Aplicação de uma meta-heurística não específica em um problema de árvores em grafos

André Renato Villela da Silva

Departamento de Computação
Instituto de Ciência e Tecnologia - PURO
Universidade Federal Fluminense
R. Recife s/n, Jardim Bela Vista - Rio das Ostras/RJ CEP 28890-000
avillela@ic.uff.br

Resumo

Este trabalho trata de uma generalização do clássico problema da Árvore Geradora Mínima, chamado de Problema da Árvore de Custo Mínimo de k arestas. Neste problema, uma árvore com exatamente k arestas e custo mínimo deve ser encontrada dentre todas as possíveis árvores de um grafo G. Já existem na literatura algumas meta-heurísticas eficazes para este problema. O objetivo deste trabalho é comparar uma meta-heurística proposta para outro tipo de problema com as meta-heurísticas da literatura. Os resultados obtidos são bastante promissores.

Palavras-chave: Otimização em Grafos, Problemas de Árvores Geradoras, Otimização Combinatória, Meta-Heurísticas, Algoritmos Evolutivos

Área principal: Meta-heurísticas

Abstract

This works deals with a generalization of the classical Minimum Spanning Tree problem, called k-Cardinality Tree Problem (KCTP). In KCTP a tree with exactly k edges and minimal cost should be found among all possible trees of a graph G. There already are some effective metaheuristics for this problem in the literature. The objective of this work is to compare a metaheuristic proposed for another kind of problem with those metaheuristics from literature. Obtained results are very promising.

Keywords: Graph Optimization, Spanning Tree Problems, Combinatorial Optimization, Metaheuristics, Evolutionary Algorithms

Main area: Metaheuristics

1 Introdução

Este trabalho aborda o problema originalmente conhecido como K-Cardinality Tree Problem (KCTP), também chamado de Problema da Árvore de Custo Mínimo de k Arestas ou Árvore Mínima de Cardinalidade k. O objetivo do problema é encontrar uma árvore de tamanho k, formada a partir de arestas de um grafo $G = (V, A)$, cujo somatório do peso das arestas escolhidas seja mínimo. Esta versão do problema é chamada de ponderada nas arestas. Caso os pesos estejam associados aos vértices, diz-se que a versão do problema é ponderada em vértices.

[Hamacher et al., 1991] definiram formalmente o KCTP assim: seja um grafo $G = (V, A)$, não-direcionado e ponderado, com uma função de atribuição de pesos $w : A \rightarrow \mathbb{R}$. Seja $\mathscr{T}$ o conjunto de todas as árvores que podem ser formadas de G, com exatamente k arestas ($k \in \mathbb{N}|0 < K < |V|$), deseja-se encontrar a árvore $T_k \in \mathscr{T}$ que minimize a função f(.) a seguir.

$$f(T_k) = \sum w(a)_{\forall a \in A(T_k)} \tag{1}$$

onde $A(T_k)$ representa o conjunto de arestas que formam a árvore T_k. A Figura 1 mostra o exemplo de uma árvore de 4 arestas, cujo somatório do custo das aresta é mínimo para o grafo em questão.

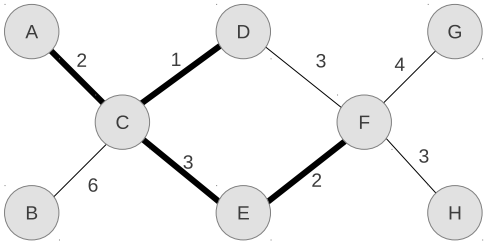

Figura 1: Exemplo de árvore geradora mínima com 4 arestas para este grafo. Em destaque as arestas escolhidas.

Muitos trabalhos comprovam que o KCTP é NP-árduo [Fischetti et al., 1994] mesmo quando o grafo G é planar [Marathe et al., 1996], ou $G = K_n$ (grafo completo de ordem n, ou mesmo quando $w(a) \in 1, 2, 3 \forall a \in A$. No entanto, o CKTP pode ser resolvido em tempo polinomial quando G for uma árvore [Zelikovsky e Lozevanu, 1993] ou quando o valor de k for muito pequeno ($k = 2$, por exemplo).

O KCTP foi originalmente proposto por [Hamacher et al., 1991], encontrando aplicações em nas seguintes áreas:

- exploração de petróleo [Hamacher et al., 1993]
- roteamento e telecomunicações [Garg e Hochbaum, 1997, Cheung e Kumar, 1994]
- mineração [Philpott e Wormald, 1997]
- decomposição de matrizes [R. Borndorfer, 1997, Borndorfer et al., 1998]
- *layout* de facilidades [Foulds e Hamacher, 1992, Foulds et al., 1998]

O restante deste trabalho está organizado da seguinte forma. A seção 2 traz uma breve revisão da literatura incluindo as diversas abordagens aplicadas ao KCTP. A seção 3 apresenta os algoritmos heurísticos para a solução do KCTP. Os resultados computacionais são discutidos na seção 4. As conclusões deste trabalho encontram-se na seção 5. Na seção 6 são apresentadas propostas de trabalhos futuros.

2 Revisão da literatura

Existem diversos trabalhos na literatura sobre o KCTP e suas principais variantes. Um dos primeiros trabalhos é o de [Fischetti et al., 1994]. Nele é apresentada uma formulação matemática para o KCTP através de um Programa Linear Inteiro Misto baseado na técnica de eliminação de subciclos. A modelagem foi implementada posteriormente por [Ehrgott e Freitag, 1996] através de um algoritmo de *branch-and-cut*. Contudo, apenas instâncias com no máximo 30 nós eram resolvidas otimamente. As ideias apresentadas nestes artigos foram utilizadas também para propor versões de algoritmos aproximativos, como em [Kataoka et al., 2000, Garg e Hochbaum, 1997]. Outras abordagens exatas podem ser vistas em [Quintão et al., 2010] e [Simonetti et al., 2011].

Métodos heurísticos também foram propostos em vários artigos da literatura. Alguns deles utilizam critérios gulosos compostos ou não de técnicas duais, como em [Blum, 2007]. Um algoritmo baseado em buscas locais pode ser visto em [Blum e Ehrgott, 2003]. Notadamente, algumas meta-heurísticas conseguiram os melhores limites primais para diversas instâncias do problema. [Brimberg et al., 2006] apresenta um VNS para a versão KCTP ponderada em nós. Outras meta-heurísticas como Colônia de Formigas [Blum e Blesa, 2000, Bui e Sundarraj, 2004], Busca Tabu [Blum e Blesa, 2000, Joernsten e Lokketangen, 1997] e Algoritmos Evolutivos [Blum e Blesa, 2000, Bui e Sundarraj, 2004] também merecem destaque por serem consideradas as melhores técnicas heurísticas conhecidas.

3 Método proposto

O método proposto neste trabalho é uma adaptação do Algoritmo Evolutivo denominado EA_{ages}, apresentado em [Silva e Ochi, 2009]. Trata-se de um Algoritmo Evolutivo baseado em *random-keys*, técnica muita empregada em Problemas de Escalonamento de Projetos (PEPs), como visto em [Gonçalves et al., 2009, Samanlioglu et al., 2009, Snyder e Daskin, 2006].

De modo simplificado, *random-keys* são como prioridades atribuídas a cada um dos elementos a serem escalonados, de forma que o algoritmo de escalonamento procure executar os elementos que tenham prioridades mais elevadas, desde que todas as restrições associadas a estes elementos já tenham sido cumpridas. Normalmente as restrições presentes em PEPs são restrições de precedência (uma tarefa só pode ser executada quando todas as suas predecessoras já tiverem sido) e de recursos (uma tarefa só pode ser executada, se houver recursos disponíveis em quantidade suficiente para tanto). Mais informações sobre PEPs podem ser obtidas em [Damaka et al., 2009, Blazewicz et al., 1983, Zhu et al., 2006]. Portanto, o algoritmo de escalonamento que funcione a partir dessas prioridades pode criar, a cada etapa, uma lista contendo as tarefas disponíveis e ordená-la de acordo com as suas prioridades.

As principais vantagens em se utilizar *random-keys* residem (i) na facilidade de manipulação dos valores das prioridades, já que podem ser utilizados quaisquer valores $v \in \mathbb{R}^+$, uniformizados ou não, e (ii) na possibilidade de realizar o escalonamento com qualquer lista de prioridades, uma vez que é responsabilidade do algoritmo de escalonamento garantir que nenhuma restrição do problema seja violada.

3.1 Algoritmo Evolutivo EA_{ages}

O algoritmo EA_{ages} trabalha com uma população de indivíduos subdividida em três classes: A, B e C. Inicialmente, os indivíduos são gerados de acordo com uma regra que leva em conta apenas os dados de entrada do problema. Para se obter indivíduos distintos, uma variável aleatória λ é somada a cada uma das prioridades que forem inicialmente calculadas. A partir de então, os indivíduos são separados entre as três classes, de forma que os melhores fiquem na classe A, os piores na classe C e o demais indivíduos na classe B.

Para a geração de novos indivíduos, são escolhidos aleatoriamente um pai da classe A e outro da classe B. O cruzamento é realizado através da média aritmética das prioridades de cada elemento dos pais. Soma-se a esses resultados um pequeno valor λ semelhante ao utilizado na geração da população inicial. O objetivo é evitar que o decorrer das gerações estabilize o valor das prioridades, estagnando o algoritmo evolutivo. Um operador de mutação é acionado com uma probabilidade p com o mesmo fim. A mutação muda arbitrariamente o valor das prioridades de um indivíduo.

Ao final de cada geração é aplicado o critério de seleção natural. Pais e filhos são ordenados de acordo com a sua aptidão, mas somente continuam existindo na próxima geração os melhores indivíduos encontrados. Caso ocorram várias gerações sem melhoria do *best* (melhor indivíduo encontrado pelo algoritmo até o presente momento), a população é considerada estagnada.

Esta convergência prematura é um dos problemas mais frequentes em algoritmos evolutivos. O EA_{ages} apresenta algumas ferramentas para tentar sair desta situação. A primeira delas consiste na aplicação de uma busca local computacionalmente mais custosa em todos os indivíduos da classe A. Caso algum deles consiga superar o valor do *best*, a população deixa de ser considerada estagnada. Se isto não ocorrer, o algoritmo tenta um novo ciclo de gerações, seguidas de uma nova fase de busca local nos indivíduos da classe A. Quando um número r de ciclos for executado sem produzir melhorias, entra em ação a segunda ferramenta de diversificação. Toda a população corrente é descartada e uma nova população é gerada. Porém, isto não ocorre como na população inicial. O melhor indivíduo produzido até o momento serve como base para a reconstrução da nova população. Desta forma, um conjunto de novos indivíduos semelhantes ao *best* funcionarão como um processo de intensificação do algoritmo ao redor deste indivíduo. Ao mesmo tempo em que isto ocorre, a reconstrução também tenta sair do estado de estagnação através da remoção de indivíduos que não mostraram potencial de melhoria.

Cada período do algoritmo entre uma reconstrução e outra da população é chamado de *era*. As eras se sucedem até que um limite de tempo seja atingido. O algoritmo 1 apresenta um pseudo-código do EA_{ages}.

Algoritmo 1 EA_{ages}

```
P ← GeraPopulacaoInicial();
ciclos ← 0;
while tempo() < TEMPOLIMITE do
  while ciclos < r do
    P' ← GeraFilhos();
    P' ← Mutacao(P');
    P ← SelecaoNatural(P, P');
    if Estagnada(P) then
      P ← BuscaLocal(P);
      ciclos ← ciclos + 1;
    end if
  end while
  ciclos ← 0;
  P ← Reconstrucao(best);
end while
```

3.2 As adaptações propostas

O algoritmo EA_{ages} mostrou-se bastante eficiente para diversas instâncias do problema de escalonamento de projeto para o qual foi proposto. Em diversas instâncias de pequeno e médio porte foi possível obter soluções com menos de 5% de diferença da solução ótima. O principal objetivo

deste trabalho é analisar a eficiência do EA_{ages} quando aplicado a um tipo de problema distinto dos problemas de escalonamento de projetos. Logicamente, o código do algoritmo precisa ser modificado nos procedimentos de leitura dos dados de entrada e no cálculo da aptidão de cada indivíduo durante a execução do algoritmo evolutivo. Enquanto a leitura dos dados deve ser feita de acordo com a organização apresentada pelo conjunto de instâncias de teste, o cálculo da aptidão consiste em um somatório simples das arestas selecionadas pelo evolutivo.

A geração da população inicial também precisa sofrer modificações, pois as prioridades iniciais levam em consideração apenas os pesos das arestas separadamente. Uma vez definida a lista de prioridades, em qualquer ponto do algoritmo onde seja necessário calcular a aptidão do indivíduo, é então acionado o algoritmo de processamento das prioridades. A aresta com maior prioridade será escolhida inicialmente. Todas as arestas consideradas possíveis têm sua prioridade ordenada. A primeira aresta da lista é sempre a escolhida e o processo continua até que sejam escolhidas k arestas. São consideradas arestas possíveis aquelas que são vizinhas de arestas já escolhidas e não formam ciclos na solução que está sendo construída. A Figura 2 mostra um grafo com prioridades já estabelecidas por um critério arbitrário e como as arestas são escolhidas de acordo com estas prioridades.

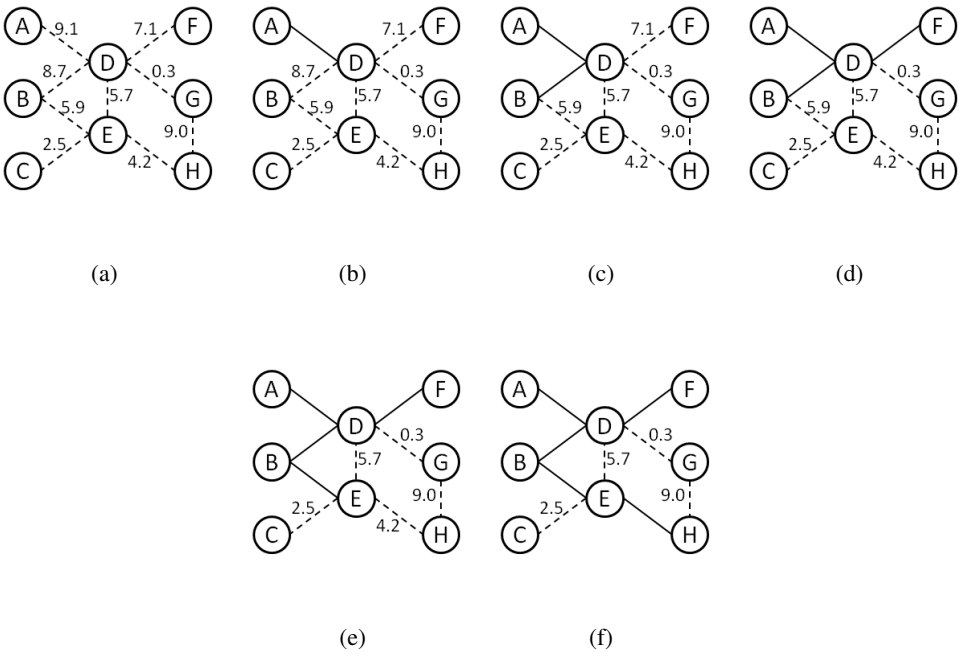

Figura 2: (a) O grafo com as prioridades. $k = 5$ para este exemplo. A aresta AD tem a maior prioridade e será escolhida inicialmente. (b) A aresta GH tem a maior prioridade, mas ela não será escolhida por não ser uma aresta possível. A aresta possível como maior prioridade é BD. (c) A Aresta DF é escolhida. (d) A aresta BE é escolhida. (e) A aresta DE tem a maior prioridade, porém ela formará um ciclo e não poderá ser escolhida. A última aresta escolhida será EH. (f) Solução final do algoritmo de processamento da lista de prioridade.

A busca local utilizada em pontos específicos do EA_{ages} consiste na troca da prioridade de duas arestas quaisquer. Na versão original do algoritmo havia um critério adicional para que um número menor de trocas fosse realizado. Esse critério, no entanto, é inerente apenas ao problema de escalonamento para o qual a troca foi originalmente proposta, não fazendo sentido ser aplicado no KCTP. Outras pequenas modificações foram realizadas no algoritmo, porém apenas nos parâmetros referentes à população de indivíduos. O tamanho da população aumentou para 3000 indivíduos

e as classes A, B e C correspondem a 5%, 15% e 80% da população, respectivamente. Esses parâmetros foram definidos após testes preliminares contendo diversas configurações diferentes. Como o objetivo é testar a aplicação do EA_{ages} a um problema distinto, nenhuma outra alteração foi realizada visando não comprometer a análise dos resultados.

4 Resultados computacionais

Para os teste computacionais foi utilizado um processador Intel I7-920 2.65GHz, com 12 GB de RAM e sistema operacional Linux Ubuntu 12.04 (*kernel* 3.2.0-40). O código foi implementado em linguagem C++, utilizando o compilador G++ versão 4.6.3.

As instâncias empregadas nos testes foram algumas daquelas presentes na KCTLib [Blum, 2012], que normalmente é o *benchmark* utilizado pelos artigos da área. Para que as comparações sejam justas, foram escolhidas apenas versões de meta-heurísticas para comporem os testes. Os algoritmos ACO, EC, TS foram propostos em [Blum e Blesa, 2000]. Os algoritmos ACO-BS e HyEA foram propostos em [Bui e Sundarraj, 2004] e [Blum, 2006], respectivamente. Estes algoritmos foram selecionados por serem aqueles capazes de produzir os melhores limites primais da literatura.

A adaptação do algoritmo EA_{ages}, proposta neste artigo, foi executada 20 vezes para cada um dos valores de k utilizados em cada uma das instâncias. Foram anotadas a média das soluções obtidas a cada execução e a melhor das soluções obtida ao longo das 20 execuções.

As Tabelas 1, 2, 3, 4, 5 e 6 apresentam os resultados dos testes. A primeira coluna indica o valor do parâmetro k avaliado a cada linha da tabela. A segunda e a terceira colunas mostram o valor da melhor solução obtida pelos algoritmos da literatura e quais algoritmos foram os primeiros a obter estes valores. A quarta e a quinta colunas mostram os resultados médios e o melhor resultado obtido pelo EA_{ages}. As duas últimas colunas mostram as diferenças percentuais entre os valores das duas colunas anteriores comparando-os ao melhor resultado da literatura.

k	KCTPLib	Alg.	Média	Melhor	difMédia(%)	difMelhor(%)
20	257	ACO,EC,TS	257,0	257	0,00	0,00
40	642	ACO,EC,TS	642,0	642	0,00	0,00
60	977	ACO,TS	991,8	977	1,52	0,00
80	1335	ACO	1395,2	1335	4,52	0,00
100	1761	HyEA	1777,4	1762	0,94	0,06
120	2235	ACO,EC,TS	2239,4	2235	0,20	0,00
140	2781	EC	2784	2781	0,11	0,00
160	3417	ACO,EC	3417,8	3417	0,02	0,00
180	4158	ACO,EC	4159,1	4158	0,03	0,00
200	5040	EC	5040,6	5040	0,01	0,00
220	6176	EC,TS	6176,0	6176	0,00	0,00

Tabela 1: Instância bb15x15-1

De maneira geral os resultados obtidos pelo EA_{ages} são muito bons, pois apresentam uma diferença percentual muito pequena em relação aos melhores resultados conhecidos. Em muitas execuções, os resultados apresentam valor idêntico. É importante ressaltar que a comparação do EA_{ages} é feita contra cinco das melhores heurísticas existentes. Como não há uma clara predominância de um algoritmo da literatura sobre o outro, se o algoritmo proposto neste trabalho for comparado com cada um deles separadamente, o que parece ser mais justo, as diferenças percentuais tendem a ser ainda menores.

Outro ponto que deve ser levado em consideração é o resultado obtido pelo EA_{ages} para a instância G1000-4-05 com o parâmetro $k = 100$ (primeira linha da última tabela). Embora o resultado

k	KCTPLib	Alg.	Média	Melhor	difMédia(%)	difMelhor(%)
20	253	ACO,EC,TS	253,0	253	0,00	0,00
40	585	ACO,EC	585,0	585	0,00	0,00
60	927	ACO	927,7	927	0,08	0,00
80	1290	ACO	1290,0	1290	0,00	0,00
100	1686	ACO	1704,9	1686	1,12	0,00
120	2120	ACO	2121,5	2120	0,07	0,00
140	2634	ACO	2634,4	2634	0,02	0,00
160	3248	HyEA	3248,3	3248	0,01	0,00
180	3915	TS	3915,4	3915	0,01	0,00
200	4718	EC,TS	4718,0	4718	0,00	0,00

Tabela 2: Instância bb15x15-2

k	KCTPLib	Alg.	Média	Melhor	difMédia(%)	difMelhor(%)
100	1503	ACO-BS	1511,2	1503	0,55	0,00
200	3442	HyEA	3489,9	3447	1,39	0,15
300	5817	HyEA	5978,3	5873	2,77	0,96
400	8691	HyEA	8894,9	8757	2,35	0,76
500	12056	HyEA	12260,0	12111	1,69	0,46
600	15916	HyEA	16108,6	15974	1,21	0,36
700	20511	HyEA	20628,5	20532	0,57	0,10
800	26053	ACO-BS	26139,2	26070	0,33	0,07
900	32963	TS	32967,4	32963	0,01	0,00

Tabela 3: Instância steind5

médio tenha ficado acima do valor da literatura, em pelo menos um das execuções foi possível obter um resultado ainda melhor. Tanto em valores absolutos quanto em relativos, a diferença foi pequena, mas mostra o potencial que o EA_{ages} tem, mesmo não tendo sido projetado especificamente para o KCTP.

Em relação ao tempo computacional gasto pelos algoritmos, infelizmente não é possível fazer uma comparação mais clara e objetiva. Os autores dos outros algoritmos não apresentaram os valores exatos utilizados como critério de parada. Os algoritmos propostos em [Blum e Blesa, 2000] utilizaram um limite de cinco vezes o tempo computacional gasto por uma implementação do clássico algoritmo de Prim aplicado ao grafo como todo. Este critério nem sempre propicia um bom limite pois, como é sabido, a eficiência do algoritmo de Prim em termos práticos depende muito da esparsidade do grafo de entrada.

Alguns métodos exatos e aproximativos chegam a consumir dezenas de milhares de segundos em grafos densos ou com elevado número de arestas. O algoritmo EA_{ages} encontrou as soluções apresentadas entre 100 e 300 segundos, na maior parte das execuções. O tempo computacional gasto com a geração de um indivíduo pelo EA_{ages} é diretamente proporcional ao valor de k e da quantidade total de arestas do grafo, devido à forma como o algoritmo processa as listas de prioridades. De fato, aproximadamente 75% do tempo total de execução do algoritmo é consumido com o processamento da lista de prioridades. Vale lembrar que uma lista com quaisquer valores sempre pode resultar em um indivíduo factível. Consequentemente, qualquer alteração nos valores presentes em um lista pode implicar na construção de um indivíduo completamente diferente. Assim, sempre que uma prioridade é modificada pela mutação ou pela busca local é necessário reordenar a lista e executar novamente o algoritmo de processamento da mesma. Qualquer tipo de otimização

k	KCTPLib	Alg.	Média	Melhor	difMédia(%)	difMelhor(%)
50	208	ACO,EC,TS	208,0	208	0,00	0,00
100	481	TS	481,6	481,0	0,14	0,00
150	802	ACO	804,8	802,0	0,35	0,00
200	1182	ACO-BS	1182,9	1182	0,08	0,00
250	1625	HyEA	1627,5	1625	0,15	0,00
300	2148	ACO-BS	2149,1	2148	0,05	0,00
350	2795	HyEA	2796,5	2796	0,06	0,04
400	3571	EC,TS	3571,3	3571	0,01	0,00
450	4553	EC	4553,4	4553	0,01	0,00

Tabela 4: Instância steinc15

k	KCTPLib	Alg.	Média	Melhor	difMédia(%)	difMelhor(%)
100	1523	ACO-BS	1561,0	1539	2,50	1,05
200	3308	HyEA	3386,0	3333	2,36	0,76
300	5325	HyEA	5451,1	5377	2,37	0,98
400	7581	ACO-BS	7740,3	7689	2,10	1,42
500	10052	ACO-BS	10245,2	10196	1,92	1,43
600	12708	ACO-BS	12944,9	12889	1,86	1,42
700	15675	ACO-BS	15900,2	15860	1,44	1,18
800	19023	HyEA	19219,9	19180	1,04	0,83
900	22827	HyEA	22904,6	22869	0,34	0,18

Tabela 5: Instância G1000-4-1

nas estruturas de dados utilizadas pode resultar em grande economia de tempo.

5 Conclusão

O presente trabalho abordou o problema da Árvore Geradora de Custo Mínimo com k Arestas (k-Cardinality Tree Problem, KCTP). O problema é uma generalização do famoso problema da Árvore Geradora Mínima e encontra aplicações em diversas áreas.

Foi empregado o algoritmo EA_{ages}, proposto para um problema de escalonamento da projetos, com algumas poucas modificações nas suas sub-rotinas. O objetivo é avaliar se um algoritmo não-específico pode ter resultados competitivos se comparado a outros algoritmos da literatura especificamente elaborados para tratar o problema.

k	KCTPLib	Alg.	Média	Melhor	difMédia(%)	difMelhor(%)
100	1652	HyEA	1657,5	1649	0,33	-0,18
200	3620	HyEA	3663,5	3627	1,20	0,19
300	5801	HyEA	5872,8	5821	1,24	0,34
400	8206	HyEA	8303,0	8262	1,18	0,68
500	10793	HyEA	10931,7	10861	1,29	0,63
600	13584	HyEA	13721,7	13679	1,01	0,70
700	16682	HyEA	16794,8	16747	0,68	0,39
800	20076	HyEA	20163,0	20123	0,43	0,23
900	24029	ACO-BS	24062,7	24035	0,14	0,02

Tabela 6: Instância G1000-4-5

Os testes utilizando instâncias da literatura mostraram resultados bastante satisfatórios que indicam que o EA_{ages} tem potencial para tratar outros tipos de problema. As diferenças entre os métodos testados é bastante pequena e em uma das instâncias, foi obtido um resultado melhor que o da literatura. Alguns aprimoramentos no código ainda precisam ser feitos para que os tempos computacionais sejam melhorados, principalmente no processamento da lista de prioridades.

6 Trabalhos Futuros

Sem dúvida, alguns aspectos do EA_{ages} ainda pode ser aprimoradas para que sua aplicação ao KCTP seja ainda mais eficiente. A primeira delas certamente se refere ao algoritmo de geração da árvore a partir da lista de prioridades. Embora seja um algoritmo simples, a estrutura de dados ainda precisa ser um pouco mais otimizada para que o tempo computacional seja reduzido. Grande parte do tempo consumido pelo EA_{ages} se deve a este algoritmo, logo, pequenas melhorias podem produzir uma considerável economia de tempo ao longo de toda a execução.

O operador de mutação está sendo pouco eficiente. Em menos de 5% dos casos nos quais ele é executado, o indivíduo mutante é melhor do que o original. Além disso, a melhoria média é de aproximadamente 3%. As opções consistem em criar novos operadores que incorporem algum tipo de conhecimento sobre o problema ou implementar operadores computacionalmente mais custosos, porém que sejam empregados em situações muito específicas.

As buscas locais apresentam resultados um pouco melhores do que a mutação. Se forem adotados critérios adicionais para realizar as trocas de prioridades, também será possível economizar um pouco de tempo, permitindo que o desempenho geral do EA_{ages} seja ainda melhor.

7 Agradecimentos

Este trabalho foi parcialmente financiado pela FAPERJ, através do auxílio E-26/112.411/2012.

Referências

[Blazewicz et al., 1983] Blazewicz, J., Lenstra, J., e Kan, A. (1983). Scheduling projects to resource constraints: Classification and complexity. *Discrete Applied Mathematics*, 5:11–24.

[Blum, 2006] Blum, C. (2006). A new hybrid evolutionary algorithm for the k-cardinality tree problem. Technical report, Universitat Politecnica de Catalunya, Barcelona (Espanha).

[Blum, 2007] Blum, C. (2007). Revisiting dynamic programming for finding optimal subtrees in trees. *European Journal of Operational Research*, 177:102–115.

[Blum, 2012] Blum, C. (2012). Kctlib: The k-cardinality tree library. `http://iridia.ulb.ac.be/~cblum/kctlib/`. [Online; acessado em 10/12/12].

[Blum e Blesa, 2000] Blum, C. e Blesa, M. (2000). New metaheuristic approaches for the edge-weighted k-cardinality tree problem. *Computers & Operations Research*, 32:1355–1377.

[Blum e Ehrgott, 2003] Blum, C. e Ehrgott, M. (2003). Local search algorithms for the k-cardinality tree problem. *Discrete Apllied Mathematics*, 128:511–540.

[Borndorfer et al., 1997] Borndorfer, R., Ferreira, C., e Martin, A. (1997). Matrix decomposition by branch-and-cut. Technical report, Konrad-Zuse-Zentrum fur Informationstechnik (Alemanha).

[Borndorfer et al., 1998] Borndorfer, R., Ferreira, C., e Martin, A. (1998). Decomposing matrices into blocks. *SIAM Journal on Optimization*, 9:236–269.

[Brimberg et al., 2006] Brimberg, J., Urosevic, D., e Mladenovic, N. (2006). Variable neighborhood search for the vertex weighted k-cardinality tree problem. *European Journal of Operational Research*, 171(1):74–84.

[Bui e Sundarraj, 2004] Bui, T. e Sundarraj, G. (2004). Ant system for the k-cardinality tree problem. In *Proc. of the Genetic and Evolutionary Computation Conference (GECCO 2004)*, Berlim (Alemanha).

[Cheung e Kumar, 1994] Cheung, S. e Kumar, A. (1994). Efficient quorumcast routing algorithms. In *Proc. of INFOCOM4*, Silver Spring(USA). MD: IEEE Society Press.

[Damaka et al., 2009] Damaka, N., B.Jarbouia, P.Siarryb, e T.Loukila (2009). Differential evolution for solving multi-mode resource-constrained project scheduling problems. *Computers & Operations Research*, 36:2653–2659.

[Ehrgott e Freitag, 1996] Ehrgott, M. e Freitag, J. (1996). K tree/k subgraph: a program package for minimal weighted k-cardinality tree subgraph problem. *Eur. J. Operational Res.*, 93:214–225.

[Fischetti et al., 1994] Fischetti, M., Hamacher, W., Jornsten, K., e Maffioli, F. (1994). Weighted k-cardinality trees: complexity and polyhedral structure. *Networks*, 24:11–21.

[Foulds e Hamacher, 1992] Foulds, L. e Hamacher, H. (1992). A new integer programming approach to (restricted) facilities layout problems allowing exible facility shapes. Technical report, University of Waikato, Department of Management Science.

[Foulds et al., 1998] Foulds, L. R., Hamacher, H., e Wilson, J. (1998). Integer programming approaches to facilities layout models with forbidden areas. *Annals of Operations Research*, 81:405–417.

[Garg e Hochbaum, 1997] Garg, N. e Hochbaum, D. (1997). An o(log k) approximation algorithm for the k minimum spanning tree problem in the plane. *Algorithmica*, 18:111–121.

[Gonçalves et al., 2009] Gonçalves, J., Mendes, J., e Resende, M. (2009). A random key based genetic algorithm for the resource constrained project scheduling problems. *International Journal of Production Research*, 36:92–109.

[Hamacher et al., 1991] Hamacher, H., Jornsten, K., e Maffioli, F. (1991). Weighted k-cardinality trees. Technical report, Politecnico di Milano, Dipartimento di Elettronica (Itália).

[Hamacher et al., 1993] Hamacher, H., Jornsten, K., e Maffioli, F. (1993). Optimal relinquishment according to the norwegian petrol law: a combinatorial optimization approach. Technical report, Norwegian School of Economics and Business Administration, Bergen (Noruega).

[Joernsten e Lokketangen, 1997] Joernsten, K. e Lokketangen, A. (1997). Tabu search for weighted k-cardinality trees. *Asia Pacific Journal of Operational Research*, 14:9–26.

[Kataoka et al., 2000] Kataoka, S., Araki, N., e Yamada, T. (2000). Upper and lower bounding procedures for the minimum rooted k-subtree problem. *European Journal of Operational Research*, 122:561–569.

[Marathe et al., 1996] Marathe, M., Ravi, R., Ravi, S., Rosenkrantz, D., e Sundaram, R. (1996). Spanning trees short or small. *SIAM Journal on Discrete Mathematics*, 9:178–200.

[Philpott e Wormald, 1997] Philpott, A. e Wormald, N. (1997). On the optimal extraction of ore from an open-cast mine. Technical report, University of Auckland (Nova Zelândia).

[Quintão et al., 2010] Quintão, F., da Cunha, A. S., Mateus, G. R., e Lucena, A. (2010). The k-cardinality tree problem: Reformulations and lagrangian relaxation. *Discrete Applied Mathematics*, 158(1):1305–1314.

[Samanlioglu et al., 2009] Samanlioglu, F., Jr., W. F., e Kurz, M. (2009). A memetic random-key genetic algorithm for a symmetric multi-objective traveling salesman problem. *Computers & Industrial Engineering*, 55(2):439–449.

[Silva e Ochi, 2009] Silva, A. e Ochi, L. (2009). New sequential and parallel algorithm for dynamic resource constrained project scheduling problem. In *Proc. of the International Workshop on Nature Inspired Distributed Computing (NIDISC'09)*, Roma (Itália).

[Simonetti et al., 2011] Simonetti, L., Protti, F., Frota, Y., e de Souza, C. (2011). New branch-and-bound algorithms for k-cardinality tree problems. *Electronic Notes in Discrete Mathematics*, 37(1):27–34.

[Snyder e Daskin, 2006] Snyder, L. e Daskin, M. S. (2006). A random-key genetic algorithm for the generalized traveling salesman problem. *European Journal of Operational Research*, 174(1):38–53.

[Zelikovsky e Lozevanu, 1993] Zelikovsky, A. e Lozevanu, D. (1993). Minimal and bounded trees. In *Tezele Cong. XVIII Acad. Romano-Americane*, Kishinev.

[Zhu et al., 2006] Zhu, G., Bard, J., e Tu, G. (2006). A branch-and-cut procedure for the multimode resource-constrained project-scheduling problem. *Journal on Computing*, 18(3):377–390.