Proposta e Avaliação Experimental de Heurísticas GRASP para um Problema de Escalonamento de Veículos

Viviane de Aragão Trindade
Instituto de Computação – Universidade Federal Fluminense
Rua Passo da Pátria, 156, Bloco E, 3 andar
24210-240, Niterói, Rio de Janeiro, Brasil
e mail: vtrindade@ic.uff.br

Luiz Satoru Ochi (*)
Instituto de Computação – Universidade Federal Fluminense
Rua Passo da Pátria, 156, Bloco E, 3 andar
24210-240, Niterói, Rio de Janeiro, Brasil
e mail: (*) satoru@ic.uff.br

Resumo

Este artigo apresenta versões da heurística GRASP aplicadas a um problema de roteamento para um conjunto de sondas de manutenção de poços petrolíferos terrestres na região nordeste do Brasil. Este problema é descrito por um conjunto de sondas disponíveis e um conjunto de poços que necessitam de serviços de manutenção. Cada poço com este status possui associado uma vazão diária de óleo e uma estimativa do número de dias previsto para a sua recuperação. O objetivo deste problema, é a de obter um escalonamento otimizado no uso das sondas de modo que se minimize a quantidade total de óleo que deixou de ser coletado devido à interrupção da extração nestes locais. São propostas diferentes versões de algoritmos de construção, busca local e reconexão por caminhos a serem aplicados num GRASP para esta aplicação. O desempenho dos algoritmos propostos é mostrado através de diferentes tipos de avaliações computacionais. Adicionalmente é apresentada uma análise probabilística mostrando a robustez dos algoritmos em termos de convergência para valores alvos pré-definidos.

palavra chaves: metaheuristicas, otimização, problemas de escalonamento de veículos.

Abstract

This paper presents versions of GRASP heuristic applied to a Vehicle Routing Problem where the vehicles are associated to a set of workover rigs of maintenance of terrestrial oil wells in the northeast region of Brazil. This problem is described by a set of available workover rigs and a set of oil wells that need maintenance services. Each well with this status has associated an amount daily of oil production that can be collected and an estimate of the number of days to conclude the maintenance service. The objective of this problem is to get a scheduling optimized in the use of the workover rigs that minimize the production loss associated with the wells awaiting for service. We present different proposals of construction, local search and path relinking algorithms for GRASP applied to this problem. The performance of the proposed algorithms is shown for different types of computational experiments. Additionally we present probabilistic analysis showing the robustness of the algorithms in terms of convergence for fixed target values.

keywords: metaheuristics, optimization, vehicle scheduling problems.

(*) autor para correspondência

1. Introdução

As metaheurísticas tem se mostrada uma das ferramentas mais eficientes na solução aproximada de problemas de elevada complexidade computacional (problemas NP Completo e NP Difícil). Dentre as diferentes metaheurísticas existentes na literatura, algumas tem se destacado como é o caso de: *Tabu Search* (TS), *Greedy Randomized Adaptive Search Procedure* (GRASP), *Variable Neighborhood Search* (VNS), e Algoritmos Genéticos Híbridos (AGH) também conhecidos como Algoritmos Evolutivos (AEs). Métodos TS e AEs já são bastante explorados pela literatura na solução de problemas de roteamento de veículos (PRV). Quanto ao GRASP, embora já bastante aplicado e com sucesso na solução de diferentes problemas de otimização combinatória, para o PRV ainda temos poucas contribuições usando esta técnica (veja [5, 7, 8]).

Este artigo tem como objetivo, desenvolver e analisar experimentalmente, diferentes versões da heurística GRASP para a solução de um problema de escalonar de forma eficiente um conjunto de sondas de manutenção de poços petrolíferos terrestres da região nordeste do Brasil. A retirada de hidrocarbonetos (óleo e gás) do subsolo seja em terra (*onshore*) ou mar (*offshore*) é uma atividade de elevado custo. E por se tratar de recursos não renováveis as jazidas necessitam ser explotadas por métodos comprovadamente eficientes a fim de serem mais bem aproveitadas. A bacia de potiguar terrestre de petróleo e gás, possui 77 campos petrolíferos situados no RN e parte do CE, abrangendo aproximadamente 4.500 poços com produção em torno de 18.300 metros cúbicos por dia. (veja [2, 3]).

Um dos problemas que afetam o funcionamento normal dos campos de petróleo terrestre, diz respeito à ocorrência de falhas nos equipamentos localizados no fundo dos poços. Quando ocorre uma falha no poço, uma equipe de manutenção é designada para a correção da mesma, a fim de recolocar o poço paralisado novamente em funcionamento o mais breve possível. Esse conjunto que abrange a equipe de manutenção e os equipamentos necessários à execução da mesma, é denominado de *Sondas de Produção Terrestre* – SPT [2, 3].

Serviços como os de limpeza e manutenção dos equipamentos alocados nestes poços são essenciais para evitar paradas na sua extração. Os serviços de manutenção são realizados por sondas, que estão disponíveis em um número limitado de acordo com o número de poços que necessitam de serviços. Por causa do alto custo de operação dessas sondas, elas estão disponíveis em um número muito pequeno se comparado à quantidade de poços que demandam serviços. Por isso, a manutenção imediata nem sempre é possível e uma fila de poços se formará a espera de atendimento por alguma sonda.

A empresa responsável por gerenciar essas sondas recebe pedidos de serviços para determinados poços. O escalonamento dessas sondas é feito para um horizonte de atendimento de T dias. No entanto, algumas restrições devem ser consideradas. Por exemplo, embora, a princípio, a empresa possa estimar o tempo para determinado conserto de um poço Pj, este tempo pode na prática se tornar bem maior. Portanto, alguns poços podem não ser atendido no atual período de planejamento e estes devem então entrar no próximo período de planejamento. Entretanto, existe também uma restrição de que cada serviço (poço) deve ser necessariamente atendido em um tempo máximo (TMAX) de dias previamente estipulados.

O *Problema de Gerenciamento de Sondas Terrestres (PGST)* consiste em encontrar para cada horizonte de T dias o melhor escalonamento para as sondas disponíveis, minimizando a perda total de produção de petróleo associada aos poços que estão aguardando por serviços.

A perda de produção de petróleo de um poço é avaliado como a média de vazão diária do poço multiplicado pelo número de dias que esse poço teve sua produção interrompida [2, 3].

Para o nosso caso em particular vamos considerar as sondas como sendo homogêneas; isto é, todas as sondas podem realizar todos os serviços, contudo os algoritmos aqui propostos podem facilmente serem adaptados para um ambiente heterogêneo. O PGST pode também ser visto como um problema de encontrar roteiros otimizados para cada sonda de forma que atenda às restrições associadas e a perda do óleo não coletado seja minimizado.

O PGST é um problema de elevada complexidade computacional (NP-Completo), incentivando com isso o desenvolvimento de algoritmos aproximados ou heurísticos. Algumas heurísticas já foram propostos para resolver esse problema; uma heurística usando conceitos de *Ant Colony System* [3], e um VNS [2]. Nestes trabalhos [2, 3], os autores trabalham com conjuntos de

instâncias baseadas em características do modelo real brasileiro; no entanto estas instâncias não estão disponíveis em bibliotecas públicas dificultando possíveis comparações com outras propostas. Dos artigos da literatura, o algoritmo apresentado em [2] é o algoritmo mais elaborado. Neste, os autores apresentam uma heurística usando conceitos da metaheuristica VNS (*Variable Neighborhood Search*) obtendo resultados bem superiores que os encontrados em [3].

Neste trabalho, colocamos como proposta, o desenvolvimento e análise experimental de versões da heurística GRASP para a solução do PGST. O objetivo maior, é avaliar o impacto da fase de construção, busca local e módulo de reconexão por caminhos no desempenho final do GRASP. Para isso, são propostos formas alternativas de gerar e refinar soluções explorando diferentes filosofias.

Este artigo apresenta na seção 2, uma formulação do PGST como um Problema de Programação Inteira, na seção 3 são apresentadas alternativas para a heurística GRASP. A seção 4 apresenta os resultados computacionais incluindo uma análise probabilística mostrando a robustez das heurísticas propostas em relação a sua convergência para valores alvos pré-definidos. Finalmente em 5 são mostradas as conclusões e em 6, as referências bibliográficas.

2. Formulação Matemática

Embora o PGST seja muito complexo (NP-Difícil), limitando com isso o uso de métodos exatos, devemos reconhecer a importância de analisar formulações eficientes para este problema. Apresentamos nesta seção, uma formulação matemática do PGST descrito como um problema de programação inteira com restrições lineares e não lineares. Considere a seguinte notação:

$N = \{1, 2, ..., n\}$ = conjunto de poços a serem atendidos no atual período de planejamento.

$K = \{1, 2, ..., k\}$ = conjunto de sondas disponíveis no período.

Matriz $T = (t_{ij})_{nxn}$, onde t_{ij} = tempo estimado (em dias) para transportar uma sonda entre os poços i e j de N.

HOR = período em dias associado a um horizonte de planejamento.

$N_k \subseteq N$; subconjunto de poços de N que podem ser atendidos pela sonda $k \in K$. (no nosso caso $N_k = N$, para todo $k \in K$)

te_i = tempo estimado (em dias) para conserto do poço i contados a partir da chegada da sonda em i.

V_i = vazão diária de petróleo do poço i.

PRAZO = tempo limite máximo (em dias) para efetuar o conserto de um poço (contabilizado a partir do seu pedido).

FOLGA(i) = tempo restante (em dias) para efetuar o conserto do poço i, onde:

FOLGA(i) = PRAZO – ATRASO(i), onde ATRASO(i) = número de dias em que o poço i ficou parado no período anterior sem atendimento (caso onde o poço i não pode ser atendido no período anterior).

Variáveis:

i) y_j = número de dias em que o poço j ficou parado no atual período, $j \in N$.

$$y_j \in Z^+ , \ \forall j \in N$$

ii) $$x^k_{ij} = \begin{cases} 1, \text{ se a sonda k atende o poço j logo após o poço i.} \\ 0, \text{ caso contrário.} \end{cases}$$

$x^k_{ij} \in \{0,1\}$, $\forall i, j\ (i \neq j) \in N$ e $\forall k \in K$

Formulação Matemática do PGST:

$$\text{Minimizar} \sum_{i \in N} v_i \, . y_i \qquad (2.1)$$

s.a.:

$$y_j \leq \text{FOLGA}(j) \, , \forall \, j \in N \qquad (2.2)$$

$$\sum_{k \in K} x^k_{ij} \leq 1, \forall \, i, j \in N, \text{ onde } i \neq j \qquad (2.3)$$

$$y_j \geq y_j * x^k_{ij} + t_{ij} * x^k_{ij} + te_j \, , \, i \neq j \, , \forall \, j \in N \qquad (2.4)$$

$$\left.\begin{array}{l} \sum_{i \in N} t_{ij} * x^k_{ij} + te_j \leq \text{HOR} \, , \, i \neq j \, , \forall \, k \in K \\ \text{ou} \\ y_j * x^k_{ij} \leq \text{HOR} \, , \, i \neq j, \forall \, k \in K \end{array}\right\} \qquad (2.5)$$

$$y_j \geq 0 \text{ e inteiro} \, , \, j \in N \qquad (2.6)$$

$$x^k_{ij} \in \{0,1\} \, , \forall \, i, j \, (i \neq j) \in N \text{ e } \forall \, k \in K \qquad (2.7)$$

No modelo (2-1) – (2-7) a função objetivo (2.1) requer a melhor alocação dos poços ao conjunto das sondas de modo que a quantidade total de óleo que deixou de ser coletado seja minimizado. O conjunto de restrições (2.2) exige que cada poço com pedido de manutenção seja atendido dentro dos prazos estipulados, ou seja, em um tempo menor ou igual ao tempo máximo previamente definido. As desigualdades (2.3) obrigam que cada aresta {i, j} é percorrido por no máximo uma sonda dentro do atual horizonte de planejamento, isso implica que cada poço deve ser atendido por no máximo uma sonda. As desigualdades (2.4) indicam que o número de dias onde um poço j fica parado é no mínimo igual a seguinte soma: número de dias em que o poço antecessor imediato na rota ficou parado (y_i) + tempo para transportar sonda entre (i, j) + tempo estimado para o conserto do poço j.

As restrições (2.5) exigem que o número de dias gastos por cada sonda $k \in K$ no período seja menor ou igual ao horizonte de planejamento. Finalmente (2.6) e (2.7) são as variáveis do problema.

Observamos que o modelo (2.1) – (2.7) é não linear devido às restrições (2.4) e (2.5). A linearização destas restrições é possível, mas como contra partida, irá aumentar ainda mais as dimensões do problema. Na seção seguinte, propomos o desenvolvimento de algoritmos heurísticos no sentido de se obter soluções próximas ao ótimo, mas exigindo um tempo computacional viável na solução de instâncias de elevadas dimensões.

3. Algoritmos Propostos

O método *Greedy Randomized Adaptive Search Procedure* - GRASP [1, 4, 7, 8] é um processo iterativo do tipo *multi-start,* no qual cada iteração é composta de uma fase de construção e outra de busca local. A fase de construção tem como objetivo gerar uma solução viável e se caracteriza como um processo iterativo, adaptativo, randômico e guloso. Ele é iterativo, pois uma solução é

construída elemento a elemento; a escolha de cada elemento da solução é feita através de uma função gulosa adaptativa que estima o benefício de cada candidato se inserido na atual solução, estes são colocadas numa lista restrita de candidatos (LRC) composto pelos k melhores candidatos, onde k é um parâmetro de entrada. Seleciona-se então um elemento de LRC aleatoriamente. A função gulosa é também adaptativa pois a LRC é atualizada a cada iteração desta fase de construção, em função das escolhas das iterações anteriores. A solução encontrada pela fase de construção do GRASP não corresponde necessariamente a um ótimo local de um problema de otimização, desta forma é aconselhável aplicar um método de refinamento desta solução através de um algoritmo de busca local. Uma boa referência sobre aplicações de GRASP pode ser encontrada em [5]. Neste artigo, para resolver aproximadamente o PGST, propomos diferentes versões da heurística GRASP. As propostas consistem de dois algoritmos construtivo e três algoritmos de busca local. Cada combinação: *construtivo + busca local* gerou uma versão GRASP totalizando portanto seis versões. Numa segunda etapa propomos um módulo de reconexão por caminhos (*path relinking*) incorporadas a algumas versões GRASP. O objetivo maior do nosso trabalho, é avaliar o impacto de cada módulo no desempenho final do GRASP. Para isso efetuamos simulações em diferentes configurações possíveis analisando os resultados experimentalmente sob diferentes ângulos. Descrevemos a seguir cada algoritmo de construção, busca local e reconexão por caminhos aqui proposto.

3.1. Algoritmo Construtivo 1 (C1)

Esta heurística construtiva é baseada no conceito de *inserção do vizinho mais próximo*. Para isso, inicialmente definimos uma fórmula para determinar o *nível de prioridade* de cada poço ser o próximo a ser inserido numa rota parcial. Considere R_k uma rota em construção, sempre iniciando num poço origem (poço onde a sonda associada S_k está localizada no início do planejamento) e seja p_i, o poço inserido em R mais recentemente. Então para todo poço p_j ainda não alocado, o nível de prioridade de p_j para a rota da sonda S_k será dado por $pri(p_j) = [v_j]/[d_{pj,sonda(Sk)} + te_j]$; onde v_j = vazão diária do poço p_j, $d_{pj,sonda(Sk)}$ = distância do poço p_j até a origem da rota da sonda S_k, e te_j = tempo estimado para o conserto do poço p_j a partir da chegada da sonda. Definido o cálculo de prioridade, os poços são ordenados de forma decrescente, para cada sonda, portanto o número de listas ordenadas será igual ao número de sondas disponíveis. Assim para cada lista (cada sonda), a cada etapa de inserção de um novo poço, criamos uma *Lista Restrita de Candidatos* (LRC), composta dos k primeiros poços da lista ordenada associada, , onde k é definido como alfa por cento do tamanho da lista total de poços ainda não alocados (alfa sendo um parâmetro de entrada). Para possibilitar que diferentes soluções sejam gerados por este procedimento, ao invés de sempre selecionarmos o primeiro poço de LRC, um poço p_f de LRC será escolhido aleatoriamente e inserido na rota da sonda associada S_k. Em seguida, as listas de cada sonda são atualizadas (retirando o poço selecionado), e alternadamente, partimos para selecionar o próximo poço de outra sonda. Observe que esta forma de construção se adapta as características da heurística GRASP, ou seja, um construtivo: iterativo (os poços são inseridos um a um); adaptativo pois no processo de inserção de um novo poço, a LRC é dependente das escolhas anteriores, randômica pois nem sempre o melhor candidato de uma LRC é escolhida e gulosa, pois se tomarmos LRC composto de um elemento, o melhor candidato será selecionado. O procedimento de alocar um poço para cada sonda de forma alternada, espera, obter um balanceamento de carga entre as sondas e com isso, tentar otimizar o objetivo de minimizar a quantidade de óleo que deixou de ser coletado no período de manutenção dos poços.

3.2. Algoritmo Construtivo 2 (C2)

Neste construtivo, colocamos como prioridade o item vazão de cada poço, para isso, é gerada uma única lista com todos os poços que ainda não foram alocados para nenhuma das sondas (Lista de Poços), ordenada de forma decrescente de acordo com a vazão diária desses poços.

Então a cada etapa de inserção de um novo poço, uma Lista Restrita de Candidato (LRC), é formada com os k primeiros poços dessa lista. A seguir é sorteado um dos poços de LRC, mas ao contrário do construtivo 1, esse poço é alocado para a sonda que primeiro pode atendê-lo, pois assim, estaremos minimizando a perda desse poço que é um dos k poços remanescentes de maior vazão.

Esse procedimento é feito até que todos os poços que estão na fila à espera de atendimento sejam alocados ou até que todas as sondas estejam saturadas no atual horizonte de planejamento.

3.3. Algoritmo de Busca Local 1 (BL1)

Essa primeira busca local trabalha com duas estruturas de vizinhanças que são executadas alternadamente em *loop*, até que não haja melhora em nenhuma das duas vizinhanças.

Antes de começar a busca, uma estrutura com os *k*-vizinhos mais próximos de cada poço é montada e passada para os procedimentos de vizinhança (onde *k* é um parâmetro de entrada).

A partir de uma solução inicial, na primeira vizinhança um poço é permutado com os poços que estão dentre seus *p*-vizinhos mais próximos e que se encontram na mesma sonda, mantendo-se ao final de todas as trocas, aquela que resultar em uma menor vazão perdida para a sonda. Caso nenhuma troca proporcione melhora, a composição original da sonda é mantida. Isso se repete para cada poço presente na rota associada a uma sonda. Esse processo é repetido para todas as rotas (sondas).

A Segunda vizinhança é similar à primeira, porém cada poço p_a é permutado com os poços que estão dentre seus *k*-vizinhos mais próximos e se encontram em alguma sonda diferente do poço p_a.

3.4. Algoritmo de Busca Local 2 (BL2)

Considere novamente uma solução inicial composta de q rotas (sondas) e seja um poço p_k escalonado para uma sonda S_r (r = 1, 2, .., q). A busca local funciona da seguinte maneira: Um poço p_k tem sua alocação testada em todas as posições possíveis das outras $(q - 1)$ sondas (sondas diferentes de S_r); a alocação que obtiver a maior economia (maior redução da vazão perdida) em relação à solução corrente será a alocação implementada. Esta análise é feita para cada poço de todas as sondas.

3.5. Algoritmo de Busca Local 3 (BL3)

Essa busca possui duas vizinhanças que são executadas uma após a outra, em *loop*, até que não haja mais melhora na segunda vizinhança. A primeira vizinhança pega cada poço p_j, testa cada posição desta mesma sonda para este poço na ordenação atual e o aloca na melhor posição possível. Depois de executar a primeira vizinhança para todas as sondas, é chamada a segunda vizinhança da busca local, que nada mais é do que a BL2. A razão pela qual a busca local termina considerando somente a não melhora na segunda vizinhança, é porque o resultado da arrumação da primeira vizinhança dará sempre o mesmo resultado caso nenhum poço seja trocado de sonda, e quando a segunda vizinhança não atualizar a solução corrente, nenhum poço será trocado de sonda, pois a alocação da sonda base é mantida.

GRASP	Construtivo	Busca Local
GRASP 1	C1	BL1
GRASP 2	C2	BL1
GRASP 3	C1	BL2
GRASP 4	C2	BL2
GRASP 5	C1	BL3
GRASP 6	C2	BL3

Tabela 3.1: Tabela dos algoritmos GRASP propostos.

3.6. Reconexão por Caminhos (*Path Relinking*) (RC)

O procedimento de reconexão por caminhos (RC) foi proposto originalmente para o método de Busca Tabu (*Tabu Search*) [6]. O procedimento consiste em analisar todas as soluções intermediárias entre duas soluções de boa qualidade procurando com isso encontrar uma terceira que seja melhor que as soluções extremos. Normalmente a RC é usada da seguinte forma: armazena-se sempre na execução de uma heurística, um conjunto elite formado pelas *p* (*p* é um dado de entrada) melhores soluções distintas geradas até o momento. De tempos em tempos, escolhe-se a melhor solução de uma iteração da heurística considerada (no nosso caso, o GRASP) que passa a ser chamada

de *solução base*. Cada solução do conjunto elite é denominada *solução alvo*. A RC, passa então a analisar as soluções intermediárias entre a *solução base* e uma *solução elite* num ou nos dois sentidos. No nosso problema, a adaptação do módulo RC, é feita da seguinte forma: selecionada uma solução base e uma alvo, inicialmente aloca-se, se for o caso, a primeira componente da solução alvo na solução base, para cada sonda, gerando uma solução intermediária. Aplica-se busca local sobre esta solução intermediária. Comparamos esta solução refinada (nova semente) com a pior solução do CE, atualizando este conjunto se for o caso. Sobre a nova semente, alocamos na segunda posição, a segunda componente da solução alvo (novamente para cada sonda) e repetimos a busca local. Este procedimento segue até que a última componente da solução alvo seja alocada para a atual semente. Em seguida fazemos o trajeto contrário, da *solução alvo* para a *solução base*. A RC pode ser feita para todas as soluções do CE ou para uma delas escolhida aleatoriamente. Essa versão, que chamaremos de GRASP 7, é constituída do GRASP 6 mais o módulo de RC (GRASP 6+RC). A RC no nosso caso é executado $2p$ vezes (onde p = cardinalidade do conjunto elite) a cada vez que este módulo é ativado ou seja analisamos as soluções intermediárias nos dois sentidos. A ativação do RC se dá quando a iteração do GRASP é múltipla de 50 ou quando 50% do conjunto elite for alterado desde a última ativação (mas a primeira ativação se dá, obrigatoriamente, após a iteração 50).

4. Resultados computacionais e análise de desempenho

Devido à inexistência (de nosso conhecimento) de instâncias para o PGST em bibliotecas públicas, geramos inicialmente dois conjuntos de problemas testes aleatoriamente, variando parâmetros tais como: o número de vértices (poços); vazão de cada poço; a distancia entre dois poços ou o tempo (dias) gasto para percorrer cada par de poços e o horizonte de planejamento. Neste trabalho para efeito de simplificação tomamos a distancia igual ao tempo e o número de sondas foi fixado em k = 3.

O número de vértices (poços) recebe valores iguais a: 50, 100, 200, 300 e 500 e para cada dimensão foram geradas duas instancias distintas; A e B. Foram, ainda criados dois grupos de instâncias (G1 e G2) contendo, cada um, todas as instâncias citadas. O GRASP 7 só foi testado até o momento com as instâncias 50 A, 100 A e 200 A dos dois grupos de instâncias.

No primeiro grupo (G1), a média das distâncias (tempo de percurso em dias) entre dois poços é dominante (é maior do que a média das vazões dos poços), enquanto na segunda (G2), ocorre o inverso. Em G1 a vazão recebe valores de 1 a 10 e em G2 a vazão recebe valores de duas a três vezes maior do que a média das distâncias entre os poços.

As distâncias entre os poços foram geradas aleatoriamente da seguinte forma: inicialmente os n vértices são gerados aleatoriamente no espaço R^2. De posse das coordenadas de cada vértice foi usada à métrica euclidiana para determinar a distância entre eles. Finalmente o tempo de serviço foi considerado uniforme, com valor constante igual a 1 para todos os poços.

As versões do GRASP descritos na Tabela 3.1 foram implementados em C, compilados com gcc e testados em um Athlon XP 2.4 GHz com 512 Mbytes de RAM. Para cada instância, cada algoritmo GRASP foi executado inicialmente três vezes utilizando os seguintes parâmetros:

- Alfa : 10% (referente ao tamanho da LRC na fase de construção).
- Critério de Parada do GRASP: Número Máximo de Iterações : 200
- K (referente aos k vizinhos mais próximos para a Busca Local 1): 20
- A cada execução de cada GRASP foi usado uma semente aleatória.
- Tamanho do conjunto elite: 2 (para o GRASP 7)

4.1. Bateria de testes G1: (Distância > Vazão)

As tabelas 4.1 e 4.4 apresentam o resultado das simulações para as instâncias onde o fator "*distância*" domina o item "*vazão*". Ou seja, estamos considerando o caso onde o custo de locomover uma sonda entre dois poços adjacentes numa rota é em média maior que o custo de manter um poço desativado para reparos. Neste caso, a tendência das soluções é a de apresentar rotas mais curtas já que

o custo de percurso será mais oneroso e significativo que o custo associado à perda da quantidade de óleo que deixou de ser coletado deste poço. Como metaheurísticas incluindo o GRASP podem gerar soluções distintas a cada execução, mesmo utilizando os mesmos parâmetros de entrada, cada instância foi executada três vezes por cada algoritmo e a melhor solução e o desempenho médio de cada versão são ilustradas nas tabelas 4.1, 4.2 e 4.3.

Instancia	GRASP 1	GRASP 2	GRASP 3	GRASP 4	GRASP 5	GRASP 6
50 A	12.825	13.316	11.722	11.834	11.648	11.702
50 B	12.486	13.170	**12.070**	**12.070**	**12.070**	**12.070**
100 A	82.401	80.943	63.383	65.154	63.263	62.995
100 B	95.945	97.571	75.372	74.923	75.118	**73.449**
200 A	694.655	694.965	447.110	444.107	435.098	432.076
200 B	687.404	626.392	452.694	440.545	435.353	**434.344**
300 A	2.037.415	2.052.170	1.225.786	1.225.709	1.196.646	**1.189.205**
300 B	2.021.849	2.164.601	1.239.339	1.205.379	1.195.500	**1.188.192**
500 A	8.792.950	8.850.227	4.346.378	4.196.810	4.291.978	**4.184.876**
500 B	9.007.707	9.019.847	4.390.105	4.231.153	4.321.717	**4.199.891**

Tabela 4.1: Desempenho das heurísticas GRASP em relação a melhor solução obtida para cada instância em 3 execuções. S*oluções em negrito representam as melhores soluções de cada instância considerando resultados de todas as versões (best).*

Instancia	GRASP 1	GRASP 2	GRASP 3	GRASP 4	GRASP 5	GRASP 6
50 A	13.168,00	13.591,00	11.802,67	11.952,00	11.687,00	11.735,00
50 B	13.278,33	13.814,33	12.123,67	12.121,00	12.075,67	**12.070,00**
100 A	84.023,67	82.887,00	64.088,00	65.506,67	63.904,67	63.197,67
100 B	98.752,67	101.655,33	75.820,67	75.752,67	75.900,33	**73.693,67**
200 A	715.647,33	705.337,67	453.110,33	451.559,00	441.200,33	438.220,33
200 B	690.284,33	687.948,67	457.009,33	**441.980,00**	443.459,33	442.881,67
300 A	2.108.937,33	2.099.468,67	1.235.044,00	1.227.108,00	1.211.822,00	**1.195.462,33**
300 B	2.059.124,67	2.195.714,67	1.251.779,33	1.224.013,67	1.215.724,67	**1.202.638,67**
500 A	9.294.810,00	9.017.724,00	4.396.498,67	4.259.089,33	4.330.350,67	**4.210.615,00**
500 B	9.325.767,33	9.042.123,00	4.416.238,00	4.262.170,00	4.350.284,67	**4.215.083,33**

Tabela 4.2: Desempenho das heurísticas em relação à média das soluções obtidas para cada instância em 3 execuções. *Soluções em negrito representam as melhores médias de cada instância (best-av).*

Instancia	GRASP 1	GRASP 2	GRASP 3	GRASP 4	GRASP 5	GRASP 6
50 A	10,10%	14,32%	0,64%	1,60%	**0%**	0,46%
50 B	3,45%	9,11%	**0%**	**0%**	**0%**	**0%**
100 A	30,81%	28,49%	0,62%	3,43%	0,43%	**0%**
100 B	30,63%	32,84%	2,62%	2,01%	2,27%	**0%**
200 A	60,77%	60,84%	3,48%	2,78%	0,70%	**0%**
200 B	58,26%	44,22%	4,22%	1,43%	0,23%	**0%**
300 A	71,33%	72,57%	3,08%	3,07%	0,63%	**0%**
300 B	70,16%	82,18%	4,30%	1,45%	0,62%	**0%**
500 A	110,11%	111,48%	3,86%	0,29%	2,56%	**0%**
500 B	114,47%	114,76%	4,53%	0,74%	2,90%	**0%**

Tabela 4.3: Desempenho médio de cada heurística: os valores das células representam o desvio em relação à solução *best* de cada instância.

Instancia	GRASP 1	GRASP 2	GRASP 3	GRASP 4	GRASP 5	GRASP 6
50 A	1,90	**1,78**	1,95	1,89	2,65	2,55
50 B	1,79	**1,64**	2,06	1,97	2,72	2,65
100 A	9,24	**8,34**	17,58	16,66	22,83	22,18
100 B	9,02	**8,26**	17,28	16,46	22,24	21,57
200 A	46,01	**42,12**	150,74	141,97	194,82	189,91
200 B	46,78	**42,17**	149,09	142,33	199,13	191,40
300 A	122,97	**115,71**	554,92	526,21	714,60	701,68
300 B	124,40	**113,72**	562,94	521,60	723,18	701,25
500 A	480,34	**427,81**	2.727,74	2.632,65	3.255,42	3.190,18
500 B	478,13	**426,97**	2.719,42	2.580,13	3.241,43	3.211,42

Tabela 4.4: Desempenho das heurísticas GRASP em relação ao tempo médio em segundos exigidos para cada instância em 3 execuções.

Pelos resultados ilustrados nas tabelas 4.1 a 4.4, verificamos que em termos da qualidade da solução gerada, a versão GRASP 6 (usando o algoritmo construtivo 2 e busca local 3) apresentou os melhores resultados seguido das versões 4, 5 e 3 que tiveram resultados similares. As versões 1 e 2 apresentaram desempenho comparativo muito fraco mostrando com isso, que a busca local 1, é menos eficaz que os demais algoritmos de busca propostos. Em relação aos tempos computacionais exigidos, os resultados foram similares 1 e 2; 3 e 4; 5 e 6; não existindo nenhuma discrepância entre cada par. Em termos globais as versões 1 e 2 foram as mais rápidas, seguidas pelas versões 3 e 4, e 5 e 6.

4.2. Bateria de Testes G2 (Distância < Vazão)

Uma segunda bateria de testes foi efetuada, desta vez com instâncias onde o fator "*vazão*" domina o item "*distância*", neste caso a tendência, é obtermos rotas mais longas priorizando o item vazão ao invés da distância percorrida (ou tempo de percurso de uma sonda).

Instancia	GRASP 1	GRASP 2	GRASP 3	GRASP 4	GRASP 5	GRASP 6
50 A	676.943	702.175	598.710	605.400	604.550	**593.432**
50 B	676.980	696.477	597.516	616.689	603.337	**589.398**
100 A	2.104.559	2.266.909	1.749.290	1.736.718	1.745.308	**1.696.510**
100 B	2.076.185	2.306.825	1.711.231	1.734.453	1.740.871	**1.687.309**
200 A	7.605.556	8.289.540	5.645.596	5.496.882	5.375.232	**5.343.001**
200 B	7.841.123	8.382.062	5.457.360	5.505.585	5.402.544	**5.299.003**
300 A	16.170.167	18.198.230	10.550.477	10.127.763	10.284.116	**10.093.202**
300 B	16.483.948	17.489.785	10.452.490	**10.015.272**	10.356.801	10.023.292
500 A	44.934.753	48.689.089	24.932.346	24.209.862	24.636.343	**23.794.935**
500 B	48.435.461	47.612.494	25.044.490	**23.616.120**	24.473.129	23.779.950,00

Tabela 4.5: Desempenho das heurísticas GRASP em relação a melhor solução obtida para cada instância em 3 execuções *(*) soluções em negrito representam as melhores soluções de cada instância (best).*

Instancia	GRASP 1	GRASP 2	GRASP 3	GRASP 4	GRASP 5	GRASP 6
50 A	685.198,67	725.222,00	604.028,67	614.970,00	606.071,00	**600.780,00**
50 B	684.895,00	711.391,00	604.425,67	622.047,67	604.205,33	**593.017,00**
100 A	2.140.430,00	2.340.275,33	1.775.255,00	1.744.508,00	1.752.779,67	**1.729.284,33**
100 B	2.150.962,67	2.397.279,00	1.754.728,33	1.742.998,33	1.750.340,33	**1.708.218,67**
200 A	7.736.824,00	8.475.391,33	5.678.421,33	5.515.350,67	5.466.216,00	**5.382.195,00**
200 B	7.919.001,33	8.548.460,67	5.586.859,67	5.523.141,00	5.426.734,67	**5.346.589,00**
300 A	16.690.280,33	18.311.204,67	10.613.464,67	10.191.555,00	10.334.656,00	**10.128.297,33**
300 B	16.754.481,33	18.122.893,00	10.480.886,00	10.153.309,33	10.435.694,00	**10.106.805,67**
500 A	46.484.961,00	49.233.593,00	25.238.174,00	24.309.184,33	24.745.823,33	**23.939.324,67**
500 B	48.763.842,67	48.976.503,67	25.218.673,67	**23.951.973,67**	24.754.091,33	23.964.174,67

Tabela 4.6: Desempenho das heurísticas em relação à média das soluções obtidas para cada instância em 3 execuções. *(*) soluções em negrito representam as melhores médias de cada instância (best-av).*

Instancia	GRASP 1	GRASP 2	GRASP 3	GRASP 4	GRASP 5	GRASP 6
50 A	14,07%	18,32%	0,89%	2,02%	1,87%	**0%**
50 B	14,86%	18,17%	1,38%	4,63%	2,36%	**0%**
100 A	24,05%	33,62%	3,11%	2,37%	2,88%	**0%**
100 B	23,05%	36,72%	1,42%	2,79%	3,17%	**0%**
200 A	42,35%	55,15%	5,66%	2,88%	0,60%	**0%**
200 B	47,97%	58,18%	2,99%	3,90%	1,95%	**0%**
300 A	60,21%	80,30%	4,53%	0,34%	1,89%	**0%**
300 B	64,59%	74,63%	4,37%	**0%**	3,41%	0,08%
500 A	88,84%	104,62%	4,78%	1,74%	3,54%	**0%**
500 B	105,09%	101,61%	6,05%	**0%**	3,63%	0,69%

Tabela 4.7: Desempenho médio de cada heurística: os valores das células representam o desvio em relação à solução *best* de cada instância.

Instancia	GRASP 1	GRASP 2	GRASP 3	GRASP 4	GRASP 5	GRASP 6
50 A	**1,74**	1,82	2,09	1,96	2,14	2,10
50 B	**1,70**	1,76	2,09	1,91	2,11	2,10
100 A	**7,43**	8,25	17,39	15,38	17,06	17,52
100 B	**7,42**	8,25	17,36	15,16	16,61	17,01
200 A	**40,73**	41,58	147,14	133,19	150,29	160,95

200 B	**41,22**	41,84	148,81	1336,12	150,59	159,92
300 A	117,27	**114,62**	521,73	502,07	611,16	611,33
300 B	118,40	**111,59**	508,11	497,26	607,71	595,47
500 A	479,74	**437,93**	2.733,55	2.623,82	3.217,42	3.245,67
500 B	482,65	**433,19**	2.681,26	2.568,63	3.201,41	3.182,22

Tabela 4.8: Desempenho das heurísticas GRASP em relação ao tempo médio em segundos exigidos para cada instância em 3 execuções.

	HEURÍSTICA	Desvio médio em relação ao valor best
1	**GRASP 6**	0%
2	**GRASP 4**	1%
3	**GRASP 5**	2,5%
4	**GRASP 3**	4,5%
5	**GRASP 1**	93%
6	**GRASP 2**	95,5%

Tabela 4.9: *Ranking* médio das duas baterias de testes dos algoritmos GRASP.

Os resultados das tabelas 4.5 a 4.8, mostram novamente uma superioridade da versão GRASP 6 em relação à qualidade das soluções geradas. Mas neste caso, houve uma competitividade maior entre GRASP 6 e GRASP 4, principalmente para instâncias maiores (300 e 500 vértices). Isso nos mostra, que metaheurísticas são (embora menos que as heurísticas tradicionais) ainda sensíveis aos parâmetros de entrada do problema. Mas de uma forma geral, percebemos que não ocorreu nenhuma anomalia nesta bateria quando comparada com a bateria anterior. Ou seja, a versão 6 foi novamente a melhor e as de pior desempenho foram às versões 1 e 2. Uma ordenação do desempenho médio das seis versões é mostrado na tabela 4.9.

4.3. Resultados comparativos do GRASP com Reconexão por Caminhos

Nesta seção incluímos o módulo de Reconexão por Caminhos (RC) na melhor versão GRASP gerada até o momento (GRASP 6) gerando a versão GRASP 7 = GRASP 6 + RC. O impacto desta proposta pode ser vista nas tabelas 4.10 e 4.11. Verificamos pelos resultados das tabelas 4.10 e 4.11, que a introdução deste módulo traz benefícios na qualidade da solução mesmo considerando a melhor versão dentre as seis analisadas na seção anterior, contudo como contra partida, existe um acréscimo significativo nos tempos computacionais exigidos pela versão GRASP 7.

Instância	**GRASP6**	**GRASP7**	**GRASP6**	**GRASP7**	**GRASP6**	**GRASP7**
	best	***best***	***average***	***average***	***time(sec)***	***time(sec)***
50A	11.702	**11.623**	11.735,00	**11.623**	**2,55**	73,03
100 A	62.995	**62.562**	63.197,67	**62.905,67**	**22,18**	1.410,49
200 A	432.076	**425.386**	438.220,33	**431.451,33**	**189,91**	24.985,70

Tabela 4.10 Resultados do GRASP 6 e 7 com instâncias de G1 onde: *best* = melhor solução; *average* = média das soluções; e time(sec) = tempo médio em segundos.

Instância	**GRASP6**	**GRASP7**	**GRASP6**	**GRASP7**	**GRASP6**	**GRASP7**
	best	***best***	***average***	***average***	***time(sec)***	***time(sec)***
50A	593.432	**589.398**	600.780,00	**593.556,67**	**2,10**	68,94
100 A	1.696.510	**1.688.841**	1.729.284,33	**1.706.073,33**	**17,52**	1.238,54
200 A	5.343.001	**5.251.697**	5.382.195,00	**5.262.270,33**	**160,95**	23.548,99

Tabela 4.11 Resultados do Grasp 6 e Grasp 7 com instâncias de G2 onde: *best* = melhor solução; *average* = média das soluções; e *time(sec*) = tempo médio em segundos

4.4. Análise Probabilística dos algoritmos GRASP

Uma das limitações de heurísticas tradicionais de construção e busca local, bem como de algumas metaheurísticas, é a sua grande sensibilidade com os parâmetros de entrada do problema. Ou seja, muitas vezes uma pequena mudança nos dados de entrada podem provocar mudanças drásticas no comportamento destes algoritmos. Nesta seção mostramos o comportamento médio dos algoritmos

aqui propostos e mostramos como as melhores versões apresentam um nível de robustez promissor. Para avaliar o desempenho, colocamos como valor alvo a ser atingido pelos algoritmos, valores sub ótimos ou próximos do valor de uma solução ótima (no nosso caso devido ao desconhecimento do ótimo, da solução *best*) de cada instância. Para isso, cada versão foi executada independentemente 100 vezes para cada instância utilizando sempre os mesmos parâmetros de entrada. O critério de parada do GRASP foi modificada para quando este atingir uma solução com valor menor ou igual ao valor alvo ou atingir um tempo limite, que foi calculado em aproximadamente cinco vezes o maior tempo de execução considerando todas as heurísticas GRASP para a instância em questão. Para cada execução i, armazena-se o tempo de cpu (t_i) O resultado de cada algoritmo é então plotado associando com o i-ésimo menor tempo de cpu, uma probabilidade $p_i = (i – ½)/100$, gerando pontos no espaço R^2 da forma $z_i = (t_i, p_i)$, para i = 1, 2, ..., 100 [1]. Esta análise foi feita para todas as instâncias aqui analisadas, mas como o comportamento foi muito similar entre estes, somente ilustramos o gráfico das instâncias 50B e 200 B. Como mostra a figuras 4.1, "*quanto mais a esquerda estiver a curva*" melhor será o desempenho do algoritmo. Assim na figura A, o GRASP 6, no tempo t = 1 segundo, possui uma probabilidade de convergência de 100% enquanto o GRASP 3 possui apenas uma probabilidade de convergência de 50% neste tempo. Na Figura 4.1, as versões GRASP 1 e 2 não atingiram o alvo em nenhuma execução no tempo limite estabelecido, daí não estarem presentes nesta figura. Em relação ao GRASP 7 = GRASP 6 + RC, esta versão não foi considerada nesta análise pois embora tenha mostrado melhor desempenho entre todas as versões anteriores, sua inclusão na figura 4.1 só se justificaria caso colocássemos um alvo muito mais difícil, entretanto neste caso, muitas das versões anteriores teriam muita dificuldade ou impossibilidade em atingir alvos desta natureza e estas teriam que ser descartadas deste estudo. Por outro lado, em alvos considerados fáceis, o GRASP 7, não chegaria a ativar o seu módulo de RC, que só é ativado inicialmente após a iteração 50.

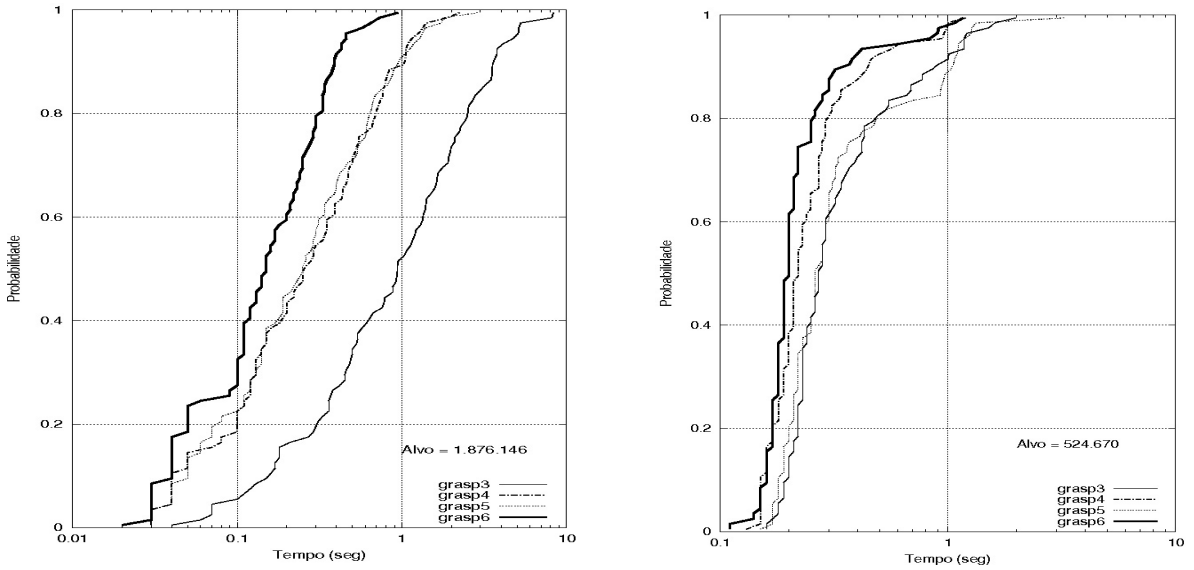

Figura 4.1: Análise probabilística dos algoritmos GRASP(3-6) para atingir valores alvos. Na figura (A) para a instância 50B, com valor alvo = 1.876,146 e na figura (B) para instância 200 A, com valor alvo = 524.670.

5. Conclusões

Este trabalho apresentou sugestões de várias versões para a metaheurística GRASP aplicadas à solução de um problema de gerar percursos otimizados para um problema real de escalonar sondas de manutenção de poços petrolíferos terrestres da região nordeste do Brasil. O objetivo principal, foi a de analisar a importância dos algoritmos de construção, busca local e reconexão por caminhos no desempenho desta metaheurística. Mostramos que o método GRASP é fortemente dependente

principalmente da fase de busca local e que uma boa combinação construção + busca local (GRASP 4, 5 e 6) pode produzir soluções aproximadas altamente competitivas em diferentes problemas de otimização combinatória. Além disso mostramos que a inclusão de um módulo de reconexão por caminhos pode melhorar ainda mais a qualidade das soluções obtidas embora exigindo um tempo computacional maior. A análise probabilística empírica mostrou que determinadas versões sofrem muito pouco com mudanças nos seus parâmetros de entrada, mostrando assim uma robustez confiável em termos de convergência para valores alvos sub ótimos. Como sugestões de trabalhos futuros, incluímos uma versão cooperativa do GRASP ora em desenvolvimento, que acaba com uma das principais limitações do GRASP que são as suas iterações independentes. Outros estudos incluem a possibilidade de elaborarmos outras versões incluindo módulos de *data mining* e novas formas de reconexão por caminhos sem contudo onerar demasiadamente os tempos finais do algoritmo.

Agradecimentos

Ao CNPq e CAPES que financiaram parcialmente este trabalho

6. Bibliografia

[1] R. M. Aiex, M. G. C. Resende and C. C. Ribeiro (2002), Probability distribution of solution time in GRASP: an experimental investigation. Journal of Heuristics, 8, 343-373.

[2] D. Aloise, Daniel Aloise, C. T.M. Rocha, J. C. Ribeiro Filho, L. S.S. Moura, C. C.Ribeiro (2004). , Scheduling Workover Rigs for Onshore Oil Production. (artigo submetido).

[3] D. Aloise, T.F. Noronha, R.S. Maia, V.G. Bittencourt, and D.J. Aloise (2002). Heurística de colônia de formigas com path-relinking para o problema de otimização da alocação de sondas de produção terrestre. Em Anais do XXXIV Simpósio Brasileiro de Pesquisa Operacional (SBPO), Rio de Janeiro.

[4] T. Feo and M. G. C. Resende (1995). Greedy Randomized Adaptive Search Procedure. Journal of Global Optimization, 6: 109-133.

[5] F. Glover, M. Laguna and R. Marti (2000), Fundamentals of scatter search and path relinking. Control and Cybernetics, 39, 653-684.

[6] P. Festa and M. G. C. Resende (2002). GRASP: An annotated bibliography, In C.C. Ribeiro and P. Hansen, editors, Essays and surveys in metaheuristics, pp. 325-367. Kluwer.

[7] G. C. Silva, S. L. Martins and L. S. Ochi (2004). Experimental comparison of Greedy Randomized Adaptive Search Procedures for the Maximum Diversity Problem, In Lecture Notes on Computer Science (LNCS) 3059, pp. 498-512, Springer.

[8] M. J. F. Souza, N. Maculan and L. S. Ochi (2003). A GRASP-Tabu Search algorithm for solving School Timetabling Problems, In Combinatorial Optimization Book Series, Metaheuristics: Computer Decision-Making, vol. 15, chapter 31, 659-672, D. Z. Du and P. M. Pardalos (serie editors), Kluwer.