Metaheurística GRASP com Memória Adaptativa para a solução do Problema da Árvore Geradora Mínima Generalizado

Cristiane Maria Santos Ferreira[1], **Luiz Satoru Ochi**[1]

[1]Instituto de Computação – Universidade Federal Fluminense (UFF)
Caixa Postal 15.064 – 91.501-970 – Niterói – RJ – Brazil

{cferreira,satoru}@ic.uff.br

Abstract. *The Generalized Minimum Spanning Tree Problem consists of given a graph G whose vertices are divided into clusters, finding a tree spanning all clusters of G, in such a way that minimizes the total cost of the edges. Applications of this problem can be found in telecommunications networks, power distribution networks and in agricultural irrigation systems. This paper presents a GRASP metaheuristic with adaptive memory to solve this problem. Experimental results illustrate the effectiveness of the proposed method.*

Resumo. *O Problema da Árvore de Cobertura Mínima Generalizado (PAGMG) consiste em, dado um grafo G cujos vértices estão divididos em grupos, encontrar uma árvore que cubra todos os grupos de G, de forma que a soma do custo das arestas de T seja mínima. Aplicações deste problema são encontradas em redes de telecomunicações, redes de distribuição de energia elétrica, e em sistemas de irrigação agrícola. Este trabalho apresenta uma metaheuristica GRASP com memória adaptativa para a solução do PAGMG. Resultados experimentais mostram a eficiência do método proposto.*

1. Introdução

Este trabalho aborda uma generalização do clássico Problema da Árvore Geradora Mínima, em que o conjunto de vértices está dividido em grupos e o objetivo é determinar uma árvore de custo mínimo que cubra exatamente um vértice de cada grupo. Este problema é denominado Problema da Árvore Geradora Mínima Generalizado (*Generalized Minimum Spanning Tree Problem* - PAGMG) e foi introduzido por [Myung et al. 1995].

Aplicações para o PAGMG podem ser encontradas nas áreas de *design* de redes de telecomunicações, onde redes locais precisam ser interconectadas para formar redes maiores [Feremans et al. 2000], na localização de instalações e mesmo na descrição de processos físicos que compõem redes [Kansal and Torquato 2001].

O presente trabalho tem como objetivo propor duas versões da heurística GRASP para o PAGMG. Um GRASP tradicional e um GRASP incorporando conceito de memória adaptativa, cuja função é armazenar informações relevantes das iterações já efetuadas pelo algoritmo, com o intuito de tentar melhorar o trabalho de busca nas iterações seguintes. O módulo proposto para minerar informações relevantes é baseado na noção de *conjunto elite* e na técnica de *Reconexão de Caminhos*.

O trabalho está organizado como segue: a seção 2 traz uma descrição do problema e uma revisão dos trabalhos da literatura; na seção 3 são descritas as versões GRASP propostas; na seção 4 encontram-se os resultados computacionais e, por fim, na seção 5 são apresentadas as conclusões e algumas sugestões de trabalhos futuros.

2. O Problema da Árvore Geradora Mínima Generalizado

O PAGMG pode ser definido sobre um grafo não-direcionado $G(V, E)$ onde V representa o conjunto de vértices e E o conjunto de arestas. Cada aresta e possui um custo $c_e \in R^+$. Neste problema, o conjunto de vértices V é particionado em m grupos disjuntos $V_1, ..., V_m$ e o objetivo é determinar uma árvore de custo mínimo que cubra exatamente um vértice de cada grupo.

Diferentemente do Problema da Árvore Geradora Mínima, o PAGMG é classificado como $\mathcal{NP}$-difícil [Myung et al. 1995] e de nosso conhecimento existe apenas um número reduzido de trabalhos relacionados até o presente momento.

A Figura 1 ilustra o exemplo de uma solução para o PAGMG em um grafo cujo conjunto de vértices está particionado em quatro grupos. No exemplo, as arestas (3,5), (3,7) e (7,11) compoem a solução.

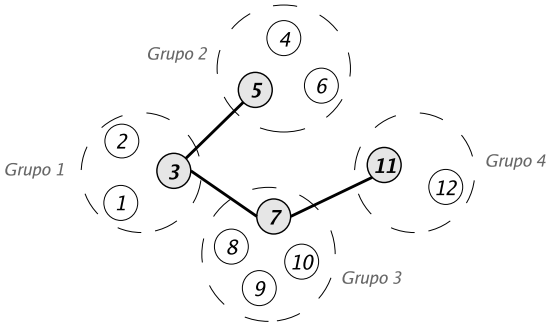


Figura 1. Uma solução para o PAGMG

Aplicações do problema podem ser encontradas, por exemplo, na área de telecomunicações, onde as redes regionais precisam ser interconectadas por uma árvore que contenha uma conexão para cada sub-rede. Para essa interconexão, um vértice de cada sub-rede deve ser selecionado como *gateway* [Myung et al. 1995]. O PAGMG também se aplica a vários problemas relacionados à localização de instalações que precisam estar conectadas por meio de rodovias ou *links* de comunicação. Um exemplo acontece quando uma empresa quer estabelecer centros regionais de distribuição para suas lojas e precisa selecionar um ponto em cada região a fim de construir uma rede de comunicação interconectando esses centros [Shyu et al. 2003]. Outra aplicação é descrita por [Dror et al. 2000] na área de irrigação agrícola.

2.1. Trabalhos Relacionados

O PAGMG foi primeiramente abordado por [Myung et al. 1995], que demonstrou que o problema é fortemente $\mathcal{NP}$-difícil. Myung também propôs quatro formulações de programação inteira mista e um algoritmo *branch-and-bound* que resolveu para a otimalidade, instâncias com até 100 vértices.

Dentre os principais trabalhos, pode-se citar [Feremans et al. 2000], que na sua tese de doutorado fez uma investigação da estrutura poliédrica do PAGMG. A tese também propôs um algoritmo *branch-and-cut* e quatro famílias de desigualdades válidas: desigualdades de Hammock, de ciclos ímpares, de casamento de ciclos ímpares e de "buracos" ímpares. O *branch-and-cut* conseguiu resolver todas as instâncias euclideanas com até 160 vértices e todas as instâncias com custos aleatórios com até 200 vértices. Também resolveu 150 das 169 instâncias adaptadas do TSPLIB, considerando um tempo limite de duas horas. Comparando com o algoritmo de [Myung et al. 1995], o algoritmo de Feremans apresentou limites melhores e conseguiu resolver instâncias bem maiores.

[Golden et al. 2005] apresentam uma comparação minuciosa de alguns algoritmos construtivos e duas heurísticas. Os construtivos comparados são adaptações dos clássicos algoritmos de Kruskal, Prim e Sollin para o Problema da Árvore Geradora Mínima. As adaptações de Kruskal e Sollin apresentaram resultados melhores que Prim, e dentre os dois, há uma ligeira vantagem para o algoritmo baseado em Kruskal. As heurísticas propostas consistem de: uma busca local bastante simples, porém eficiente, e um algoritmo genético.

Os autores compararam os algoritmos nas instâncias adaptadas do TSPLIB, das quais 150 possuem o valor ótimo conhecido. As soluções do procedimento de busca local e do algoritmo genético ficaram, em média, a $0,07\%$ e $0,01\%$ da otimidalidade, respectivamente. O algoritmo genético também apresentou um desempenho ligeiramente melhor nos testes com instâncias cujo ótimo não é conhecido e com instâncias aleatórias propostas pelos autores. Vale colocar que tal algoritmo consome aproximadamente o dobro do tempo da busca local.

3. GRASP

GRASP (*Greed Randomized Adaptive Search Procedure*) é uma meta-heurística *multi-start*, proposta por [Feo et al. 1994], em que cada iteração é composta de uma fase de construção e de uma fase aprimoramento (busca local). Durante o processo, a solução incubente é armazenada e atualizada sempre que a fase de aprimoramento resulta em uma solução melhor. Ao final de um número estabelecido de iterações, o algoritmo retorna a melhor solução encontrada.

A fase de construção gera uma solução viável para o problema através de um procedimento parcialmente guloso e parcialmente aleatório. A cada estapa da construção, a seleção dos elementos que vão compor a solução é feita aleatoriamente em um subconjunto do melhores elementos candidatos, chamado Lista de Candidatos Restrita. A fase de aprimoramento é uma busca local. O Algoritmo 1 apresenta o pseudo-código de uma versão tradicional do GRASP para um problema de minimização. A função `custo(`s`)` retorna o custo da solução s e s^* é a solução incubente.

No GRASP, a fase de construção é iterativa, adaptativa, semi-gulosa e randômica. Contudo as iterações GRASP são totalmente independentes, não havendo nenhum tipo de memória nem aprendizagem no decorrer da execução. Daí muitas vezes ele ser chamado de *multistart*. Conseqüentemente, nenhuma informação relevante obtida nas iterações anteriores são incorporadas no processo de busca das iterações remanescentes.

Muitos pesquisadores vêm então utilizando técnicas adicionais, no intuito de aproveitar de alguma forma resultados obtidos para tentar encontrar melhores soluções, como

Algoritmo 1 GRASP

```
s* ← ∅;
para todo i ∈ {1, ..., max_iter} faça
    s_0 ← constroi_solucao(α);
    s ← busca_local(s_0);
    se custo(s) ≤ custo(s*) então
        s* ← s;
    fim se
fim para
retornar s*;
```

acontece com o GRASP reativo [Prais and Ribeiro 2000] e com o uso de reconexão de caminhos [Resende and Ribeiro 2003], [Silva et al.], [Goncalves et al. 2005]por exemplo.

Neste trabalho, são propostos duas versões GRASP para o PAGMG: um GRASP puro (GRASP), que segue o modelo tradicional explicado anteriomente; e um GRASP com memória adaptativa, que é a versão incluindo o módulo de reconexão de caminhos (RC) (GRASP+MA).

Nesta seção serão apresentados os algoritmos que compoem as duas versões GRASP propostas: a heurística de construção, o procedimento de busca local e o mecanismo de reconexão de caminhos.

3.1. Heurística de Construção

Foram implementadas diferentes heurísticas construtivas, dentre elas adaptações dos algoritmos de Kruskal, Prim e uma heuristica aqui proposta, baseada no cálculo das distâncias médias (HDM) para o PAGMG. Resultados preliminares demonstraram que a HDM apresenta o melhor desempenho médio. Essa heurística HDM será explicada a seguir.

O objetivo da HDM é selecionar um vértice $\gamma[i]$ em cada grupo V_i levando em consideração os grupos a que V_i poderá estar conectado na solução. A seleção de $\gamma[i]$ é feita com base em sua distância média a um conjunto $S_i \subseteq \{V \setminus V_i\}$, dado que S_i é composto por vértices que se encontram a uma distância máxima D_{max} do grupo V_i.

Ao início da construção, define-se aleatoriamente uma ordem para percorrer os grupos. Em seguida, a cada grupo V_i percorrido, constrói-se o conjunto S_i. A seguir, calcula-se a distância de cada vértice $v \in V_i$ aos elementos de S_i. A distância $d_{\{v,S_i\}}$ de um vértice v a S_i é o custo médio das arestas entre v e os vértices presentes em S_i, ou seja, $d_{\{v,S_i\}} = \frac{\sum_{u \in S_i} c_{(u,v)}}{|S_i|}$.

A seleção do vértice $\gamma[i]$ é feita aleatoriamente dentre os elementos de uma *Lista de candidatos Restritos* (LCR), formada pelo subconjunto dos melhores candidatos como feito tradicionalmente na fase de construção do GRASP. Neste caso, a LCR é composta de cada vértice v cuja distância $d_{\{v,S_i\}}$ é menor ou igual a $d_{min} + \alpha(d_{max} - d_{min})$, onde d_{min} e d_{max} são a menor e a maior distância de todos os vértices do grupo V_i, respectivamente. O parâmetro $\alpha \in [0, 1]$ indica o quão gulosa será essa seleção. Para $\alpha = 1$, por exemplo, o comportamento do algoritmo é totalmente aleatório [Resende and Ribeiro 2003]. Uma

observação importante é que na construção de S_i, nos grupos que já possuem vértices fixados na solução, apenas estes integrarão S_i.

3.2. Busca Local

A busca local implementada foi utilizada no PAGMG primeiramente por Golden et. al. [Golden et al. 2005]. A fim de facilitar o cálculo da vizinhança, considera-se cada solução como um conjunto de m vértices, um de cada grupo. A avaliação das soluções é feita calculando-se a árvore de custo mínimo que cubra os vértices presentes na solução. Dada uma solução inicial s, o procedimento pode ser descrito basicamente nas três etapas abaixo:

1. Aleatoriamente define-se uma ordem em que os grupos serão visitados
2. A cada visita a um grupo V_i, calcula-se todos os vizinhos de s em relação a V_i. A solução corrente passa a ser o melhor vizinho s', caso este seja melhor que s.
3. Repete-se o passo 2 até que m grupos sejam visitados sem que haja melhoras na solução corrente.

Considerando que s é composta pelos vértices $\{\gamma_1, ..., \gamma_m\}$, sua vizinhança em relação ao grupo V_i são todas as soluções $s' = \{\gamma'_1, ..., \gamma'_m\}$ em que $\gamma'_i \neq \gamma_i$ e $\gamma'_j = \gamma_j, \forall j \in \{1, .., m\} \setminus i$. Ou seja, em relação ao grupo V_i, s possui $|V_i| - 1$ vizinhos.

3.3. Reconexão de Caminhos

O mecanismo de Reconexão de Caminhos (*Path Relinking*), proposto originalmente para a busca tabu e *scatter search* por Glover [Glover et al. 2000], tem como objetivo encontrar soluções intermediárias entre duas boas soluções extremas. O algoritmo parte de uma *solução base* s_0, e passo-a-passo a transforma em s_d, chamada *solução guia*. Nesse trajeto, entende-se que pode ser encontrada uma solução melhor que s_0 e s_d.

No procedimento implementado neste trabalho, as soluções são representadas por conjuntos de m vértices e a sua avaliação é feita pelo cálculo da árvore geradora mínima - a mesma representação usada na busca local.

Se s_0 e s_d são duas soluções com d vértices diferentes entre si, um movimento de s_0 para s_d é a substituição de um vértice em s_0 por um vértice de s_d. A reconexão de caminhos prodece da seguinte forma: partindo de s_0, um movimento para s_d é gerado e tem-se uma solução intermediária s_1; a seguir mais um movimento é gerado, de s_1 a s_d; e assim por diante até que se chegue a uma solução s_{d-1}, após $d-1$ movimentos realizados.

A Figura 2 ilustra um exemplo do funcionamento do mecanismo, nesse caso s_d é o grafo ilustrado na Figura 1. Como $s_0 = \{3, 6, 8, 12\}$ e a solução guia $s_d = \{3, 5, 7, 11\}$ há três diferenças entre as duas soluções, $d = 3$. Há, portanto, três candidatos para gerar s_1: $\{3, 6, 8, 11\}$, $\{3, 5, 8, 12\}$ e $\{3, 6, 7, 12\}$. A melhor opção é a troca do vértice 6 por 5. Na próxima iteração, os candidatos são $s_1^1 = \{3, 5, 8, 11\}$ e $s_1^2 = \{3, 5, 7, 12\}$. A primeira opção é selecionada para ser s_2. Para o movimento seguinte, a única possibilidade é a troca do vértice 8 por 7. Com essa mudança, tem-se a solução guia s_3 e o procedimento termina.

Neste trabalho, o GRASP com reconexão de caminhos mantém um conjunto elite CE contendo as t melhores distintas soluções encontradas durante a execução até o momento. Esse conjunto é atualizado a cada iteração, caso a solução gerada pela busca

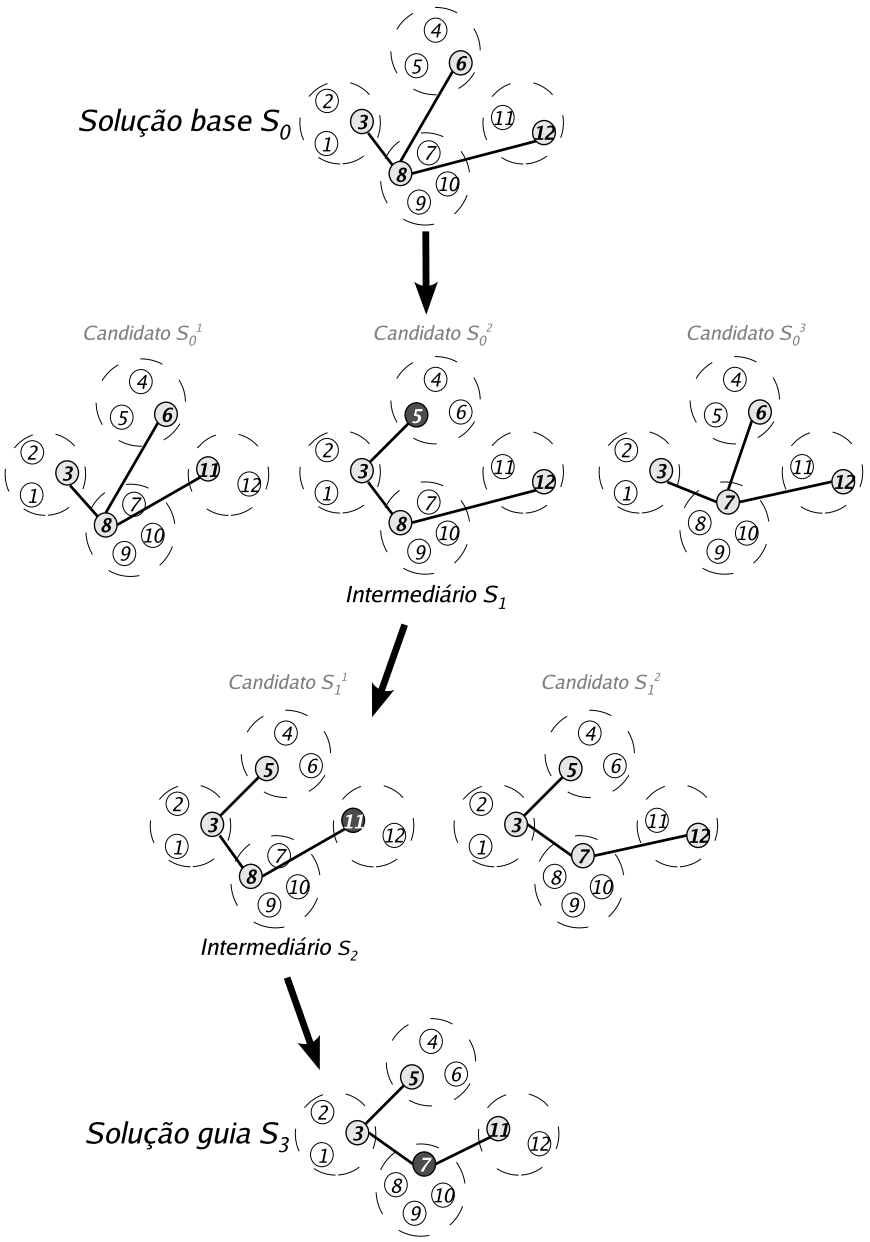


Figura 2. Ilustrando o mecanismo de reconexão de caminhos.

local seja melhor que a pior solução em CE. Sempre que o algoritmo atinge `max_iter` iterações sem atualização do conjunto elite, são removidas todas as suas soluções (com exceção da melhor).

A reconexão de caminhos é ativada a partir da iteração μ (dado de entrada), e ocorre sempre que a solução da busca local é no máximo $p\%$ pior que a melhor solução do conjunto elite. Aplica-se o procedimento entre s e a solução de CE que maximize d (em outras palavras, fazer a reconexão com a solução elite *mais diferente de* s). A solução base por definição será sempre a melhor dentre as duas soluções extremas. Como estratégia de intensificação, o valor de p é proporcional ao número de iterações sem atualização de CE, ou seja, quanto mais difícil for encontrar uma solução com a qualidade das soluções elite, mais próvavel ocorrer reconexão de caminhos.

4. Resultados Computacionais

Nesta seção, estão apresentados os resultados comparativos obtidos por meio de testes empíricos realizados com o procedimento de busca local de [Golden et al. 2005] e as versões GRASP aqui propostas. Os testes computacionais foram realizados em um computador Pentium IV, com 3.2GHz e 1GB de memória principal, rodando um sistema operacional Linux versão 2.6.8-1.521. Os programas foram implementados na linguagem de programação C++ e compilados com g++ versão 4.0.2 utilizando as opções de compilação -o3 e march=pentium4.

Os testes foram realizados sobre 169 instâncias com $48 \leq |V| \leq 226$, apresentadas em [Feremans et al. 2000]. Essas instâncias foram geradas a partir de instâncias do repositório do Problema do Caixeiro Viajante [Reinelt 1991]. O agrupamento dos vértices foi realizado por [Fischetti et al. 1995] para o Problema do Caixeiro Viajante Generalizado, através de duas técnicas: *Center Clustering* e *Grid Clusterization*. [Feremans et al. 2000] apresentou ainda cinco instâncias geográficas.

Dentre as instâncias utilizadas, 150 possuem valor ótimo conhecido. Para essas instâncias, cada algoritmo foi executado 10 vezes para cada instância. O critério de parada estipulado foi encontrar a solução ótima ou um tempo limite de 300s de execução. Todos os algoritmos conseguiram encontrar as soluções ótimas. O tempo médio necessário para a busca local de Golden foi de 1,06s, para o GRASP tradicional foi 2,81 e para o GRASP+MA foi de 0,28s.

Nos testes com instâncias cujo valor ótimo não é conhecido, o critério de parada foi atingir um custo igual ou menor ao da melhor solução conhecida, ou um tempo de execução de 300s. Os resultados estão descritos na Tabela 1. A primeira coluna apresenta o nome da instância; nas três colunas seguintes tem-se o número de vértices, grupos e arestas, respectivamente; a coluna cinco traz o tempo da busca local de Golden; as duas últimas colunas apresentam os tempo médio do GRASP tradicional e do GRASP com Memória Adaptativa (GRASP+MA), respectivamente.

Nas instâncias consideradas fáceis, em que os algoritmos conseguem encontrar o alvo em menos de 1s, percebe-se que não há grandes diferenças entre os três algoritmos. Isso acontece porque os algoritmos encontram as soluções alvo antes mesmo que o mecanismo de reconexão de caminhos seja ativado no GRASP+MA. As diferenças aparecem nas instâncias mais difíceis. Somente o GRASP+MA consegue encontrar todos os alvos no tempo estipulado.

Adicionalmente foi realizada uma bateria de testes para a análise da função de distribuição cumulativa da probabilidade dos algoritmos atingirem uma determinada solução alvo em relação ao tempo. O objetivo é verificar o tempo necessário para que os algoritmos encontrem soluções de boa qualidade.

Em cada experimento, foram computados os tempos de 100 execuções independentes para cada algoritmo. Nos gráficos da Figura 3, estão apresentados os resultados obtidos com a instância 40krob200. Em cada curva, o i-ésimo menor tempo de execução rt_i foi associado à probabilidade $p_i = (i - 0,5)/100$, gerando pontos (rt_i, p_i) para $i = 1, ..., 100$.

A análise dos gráficos pode ser feita considerando o alinhamento das curvas: a

Instância	$\|V\|$	$\|E\|$	m		Busca Local		GRASP		GRASP+MA	
				Alvo	Custo	T (s)	Custo	T (s)	Custo	T (s)
40d198	198	18841	40	7044	7044	1,27	7044	0,77	7044	**0,26**
41gr202	202	19532	41	242	242	7,53	242	10,08	242	**0,45**
45ts225	225	24650	45	62269	62268	39,92	62268,2	8,97	62268,8	**1,21**
46pr226	226	24626	46	55515	55515	0,014	55515	**0,01**	55515	0,013
67d198	198	19101	67	8283	8283	39,48	8284,6	279,93	8283	**6,75**
68gr202	202	19826	68	293	293	1,19	293	**0,51**	293	0,67
75ts225	225	24900	75	79019	79019	88,89	79019	24,78	79019	**0,32**
84pr226	226	25118	84	62527	62527	**0,08**	62527	0,20	62527	0,15
40d198	198	18772	40	7098	7098	**0,12**	7098	1,65	7098	0,29
41gr202	202	19303	41	232	232	19,82	232,2	140,3	232	**2,07**
45ts225	225	24726	45	60588	60641,6	300	60636,6	271,2	60588	**61,02**
50pr226	226	24711	50	56721	56721	0,033	56721	**0,01**	56721	**0,01**
32d198	198	18372	32	6501	6501	**0,028**	6501	0,04	6501	0,04
31gr202	202	18872	31	203	203	0,33	203	0,23	203	**0,17**
35ts225	225	24544	35	50813	50813	1,52	50813	2,00	50813	**0,56**
33pr226	226	24355	33	48249	48249	**0,01**	48249	**0,01**	48249	0,03
25d198	198	18149	25	6185	6185	0,05	48249	**0,01**	6185	0,09
21gr202	202	17904	21	177	177	**0,22**	177	0,50	177	0,52
25ts225	225	24300	25	40339	40339	0,28	40339	**0,18**	40339	0,23

Tabela 1. Resultados comparativos para instâncias cujo valor ótimo não é conhecido.

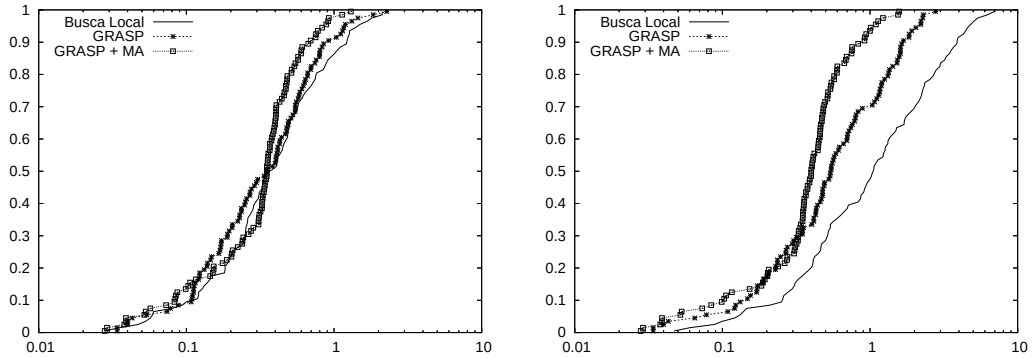
curva mais à esquerda indica o algoritmo que converge mais rápido para o valor alvo. Pode-se perceber que a busca local de [Golden et al. 2005] gasta um tempo maior para encontrar as soluções alvo, seguida da versão do GRASP tradicional. As curvas começam a divergir em torno de 0,4s, tempo médio em que a reconexão de caminhos é ativada. Na média, verificando os gráficos das figuras 3 e 4, percebe-se que o melhor desempenho (curva mais a esquerda) foi do GRASP+MA.

A influência do mecanismo de memória adaptativa através da reconexão de caminhos se torna mais evidente nos gráficos da Figura 4, que apresenta os resultados para a instância 45ts225. Nesse caso, os algoritmos têm mais dificuldade para encontrar boas soluções. Na Figura 4(b), foram desconsideradas as execuções em que os algoritmos não conseguiram encontrar os alvos.

Mesmo considerando um alvo fácil, há uma diferença considerável nos tempos gastos pelos algoritmos. A versão do GRASP+MA converge bem mais rápido, há 100% de probabilidade do algoritmo encontrar a solução alvo em menos de 27s, enquanto para os outros dois algoritmos essa probabilidade não passa de 17%. Para um alvo difícil (figura 4b), os algoritmos sem reconexão de caminhos não conseguem encontrar os alvos no tempo estipulado.

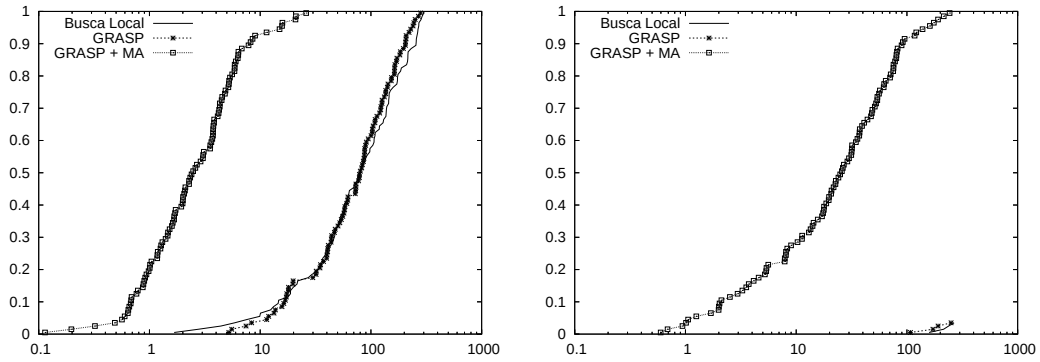
5. Conclusões e Trabalhos Futuros

Este trabalho abordou a utilização da heurística GRASP na resolução do PAGMG. Diante do fato de que o funcionamento do GRASP tradicional não se baseia em aprendizagem sobre a sua execução, o objetivo foi analisar a influência da utilização de memória para



(a) Alvo fácil (b) Alvo difícil

Figura 3. Análise empírico-Probabilística dos algoritmos com a instância 40krob200



(a) Alvo fácil (b) Alvo difícil

Figura 4. Análise empírico-Probabilística dos algoritmos com a instância 45ts225

armazenar informações das melhores soluções obtidas até o momento (conjunto elite) sobre o desempenho do GRASP. Neste contexto, foram propostas duas versões da heurística: uma tradicional, composta por um algoritmo construtivo proposto neste trabalho e um algoritmo de busca local da literatura (GRASP); e uma versão que difere da primeira por utilizar o mecanismo de reconexão de caminhos (GRASP+MA). Esse mecanismo é aplicado sobre um conjunto das melhores soluções encontradas no decorrer do algoritmo.

Os resultados comparativos com o algoritmo da literatura indicam que a aplicação de memória adaptativa, com a reconexão de caminhos, faz com que o GRASP convirga mais rápido para boas soluções, especialmente no caso de soluções alvo difíceis.

Como trabalhos futuros, propõe-se a implementação de um GRASP com memória adaptativa, que inicialmente treine várias heurísticas construtivas e buscas locais e após o treinamento utilize somente as combinações que geraram soluções melhores nas iterações remanescentes. Dessa forma, procura-se encontrar para cada instância, a melhor combinação de métodos construtivo + busca local.

Propõe-se também a análise da utilização de memória em outros elementos do GRASP, como o parâmetro α, e parâmetros específicos dos algoritmos construtivos.

Referências

Dror, M., Haouari, M., and Chaouachi, J. S. (2000). Generalized spanning trees. *European Journal of Operational Research*, 120:583–592.

Feo, T., Resende, M., and Smith, S. (1994). A greedy randomized adaptive search procedure for maximum independent set. *Operations Research*, 42:860–878.

Feremans, C., Labbé, M., and Laporte, G. (2000). The generalized minimum spanning tree problem: Polyhedral analysis ans branch-and-cut algorithm. Technical report, Université Libre de Bruxelles, Bruxelas, Bélgica. Available at `http://smg.ulb.ac.be`.

Fischetti, M., Salazar, J. J., and Toth, P. (1995). The symmetric generalized traveling salesman polytope. *Networks*, 26:113–123.

Glover, F., Laguna, M., and Martí, R. (2000). Fundamentals of scatter-search and path relinking. *Control ans Cybernetics*, 36:653–684.

Golden, B., Raghavan, S., and Stanojević, D. (2005). Heuristic search for the generalized minimum spanning tree. *INFORMS Journal on Computing*, 17:290–304.

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

Kansal, A. R. and Torquato, S. (2001). Globally and locally minimal weigth spanning tree networks. *Physica A*, 301:601–619.

Myung, Y. S., Lee, C. H., and Tcha, D. W. (1995). On the generalized minimum spanning tree problem. *Networks*, 26:231–241.

Prais, M. and Ribeiro, C. (2000). Reactive GRASP: An application to a matrix decomposition problem in TDMA traffic assignment. *INFORMS Journal on Computing*, 12:164–176.

Reinelt, G. (1991). TSPLIB - a traveling salesman problem library. *ORSA Journal on Computing*, 3:376–384.

Resende, M. and Ribeiro, C. (2003). GRASP and path-relinking: Recent advances and applications. In Ibaraki, T. and Yoshitomi, Y., editors, *Proceedings of the Fifth Metaheuristics International Conference (MIC2003)*, pages 1–6.

Shyu, S. J., Yin, P. Y., Lin, B. M. T., and Haouari, M. (2003). Ant-tree: An ant colony optimization approach to the generalized minimum spanning tree problem. *Journal of Experimental and Theoretical Artificial Inteligence*, 15:103–112.

Silva, G., Andrade, M. R. Q., Ochi, L. S., Martins, S. L., and Plastino, A. New heuristics for the maximum diversity problem. STATUS: To appear in Journal of Heuristics - SPRINGER.