UNIVERSIDADE FEDERAL DE MINAS GERAIS
Instituto de Ciências Exatas
Departamento de Ciência da Computação

DCC207 -- Algoritmos 2
Prof. Renato Vimieiro

Trabalho Prático 2 – Soluções para problemas difíceis

Objetivos

Nesse trabalho serão abordados os aspectos práticos dos algoritmos para solucionar problemas difíceis. Avaliaremos as implementações dos algoritmos para computação de rotas no problema do caixeiro viajante. Especificamente, avaliaremos uma solução exata, baseada em branch-and-bound, e as duas soluções aproximadas vistas em sala de aula para o TSP euclidiano, twice-around-the-tree e algoritmo de Christofides.

O objetivo principal é que os alunos presenciem as dificuldades inerentes da implementação dos algoritmos vistos em sala de aula. Os alunos deverão tomar decisões e/ou investigar quanto a melhor representação dos dados, e estruturas mais adequadas para a implementação dos algoritmos.

Tarefas

Os alunos deverão implementar um algoritmo branch-and-bound, o algoritmo twice-around-the-tree, e o algoritmo de Christofides para solucionar o problema do caixeiro viajante geométrico.

As implementações poderão ser feitas em Python3 (preferencial) ou C++11. Em ambos os casos, só poderão ser usadas as bibliotecas constantes nas distribuições padrão das linguagens. No caso de Python3, pode-se assumir como distribuição padrão a Anaconda 22.9 e/ou Python +3.9 com NumPy, SciPy e Pandas. No caso de C++11, deve-se assumir o padrão da linguagem. Exclusivamente para C++, deverá ser enviado, em conjunto com o código-fonte, um arquivo makefile com as instruções de compilação do programa.

Para a manipulação de grafos e algoritmos relacionados, mas não os mencionados aqui, você pode usar as bibliotecas Networkx (Python), iGraph (C++/Python). O uso de bibliotecas adicionais deve ser discutido com o professor.

Em seguida, os alunos deverão avaliar o desempenho dos algoritmos segundo três aspectos: tempo, espaço, e qualidade da solução. Cada algoritmo deverá ser executado com cada instância do conjunto de teste, e as variáveis de desempenho devem ser coletadas para cada execução. O tempo de processamento deve ser limitado a 30min. Após esse prazo a execução do algoritmo deve ser abortada e os dados referentes à execução colocados como NA (não-disponível).

Como conjunto de teste, serão usadas instâncias compiladas na biblioteca TSPLIB (http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/). Em particular, serão consideradas apenas as instâncias cuja função de custo seja a distância euclidiana em 2D. A lista de instâncias a serem usadas está disponível no arquivo anexo a essa descrição. Nesse arquivo, estão listadas as instâncias, seu tamanho (número de nós), e um limiar de qualidade em que se encontra a solução ótima segundo a biblioteca. O arquivo original com esses dados está disponível em http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/tsp95.pdf (o ótimo está listado em http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/STSP.html). O pdf no link anterior também contém a descrição do formato dos arquivos de entrada que estão disponíveis em http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/tsp/.

Finalmente, deverá ser escrito um relatório em formato de artigo científico em que:

1. Se introduz o problema e o trabalho realizado.
2. Descreve as implementações utilizadas. Deve-se apresentar com detalhes a escolha da estimativa de custo; estruturas de dados usadas e o porquê; escolha de best-first ou depth-first no algoritmo de branch-and-bound; além de qualquer outro detalhe de implementação e dos algoritmos em si.
3. Apresentar os experimentos e discutir os resultados. Você deve avaliar os limites de cada algoritmo/implementação, tentando buscar uma relação entre tamanho da instância e desempenho. Deve também comparar os algoritmos entre si. Tente responder quando cada implementação se sai melhor ou deveria ser usada.
4. Apresente as conclusões do seu trabalho. Mostre o quê pôde ser percebido com seus experimentos.

O artigo deverá ser escrito em Latex, usando o formato de artigos de conferência da Sociedade Brasileira de Computação (o template se encontra no site da SBC e no editor de Latex online Overleaf; consulte o professor se tiver dúvida).

Caso não tenha familiaridade com escrita de artigos, sugiro a leitura do material preparado pela Profa. Mirella disponível em https://homepages.dcc.ufmg.br/~mirella/doku.php?id=escrita.

O que entregar?

Devem ser entregues todos os arquivos fonte usados na implementação, bem como uma tabela completa com todos os resultados coletados nos experimentos. Caso o relatório não esteja junto com o código fonte, então esse deverá ser entregue à parte em formato pdf. O trabalho deverá ser entregue em um repositório do GitHub a ser tornado público somente após a entrega. O link para o repositório será entregue no Teams.

Política de Plágio

Os alunos podem, e devem, discutir soluções sempre que necessário. Dito isso, há uma diferença bem grande entre implementação de soluções similares e cópia integral de ideias. Trabalhos copiados na íntegra ou em partes de outros alunos e/ou da internet serão prontamente anulados. Caso haja dois trabalhos copiados por alunos/grupos diferentes, ambos serão anulados. Recomenda-se particular atenção ao uso de ferramentas auxiliares para construção de código (e.g. Copilot) e de referências online como GeekforGeeks. Aproveitamento de códigos dessas origens são particularmente anulados com frequência.

Datas

Entrega final Teams: 10/12/2023 às 23h59

Notem que os experimentos podem requerer bastante tempo (horas) para serem executados, dependendo da implementação. Esse tempo deve ser considerado na organização do trabalho para evitar problemas com a entrega final, sobretudo com a escrita do relatório.

Política de atraso

Haverá tolerância de 30min na entrega dos trabalhos. Submissões feitas depois do intervalo de tolerância serão penalizados.

- Atraso de 1 dia: 50%
- Atraso de +2 dias: não aceito

Serão considerados atrasos de 1 dia aqueles feitos após as 0h30 do dia de entrega. A partir daí serão contados o número de dias passados da data de entrega.

Referências

https://en.wikipedia.org/wiki/Blossom_algorithm
https://www.eecs.tufts.edu/~gdicks02/Blossom/Blossom_Algorithm.html
https://networkx.github.io/documentation/latest/index.html
https://igraph.org
https://homepages.dcc.ufmg.br/~mirella/doku.php?id=escrita
http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/