Lista de exercícios 1

Nos vamos usar o problema da ordenação linear (ingl. linear ordering problem, LOP) como exemplo recorrente nas listas nesse semestre. Segue a definição do problema

---

Ordenação linear

Instância Um matriz $W = (w_{ij}) \in \mathbb{R}^{n \times n}$.

Solução Uma permutação π de $[n]$.

Objetivo Maximizar o pesos no triângulo superior da matriz W permuntando linhas e colunas por π, i.e. maximizar

$$\Delta(\pi) = \sum_{i \in [n]} \sum_{j \in [i+1,n]} w_{\pi_i, \pi_j}.$$

Exercício 1 (Vizinhanças, 3 pt)
Considere as seguintes três vizinhanças para o LOP, para uma dada permutação π:

a) A vizinhança "2-swap" que troca a posição de dois elementos de π.

b) A vizinhança "k-reverse" que inverte a ordem de $k > 2$ posições consecutivas (para um k fixo, então a reposta depende do k).

c) A vizinhança "2-move-shift" que, para duas posições i e j remove π_i, coloca π_j na posição i (onde π_i estava antes) anterior de i, e depois re-insere π_i em outra posição.

Para as três vizinhanças: ela é simétrica? Conectada? Fracamente otimamente conectada? Exata? Responde as dúvidas por uma demonstração ou um contra-exemplo.
Exemplos: Para $\pi = (1\,2\,3\,4\,5)$ obtemos com

- um 2-swap das posições 2 e 4: $\rho = (1\,4\,3\,2\,5)$;
- um 3-reverse do segmento $[1, 3]$: $\rho = (3\,2\,1\,4\,5)$;
- um 2-move-shift das posições $i = 2$ e $j = 4$, onde i sofre um shift para direita por 3 posições $\rho = (1\,4\,3\,5\,2)$.

Exercício 2 (Busca local para um problema polinomial, 3pt)
Estude a busca local na vizinhança 1-flip no problema $1 \mid\mid \sum U_i$. Neste problema temos n tarefas com tempos de processamento p_i, e prazos d_i, $i \in [n]$ que devem ser executadas numa única máquina. Uma solução do problema é uma permutação π das tarefas, e o objetivo é encontrar a permutação que minimiza o número de tarefas atrasadas (i.e. que terminam depois do prazo) $U_{\text{sum}} = \sum_{i \in [n]} U_i$, onde $U_i = 1$ caso a tarefa i está atrasada, e 0 caso contrário.
Para uma permutação π, o tempo de término é definido recursivamente por $C_{\pi_1} = p_{\pi_1}$ e $C_{\pi_{i+1}} = C_{\pi_i} + p_{\pi_{i+1}}$ para $i \in [n-1]$. Uma solução é representada pelo vetor $U \in \{0,1\}^n$. Para saber as tarefas realmente não atrasadas, serve o seguinte algoritmo. Ordene as tarefas por (U_i, d_i) lexicograficamente (i.e. primeiro as tarefas não atrasadas, em ordem não-decrescente dos prazos, e depois as tarefas atrasadas, em ordem não-decrescente dos prazos). Depois calcula o tempo de término de toda tarefa e conta o número de tarefas realmente não atrasadas.

a) Compare as estratégias "first improvement" e "best improvement" na vizinhança 1-flip. Repete o experimento para cada estratégia 100 vezes com instâncias iniciais aleatórias. Gera as instâncias aleatórias da seguinte forma: para cada $n \in \{100, 200, \ldots, 500\}$, gera primeiramente os tempos

$p_1, \ldots, p_n$ tirados de $U\{1, 100\}$. Seja $P = \sum_{i \in [n]} p_i$. Com isso gera os prazos $d_1, \ldots, d_n$ tirados de $U\{0.1P, 0.7P\}$ (todos dados são números inteiros). Depois gera 20 permutações aleatórias para cada n. Compare a média aritmética (μ) e desvio padrão (σ) do tempo, número de iterações e função objetivo U_{sum}. Apresenta uma análise dos resultados.

b) Um algoritmo exato para o problema é o seguinte. Começa com uma permutação vazia $\pi = ()$. Ordene as tarefas por prazos não-decrescentes e processa a tarefas nessa ordem, sempre alocando no final da permutação atual. Caso a última tarefa não está no prazo, remove a tarefa mais longa da permutação atual. Supõe que isso termina com $\pi_1, \ldots, \pi_k$. A permutação completa é obtida por adicionar, em qualquer ordem, as tarefas removidas. Compara a implementação em termos de tempo e função objetivo com as buscas locais.

Entrega dos resultados: para os dois itens um arquivo texto com os resultados individuais da forma

```
alg n      no   time iterations        value
FI  100     1   17.3 23764          88123
FI  100     2   17.3 23764          88123
BI  100     1   32.2 41234          88123
EX  100     1   32.2 NA             88123
...
```

com os seguintes campos:

- "alg": categorias "FI" para "first improvement" e "BI" para "best improvement" e "EX" para "algoritmo exato";
- "n": tamanho da instância $n \in \{100, 200, \ldots, 500\}$;
- "no": número da permutação entre $1, \ldots, 20$
- "time": tempo de execução, em segundos;
- "iterations": número de iterações (nota que o algoritmo exato não tem iterações: neste caso informar "NA" na tabela);
- "value": número de tarefas atrasadas $\sum_i U_i$

Exercício 3 (Busca local para um problema NP-completo, 4pt)
Estude a busca local na vizinhança 1-shift no LOP (1-shift remove uma tarefa da permutação e insere em qualquer outra posição). Use as 50 instâncias `nnnnn_1000`. As instâncias podem ser encontrados em https://github.com/mrpritt/lop-instances), os melhores valores em https://www.inf.ufrgs.br/~mrpritt/msc/bkv-1000-export.csv.

a) Compare as estratégias "first improvement" e "best improvement". Repete o experimento para cada estratégia 100 vezes para cada instância. Compare a média aritmética (μ) e desvio padrão (σ) do tempo, número de iterações e valor $\Delta(\pi)$ das duas estratégias. Apresenta um histograma dos valores de $\Delta(\pi)$ para as duas estratégias.

b) Compare com uma busca local monótona randomizada. Testa com valores de

$$p \in \{0.0, 0.05, 0.25, 0.5, 0.75, 1.0\}.$$

Repete o experimento 15 vezes para cada instância e valor de p. Relate o tempo até encontrar a solução ótima, ou o melhor valor encontrada com de um tempo limite de 15 segundos e o número de iterações numa tabela da forma $\mu \pm \sigma$.

c) Determine a complexidade empírica do algoritmo: a) o tempo para encontrar a melhor solução conhecida, e b) o número de iterações para encontrar a solução ótima em função do tamanho da instância. Neste caso é necessário que selecionar algumas instâncias de tamanhos diferentes[1].

Observações:

- Para um desempenho razoável considere implementar um cálculo eficiente da função objetivo.
- Para escolher um vizinho melhor ou arbitrário uniformemente sem armazenar todos vizinhos pode-se usar *reservoir sampling*: ao encontrar o i-ésimo vizinho candidato, aceitá-lo com probabilidade $1/i$.

Entrega dos resultados:

- Para o primeiro item um arquivo texto com os resultados individuais da forma

  ```
  alg instance    rep time iterations     value
  FI  ta001       1   17.3 23764          88123
  FI  ta001       2   17.3 23764          88123
  BI  ta001       1   32.2 41234          88123
  ...
  ```

  com os seguintes campos:
  - "alg": categorias "FI" para "first improvement" e "BI" para "best improvement";
  - "instance": nome da instância;
  - "rep": o número da replicação entre $1, \ldots, 100$
  - "time": tempo de execução, em segundos;
  - "iterations": número de iterações;
  - "value": valor total do triãngulo superior $\Delta(\pi)$
- Para o segundo item um arquivo texto com os resultados individuais, da forma

  ```
  p   instance    rep time iterations value
  0.0 ta001       1   30.1 2943       88123
  0.0 ta001       2   33.8 1234       88124
  ...
  ```

  com os seguintes campos:
  - "p": probabilidade na BLMR;
  - "instance": nome da instância;
  - "rep": o número da replicação entre $1, \ldots, 15$
  - "time": tempo de execução, em segundos;
  - "iterations": número de iterações;
  - "value": valor total do triãngulo superior $\Delta(\pi)$

Data de entrega: 25 de junho de 2023.

[1] Sugestão: usar a instâncias de xLOLIB2 (melhores valores em https://www.inf.ufrgs.br/~mrpritt/msc/xlolib2-bkv.csv). Caso o algoritmo não chega nas soluções em tempo hábil, da par medir o tempo para chegar no valor αv para melhor valor v e algum $\alpha \in (0, 1)$ adequado)

Regras para listas de exercícios

1. Os exercícios podem ser resolvidos em colaboração com outros, mas a entrega é individual informando os eventuais colaboradores.
2. A entrega é eletrônica, num única arquivo com todos artefatos (relatório, não escrito a mão, em formato PDF, código fonte, e tabelas).
3. Para receber pontos as respostas devem ser justificadas (i.e. provadas quando não são obvias).
4. Somente entregem respostas que vocês sabem explicar pessoalmente.