GRASP com Memória Adaptativa Aplicado ao Problema de Roteamento e *Scheduling* de Sondas de Manutenção

Tiago Araújo Neves[1], **Luiz Satoru Ochi**[1]

[1]Instituto de Computação – Universidade Federal Fluminense (UFF)
Rua Passo da Pátria 156 – Bloco E – 3^o andar – CEP: 24210-240
São Domingos – Niterói - RJ - Brazil

{tneves,satoru}@ic.uff.br

Abstract. *This paper presents some proposals to improve the performance of the GRASP metaheuristic applied to the Workover Rig Routing and Scheduling Problem (WRRSP). This problem consists on create schedules to workover rig set, responsable for doing mantenance on oil wells. This paper focuses on the analysis of GRASP algorithm added with Adaptive Memory Procedures (GRASP+MA). Computational results show that the GRASP+MA outperforms the standard GRASP, indicating that the icnlusion of a learning procdure can significantly improve the performance of this kind of metaheuristic.*

Resumo. *Este artigo apresenta algumas propostas para melhorar o desempenho da metaheurística GRASP applicado para o Problema de Roteamento Scheduling de Sondas de Manutenção (PRSSM). Este problema consiste em gerar roteiros para sondas de manutenção nos poços de petróleo. Este trabalho enfoca análises do algoritmo GRASP adicionando procedimentos de Memória Adaptativa (GRASP+MA). Resultados computacionais mostram que o GRASP+MA supera o desempenho do GRASP tradicional, indicando que o uso de fases de aprendizado pode melhorar significativamente o desempenho deste tipo de metaheurísticas.*

1. Introdução

A área de otimização combinatória tem recebido, nas últimas décadas, contribuições relevantes tanto no aspecto técnico como no algorítmico. Uma das contribuições mais relevantes é a introdução das chamadas Metaheurísticas ou Heurísticas Inteligentes tais como: Algoritmos Genéticos (AGs) e suas variantes, Redes Neuronais (RN), *Simulated Annealing* (SA), *Tabu Search* (TS), *Greedy Randomized Adaptive Search Procedures* (GRASP) entre outros.

De forma resumida, podemos dizer que metaheurísticas são heurísticas genéricas com ferramentas que permitem que algoritmo heurístico associado seja capaz de escapar de ótimos locais ainda distantes de uma solução ótima global em problemas de otimização.

No entando algumas metaheurísticas como o GRASP, embora muito competitiva na solução de diferentes problemas das classes NP-completo e NP-Difícil, não fazem uso de informações das iterações anteriores no processo de busca. Ou seja, embora no módulo de construção, ela seja adaptativa, suas iterações são totalmente independentes [Festa and Resende 2002].

De uma forma geral, as técnicas de memória adaptativa consistem em fazer bom uso de memória para armazenar informações relevantes sobre as regiões já pesquisadas do espaço de busca e utilizar estas informações de maneira estratégica, nas iterações remanescentes do algoritmo.

Existem diferentes propósitos ou metas estabelecidas com a inserção de procedimentos de memória adaptativa em uma heurística iterativa: Pode-se buscar a calibração de parâmetros do algoritmo; evitar movimentos que não se mostraram eficazes ou incentivar movimentos que se mostraram muito eficazes nas iterações anteriores; construir ou atualizar um conjunto de soluções elite; calibrar determinada combinação de módulos (exemplo: algoritmos de construção + busca local) numa dada instância de um problema; criar determinados padrões (por exemplo, segmentos de uma solução, fixação de variáveis de uma solução, etc) com o propósito de filtrar as direções de busca nas iterações remanescentes.

O conceito de memória adaptativa tem relação com algoritmos de aprendizado e, portanto, possui normalmente um caráter dinâmico. Ou seja, as informações contidas na memória devem ser atualizadas sempre que possível. Para isso, busca-se considerar informações relevantes a cada iteração.

Em alguns casos é conveniente considerar somente informações recentes (memória de curto prazo). Em outros, manter informações sobre todo o espaço já verificado é extremamente necessário (memória de longo prazo). Todas estas questões devem ser levadas em conta no momento de atualizar a estrutura de memória utilizada.

As técnicas de memória adaptativa normalmente possuem uma descrição muito sucinta, porém, seu uso ideal na prática não é tão trivial. Questões tais como quais atributos das soluções devem ser armazenados, quais técnicas devem ser utilizadas para combiná-los e por quanto tempo um atributo deve ser levado em consideração, ainda são temas polêmicos e normalmente tratados caso a caso [Glover and Laguna 1997, Taillard et al. 2001, Glover et al. 2000, Prais and Ribeiro 2000, Talbi 2002].

Neste contexto, apresentamos neste artigo, uma proposta para tornar as iterações do GRASP adaptativas, através do uso de alguns tipos de memória [Silva et al. 2007] [Gonçalves et al. 2005]. Este processo é na verdade uma fase de aprendizado do algoritmo para analisar e calibrar determinados parâmetros e módulos, visando obter um processo de busca mais eficiente nas próximas iterações do algoritmo. A proposta é desenvolvida na solução de um Problema de Roteamento e *Scheduling* de Sondas de Manutenção de poços terrestres de petróleo encontradas na região nordeste do Brasil [de Noronha and Aloise 2001].

2. Descrição do Problema Abordado

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

Como dito em [Aloise et al. 2002], em muitos casos, a elevação dos fluídos é feita de forma artificial, com o uso de equipamentos especiais, devido ao fato de que os poços

nem sempre possuem pressão suficiente para que os fluídos atinjam a superfície. Por isso, serviços como os de limpeza e manutenção dos equipamentos de elevação de fluídos são essenciais para se evitar paradas no processo de extração. Esses serviços são realizados por equipamentos denominados sondas, que estão disponíveis em um número limitado e muito pequeno se comparado à quantidade de poços que demandam serviços, devido ao seu alto custo de operação. As sondas ainda encontram-se geograficamente dispersas na bacia petrolífera, o que faz com que cada uma delas parta de um local diferente para atender os poços. Por estes motivos, a manutenção imediata nem sempre é possível, o que provoca a ocorrência de uma fila de poços à espera de atendimento.

Quando é detectada uma falha nos equipamentos de algum poço, para que este volte ao seu funcionamento o mais breve possível, é feito um pedido de conserto. Em seguida é feito um planejamento de atendimento para cada sonda disponível em um dado período de tempo pré-estabelecido e, então, essas sondas são designadas para a manutenção dos poços que aguardam conserto na ordem estabelecida por um escalonamento. Um escalonamento para uma sonda (também chamado de rota), nada mais é do que uma especificação de quais poços esta sonda deve atender e em que ordem eles devem ser atendidos.

Desta forma, podemos colocar como objetivo do PRSSM, encontrar o melhor escalonamento para as sondas disponíveis, minimizando a perda total de produção de petróleo associada aos poços que estão aguardando por atendimento. Entende-se por perda de produção de petróleo de um poço como sendo a vazão deste poço (quantidade de óleo que se pode extrair) multiplicado pelo tempo que a produção deste poço ficou interrompida. Além disso, para gerar uma boa solução do problema, deve-se fazer um escalonamento de sondas para o conjunto de poços considerando informações como: vazão do poço, sua posição em relação aos demais poços e o prazo limite para o seu atendimento.

O PRSSM pode ser descrito na estrutura de um grafo completo não direcionado $G = (P, E)$ onde $P = \{p_1, p_2, ..., p_n\}$ representa os poços (vértices) que estão à espera de manutenção $E = \{(p_i, p_j) : i \neq j / p_i, p_j \in P\}$ representa as arestas ligando os pares de poços de P. Considere ainda o conjunto $S = \{s_1, s_2, ..., s_k\}$ o conjunto de sondas disponíveis para atender aos poços que aguardam a manutenção e o conjunto de suas respectivas origens (pontos de partidas) $O = \{o_1, o_2, ..., o_k\}$ onde k representa o número de sondas disponíveis. Cada poço possui: uma vazão v_i, que representa a quantidade de óleo produzido pelo poço p_i por unidade de tempo, um tempo de serviço ts_i, que representa o tempo estimado para que uma sonda realize a manutenção neste poço e um tempo limite de atendimento l_i, que representa o tempo limite para que o poço p_i seja atendido. Esta última restrição, surge do fato de que cada poço independente da sua vazão, deve ser atendido dentro de um limite de tempo pré-definido pela empresa. Cada poço só pode ser atendido uma única vez e por uma única sonda. O objetivo é minimizar a vazão perdida (que deixa de ser coletada) satisfazendo as restrições do problema.

3. Algoritmos Propostos

Para avaliar a eficiência das técnicas de memória adaptativa foram construídas duas heurísticas: um GRASP tradicional (GRASP Puro ou simplesmente GRASP) e uma versão que faz uso de memória chamado de GRASP+MA (GRASP + Memória Adaptativa). Ambos utilizam o mesmo método construtivo e a mesma busca local.

O construtivo C1 [Trindade 2005], mostrado no Algoritmo 1, como todo construtivo do GRASP, é iterativo, adaptativo, aleatório e semi-guloso. A proposta é baseada no conceito de inserção do vizinho mais próximo. Para isso, a cada iteração, o algoritmo constrói uma Lista Restrita de Candidatos (LCR) que é uma lista com os melhores candidatos da iteração atual. Para construir esta lista, primeiramente deve-se determinar o nível de prioridade de cada poço ainda não inserido na solução. Esta prioridade é calculada da seguinte maneira: suponha uma rota r em construção, uma sonda para atendê-la s_k e o último poço inserido nesta rota p_i. A prioridade de um poço p_j é dada pela equação $prioridade(p_j) = v_j/(t_{ij} + ts_j)$, onde v_j é a vazão do poço p_j, t_{ij} o tempo de viajem do último poço da rota p_i até p_j e ts_j o tempo de serviço de p_j. Caso nenhum poço tenha sido inserido na rota r até o momento, assume-se que t_{ij} é o tempo de viajem entre o ponto de origem da sonda s_k (o_k) e o atual poço p_j.

Tendo associado uma prioridade a todos os poços, estes são ordenados em ordem decrescente de acordo com essa prioridade, e uma lista contendo os melhores candidatos (LRC) é retornada. O tamanho desta lista é definido a partir do parâmetro α (coeficiente de aleatoriedade do GRASP) que assume valores entre 0 (zero) e 1 (um), sendo que, ao assumir o valor 0, esta lista conterá apenas um único candidato, tornando o método guloso e, ao assumir o valor 1, a lista conterá todos os possíveis candidatos, tornando o método totalmente aletório.

Por fim, é escolhido, por sorteio, um dos poços presentes na LRC e este é inserido no final da rota analisada. O algoritmo, a cada passo, insere um poço em uma sonda diferente, construindo assim as rotas de maneira gradativa. O fato de o poço inserido ser escolhido por sorteio possibilita que o algoritmo gere soluções diferentes, o que facilita que o método escape de ótimos locais.

Algoritmo 1 Procedimento C1 ()

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

A busca local utilizada em ambos os GRASP (tradicional e adaptativo), chamada de BL4 é descrita no Algoritmo 2. Esta busca local possui duas estruturas de vizinhança, i) realocação de poços dentro da mesma sonda e ii) realocação de poços para sondas diferentes. A BL4 é uma busca gulosa e exaustiva, logo, os poços sempre são colocados na melhor posição possível a cada iteração.

Primeiramente, a BL4 realoca os poços dentro das próprias sondas, encontrando um bom arranjo dos poços para cada sonda. O procedimento que implementa esta vizinhança é o procedimento *vizinhança1* (linha 4). Feito isso, é realizada a realocação de poços para sondas diferentes. Esta vizinhança é implementada pelo procedimento *vizinhança2* (linha 5). No procedimento *vizinhança2*, para cada poço, testa-se a inserção

Algoritmo 2 BL4 (Solução entrada)

```
1: valor = 0
2: enquanto valor ≠ custo(entrada) faça
3:     valor = custo(entrada)
4:     Vizinhança1(entrada)
5:     Vizinhança2(entrada)
6: fim enquanto
```

deste, em todas as posições de todas as demais sondas sendo a melhor posição escolhida como nova posição do poço. Em ambas as vizinhaças, a solução de entrada só é alterada caso seja encontrado uma solução de menor custo, do contrario, a solução dada como entrada permanece inalterada. O busca local BL4 repete estas duas vizinhanças até que nenhuma delas consiga melhorar a solução incumbente.

O GRASP tradicional é apresentado no Algoritmo 3 e funciona da seguinte maneira: primeiramente uma solução é criada pelo algoritmo construtivo C1. Em seguida é aplicada a busca local BL4 na solução gerada pelo construtivo. O GRASP repete estes dois passos até que um critério de parada estabelecido seja atingido. Ao final de sua execução, o GRASP retorna a melhor solução encontrada.

O GRASP+MA, que é mostrado no Algoritmo 6, faz uso de uma série de técnicas bem conhecidas de memória adaptativa, que serão citadas e explicadas a seguir.

Algoritmo 3 Procedimento GRASP()

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

Auto configuração do parâmetro α (coeficiente de aleatoriedade do algoritmo construtivo do GRASP): um dos problemas fundamentais de sintonia fina do GRASP é a escolha do α [Prais and Ribeiro 2000][Resende and Ribeiro 2002]. É preciso encontrar um valor que produza boas soluções mas que também possibilite que o algoritmo não fique preso em ótimos locais. Para resolver este problema foi proposto o modelo de GRASP reativo, que configura automaticamente o valor deste parâmetro de acordo com a instância trabalhada, através de uma etapa preliminar de aprendizado, onde uma série de tentativas de construção com diferentes valores para α são efetuados. Ao final das tentativas, escolhe-se o valor que produziu os melhores resultados neste treinamento, e, no restante do tempo (ou nas iterações remanescentes), o GRASP executará com este valor agora fixado.

Conjunto Elite e Reconexão por Caminhos (Path Relinking): A Reconexão por Caminhos (RC) é uma técnica proposta inicialmente para o *Tabu Search* e *Scatter Search* por Glover e Laguna [Glover and Laguna 1997][Glover et al. 2000]. Ela utiliza duas so-

luções extremas de boa qualidade, uma denominada base e a outra guia, para tentar encontrar uma solução intermediária que seja melhor que as duas soluções da extremidade. A idéia é fazer com que, passo a passo, a solução base fique mais parecida com a solução guia. O conjunto elite é usado como repositório de soluções base e/ou guia as quais a RC será aplicada.

Neste trabalho a RC é utilizada de maneira hierárquica. Durante os 10% finais do tempo de execução total (critério de parada do GRASP), todas as soluções produzidas pelas iterações GRASP são submetidas à RC com cada um dos membros presentes no conjunto elite sempre tomando como base a solução de melhor qualidade e como guia a solução de pior qualidade. Se durante esse procedimento o conjunto elite for modificado, ao final dele é feito a RC entre os membros do conjunto elite, também no sentido da melhor solução para a pior.

Algoritmo 4 Procedimento executaRC(Solução sol)

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

Algoritmo 5 Procedimento reconexãoCaminhos(Solução base, Solução guia)

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

Diversificação Reativo+Conjunto Elite: também foi feita uma estratégia simples de diversificação que é a variação do parâmetro α de forma associada com a atualização

do conjunto elite durante a execução. Esta variação é feita da seguinte maneira: durante a busca, é feita uma medição de quanto tempo o conjunto elite ficou estático (sem modificação). Toda vez que este tempo atinge um valor t (parâmetro de entrada), o valor de α é aumentado em $0,1$, tornando o método menos guloso. Quando o Conjunto Elite é atualizado, α retorna ao valor que possuia antes da primeira modificação. Esta estratégia tem como objetivo fazer com que o GRASP+MA escape de ótimos locais, uma vez que usando por muito tempo o mesmo valor para o parâmetro α, o algorito construtivo C1 tende a produzir soluções iguais à soluções analisadas anteriormente.

Algoritmo 6 procedimento GRASP+MA(tempo t)

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

4. Resultados Computacionais

Todos os componentes de softwares implementados neste trabalho foram feitos na linguagem C++, compilados com o compilador G++ versão 4.0.2 e executados em um computador com processador Pentium 4 HT 3.2 GHz com 1 GB de memóra RAM rodando um sistema operacinal Linux (Fedora Core) kernel versão 2.6.11.

Devido à inexistência, no nosso conhecimento, de instâncias com características específicas, como, por exemplo, limites de tempo individuais e variação do tempo de atendimento de cada poço, foi construído um gerador de instâncias para o PRSSM abordado neste trabalho. Esse gerador cria instâncias onde cada atributo é definido de maneira aleatória seguindo a distribuição normal de probabilidade. Foram geradas 10 instâncias com 100 poços e 10 com 500 poços. Também foram gerados 2 conjuntos de sondas, um contendo 5 sondas e outro contendo 10 sondas.

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

Como critério de parada, foi utilizado tempo de execução: 1 hora para as instâncias de 100 poços e 4 horas para instâncias de 500 poços. O parâmetro α foi fixado em $0,1$ para todas as instâncias no algoritmo GRASP e o parâmetro t (intervalo de tempo de variação do α) foi fixado em 5 minutos para todas as instâncias. Tais valores foram definidos após testes preliminares efetuados.

	GRASP		GRASP+AM			
Instância	Melhor	Média	Melhor	Média	Dif. Melhor	Dif. Média
100/5-1	135217	135976,2	**134885**	135250,2	0,2	0,5
100/5-2	112599	112953,5	**111949**	111999,1	0,6	0,9
100/5-3	124852	125539,8	**124160**	124847,8	0,6	0,6
100/5-4	123297	124211,6	**122246**	122827,6	0,9	1,1
100/5-5	129779	130248,0	**128874**	129248,7	0,7	0,8
100/5-6	115140	115448,5	**114685**	114794,7	0,4	0,6
100/5-7	133741	134522,2	**132911**	132917,1	0,6	1,2
100/5-8	129387	129786,6	**128612**	128805,3	0,6	0,8
100/5-9	117506	117789,4	**117343**	117603,4	0,1	0,2
100/5-10	123947	124112,0	**123144**	123301,2	0,7	0,7
500/5-1	1862305	1905405,2	**1853045**	1878979,4	0,5	1,4
500/5-2	1856058	1877151,3	**1818763**	1838369,5	2,1	2,1
500/5-3	1891220	1905702,8	**1802280**	1849696,1	4,9	3,0
500/5-4	1939533	1955037,2	**1872395**	1921074,7	3,6	1,8
500/5-5	1894201	1907484,6	**1872539**	1877458,6	1,2	1,6
500/5-6	1799649	1817957,9	**1785956**	1805513,8	0,8	0,7
500/5-7	1898304	1917840,7	**1860601**	1872882,7	2,0	2,4
500/5-8	1850295	1887152,5	**1841788**	1872877,3	0,5	0,8
500/5-9	**1872865**	1908203,8	1883346,4	1900247,6	-0,6	0,4
500/5-10	1925922	1962193,3	**1865705**	1916187,9	3,2	2,4

Tabela 1. GRASP X GRASP+MA para instâncias com 5 sondas

A Tabela 1 mostra um comparativo entre os algoritmos GRASP e GRASP+MA para as instâncias com 100 e 500 poços com 5 sondas. Na coluna Instância mostra-se o nome das instâncias tratadas, a coluna Melhor mostra o melhor valor da função objetivo obtido por cada algoritmo e a coluna Média mostra a média dos valores da função objetivo obtidos nas cinco execuções. Em negrito estão os melhores valores encontrados para cada instância. A coluna Dif. Melhor mostra a diferença percentual dos melhores valores obtidos pelo GRASP e pelo GRASP+MA. A coluna Dif. Média mostra a diferença percentual das médias dos dois algoritmos. Esta diferença precentual é calculada da seguinte maneira: seja vg o valor atingido pelo GRASP e vma o valor atingido pelo GRASP+MA. A diferenạ percentual é dada pela fórmula $(vg - vma) / vma \times 100$. Quando esta diferença aparece com valor negativo, significa que o GRASP obteve melhor resultado, e, quando ela aparece com valor positivo, significa que o GRASP+MA obteve melhor resultado. A Tabela 2 mostra os resultados para as instâncias de 10 sondas.

É importante comentar o fato de que o GRASP+MA conseguiu os melhores resultados para todas as instâncias exceto duas (500/5-9 e 500/10-4) e que ele conseguiu as melhores médias para todas as intâncias, sem exceção. Também é importante ressaltar o fato de que a média atingida pelo GRASP+MA muitas vezes é menor que o melhor valor alcançado pelo GRASP. Este fato confirma a superioridade do GRASP+MA sobre o GRASP visto que o PRSSM é um problema de minimização.

	GRASP		GRASP+AM			
Instância	Melhor	Média	Melhor	Média	Dif. Melhor	Dif. Média
100/10-1	83245	83599,5	**82542**	82596,4	0,9	1,2
100/10-2	71723	72109,1	**71063**	71118,6	0,9	1,4
100/10-3	75995	78579,8	**75297**	75542,8	0,9	4,0
100/10-4	71526	73941,4	**70815**	70922,6	1,0	4,3
100/10-5	78284	79657,8	**77547**	77567,8	1,0	2,7
100/10-6	71130	74341,9	**70330**	70354,8	1,1	5,7
100/10-7	82968	83337,2	**82261**	82303,2	0,9	1,3
100/10-8	77312	79588,2	**76642**	76678,0	0,9	3,8
100/10-9	73188	74976,7	**72488**	72497,6	1,0	3,4
100/10-10	75586	76874,1	**74868**	74874,3	1,0	2,7
500/10-1	966966	973286,6	**943335**	950639,0	2,5	2,4
500/10-2	940852	943352,4	**925540**	934259,3	1,7	1,0
500/10-3	952534	959979,0	**941958**	947480,8	1,1	1,3
500/10-4	**957145**	984200,5	969484	978300,2	-1,3	0,6
500/10-5	991807	996290,2	**970355**	979093,3	2,2	1,8
500/10-6	928871	938368,6	**914980**	923819,4	1,5	1,6
500/10-7	952842	964152,7	**944301**	952262,1	0,9	1,2
500/10-8	953220	960610,8	**945750**	952073,7	0,8	0,9
500/10-9	963855	972176,6	**958511**	964221,6	0,6	0,8
500/10-10	1000136	1005396,1	**979184**	984955,6	2,1	2,1

Tabela 2. GRASP X GRASP+MA para instâncias com 10 sondas

5. Conclusões e Trabalhos Futuros

Foram propostas neste trabalho, duas heurísticas para a resolução do problema proposto, sendo uma na sua versão tradicional (GRASP), e uma que faz uso de memória adaptativa (GRASP+MA).

Para comprovar a eficiência empírica das técnicas de memória adaptativa foram comparados uma heurística na sua versão tradicional, o GRASP, com uma versão que faz uso de memória adaptativa, o GRASP+MA. O GRASP+MA obteve melhores médias para todas as instâncias e melhores resultados para a grande maioria das instâncias.

O GRASP+MA apresenta uma média de tempo para encontrar a melhor solução muito próxima do tempo limite de execução. Isto indica que a técnica de Reconexão por Caminhos é a principal responsável pela qualidade das soluções encontradas por esta metaheurística uma vez que esta técnica começa a ser executada apenas quando 90% do tempo limite já transcorreu. Estudos utilizando a Reconexão por Caminhos com maior frequência durante a busca parecem promissores.

A partir dos resultado empíros obtidos, pode ser comprovado o que a literatura afirma sobre as técnicas de memória adaptativa. Ao usar estas técnicas, a qualidade das soluções obtidas realmente tende a melhorar, porém, o uso destas técnicas com tempo insuficiente pode ocasionar a não convergência dos métodos utilizados.

Como trabalhos futuros podemos sugerir testes com a Reconexão por Caminhos com maior frequência durante a busca no GRASP+MA; propostas de outras técnicas de

memória adaptativa e a utilização das propostas deste trabalho na solução de outros problemas de Otimização Combinatória.

Referências

Aloise, D., Noronha, T., Maia, R. S., Bittencourt, V., and Aloise, D. (2002). Heurística de colônia de formigas com path-relinking para o problema de otimização da alocação de sondas de produção terrestre. In *Anais do XXXIV Simpósio Brasileiro de Pesquisa Operacional (SBPO)*.

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

Festa, P. and Resende, M. (2002). *Essays and Surveys on Metaheuristics*, chapter GRASP: An annotated bibliography, pages 325–367. Kluwer Academic Publishers.

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

Glover, F., Laguna, M., and Martí, R. (2000). Fundamentals of scatter search and path relinking. *Control and Cybernetics*, 39:653–684.

Gonçalves, L. B., Martins, S. L., and Ochi, L. S. (2005). A grasp with adaptive memory for a period vehicle routing problem. In *Proceedings of the International Conference on Computational Intelligence for Modelling Control and Automation*, volume 1, pages 721–727, Vienna, Austria. IEEE.

Prais, M. and Ribeiro, C. C. (2000). Reactive grasp: an application to a matrix decomposition problem in tdma traffic assignment. *INFORMS Journal on Computing*, 12:64–176.

Resende, M. and Ribeiro, C. C. (2002). *Handbook of Metaheuristics*, chapter Greedy randomized adaptive search procedures (GRASP), pages 219–249. Kluwer Academic Publishers.

Silva, G. C., Andrade, M. R. Q., Ochi, L. S., Martins, S. L., and Plastino, A. (2007). New heuristics for the maximum diversity problem. *To appear in Journal of Heuristics*.

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

Talbi, E.-G. (2002). A taxinomy of hybrid metaheuristics. *Journal of Heuristics*, 8:541–564.

Trindade, V. (2005). Desenvolvimento e análise experimental da metaheurística grasp para um problema de planejamento de sondas de manutenção. Master's thesis, Universidade Federal Fluminense - UFF.