MC658 - Projeto e Análise de Algoritmos III

Prof. Flávio Keidi Miyazawa
PED: Francisco Jhonatas Melo da Silva
Laboratório 3 - 1º Semestre de 2018

Algoritmo exato para o
TSP com Prioridades - TSP-P

TSP-P. São dados como entrada:

- Um grafo direcionado $D = (V, A)$;
- Nó $s \in V$ chamado de origem
- Prioridades p_v para cada $v \in V$ (por simplicidade, temos $p_s = 0$);
- Tempos t_a para todo arco $a \in A$;

O objetivo é encontrar um ciclo hamiltoniano C que começa em s, passa por todos os vértices exatamente uma vez e termina em s, tal que $p(C)$ é mínimo. A função $p(C)$ é definida como: $p(C) = \sum_{v \in V} p_v * t_C(s, v)$, onde $t_C(s, v)$ é a soma dos tempos nas arestas do caminho de s até v no ciclo C.

Técnicas. As técnicas *Heurísticas* e *Algoritmo Exato* apresentadas a seguir são, respectivamente, para serem usadas no lab 03 para resolver o mesmo problema: o TSP-P.

Heurísticas: Implementar três heurísticas diferentes para resolver o TSP-P:

- Uma heurística construtiva que seja rápida;
- Aplicar estratégias heurísticas baseadas em grafo de vizinhança e/ou busca local. Por exemplo, você poderá implementar uma heurística do tipo *multi-start local search*, busca tabu, *simulated annealing* ou GRASP; e
- BRKGA.

Algoritmo Exato: Deve-se formular **TSP-P** como um programa linear inteiro e elaborar um algoritmo de *branch and bound* (ou variação) que utiliza técnica de PLI do Gurobi. Como o algoritmo exato pode se beneficiar de boas soluções obtidas heuristicamente, é recomendado usar heurística boa e rápida para obter soluções iniciais que permitam obter uma solução ótima em menor tempo. Pode ser usado algoritmo da técnica *Heurísticas* que você já elaborou.

Dado tempo suficiente, seu algoritmo exato deve ser capaz de retornar solução ótima.

Implementação. O projeto deve ser feito na linguagem `C++`. Lhe será fornecido o `proj03` como base da implementação, e você deverá implementar as rotinas a seguir do arquivo fonte `tsp_p_algs.cpp`. Certifique-se de inserir seu RA no local indicado dentro desse arquivo. Além do arquivo `tsp_p_algs.cpp`, existem outros arquivos fontes, sendo que nestes encontram-se um programa principal, 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/`. O arquivo `README` do `proj03` tem, dentre outras, informações sobre o *Gurobi*: é importante que você o leia. Com exceção do nome, as rotinas tem o mesmo protótipo:

```
bool rotina(const Tsp_P_Instance &l, Tsp_P_Solution &s, int tl);
```

Note que os dados da instância de entrada do problema em `l` já terão sido lidos pelo projeto base. Sobre o retorno das funções:

- Se encontrou uma solução garantidamente ótima, retorna `true`;
- Se encontrou uma solução, mas não é garantidamente ótima, retorna `false`. Note que mesmo o algoritmo exato pode dar solução que não seja garantidamente ótima, pois pode ter parado por causa da **restrição do tempo**;
- Se não encontrou solução factível: retorna `false` e temos que o objeto `s` passado por referência terá `s.cost` = ∞. Note que `s` representa a solução.

E ainda, se encontrou alguma solução factível, a sequência de nós da rota deverá estar corretamente preenchida em `tour` de `s` (note que `tour` é um vetor elementos do tipo `DNode`). O primeiro nó não deve estar repetido no final. Por fim, a obtenção de sua solução deve obedecer o seguinte critério: sempre que necessário algum desempate, faça-o por ordem lexicográfica. Isto é importante no momento que formos executar testes para conferir sua solução.

Uma vez que você implemente as funções, você pode compilar seu projeto digitando `make` no diretório. Depois disso, você pode digitar `./runconstrheur`, `./runmetaheur`, `./runbrkga` ou `./runexact` no diretório para que os algoritmos correspondentes sejam executados usando como entrada as instâncias existentes no subdiretório `in`. Você também pode usar a versão `rungraph` no lugar da *string* `run`: aqui será mostrado mais informações de saída na tela em formato texto e também mostrará a solução graficamente.

As funções devem executar até o limite de tempo passado pelo parâmetro `tl`.

Depois que seu algoritmo constrói a solução, o projeto verifica se ela é uma solução factível válida, basicamente verificando se atende às definições do problema descrito. Além disso:

1. Verifica se o custo (`cost`) retornado é compatível com o `lowerBound` e o `upperBound`; e
2. Fará outras verificações que viermos a julgar necessárias.

O programa recebe um parâmetro na linha de comando:

`-c|-m|-b|-e:` opção exclusiva, que indica se será executado, respectivamente, *heurística construtiva*, *estratégias heurísticas baseadas em grafo de vizinhança e/ou busca local*, *BRKGA* ou *algoritmo exato*; e

`-t tempo_limite:` recebe o tempo limite que será passado para seu algoritmo.

Submissão e testes. Dentre os códigos-fonte, você *deve* submeter o arquivo `tsp_p_algs.cpp`, você *pode* submeter um `Makefile` alterado e *pode* submeter aqueles que você *acrescentou* ao seu projeto no SUSY.

Seu `Makefile` deverá funcionar sem erros no Ubuntu 16.04: não será aceito `Makefile` para outras plataformas. Assim, os arquivos do `proj03` que vocês podem alterar são `tsp_p_algs.cpp` (obrigatório) e `Makefile` (opcional). Os arquivos *acrescentados* (opcionais) *não* poderão sobrescrever arquivos do `proj03`. Além dos arquivos do `proj03` e de todos anteriormente citados, não deve ser necessário mais nenhum arquivo para seu projeto compilar e executar: se ainda houver algum requisito que não foi contemplado, entre em contato com PED para ver se é possível viabilizar sua solicitação.

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, 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 razoavelmente pequeno. Quando formos fazer os testes, poderemos adicionar novas instâncias de teste que não foram disponibilizadas. Portanto, você deve testar seu programa antes de submetê-lo, pois o SUSY não fará isto.

Relatório. Você *deve* entregar seu relatório utilizando o formato de artigo da SBC disponível em `http://www.sbc.org.br/documentos-da-sbc/send/169-templates-para-artigos-e-capitulos-de-livros/878-modelosparapublicaodeartigos`. Seu relatório *não deve* conter filiações detalhadas *nem abstract*. O relatório pode ter *no máximo* 12 páginas (incluindo referências).

Cada texto deve explicar as ideias do(s) respectivo(s) algoritmo(s) em alto nível, preferencialmente usando pseudo-código. Faça isto de modo que permita a alguém com conhecimentos básicos de programação entender seu algoritmo, suas ideias e acompanhar o algoritmo na codificação em `C++`.

Os textos também deverão relatar 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_p_ra999999.pdf` e deverá ser submetido no SUSY.

Prazos. O laboratório 03 deverá ser entregue até o prazo limite definido na tarefa correspondente do SUSY. Este prazo não será prorrogado por conta do prazo para indicarmos os alunos que porventura ficarem de exame.

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. O segundo melhor receberá 1.0 ponto e o terceiro melhor receberá 0.5 ponto.

Observações.

- Qualquer fraude resultará em média final zero para os envolvidos.
- Usuário e senha para submeter a tarefa no SUSY são os mesmos que você usou na tarefa anterior.
- Apenas com propósito de teste, foi criada a tarefa `lab0` em `https://susy.ic.unicamp.br:9999/mc658a/lab0`. Teste se seu usuário e senha estão funcionando corretamente. Se não estiverem, entre em contato.