Qualidade(S, s)$;
3: $s_d \leftarrow Pior\_Qualidade(S, s)$;
4: $s^* \leftarrow Best(s_0, s_d)$;
5: $Calcular(\Delta)$;
6: **enquanto** $\Delta \neq \phi$ **faça**
7: $s_{rc} \leftarrow Melhor\_Movimento(\Delta)$;
8: $Atualizar(\Delta)$;
9: $Viabilizar(s_{rc})$;
10: **se** $Custo(s_{rc}) > Custo(s^*)$ **então**
11: $s^* \leftarrow s_{rc}$;
12: **fim se**
13: **fim enquanto**

3.3.2.4 Heurísticas GRASP Avaliadas

Dentre as possíveis combinações das heurísticas de construção, busca local e Reconexão de Caminhos, para compor as heurísticas GRASP foram selecionadas as opções apresentadas na Tabela 3.2. Verifica-se na tabela que em todas as versões, as duas buscas locais são executadas, diferindo quanto à ordem em que são aplicadas. Para cada GRASP, verificou-se o efeito da utilização da Reconexão de Caminhos (G1+RC, G2+RC, G3+RC e G4+RC), cujos resultados são apresentados no próximo capítulo.

GRASP	Construção	Busca Local
G1	Pétala	BLD + BLP
G2	Pétala	BLP + BLD
G3	IMP	BLD + BLP
G4	IMP	BLP + BLD

Tabela 3.2: Heurísticas GRASP propostas

Capítulo 4

Resultados Computacionais

Para avaliar os resultados das abordagens aqui propostas, foram realizados testes empíricos onde procurou-se comparar os algoritmos de tal forma que se pudesse analisar o diferencial de cada abordagem. Tendo em vista as abordagens apresentadas neste trabalho referirem-se a problemas de duas naturezas distintas, os resultados estão apresentados conforme as mesmas.

4.1 Resultados para o PCVP

Nesta seção são apresentados resultados computacionais para avaliar a heurística proposta neste trabalho para o PCVP quando comparada à apresentada em [Bertazzi et al. 2004] para o mesmo problema. Para avaliar os dois algoritmos, foram utilizadas trinta e três instâncias da literatura disponíveis em [OR-Library]. As instâncias de 1 a 10 foram introduzidas em [Christofides e Beasley 1984], as de 11 a 23 por [Chao et al. 1995] e as demais foram propostas por [Cordeau et al. 1997]. Neste trabalho foi realizada uma nova implementação do algoritmo H_BPS, denotado H_BPS$^+$, com o intuito de se obter mais informações sobre a H_BPS.

Para implementação das heurísticas H_BPS$^+$ e G_GMO foi utilizada a linguagem de programação C, compilada com o gcc versão 3.2.2, executadas em máquinas AMD Athlon (TM) 1.4GHz com 256Mbytes de RAM. A seguir, são apresentados os resultados em relação a custos das soluções e a tempo de CPU para todas as instâncias utilizadas nos testes.

4.1.1 Qualidade das soluções

Na Tabela 4.1 são apresentadas as soluções encontradas por cada um dos algoritmos, onde os valores destacados em negrito indicam os melhores resultados para cada instância. Na primeira coluna tem-se a identificação da instância, na segunda e na terceira são fornecidos, respectivamente, o número de cidades e o número de dias do período de planejamento. Na quarta coluna são apresentados os resultados da heurística H_BPS encontrados em [Bertazzi et al. 2004]. Para estes resultados foram executadas todas as possíveis combinações dos parâmetros p_1, p_2 e f sugeridas ($p_1 \in \{1, 2, 3, \ldots, 20\}$; $p_2 \in \{1, 2, 3, \ldots, 20\}$ e $f \in N$, onde N é o conjunto das cidades). Na quinta coluna é possível observar o resultado do melhor valor obtido pela implementação realizada neste trabalho do algoritmo H_BPS (sub-coluna H_BPS$^+$), bem como a diferença entre os resultados do H_BPS e H_BPS$^+$ (sub-coluna $\%_1$), indicando o quanto pior foi o desempenho do H_BPS$^+$ em relação ao H_BPS. Finalmente, na sexta coluna rotulada G_GMO, são apresentados os resultados do algoritmo GRASP proposto para o PCVP.

Devido ao caráter aleatório do algoritmo, o G_GMO foi executado 10 vezes com diferentes sementes utilizadas na geração dos números aleatórios. Os testes foram realizados com a seguinte configuração de parâmetros: número máximo de iterações MAX_IT = 250, p = 0.5, $p_1 = 10$, $p_2 = 3$ e $f = \arg\max_{i \in I} d_{0i}$. Os valores considerados para α foram (0,02; 0,05; 0,1; 0,15 e 0,20). Na sub-coluna Custo$_2$, é apresentada a média dos resultados obtidos em todas as 10 execuções, e na coluna $\%_2$ tem-se a diferença entre este resultado e o apresentado na coluna H_BPS. Já na sub-coluna Custo$_3$, pode ser vista a melhor entre todas as soluções encontradas pelo GRASP, e na coluna $\%_3$ é mostrada a diferença desta solução em relação ao valor apresentado na quarta coluna.

A diferença de 0,69%, apresentado na última linha, indica a média dos valores apresentados na coluna $\%_1$ (diferenças entre H_BPS e H_BPS$^+$). Este resultado deve-se ao fato do número de combinações de parâmetros testados pela implementação deste trabalho (H_BPS$^+$) ser muito menor do que literatura (H_BPS). Após testes preliminares, um conjunto de 10 combinações de parâmetros foi escolhida para realização dos testes.

Pode-se verificar também que as soluções conseguidas pelo GRASP G_GMO, quando não conseguem atingir o melhor resultado da literatura, ficam bem próximas do mesmo. A diferença média de 0,17% torna-se pequena quando se considera o elevado tempo demandado pela heurística H_BPS para executar todas as combinações de parâmetros quanto comparado ao tempo exigido pelo GRASP proposto.

Inst.	$\|N\|$	$\|P\|$	H_BPS	H_BPS$^+$		G_GMO			
						Média		Melhor	
				Custo_1	$\%_1$	Custo_2	$\%_2$	Custo_3	$\%_3$
1	50	2	**432,10**	436,07	0,92	432,14	0,01	**432,10**	0,00
2	50	5	**1.105,81**	1.118,52	1,15	1.106,52	0,06	1.106,24	0,04
3	50	5	**466,71**	469,22	0,54	**466,71**	0,00	**466,71**	0,00
4	75	2	**549,05**	555,44	1,16	551,22	0,39	550,18	0,21
5	75	5	**1.382,33**	1.383,34	0,07	1.385,85	0,25	1.383,34	0,07
6	75	10	**643,50**	646,93	0,53	643,56	0,01	**643,50**	0,00
7	100	2	**643,80**	658,93	2,35	**643,80**	0,00	**643,80**	0,00
8	100	5	**1.613,80**	1.627,65	0,86	1.619,07	0,33	1.616,42	0,16
9	100	8	**722,69**	731,86	1,27	723,61	0,13	723,03	0,05
10	100	5	**1.237,77**	1.251,58	1,12	1.244,15	0,52	1.238,34	0,05
11	65	4	**490,97**	**490,97**	0,00	**490,97**	0,00	**490,97**	0,00
12	87	4	**664,10**	**664,10**	0,00	**664,10**	0,00	**664,10**	0,00
13	109	4	**830,80**	**830,80**	0,00	**830,80**	0,00	**830,80**	0,00
14	131	4	**994,60**	**994,60**	0,00	**994,60**	0,00	**994,60**	0,00
15	153	4	**1.157,07**	**1.157,07**	0,00	**1.157,07**	0,00	**1.157,07**	0,00
16	48	4	**660,12**	**660,12**	0,00	**660,12**	0,00	**660,12**	0,00
17	66	4	**776,43**	**776,43**	0,00	**776,43**	0,00	**776,43**	0,00
18	84	4	**873,73**	876,44	0,31	**873,73**	0,00	**873,73**	0,00
19	102	4	**958,51**	**958,51**	0,00	**958,51**	0,00	**958,51**	0,00
20	120	4	**1.033,58**	**1.033,58**	0,00	**1.033,58**	0,00	**1.033,58**	0,00
21	77	4	**1.375,07**	**1.375,07**	0,00	**1.375,07**	0,00	**1.375,07**	0,00
22	154	4	**4.312,31**	4.339,88	0,64	**4.312,31**	0,00	**4.312,31**	0,00
23	231	4	**8.308,51**	8.442,01	1,61	8.340,06	0,38	**8.308,51**	0,00
24	48	4	**2.064,84**	**2.064,84**	0,00	**2.064,84**	0,00	**2.064,84**	0,00
25	96	4	**3.207,44**	3.241,07	1,05	3.218,90	0,36	3.218,35	0,34
26	144	4	**4.030,54**	4.113,24	2,05	4.079,57	1,22	4..065,07	0,86
27	192	4	**4.558,94**	4.594,40	0,78	4.585,90	0,59	4.580,23	0,47
28	240	4	**4.628,89**	4.683,21	1,17	4.676,53	1,03	4.65.8,67	0,64
29	288	4	**5.534,94**	5.618,72	1,51	5.584,76	0,90	5.574,32	0,71
30	72	6	**4.435,39**	4.471,04	0,80	4.454,93	0,44	4.451,96	0,37
31	144	6	**5.376,11**	5.414,42	0,71	5.396,83	0,39	5.392,44	0,30
32	216	6	**7.282,39**	7.345,70	0,87	7.326,97	0,61	7.324,80	0,58
33	288	6	**8.280,07**	8.389,93	1,33	8.339,18	0,71	8.331,03	0,62
Média Final Percentual					0,69		0,25		0,17

Tabela 4.1: Resultado das heurísticas quanto ao custo da solução

4.1.2 Tempo de Execução

Nesta seção são apresentados os resultados quanto ao tempo de CPU, em segundos, utilizados pela heurística H_BPS$^+$ e pela abordagem GRASP G_GMO. Na Tabela 4.2 encontra-se na primeira coluna o identificador das instâncias. Na sub-coluna tempo$_0$ vê-se o tempo médio despendido para uma única execução da heurística H_BPS$^+$, tempo este conseguido pela implementação feita neste trabalho da referida heurística. Os autores da H_BPS não apresentaram os tempos computacionais, impossibilitando a comparação de tempo entre as versões da H_BPS.

Na sub-coluna t-H_BPS$_1$, assumindo um valor fixo para o parâmetro f, é apresentado o tempo que seria necessário para executar o algoritmo com as demais combinações de parâmetros sugeridas pelos autores ($p_1 \in \{1, 2, 3, \ldots, 20\}$ e $p_2 \in \{1, 2, 3, \ldots, 20\}$), totalizando 400 combinações. Na coluna t-H_HPS$_2$ é apresentado o tempo que seria gasto se considerados, além das possibilidades dos parâmetros p_1 e p_2, todos os valores que podem ser assumidos pelo parâmetro f.

Finalmente, na última coluna, apresentam-se os tempos de CPU do G_GMO. Na coluna t-G_GMO é possível verificar a média do tempo gasto pelo GRASP para uma execução com 250 iterações.

Analisando os resultados apresentados nesta tabela, pode-se concluir que a heurística da literatura H_BPS se torna inviável para o tratamento de instâncias muito grandes. Para computar todas as possibilidade de combinações de parâmetros para obter os resultados do H_BPS, o tempo de execução estimado seria igual ao número de combinações multiplicado pelo tempo$_0$, apresentado na coluna t-H_BPS$_2$. Mesmo reduzindo o número de combinações através da fixação do parâmetro f ($f = \arg\max_{i \in N} d_{i0}$), cuja utilização mostrou uma tendência a prover bons resultados, ainda assim o tempo de CPU seria muito grande (coluna t-H_BPS$_1$).

Por exemplo, para a instância 1, que possui 50 cidades, o número de combinações é igual a 20.000 e, como uma execução dura em média 0.41 segundos, o tempo estimado para executar todas as possibilidades seria 8.200,00 segundos. Fixando a primeira cidade como aquela mais distante da origem, o número de combinações diminuiria para 400, mas o tempo de CPU continua extremamente alto se comparado com o GRASP (H_BPS$^+$ 164.00 seg e GRASP 30.19 seg).

Inst.	H_BPS$^+$			t-G_GMO
	Tempo$_0$	t-H_BPS$_1$	t-H_BPS$_2$	
1	0,41	164,00	8.200,00	30,19
2	0,29	116,00	5.800,00	12,84
3	0,54	216,00	10.800,00	63,72
4	1,36	544,00	40.800,00	97,72
5	0,85	340,00	25.500,00	35,75
6	2,45	980,00	73.500,00	385,02
7	2,63	1.052,00	105.200,00	238,07
8	1,56	624,00	62.400,00	82,43
9	4,90	1.960,00	196.000,00	791,67
10	2,52	1.008,00	100.800,00	195,58
11	0,25	100,00	6.500,00	11,32
12	0,56	224,00	19.488,00	26,55
13	1,10	440,00	47.960,00	50,98
14	1,94	776,00	101.656,00	92,17
15	3,07	1.228,00	187.884,00	144,88
16	0,29	116,00	5.568,00	24,89
17	0,68	272,00	17.952,00	63,44
18	1,37	548,00	46.032,00	126,73
19	2,59	1.036,00	105.672,00	226,98
20	3,84	1.536,00	184.320,00	371,88
21	0,49	196,00	15.092,00	20,7 3
22	3,38	1.352,00	208.208,00	164,30
23	10,14	4.056,00	936.936,00	567,88
24	0,29	116,00	5.568,00	12,28
25	2,38	952,00	91.392,00	94,16
26	6,86	2.744,00	395.136,00	318,85
27	16,87	6.748,00	1.295.616,00	751,24
28	26,14	10.456,00	2.509.440,00	1.446,92
29	36,27	14.508,00	4.178.304,00	2.443,80
30	1,48	592,00	42.624,00	61,21
31	10,37	4.148,00	597.312,00	466,88
32	18,40	7.360,00	1.589.760,00	1.588,82
33	38,35	15.340,00	4.417.920,00	3.670,89
Média	6,20	2.480,24	534.404,24	444,87

Tabela 4.2: Comparação quanto ao tempo de CPU

4.2 Resultados para o PRP-UMP

Para avaliar os algoritmos propostos para o Problema de Roteamento Periódico de Unidades Móveis de Pistoneio, foram realizados experimentos em máquinas Intel Pentium 4 2.8GHz com 512MB de RAM, onde os algoritmos foram implementados usando a linguagem C, compilada no gcc versão 3.3.3 rodando sobre Linux Fedora Core 2. No caso dos testes usando método exato, a ferramenta XPRESS 2005 foi utilizada rodando sobre um AMD Athlon 1.6 GHz com 256 Mbytes de RAM.

Uma vez que não se dispõe de instâncias na literatura para o problema apresentado, foram geradas instâncias artificiais divididas em dois grupos. Inicialmente, foram geradas 12 instâncias consideradas pequenas, objetivando-se verificar o desempenho das abordagens em relação ao método exato usando a formulação matemática aqui proposta. Nestas instâncias, o número de dias do período varia de 1 a 3, o número de UMP pode ser 1 ou 2 e o número de poços está no intervalo $[5, 15]$.

O segundo conjunto de testes gerado consiste de 17 instâncias. Neste conjunto o número de dias do período se encontra no intervalo $[5, 15]$; o número de UMP's no conjunto $2, 3$; e o número de poços foi distribuído no intervalo $[50, 1000]$.

4.2.1 Comparação com a Solução Exata

Para verificação dos resultados das abordagens propostas em relação à execução de um método exato foi utilizado o primeiro conjunto de instâncias testes. Na Tabela 4.3, a primeira coluna identifica estas instâncias, onde o primeiro número indica o número de poços, o segundo corresponde ao número de dias do período de planejamento e o terceiro indica o número de UMP's. A segunda coluna mostra o resultado obtido executando-se o método exato (XPRESS), tanto no que se refere ao custo da solução como em relação ao tempo de processamento medido em segundos. Da quarta a sétima coluna são apresentados os resultados, em relação ao tempo computacional, exigidos pelas quatro versões do GRASP apresentadas na Tabela 3.2 para atingirem a solução ótima. Os dois primeiros utilizam a heurística de construção por pétala (G1 e G2) e os demais (G3 e G4) utilizam a construção baseada na heurística de inserção mais próxima. A busca local de cada aborgadem difere quanto à ordem de execução dos algoritmo de busca propostos. O custo da solução não é apresentado, pois todas as versões GRASP alcançaram o valor ótimo para todas as instâncias. Devido à simplicidade das instâncias e ao tempo gasto para determinar a solução ótima, não foi avaliado o módulo de Reconexão de Caminhos para estas instâncias.

Cada heurística GRASP foi executada dez vezes, sendo apresentado na tabela o tempo médio de todas as execuções. Para este tipo de teste o critério de parada do GRASP foi alterado de forma que o término do algoritmo fosse registrado assim que este alcançasse a solução ótima (ou um limite máximo de 200 iterações).

Embora a configuração das máquinas utilizadas para execução das heurísticas GRASP seja superior àquela usada para a execução do XPRESS, pode-se considerar, pela ordem de grandeza nos resultados referentes ao tempo de processamento, que todos os algo-

Instância	XPRESS		G1	G2	G3	G4
	custo	tempo(s)	tempo(s)	tempo(s)	tempo(s)	tempo(s)
I' [05] [1] [1]	114,00	0,06	0,000	0,000	0,000	0,000
I' [10] [1] [1]	450,00	0,43	0,010	0,010	0,010	0,010
I' [10] [2] [1]	900,00	1,51	0,020	0,030	0,030	0,030
I' [10] [2] [2]	1.300,00	216,38	0,030	0,030	0,040	0,030
I' [13] [1] [1]	600,00	2,72	0,020	0,020	0,030	0,030
I' [13] [2] [1]	1.110,00	148,53	0,040	0,040	0,040	0,040
I' [13] [2] [2]	1.610,00	823,80	0,050	0,050	0,060	0,060
I' [14] [1] [1]	300,00	0,22	0,020	0,010	0,020	0,020
I' [14] [1] [2]	550,00	3,10	0,030	0,030	0,030	0,030
I' [14] [2] [2]	1.050,00	123,93	0,050	0,050	0,050	0,050
I' [14] [3] [2]	1.350,00	3,294,11	0,050	0,070	0,050	0,040
I' [15] [3] [1]	620,00	4366,10	0,020	0,020	0,020	0,060
Média	829,5	748,40	0,028	0,030	0,032	0,033

Tabela 4.3: Tempo computacional: Método exato (XPRESS) × Heurísticas GRASP

ritmos propostos mostraram-se muito mais rápidos que o método exato, o que de fato já era de se esperar, haja vista ser exatamente esta a motivação para o uso de heurísticas para problemas desta complexidade. Este resultado tem portanto aspecto apenas comprobatório.

4.2.2 Comparativos entre as abordagens propostas

Testes preliminares foram executados com o intuito de definir, de forma empírica, os parâmetros a serem utilizados para avaliação dos algoritmos propostos. Assim, são assumidos os seguintes valores: $\alpha = 0.30$ e número máximo de iterações MAX_IT = 200. Na etapa de construção, para selecionar o primeiro poço a integrar uma rota, optou-se pela função definida em (3.21) por esta ter apresentado um melhor desempenho em simulações preliminares. Durante a busca local periódica, BLP, o valor do parâmetro δ não é fixo, assumindo diferentes valores durante uma execução. Estes valores que pertencem ao conjunto $\{5, 8, 12, 16\}$ onde cada elemento representa o percentual que será aplicado sobre o número de poços das instâncias.

Na Tabela 4.4 são apresentadas as características das instâncias utilizadas para avaliação das abordagens propostas. Na primeira coluna é apresentado o identificador da instância e na coluna seguinte o número de poços da instância, seguido, pelo número de dias do período analisado e pelo número de veículos disponíveis.

Instância	Características		
Id	nº Poços	Período	nº UMP
Inst 1	50	10	2
Inst 2	50	7	2
Inst 3	50	7	2
Inst 4	50	10	2
Inst 5	50	5	2
Inst 6	100	15	3
Inst 7	100	10	3
Inst 8	150	14	2
Inst 9	150	7	2
Inst 10	200	10	2
Inst 11	250	10	2
Inst 12	250	10	3
Inst 13	400	10	3
Inst 14	500	15	3
Inst 15	700	10	3
Inst 16	800	10	2
Inst 17	10000	10	2

Tabela 4.4: Descrição de cada instância utilizada

Para verificação do desempenho dos algoritmos analisados, cada versão GRASP foi executada 100 vezes para cada uma das 17 instâncias. O valor da melhor entre todas as soluções obtidas, considerando todas as heurísticas, para cada umas das instâncias i, é denotada como $best(i)$. Nas colunas pares da Tabela 4.5 (rotuladas Média) é mostrada a proximidade média de cada algoritmo em relação ao $best(i)$, calculada pela razão $VM\_H\,(i)\,/best(\mathrm{i})$, onde $VM\_H\,(i)$ representa a média dos valores encontrados pela versão H para a instância i. Nas colunas 3, 5, 7, 9, 11, 13, 15 e 17 (Melhor) é mostrado a aproximação da melhor solução encontrada por cada uma das heurísticas em relação ao *best* de cada instância, dada por $VMax\_H\,(i)\,/best(i)$ onde $VMax\_H\,(i)$ representa o valor da melhor solução da heurística H na instância i.

Nesta tabela pode-se observar que, entre os resultados obtidos pelas versões G1, G2, G3 e G4 o melhor resultado médio foi obtido por G4 seguido por G3 (ambas utilizando construção IMP, sendo que a versão G4 executa a BLP antes da BLD). Incluindo o módulo de Reconexão de Caminhos (RC) é possível observar melhoria de cada um dos algoritmos em relação à sua versão pura.

	G1		G1+RC		G2		G2+RC		G3		G3+RC		G4		G4+RC	
Inst.	Média	Melhor	Média	Melhor	Média	Melhor	Média	Melhor	Média	Melhor	Média	Melhor	Média	Melhor	Média	Melhor
Inst 1	0,842	0,866	0,905	0,928	0,846	0,887	0,905	0,928	0,938	0,962	0,968	1,000*	0,943	0,969	**0,972**	0,996
Inst 2	0,819	0,862	0,881	0,913	0,824	0,850	0,878	0,922	0,951	0,977	**0,974**	1,000*	0,953	0,977	0,974	0,992
Inst 3	0,858	0,884	0,887	0,920	0,862	0,893	0,887	0,911	0,965	0,991	0,971	0,991	0,968	0,991	**0,972**	1,000*
Inst 4	0,840	0,872	0,852	0,878	0,841	0,865	0,852	0,878	0,976	1,000*	0,982	1,000*	0,977	1,000*	**0,982**	1,000*
Inst 5	0,897	0,928	0,941	0,972	0,894	0,911	0,943	0,971	0,964	0,986	0,976	1,000*	0,964	0,994	**0,978**	0,998
Inst 6	0,880	0,897	0,887	0,906	0,883	0,912	0,889	0,912	0,986	0,996	0,987	0,997	0,989	0,999	**0,990**	1,000*
Inst 7	0,861	0,890	0,866	0,890	0,863	0,890	0,866	0,890	0,976	0,992	0,977	0,992	0,980	1,000*	**0,990**	1,000*
Inst 8	0,822	0,844	0,909	0,936	0,817	0,837	0,901	0,930	0,971	0,996	0,982	0,996	0,973	0,993	**0,983**	1,000*
Inst 9	0,799	0,832	0,897	0,930	0,795	0,831	0,896	0,939	0,956	0,988	**0,974**	1,000*	0,957	0,985	0,972	0,995
Inst 10	0,811	0,834	0,894	0,910	0,797	0,814	0,878	0,902	0,963	0,972	0,974	0,991	0,967	0,977	**0,976**	1,000*
Inst 11	0,792	0,807	0,859	0,886	0,776	0,797	0,852	0,886	0,980	0,998	0,980	0,998	0,983	1,000*	**0,984**	1,000*
Inst 12	0,797	0,809	0,868	0,893	0,775	0,796	0,861	0,887	0,988	0,999	0,989	0,999	0,991	1,000*	**0,992**	1,000*
Inst 13	0,813	0,837	0,882	0,904	0,793	0,827	0,856	0,883	0,987	1,000*	**0,988**	1,000*	0,985	0,990	0,985	0,990
Inst 14	0,868	0,880	0,872	0,880	0,863	0,872	0,871	0,885	0,787	0,821	0,888	0,911	**0,994**	1,000*	**0,994**	1,000*
Inst 15	0,854	0,864	0,909	0,924	0,834	0,855	0,881	0,898	0,989	0,996	**0,993**	1,000*	0,986	0,997	0,989	0,998
Inst 16	0,794	0,808	0,910	0,920	0,764	0,800	0,889	0,913	0,931	0,959	0,967	1,000*	0,926	0,940	**0,969**	0,985
Inst 17	0,750	0,783	0,906	0,937	0,730	0,742	0,900	0,924	0,940	0,956	**0,986**	0,997	0,931	0,940	0,983	1,000*
Média	0,829	0,853	0,890	0,913	0,821	0,846	0,883	0,909	0,956	0,976	0,974	0,992	0,969	0,985	0,982	0,997

Tabela 4.5: Resultados computacionais das heurísticas GRASP quanto à qualidade da solução; onde em negrito destaca-se a maior média obtida e o $(_*)$ destaca o algoritmo que obteve a melhor solução considerando o resultado de todas as heurísticas.

Para melhor ilustrar o desempenho de cada abordagem, na Figura 4.1 estão apresentados os gráficos referentes aos resultados percentuais médios em relação à melhor solução obtida para cada instância. Assim, uma torre com altura até 90% representa que o GRASP associado obteve resultados médios distantes 10% do melhor valor encontrado. Convém destacar que, mesmo para as versões puras G3 e G4, onde as soluções geradas já são de boa qualidade, na maioria dos casos o módulo de RC ainda conseguiu melhorar um pouco a solução final, o que mostra a eficiência deste procedimento.

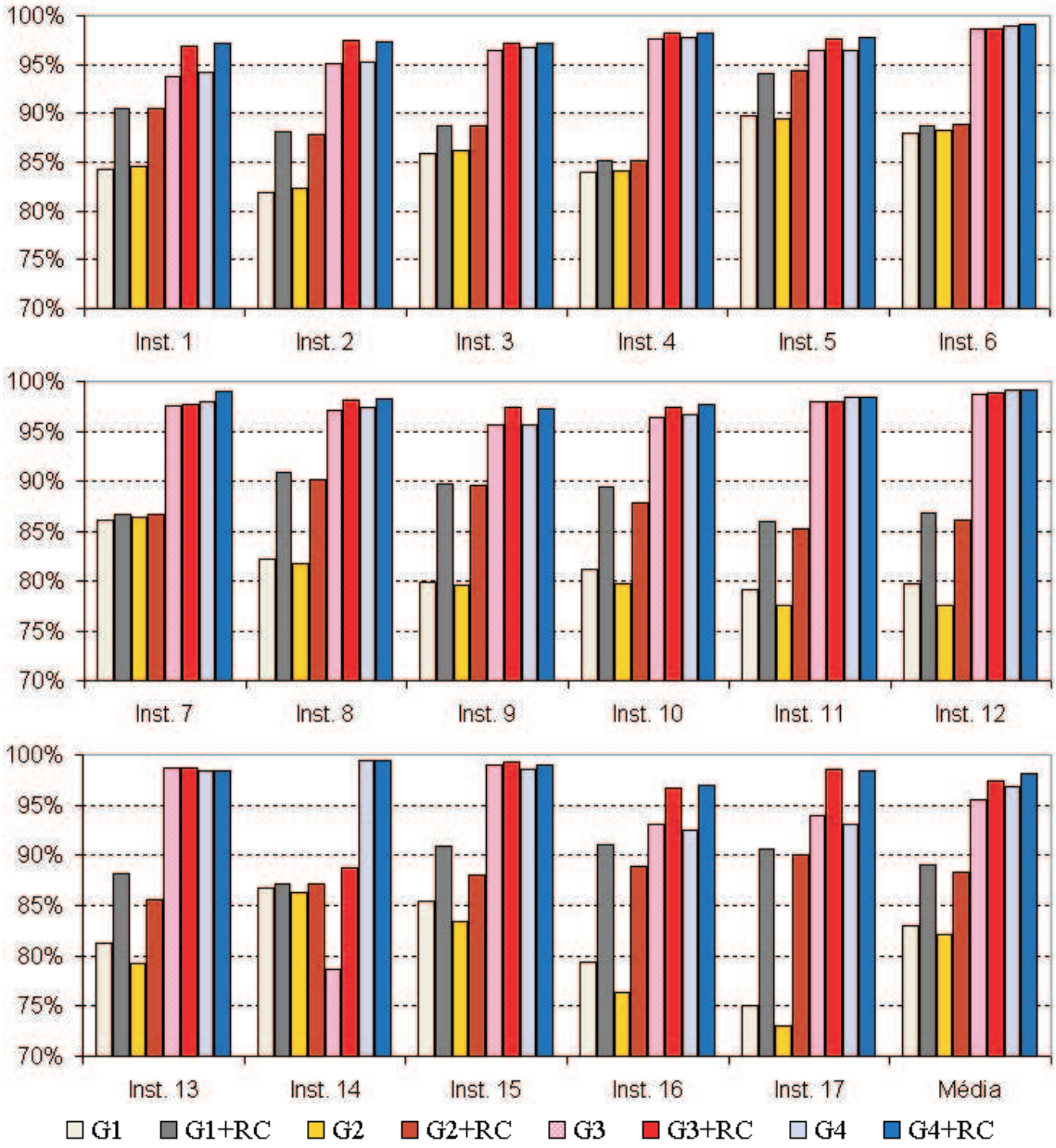


Figura 4.1: Desempenho médio dos algoritmos quanto a qualidade da melhor solução conhecida

Na Tabela 4.6 são ilustradas as melhorias em cada GRASP puro ao se incluir neste a RC (percentual). A última linha da tabela mostra o ganho médio percentual pela aplicação da RC. Por esta linha pode-se observar que a contribuição do RC foi muito maior para os casos onde o GRASP puro utiliza a construção por pétalas (G1 e G2).

Instância	G1	G2	G3	G4
Inst 1	7,51	6,95	3,25	3,08
Inst 2	7,56	6,59	2,46	2,22
Inst 3	3,38	2,91	0,67	0,49
Inst 4	1,40	1,35	0,56	0,47
Inst 5	4,90	5,44	1,29	1,37
Inst 6	0,82	0,65	0,10	0,10
Inst 7	0,60	0,32	0,07	1,04
Inst 8	10,63	10,28	1,14	0,96
Inst 9	12,30	12,73	1,92	1,66
Inst 10	10,21	10,18	1,09	1,02
Inst 11	8,51	9,82	0,02	0,02
Inst 12	8,96	11,13	0,10	0,08
Inst 13	8,52	7,92	0,07	0,00
Inst 14	0,51	0,92	12,88	0,00
Inst 15	6,45	5,59	0,36	0,34
Inst 16	14,68	16,38	3,91	4,72
Inst 17	20,80	23,28	4,89	5,62
Média	7,51	7,79	2,05	1,37

Tabela 4.6: Ganho percentual na qualidade da solução devido à aplicação da RC

Para verificar a contribuição de cada etapa dos GRASP's puros (construção e busca local), durante a execução dos testes foram computados os tempos e o acréscimo na função objetivo associados a cada um destes módulos. A Figura 4.2 ilustra o percentual na qualidade da solução atribuído às heurísticas de construção e busca local dos GRASP's. No eixo das abscissas estão indicadas as instâncias e, na última coluna, a média (M), enquanto no eixo das ordenadas encontram-se indicados os ganhos percentuais. A figura mostra que os construtivos aqui propostos apresentam na média, soluções de boa qualidade (comprimento da parte preta de cada torre). Isso é mais acentuado quando se usa o construtivo IMP (gráficos (c) e (d)). Ainda sobre as Figuras 4.2, pode-se observar que a ordem de aplicação das heurísticas de busca local tende a afetar a contribuição destas para a qualidade da solução final do GRASP, o que pode ser melhor observado nas Figuras 4.2 (a) e (b). Na versão G1, apresentada na Figura 4.2 (a), onde a busca local diária é aplicada antes da busca local periódica, observa-se que ambas as buscas contribuem significativamente para a qualidade final da solução, o que não ocorre na versão G2 (Figura 4.2 (b)), cuja ordem de chamada das heurísticas de busca é inversa, tendo as duas versões (G1 e G2) utilizado a mesma heurística de construção.

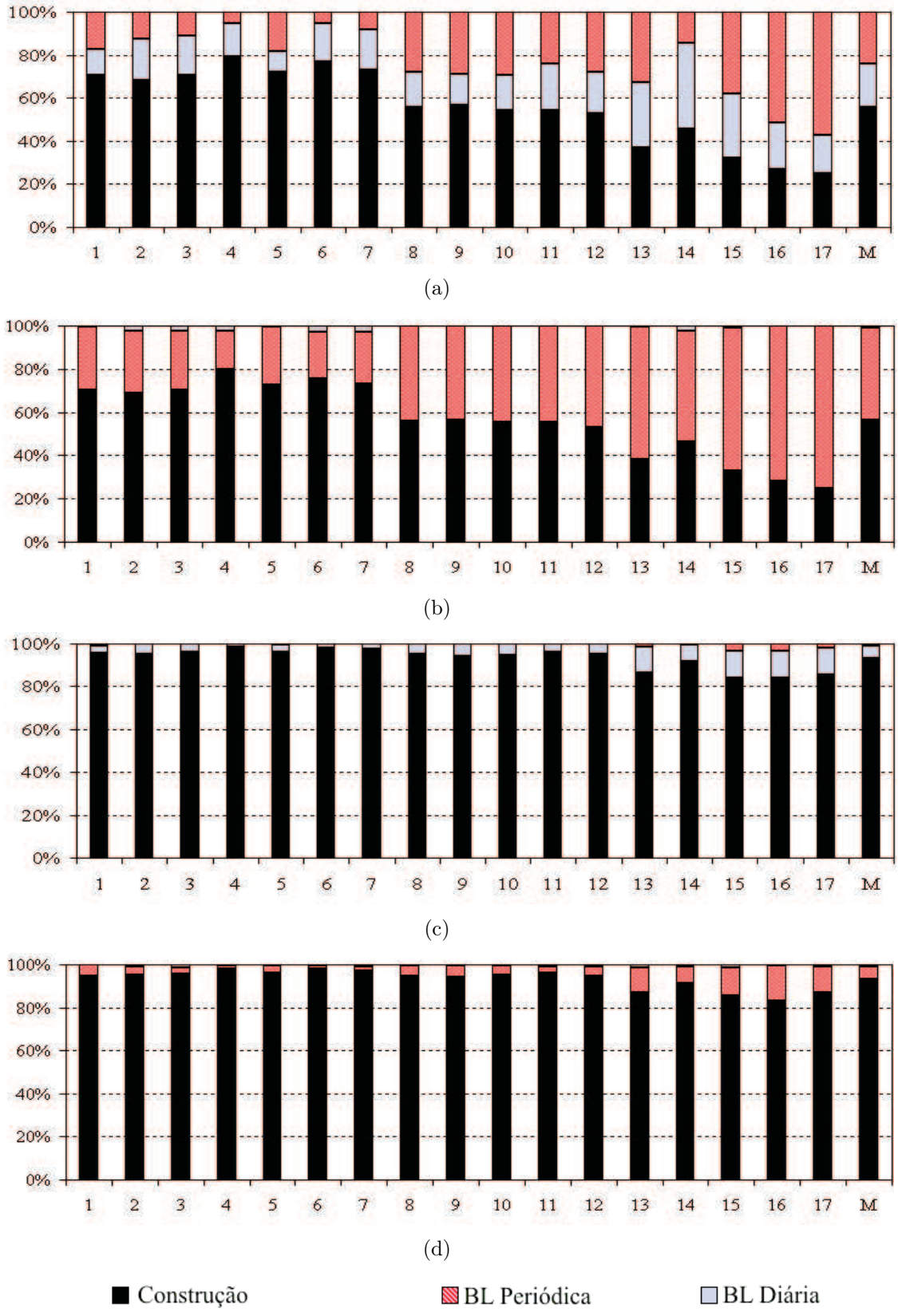


Figura 4.2: Contribuição média de cada etapa do GRASP para a qualidade da solução: (a)- G1, (b)- G2, (c)- G3 e (d)- G4

Sabe-se que, em geral, o uso de módulos adicionais, como por exemplo a Reconexão de Caminhos, numa metaheurística tende a provocar um acréscimo significativo no tempo total de processamento do algoritmo. O impacto do uso da RC é ilustrado na Tabela 4.7, que apresenta os tempos médios de cada versão GRASP com e sem RC. Os resultados mostram que mesmo tendo sido chamado em todas as iterações GRASP, a RC não provoca um acréscimo significativo no tempo final do GRASP.

Inst	G1	G1+RC	G2	G2+RC	G3	G3+RC	G4	G4+RC
1	3,84	4,48	3,75	4,39	4,66	5,31	4,33	5,00
2	3,81	4,10	4,14	4,43	5,78	6,08	5,08	5,38
3	4,19	4,42	4,34	4,56	6,13	6,35	5,46	5,68
4	4,42	4,90	4,56	5,02	6,62	7,04	5,72	6,15
5	2,25	2,38	2,14	2,28	2,47	2,62	2,13	2,27
6	45,91	49,19	47,79	51,08	57,52	60,97	53,82	57,25
7	15,65	16,72	16,63	17,73	20,14	21,26	18,77	19,89
8	88,92	92,94	86,60	90,53	100,07	104,47	97,08	101,35
9	34,83	35,73	35,00	35,93	40,82	41,79	38,08	39,06
10	124,38	126,92	122,65	125,23	141,97	144,83	138,98	141,83
11	184,58	187,40	183,72	186,59	225,48	228,49	220,45	223,48
12	152,30	155,39	154,37	157,50	218,61	222,16	221,19	224,77
13	497,68	502,68	328,36	331,74	1.060,58	1.065,86	733,79	737,36
14	1.980,75	1.994,37	2.261,20	2.274,58	13.373,46	13.391,67	12.186,41	12.205,68
15	4.109,28	4.119,22	2.922,87	2.929,95	8.869,12	8.880,22	6.574,11	6.581,91
16	6.664,71	6.674,70	7.099,63	7.109,63	9.711,30	9.724,06	9.153,86	9.166,90
17	6.086,22	6.102,49	7.821,34	7.841,50	9.411,04	9.433,12	9.161,23	9.183,08
Média	1.176,69	1.181,06	1.241,12	1.245,45	2.544,46	2.549,78	2.271,79	2.276,88

Tabela 4.7: Tempo médio de CPU em segundos

4.2.2.1 Avaliação Probabilística dos Resultados

Nesta seção, é colocada como meta a análise da regularidade das heurísticas aqui propostas. Isso se torna importante na medida 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 [Aiex et al. 2002]. 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 (em problemas de maximização) ao alvo fixado, o tempo é registrado e a execução é interrompida.

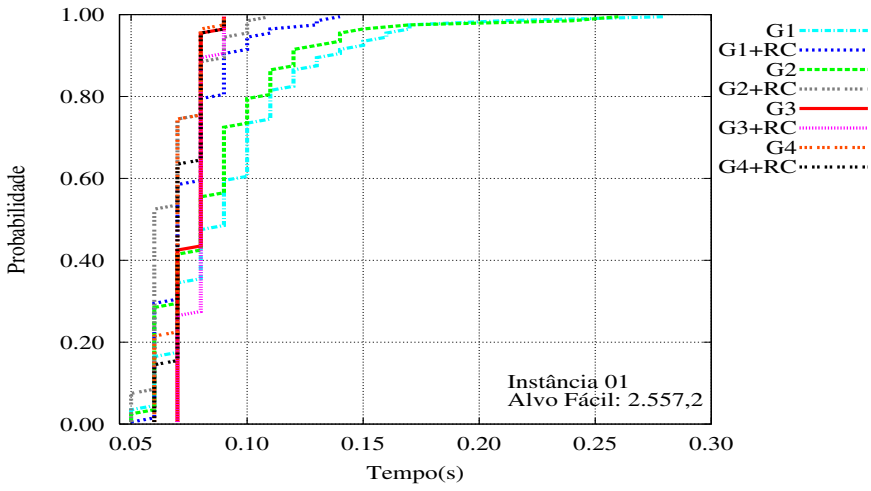
Para esta bateria de testes, cada algoritmo foi executado 100 vezes para cada instância avaliada usando os mesmos parâmetros inicialmente adotados nas análises anteriores. Após 100 execuções, os tempos foram tomados e dispostos em ordem crescente numa lista L. A cada tempo de CPU t obtido, foi associada a probabilidade $p_i = (i - \frac{1}{2})/100$, onde

i é a ordem 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 as 200 iterações, lhe são concedidas mais 100 iterações para que o mesmo possa alcançar o alvo. Caso, mesmo após as 300 iterações, o alvo não seja obtido considera-se que nesta execução o mesmo não foi capaz de atingir o mesmo.

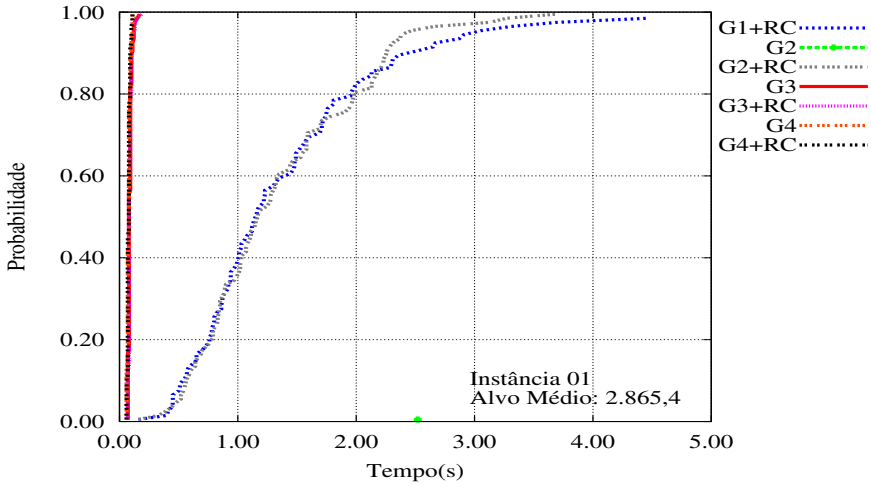
Inicialmente, para se determinar os alvos a serem usados, cada algoritmo proposto foi executado dez vezes com sementes diferentes por 200 iterações. Para cada versão GRASP, calculou-se o resultado médio atingido para cada uma das instâncias. Os alvos *fáceis* são assumidos como a média alcançada pela versão com pior desempenho. Os alvos *médios* têm valor igual a média de todos os resultados e, por fim, os alvos *difíceis* correspondem a 95% da média atingida pela versão GRASP de melhor desempenho. Tendo em vista o aumento no número de iterações para as heurísticas, observou-se que as instâncias cujo número de poços era muito elevado consumiam um tempo excessivo de processamento. Assim, para este tipo de análise, foram escolhidas apenas as instâncias 1, 6 e 8, com 50, 100 e 150 poços, respectivamente.

Na Figura 4.3 (a) é apresentado o resultado para a instância 1 com alvo fácil. Convém destacar que as versões que utilizam construção por pétala (G1 e G2), quando não empregam RC necessitam de mais de 0.25 segundos para que se possa afirmar com certeza que estas convirjam empiricamente para o alvo, ao passo que suas versões com RC necessitam de apenas a metade deste tempo. Isto demonstra a contribuição real do uso do RC para estas abordagens quando o alvo é fácil.

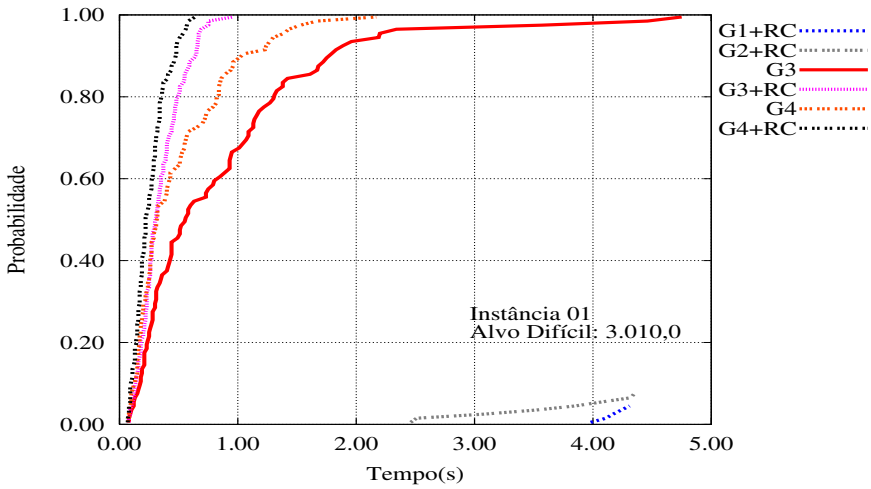
Para o alvo médio, mostrado na Figura 4.3 (b), ainda sobre a instância 1, a versão G1 sem RC não convergiu em nenhuma das 100 execuções e a versão G2 o fez apenas uma única vez. Já as versões G1+RC e G2+RC conseguiram atingir o alvo em todas as execuções, mas com tempos excessivamente elevados quando comparados às versões com construção IMP. Cabe ressaltar que, no tempo em que a probabilidade de G1+RC ou G2+RC atingirem o alvo é de ainda 0%, as versões G3 e G4, com ou sem RC já atingiram o alvo com probabilidade 1. Por outro lado, no caso do alvo difícil para instância 1, Figura 4.3 (c), as versões com construção por Pétala, mesmo utilizando Reconexão de Caminhos, conseguiram atingir o alvo em apenas algumas execuções e , além disso, pode-se perceber notadamente que as versões G3 e G4, quando utilizam RC têm muito mais chance de convergir em um tempo mais cedo.



(a)



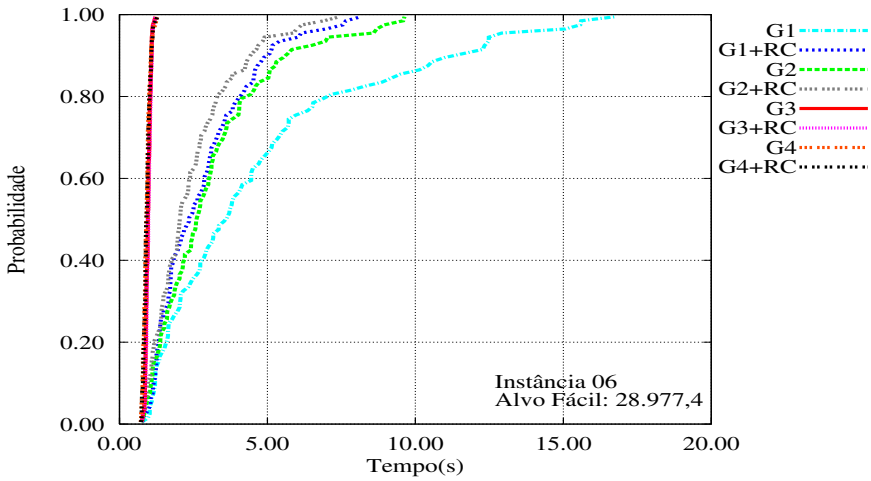
(b)



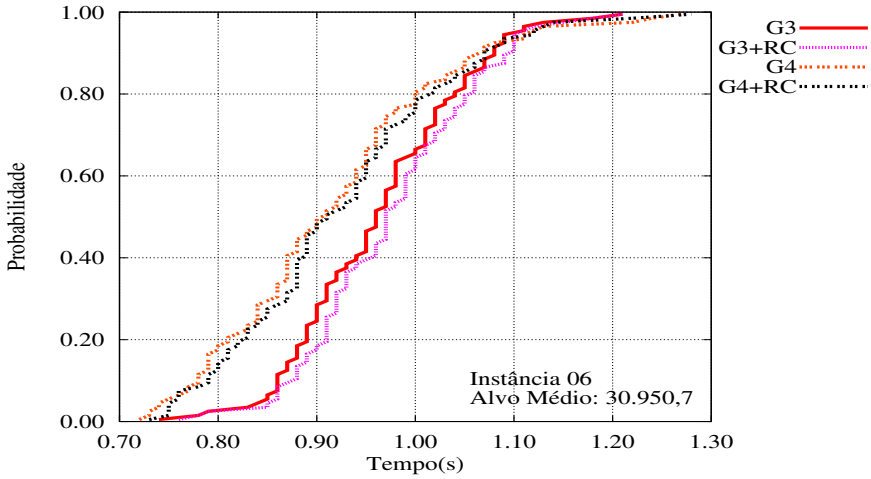
(c)

Figura 4.3: Análise Probabilística para a instância 01 [50 poços][10 dias][2 UMP's]: (a)- Alvo Fácil, (b)- Alvo Médio e (c)- Alvo Difícil

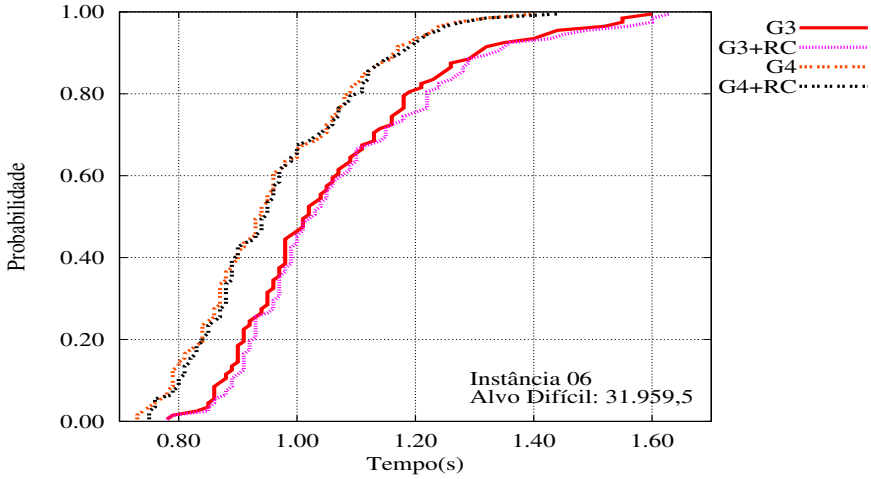
Ao tomar uma instância com o dobro do número de poços e com período e número de UMP´s maiores do que na instância 1, no caso, a instância 6, o desempenho dos algoritmos mantém a tendência. A Figura 4.3 (a) mostra que, para um alvo fácil, o desempenho ligeiramente equilibrado entre as versões com tipos diferentes de construção para os tempos iniciais apresentados para a instância 1 não se confirmam para a instância 6. Pode-se perceber que as versões GRASP G3 e G4 com e sem RC têm convergência precoce se comparadas às versões G1 e G2. Além disso, observa-se que o impacto da RC no desempenho do algoritmo G1 é muito mais significativo que para qualquer uma das abordagens, sendo que, quando G1+RC converge com 100% de acerto demandando um tempo inferior a 10 segundos, para esta mesma convergência, a versão sem RC exige mais de 15 segundos. No entanto, a eficiência do módulo de RC, a exemplo do que pode ser observado para a instância 1, não foi suficiente para que as abordagens que utilizam construção por pétala convergissem para os alvos médio e difícil.



(a)



(b)



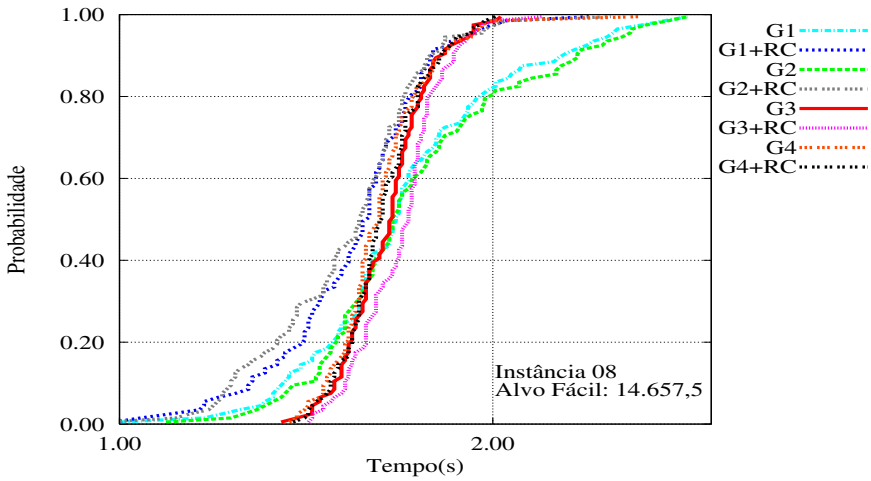
(c)

Figura 4.4: Análise Probabilística para a instância 06 [100 poços][15 dias][3 UMP's]: (a)- Alvo Fácil, (b)- Alvo Médio e (c)- Alvo Difícil

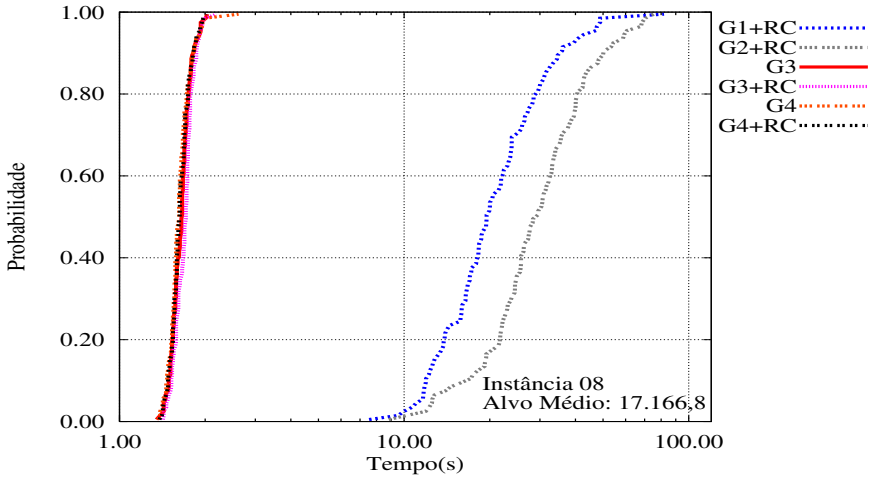
No caso da instância 8, com 150 poços, 2 UMP´s e período de 14 dias, a Figura 4.5 (a) mostra que, para um alvo fácil, se for fixada probabilidade de 0.6, os algoritmos apresentam convergência ligeiramente similar, com uma posterior queda de desempenho das versões G1 e G2. Mesmo assim, para se garantir com certeza que o alvo fácil foi atingido, os tempos exigidos pelas versões G1, G2 e G3 são praticamente os mesmos. Além disso, mais uma vez pode-se observar a contribuição do módulo de RC para todas as versões.

Para o alvo médio, como mostra a Figura 4.5 (b), observa-se que as versões G3 e G4 e as respectivas versões com RC obtiveram melhor desempenho. Diferente do que foi observado na instância 6, para os alvos médio e difícil houve convergência para as versões G1+RC e G2+RC. Entretanto, o tempo da execução mais tardia de qualquer um dos algoritmos que usam construção IMP com ou sem RC é sempre inferior à execução de mais breve convergência de qualquer algoritmo que utiliza construção por pétala.

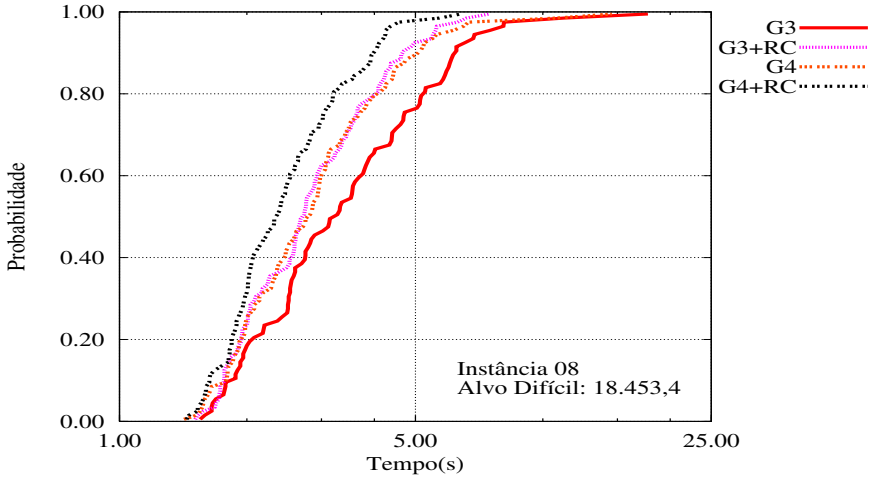
Ainda sobre a instância 8, para o alvo difícil vê-se que nenhuma das abordagens com construção por pétala convergiu para o alvo com o número de iterações dado. Além disso, observa-se na figura que a melhor versão é a G4+RC, seguida das versões G4, G3+RC e G3.



(a)



(b)



(c)

Figura 4.5: Análise Probabilística para a instância 08 [150 poços][14 dias][2 UMP's]: (a)- Alvo Fácil, (b)- Alvo Médio e (c)- Alvo Difícil

Em termos gerais, pode-se observar que as versões mais regulares, que sempre atingindo o alvo, foram: G3, G4, G3+RC e G4+RC. Outro aspecto importante é a convergência das versões com RC em relação as suas versões puras (sem RC). Nesta análise, percebe-se que a introdução do módulo de RC não onera o tempo para atingir o valor alvo e, em muitos casos, o uso da RC permite agilizar a obtenção deste valor. Esta observação, mesmo de forma empírica, mostra que, no processo de avaliar heuristicas, diferentes critérios de parada devem ser considerados.

Capítulo 5

Conclusões e Trabalhos Futuros

Neste trabalho foi apresentado um estudo de uma variante do Problema de Roteamento de Veículo Periódico - PRVP aqui denotado como Problema do Roteamento Periódico de Unidades Móveis de Pistoneio - PRP-UMP.

Na fase inicial de pesquisa bibliográfica sobre PRPV, foi encontrado um trabalho recente para um caso especial do PRVP conhecido como Problema do Caixeiro Viajante Periódico (PCVP) proposto em [Bertazzi et al. 2004]. Neste trabalho os autores apresentaram uma nova heurística que obteve os melhores resultados para um conjunto de instâncias do PCVP. Apesar da heurística proposta conter contribuições interessantes, a mesma demanda tempos computacionais muito elevados, o que pode dificultar a solução de instâncias de maiores dimensões. Este fato motivou se pensar num método que mantivesse a qualidade das soluções obtidas neste algoritmo reduzindo-se o tempo computacional exigido. Com este objetivo, incorporou-se esta heurística numa estrutura do tipo GRASP. Os resultados computacionais obtidos para instâncias da literatura analisadas mostraram que o GRASP proposto consegue, em média, soluções muito próximas às do algoritmo original, mas exigindo tempos computacionais muito menores.

Para a solução aproximada do PRP-UMP, incorporaram-se algumas das idéias do GRASP desenvolvido para o PCVP adaptando-as para a solução do PRP-UMP. Desta forma, foram propostas duas heurísticas de construção, duas de busca local, além de um procedimento de busca intensiva conhecido na literatura como Reconexão de Caminhos (RC).

Para o desenvolvimento dos algoritmos de construção, foram empregados conceitos clássicos amplamente utilizados em heurísticas para solução de problemas de roteamento, como inserção mais próxima e estrutura de pétalas. Em relação às duas buscas locais

propostas neste trabalho, uma delas tem como finalidade efetuar um refinamento nas rotas diárias já encontradas para as unidades móveis de pistoneio, enquanto a segunda busca (BLP), explora uma vizinhança mais ampla, envolvendo todo período de planejamento.

Para avaliar cada versão GRASP e cada módulo de construção e busca local, foram efetuados basicamente três tipos de experimentos. O primeiro, para comparar soluções heurísticas com a solução exata em instâncias de pequenas dimensões. Neste caso, foi proposta uma formulação matemática do PRP-UMP descrevendo-o como um Problema de Programação Linear Inteira (PPLI). Esta formulação foi utilizada para obter soluções exatas para instâncias de pequeno porte, de modo a possibilitar a comparação entre os limites obtidos pelas heurísticas com o valor ótimo. Nestes testes, verifica-se que, em todos os casos onde o ótimo pode ser encontrado, as heurísticas sempre atingiram o valor ótimo, exigindo, entretanto, tempos computacionais bem menores que os exigidos pela resolução exata.

Na segunda bateria de testes, o objetivo foi comparar o desempenho dos algoritmos GRASP propostos neste trabalho. Inicialmente, foram avaliadas quatro versões dos algoritmos GRASP, denotadas por GRASP's puros, onde em cada uma destas, uma construção diferente foi usada e as duas buscas locais foram usadas sempre em conjunto, mas em ordem diferente.

Na avaliação dos GRASP's puros, observou-se que as versões que usam a construção baseada na inserção mais próxima tiveram melhor desempenho médio que as que usam estrutura de pétalas. Em termos de busca local, foi observado nos testes empíricos que, pelo menos em algumas das instâncias, a ordem em que as duas buscas são usadas pode alterar o resultado final do GRASP. Em geral, a utilização da busca local diária antes da busca local periódica implicou em melhoria nas soluções obtidas a partir da construção, não mantendo o mesmo comportamento quando utilizada após a busca local periódica. Uma explicação para este fato seria que a busca local periódica pode realizar uma busca mais ampla, e os movimentos realizados que afetam todos os dias do período tendem também a refinar alocações diárias do problema.

Posteriormente, após a análise de desempenho das versões GRASP puras, foi verificado o impacto da introdução do módulo de Reconexão de Caminhos (RC) nestes algoritmos, criando-se assim, versões GRASP+RC. Os resultados obtidos foram muito promissores. Comparando cada GRASP puro com sua versão GRASP+RC esta última sempre produziu soluções médias de melhor qualidade em tempos computacionais pouco maiores.

Na análise da contribuição de cada módulo do GRASP puro e GRASP+RC percebeu-

se, empiricamente, que os casos onde a heurística de construção não produz soluções de boa qualidade (versões GRASP com construção por Pétalas), o papel das buscas locais e da Reconexão de Caminhos é muito mais relevante. Nestas situações, percebeu-se que, mesmo saindo de soluções iniciais de baixa qualidade, o uso conjunto das buscas locais e Reconexão de Caminhos resultou numa solução final GRASP de boa qualidade. Nos casos onde a solução inicial já era de boa qualidade, o trabalho das buscas locais e RC foi menos eficaz, mas pequenas melhorias foram obtidas.

A Análise Probabilística Empírica mostrou novamente a superioridade de algumas versões incluindo Reconexão de Caminhos pois estas quase sempre convergirem para o alvo estipulado, ao contrário de algumas versões GRASP puras, cuja taxa de convergência foi muito pequena, principalmente para alvos considerados difíceis.

O bom desempenho das versões GRASP com Reconexão de Caminhos mostra que o uso de algum tipo de memória (no caso, armazenar um conjunto elite) nesta heurística tende a produzir bons resultados mostrando um caminho promissor para versões adaptativas do GRASP.

Como trabalhos futuros, pode-se incluir o uso de outras formas de busca local ou o uso de diferentes estratégias de busca numa vizinhança como, por exemplo, o VNS. Devido ao seu bom comportamento na solução aproximada de diferentes problemas de roteamento de uma frota de veículos, a metaheurística Busca Tabu poderia ser investigada para ser empregada na resolução do PRP-UMP.

Referências

[Aiex et al. 2002] Aiex, R.; Resende, M. G. C. e Ribeiro, C. C. **P**robability distribution of solution time in GRASP: An experimentak investigations. *Journal of Heuristics*, v. 8, p. 343–373, 2002.

[Aloise et al. 2002] Aloise, D. J.; S., L.; Maia, R. S. e Bittencourt, V. G. **A**nt Colony Systems para um problema de extração de Petróleo e Otimização de Rotas de Unidades Móveis de Pistoneio. In: *XIV Congresso Brasileiro de Automática*. 2002.

[Aloise et al. 2001] Aloise, D. J.; Santos, A. C.; Barros, C. A.; Souza, M. C. e Noronha, T. F. **U**m Algoritmo GRASP Reativo Aplicado ao Problema da Unidade Móvel de Pistoneio (POE-UMP). In: *XXXIII SBPO*. 2001. p. 247–258.

[Angelelli e Speranza 2002] Angelelli, E. e Speranza, M. G. **T**he periodic vehicle routing problem with intermediate facilities. *European Journal of Operational Research*, Elsevier Science Publishers B. V., v. 137, p. 233–247, 2002.

[Bertazzi et al. 2004] Bertazzi, L.; Paletta, G. e Speranza, M. **A**n improved heuristic for the period traveling salesman problem. *Computers Operational Research*, v. 31, p. 1215–1222, 2004.

[Bodin e A. Assad 1983] Bodin, L. D. e A. Assad, M. B. **T**he state of the art in the routing and scheduling os vehicles and crews. *Computers and Operational Research*, v. 10, p. 149–180, 1983.

[Campello e Maculan 1994] Campello, R. E. e Maculan, N. **A**lgoritmos e heurísticas: desenvolvimento e avaliação de performance. NiteróiEDUFF, 1994.

[Chao et al. 1995] Chao, I.-M.; Golden, B. L. e A.Wasil, E. **A** new heuristic for the period traveling salesman problem. *Computers and Operations Research*, v. 22, p. 553–565, 1995.

[Christofides e Beasley 1984] Christofides, N. e Beasley, J. E. **T**he period routing problem. *Networks*, v. 14, p. 237–256, 1984.

[Cordeau et al. 1997] Cordeau, J.; Gendreau, M. e Laporte, G. **A** Tabu Search Heuristic for Period and Multi-Depot Vehicle Routing Problems. *Networks*, v. 30, p. 105–119, 1997.

[Dueck 1990] Dueck, G. **N**ew optimization heuristic, the gread deluge algorithm and the record-to-record travel. *Scientific Center Technical Report*, v. 104, p. 86–93, 1990.

[Feo e Resende 1995] Feo, T. A. e Resende, M. G. C. **G**reedy randomized adaptive search procedures. *Journal of Global Optimization*, v. 6, p. 109–133, 1995.

[Fisher e Jaikumar 1981] Fisher, M. L. e Jaikumar, R. **A** generalized assigment heuristic for vehicle routing. *Networks*, v. 11, p. 109–124, 1981.

[Fleurent e Glover 1999] Fleurent, C. e Glover, F. **I**mproved constructive multistart strategies for the quadratic assignment problem using adaptive memory. *INFORMS Journal on Computing*, v. 11, p. 198–204, 1999.

[Foster e Ryan 1976] Foster, B. A. e Ryan, D. M. **A**n integer programming approach to the vehicle scheduling problem. In: *Operational Research Quartely*. 1976. v. 27, p. 377–384.

[Gendreau et al. 1992] Gendreau, M.; Hertz, A. e Laporte, G. **N**ew Insertion and Postoptimization procedures for the traveling salesman problem. *Operations Research*, v. 40, p. 1086–1094, 1992.

[Glover 1996] Glover, F. **T**abu search and adaptive memory programing - Advance, applications and challenges. In: . : R. S. Barr, R. V. Helgason and J. L. Kennington, 1996. p. 1–75.

[Golden et al. 1987] Golden, B.; Levy, L. e Vohra, R. **T**he Orienteering Problem. *Naval Research Logistics*, v. 34, p. 307–318, 1987.

[Golden et al. 1995] Golden, B. L.; Chao, M. e Wasil, E. **A**n improved heuristic for the Periodic Vehicle Routinhg Problem. *Networks*, v. 26, p. 25–44, 1995.

[Keller 1989] Keller, P. C. **A**lgorithms to solve the Orienteering Problem: A comparison. *European Journal of Operational Research*, v. 41, p. 224–231, 1989.

[Laporte 1992] Laporte, G. **T**he vehicle routing problem: An overview of exact and approximate algorithms. *European Journal of Operational Research*, v. 59, p. 345–358, 1992.

[Martins et al. 1999] Martins, S.; Pardalos, P.; Resende, M. e Ribeiro, C. **G**reedy randomized adaptive search procedures for the steiner problem in graphs. *DIMACS Series on Discrete Mathematics and Theoretical Computer Science*, v. 43, p. 133–145, 1999.

[Ochi et al. 2001] Ochi, L. S.; Vianna, D. S. e Drummond, L. M. A. **A**n Asynchronous Parallel Metaheuristic for the Period Vehicle Routing Problems. *Future Generation Computer Systems*, Elsevier, v. 17, p. 379–386, 2001.

[OR-Library]**O**r-library. `http://people.brunel.ac.uk/~mastjjb/jeb/info.html`.

[Paletta 2002] Paletta, G. **T**he period traveling salesman problem: a new heuristic algorithm. *Computers and Operational Research*, v. 29, p. 1345–1352, 2002.

[Pearn e Chiens 1998] Pearn, W. L. e Chiens, R. C. **I**mproved solutions for the traveling purchaser problem. *Comput. Oper. Res.*, Elsevier Science Ltd., v. 25, p. 879–885, 1998.

[Prais e Ribeiro 2000] Prais, M. e Ribeiro, C. **A**n application to a matrix decomposition problem in TDMA traffic assignment. *INFORMS Journal on Computing*, v. 12, p. 164–176, 2000.

[Resende e Ribeiro 2005] Resende, M. e Ribeiro, C. **G**RASP with path-relinking: Recent advances and applications. In: *Metaheuristics: Progress as Real Problem Solvers*. 2005. p. 29–63.

[Russel e Gribbin 1991] Russel, R. A. e Gribbin, D. **A** multi-phase approach to the period routing problem. *Networks*, v. 21, p. 747–765, 1991.

[Tan e Beasley 1984] Tan, C. C. R. e Beasley, J. E. **A** heuristic algorithm for the period vehicle routing problem. *Omega*, v. 12, p. 497–504, 1984.

[Teeninga e Volgenant 2004] Teeninga, A. e Volgenant, A. **I**mproved heuristics for the traveling purchaser problem. *Computers and Operations Research*, Elsevier Science Ltd., v. 31, p. 139–150, 2004.