Problemas de Otimização Combinatória (2014-1)

Relatório

Para cada combinação problema vs. metaheurística, apresentar um relatório com aproximadamente 6 páginas contendo no mínimo, as seguintes informações:

- Introdução.
- Descrição clara do problema e formulação matemática do problema.
- Descrição com detalhes do algoritmo proposto:
  - Representação do problema.
  - Principais estruturas de dados
  - Heurística construtiva.
  - Vizinhança e a estratégia de escolha de vizinhos.
  - Parâmetros do método (combinações testadas e valores usados nos experimentos).
  - Critério de terminação.
- Tabela de resultados com no mínimo as seguintes colunas para cada instância:
  - Valor da solução inicial heurística.
  - Valor da melhor solução ou taxa de sucesso para problemas de satisfabilidade encontrada pelo seu algoritmo (S).
  - Tempo de execução (em segundos).
  - Solução ou taxa de sucesso para problemas de satisfabilidade do GLPK (Reportar a melhor solução encontra no tempo disponível para execução do GLPK, mesmo quando não ótima).
  - Tempo do GLPK (com limite de tempo de 1h por instância - ou mais tempo).
  - Desvio percentual ($100\frac{|S-MC|}{S}$ da melhor solução conhecida MC.
  - Análise dos resultados.
  - Conclusões.
  - Bibliografia pesquisada.

Os resultados das metaheurísticas deve ser uma média de no mínimo 10 rodadas. Além do relatório, os alunos devem entregar (no moodle) um zip contendo os códigos.

Implementação

- Todas as implementações devem aceitar uma instância no formato do problema na entrada padrão (stdin) e imprimir a melhor solução encontrada, bem como o tempo utilizado na saída padrão (stdout).
- Os principais parâmetros do método devem ser definíveis pela linha de comando.
- Critérios básicos de engenharia de software: documentação, legibilidade, etc.
- Critérios como qualidade das soluções encontradas e eficiência das implementações serão considerados na avaliação.

1. **Linear arrangement**

Instância Um grafo (não-direcionado) $G = (V, A)$.

Solução Uma ordenação π dos vértices.

Objetivo Minimizar a distância total entre vértices vizinhos $\sum_{\{u,v\} \in A} |\pi(u) - \pi(v)|$.

As instâncias estão disponíveis em `www.inf.ufrgs.br/~mrpritt/oc/la.zip`. Testes devem ser feitos com bintree10, hc10, gd96a, mesh33x33, c3y, c4y, c5y, airfoil1, 3elt, whitaker. Tamanhos e melhores valores conhecidos:

Instância	n	m	Melhor valor
bintree10	1023	1022	4267
hc10	1024	5120	523776
mesh33x33	1098	2112	32703
3elt	4720	13722	431737
airfoil1	4253	12289	322611
whitaker3	9800	28989	1307540
c3y	1327	2844	124117
c4y	1366	2915	115144
c5y	1202	2557	96952
gd96a	1096	1676	96253

Os arquivos de entrada contêm os números n e m, seguido pelos graus dos vértices $\delta_1, \ldots, \delta_n$, seguido por $2m$ números $v_1, \ldots, v_{2m}$, seguido por -1, seguido por $n+1$ números $a_1, \ldots, a_{n+1}$. Os números satisfazem $\delta_i = a_{i+1} - a_i$. Os vértices do grafo são $0, 1, \ldots, n-1$. Os vizinhos do vértice i são os vértices $v_{a_i}, \ldots, v_{a_{i+1}-1}$.
`http://tracer.lcc.uma.es/problems/minla/minla.htm`.

2. **Localização de facilidades não-capacitadas**

Instância Um conjunto de n cidades, custos f_i de abrir uma facilidade na cidade $i \in [n]$, um limite de facilidades k, custos de atendimento c_{ij} da cidade $i \in [n]$ por uma facilidade na cidade $j \in [n]$.

Solução Uma seleção de no máximo k cidades para abrir uma facilidade, e uma atribuição de cada cidade para uma cidade com facilidade aberta para atender a demanda.

Objetivo Minimizar a soma dos custos para abrir facilidades e para atender as cidades.

As instâncias estão disponíveis em `www.inf.ufrgs.br/~mrpritt/oc/ufl-2014-1.tgj` (pode descompactar com "tar -xovf ufl-2014-1.tgj"). Os testes devem ser feitos com as instâncias 923UnifS, 934PCodesS, cap131, MO1, MQ2, MT5, ga250a-2, ga500c-1, ga750a-1, ga750c-1. Detalhes sobre o formato disponível em `http://www.inf.ufrgs.br/~MRPRITT/doku.php?id=inf05010:formato-ufl`.

3. **Mapeamento de Redes Virtuais**

Instância Dois grafos não-direcionados: um grafo $S = (V^S, E^S)$, representando o substrato físico com capacidades de CPU c_v para cada nodo $v \in V^S$ e capacidade de banda c_e para cada aresta $e \in E^S$, e um grafo $N = (V^V, E^V)$, representando a rede virtual com demandas de CPU d_v de para cada nodo $v \in V^V$ com demanda de banda d_e para cada $e \in E^V$.

Solução Um mapeamento M. Todo nodo virtual v deve ser mapeado para um nodo físico s que tenha capacidade suficiente para hospedá-lo. Cada nodo físico pode hospedar no máximo um nodo virtual. Cada link e da rede virtual deve ser mapeado para uma ou mais arestas do grafo físico, essas arestas devem formar um caminho entre os nodos físicos que hospedam os nodos virtuais adjacentes a e. A soma das demandas de todos os links virtuais que são hospedadas por uma aresta física não pode ultrapassar a sua capacidade.

Objetivo Minimizar a soma da banda usada pelo mapeamento. Se um link virtual com demanda 10 é mapeado para um caminho com 3 arestas físicas, o custo desse mapeamento é 30.

Instâncias disponíveis em `www.inf.ufrgs.br/~mrpritt/oc/mrv.zip`. Cada pasta corresponde a uma instância contendo dois arquivos: *sub.gtt* e *vir.gtt*, correspondentes aos grafos físico e virtual, respectivamente. Cada arquivo contêm três tipos de linha que são identificadas pela primeira letra:

- "G" V E - linha que contém o número de vértices V e de arestas E e informações sobre a topologia da rede (que podem ser ignoradas).
- "V" I C - corresponde a um vértice, de identificador I e capacidade/demanda C.
- "E" U V C - corresponde a uma aresta, que liga U a V e tem capacidade/demanda C.

Melhores valores:

Instância	Melhor valor	Ótimo
sub1	266	sim
sub2	280	sim
sub3	421	sim
sub4	619	sim
sub5	454	não
ts1	282	sim
ts2	300	sim
ts3	499	sim
ts4	653	sim
ts5	676	sim

Dicas de referência às metaheurísticas:

a) VNS: A Tutorial on Variable Neighborhood Search, by Pierre Hansen (GERAD and HEC Montreal) and Nenad Mladenovic (GERAD and Mathematical Institute, SANU, Belgrade), 2003. `http://www.gerad.ca/fichiers/cahiers/G-2003-46.pdf`

b) GRASP: `http://www2.research.att.com/~mgcr/grasp/gannbib/tutorial/index.html`

c) GA: A genetic algorithm tutorial, by D. Whitley, Statistics and computing 4 (2), 65-85.

d) Tabu Search: Tabu Search: A Tutorial, by Fred Glover (1990), Interfaces.

e) SA: Optimization by simulated annealing: An experimental evaluation; part I, graph partitioning, by D.S. Johnson, C.R. Aragon, L.A. McGeoch, C. Schevon, Operations research 37 (6), 865-892, 1989.