Problemas de Otimização Combinatória (2013-2)

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 - ou mais tempo).
  - Desvio percentual ($100\frac{S-MC}{S}$ - para o caso de problemas de minimização) 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. **Problema de Ordenação Linear**
   - **Entrada:** Uma matriz $C_{n \times n}$ de custos.
   - **Solução:** Uma permutação π de colunas e linhas (a mesma permutação é utilizada para linhas e colunas).
   - **Objetivo:** Maximizar a soma dos valores no triângulo superior (valores acima da diagonal).
   - **Informações Adicionais:** Instâncias disponíveis em `http://www.optsicom.es/lolib/`. Testes devem ser feitos nas instâncias: IO(N-t65l11xx, N-tiw56n54), RandA1(N-t1d100.01, N-t1d150.11, N-t1d500.7, N-t1d500.25), RandA2(N-t2d150.02, N-t2d200.14), RandB(N-p50-09), SPEC(N-atp111, N-atp163).

   Melhores valores conhecidos:

Instância	Valores
N-t65l11xx	16719
N-tiw56n54	91554
N-t1d100.01	106852
N-t1d150.11	234157
N-t1d500.7	2400739
N-t1d500.25	2405718
N-t2d150.02	73624
N-t2d200.14	144384
N-p50-09	43711
N-atp111	1495
N-atp163	2073

2. **Particionamento de Conjuntos**
   - **Entrada:** Um universo U,uma família S de subconjuntos do universo e custos $c(S)$ para cada conjunto.
   - **Solução:** Uma seleção de conjuntos, em que cada elemento de U pertence a exatamente um conjunto, i.e., $\sum_{j \in S} a_{rj} x_j = 1$ com $r \in U$.
   - **Objetivo:** Minimizar o custo total dos conjuntos selecionados.
   - **Informações Adicionais:** Instâncias disponíveis em `http://people.brunel.ac.uk/~mastjjb/jeb/orlib/sppinfo.html`. Testes devem ser feitos nas instâncias: sppnw41, sppnw32, sppnw34, heart, delta, sppnw36, meteor, sppaa06, sppnw05, sppnw16, sppus01.

   Melhores valores conhecidos:

Instância	Valores
sppnw41	11307
sppnw32	14877
sppnw34	10488
heart	180
delta	126
sppnw36	7314
meteor	60
sppaa06	27040
sppnw05	132878
sppnw16	1181590
sppus01	10036

3. **Minimizar o tempo de fluxo no flow shop permutacional**
   - **Instância** Um conjunto de tarefas $J = [n]$ a serem executadas nas máquinas $M = [m]$. Uma tarefa $j \in J$ possui m operações com tempo de execução t_{ij} para $i \in M$. A i- ésima operação de cada tarefa deve ser executada na máquina i. Em cada instante, uma tarefa pode ser processada por uma única máquina, e cada máquina pode processar no máximo uma tarefa. Uma vez iniciado, o processamento de uma tarefa tem que ser executada sem interrupção (*non-preemptive*). Além disso, as tarefas devem ser processadas em cada máquina na mesma ordem.
   - **Solução** Uma permutação π das tarefas que define o escalonamento das tarefas. (A saber: a tarefa π_j termina na máquina i no momento $C_{i\pi_j} = \max\{C_{i-1,\pi_j}, C_{i,\pi_{j-1}}\}$, com $C_{0j} = C_{i,\pi_0} = 0$.)
   - **Objetivo** Minimizar o tempo de fluxo $\sum_{j\in J} C_j$ (ingl. *flowtime*) total das tarefas, sendo $C_j = C_{mj}$ o tempo de término (ingl. *completion time*) da tarefa j.
   - **Informações adicionais** Um gerador de instâncias é disponível em `http://www.mathematik.uni-osnabrueck.de/research/OR/fsbuffer/taillard.c`. Testes devem ser feitos com instâncias ta$n$, com $n = 1 + 10k$ para $k \in \{0, 1, \ldots, 11\}$.

   Melhores valores conhecidos:

Instância	Valor	Instância	Valor
ta001	14033	ta061	253266
ta011	20911	ta071	298385
ta021	33623	ta081	365463
ta031	64802	ta091	1046314
ta041	87114	ta101	1227733
ta051	125831	ta111	6698656

4. **Árvores de Steiner**
   - **Entrada:** Um grafo $G = (V, A)$ não direcionado com vértices V e arestas A e custos $c_a \geq 0$ para $a \in A$.
   - **Solução:** Um subgrafo conexo mínimo que inclui um dado conjunto de vértices necessários $T \subseteq V$.
   - **Objetivo:** Minimizar $\sum_{a\in A} c_a$.
   - **Informações Adicionais:**Instâncias disponíveis em `http://steinlib.zib.de//steinlib.php`. Testes devem ser feitos nas instâncias: b14, c01, d10, e20, mc11, brasil58, cc3-4p, hc10p, i160-003, i160-033.

   Melhores valores conhecidos:

Instância	Valores
b14	235
c01	85
d10	d10
e20	1342
mc11	11689
brasil58	13655
cc3-4p	2338
hc10p	60679
i160-003	2297
i160-033	2101