MC658 - Projeto e Análise de Algoritmos III

Prof. Flávio Keidi Miyazawa
PED: Mauro Henrique Mulati
Laboratório 1 - 2º Semestre de 2016

Backtracking e Branch and Bound para o Problema do Caixeiro Viajante (TSP)

Há algumas modificações ao longo do texto. Verifique.

TSP. São dados como entrada um grafo não orientado $G = (V, E)$ e uma função de peso $w : E \to \mathbb{R}$. O objetivo é encontrar um ciclo de custo mínimo que que passe por cada nó exatamente uma vez e retorne ao vértice inicial. Seja $n = |V|$, note que o número de arestas do ciclo é exatamente n. Não podem haver subciclos.

Técnicas. Este lab consiste da implementação de dois algoritmos para o TSP: um de *backtracking* e o outro de *branch and bound*, seguindo o que foi visto em aula. No caso *branch and bound*, é importante notar que não é para utilizar programação linear, mas sim outros tipos de limitantes superiores e inferiores, como aqueles obtidos baseados em árvore geradora mínima.

Dado tempo suficiente, seu programa deve ser capaz de retornar soluções ótimas para as instâncias do TSP utilizadas.

Implementação. O projeto deve ser feito na linguagem `C++`. Será lhe fornecida a base da implementação, e você editará apenas os métodos nomeados `bt` e `bnb` do arquivo fonte `tsp_bt_bnb.cpp`. Certifique-se de que inserir seu RA no local adequado dentro dess arquivo. Além desse arquivo, existem rotinas para auxiliar no uso da biblioteca **Lemon** e outros utilitários. Você *deverá* elaborar seu código usando esta biblioteca para representar e manipular o grafo. Documentação do Lemon pode ser encontrada em `http://lemon.cs.elte.hu/pub/doc/latest/`.

Esse esqueleto de projeto possui os arquivos principais:

- `tsp.{cpp|h}`: possui o método `main` do seu projeto. Você não deve alterá-los.
- `tsp_bt_bnb.h`: é o header de sua implementação. Você também não deve alterá-lo.

Uma vez que você implemente as duas funções, você pode compilar seu projeto digitando `make` no diretório. Depois disso, você pode digitar `./run` no diretório que seus algoritmos serão testados nas instâncias existentes no subdiretório `in`.

As funções `bt` e `bnb` devem executar até o limite de tempo passado por parâmetro. Sobre o retorno dessas funções:

- Se encontrou uma solução garantidamente ótima: retorna `true`.
- Se encontrou uma solução, mas não é garantidamente ótima; retorna `false`.
- Se não encontrou solução factível: retorna `false` e temos que o objeto `tsp` passado por referência terá: `tsp.BestCircuitValue = ` ∞.

E ainda, se encontrou alguma solução factível, a sequência de nós do ciclo deverá estar corretamente preenchido o `vector<Node> BestCircuit` da mesma estrutura `tsp`. O primeiro nó não deve estar repetido no final.

Por fim, a obtenção de sua solução deve obedecer o seguinte critério: *desempate por ordem lexicográfica (no caso do primeiro vértice, este deve ser o de menor nome especificado no arquivo de entrada)*. Isto é importante no momento que formos executar testes para conferir sua solução.

Depois que seu algoritmo constrói a solução, o projeto verifica se sua solução é válida. Checagem de qualidade de solução e tempo de execução serão feitas manualmente no momento da avaliação.

Submissão e testes. Dentre os códigos-fonte, você deverá submeter apenas o arquivo `tsp_bt_bnb.cpp` no SUSY: utilize a tarefa `lab01` em `https://susy.ic.unicamp.br:9999/mc658ab/`.

Neste lab, o SUSY receberá sua submissão, mas *não* vai compilar nem testar seu programa. Ele vai servir apenas como sistema de submissão. Depois de terminado o prazo de submissão, seu trabalho será baixado e manualmente compilado e testado juntamente com projeto base semelhante ao que você recebeu para programar. Seu programa será testado com um limite definido de tempo, e este tempo deverá ser pequeno. Poderemos ou não adicionar outras instâncias de testes.

Portanto, você deve testar seu programa antes de submetê-lo, pois o SUSY não fará isto.

Também na mesma tarefa do SUSY, você deve submeter o pdf do relatório, como instruído a seguir.

Relatório. Você deverá entregar um relatório explicando as ideias do seu algoritmo em alto nível preferencialmente usando pseudo-código. Faça isto de modo que permita alguém com conhecimentos básicos de programação entender seu algoritmo, suas ideias e acompanhar o algoritmo na codificação em `C++`.

O texto também deverá conter testes computacionais sobre algumas entradas disponibilizadas ou que você tenha elaborado/obtido, mostrando o desempenho da sua implementação comparando, por exemplo, tempo e qualidade de solução. Sugere-se que estes experimentos mostrem tabelas e gráficos, bem como a configuração do computador usado para executar o programa. Seu relatório deve ser nomeado `tsp_bt_bnb_ra999999.pdf` e deverá ser submetido na mesma tarefa no SUSY.

Prazos. O Laboratório 1 deverá ser entregue até às 10h00 do dia 29 de setembro.

Bônus. Para valorizar o empenho e dedicação em cada abordagem, o programa que obtiver o melhor desempenho (independente do algoritmo) terá 1.5 pontos (resp. 1 ponto e 0.5 ponto) na nota do laboratório.

Observações.

- Baixe o novo projeto nomeado `proj01.zip` nos arquivos auxiliares da tarefa `lab01`.
- A visualização gráfica da solução está funcionando. Para isto, utilize a opção `-v` na execução do programa.
- Qualquer fraude resultará em média final zero para os envolvidos.
- Você deve usar seu usuário e senha da DAC para submeter sua tarefa no SUSY.
- Apenas com propósito de teste, foi criada a tarefa `lab00` em `https://susy.ic.unicamp.br:9999/mc658ab/`. Teste se seu usuário e senha estão funcionando corretamente. Se não estiverem, entre em contato.