Relatório lab01 - Backtracking e Branch-and-Bound
MC658

Fábio Takahashi Tanniguchi - RA 145980

1. Backtracking

A ideia da função bt implementada é de utilizar backtracking para obter o caminho de menor custo que resolvesse o problema do caixeiro-viajante.

O algoritmo, de forma simplificada, é:

- chama a função greedy que gera uma solução gulosa para o problema;
- partindo do caminho da solução gulosa, é feito o backtracking
    - para cada vértice retirado da solução gulosa (do fim para o começo):
        - explora adjacências do vértice anterior não contempladas na solução, que são exploradas até obter um caminho tal que os pesos totais possam ser comparados;
        - caso seja encontrada uma solução melhor, são atualizados tsp.BestCircuit e no tsp.BestCircuitValue
- caso tenha conseguido chegar até esta etapa sem estourar o tempo e sem nenhum problema, retorna true.

2. Branch and Bound

A ideia da função bnb implementada é de utilizar branch-and-bound para obter o caminho de menor custo que resolvesse o problema do caixeiro-viajante.

O algoritmo, de forma simplificada, é:

- inicia a formação da árvore de ramificações, percorrendo todas as possibilidades a partir da raiz de percursos passando por todos os vértices uma única vez, sem repetição de vértices;
- chama o método greedy que gera uma solução gulosa para o problema;
- a partir do valor obtido pela solução gulosa, a árvore de ramificações é percorrida

    - caso o algoritmo chegue num vértice e a soma dos pesos já dê maior do que o total da solução gulosa, o ramo é podado;
    - caso o algoritmo chegue ao final da ramificação e a soma dê maior do que o total da solução gulosa, o ramo é podado;
    - caso o algoritmo chegue ao final da ramificação e a soma seja menor do que o total da solução gulosa, então tsp.BestCircuit e tsp.BestCircuitValue são atualizados;
- como a tsp.BestCIrcuit e tsp.BestCircuitValue são atualizados durante a análise da árvore de ramificações, não há a necessidade de percorrer o ramo restante.

3. Testes e Problemas

Com os algoritmos implementados, o teste básico foi realizado utilizando a entrada gr_small_7 previamente fornecida.

O algoritmo de backtracking não rodou devidamente, resultando em “segmentation fault”. Rodando todas as entradas previamente fornecidas ocorreu o mesmo problema. O mesmo aconteceu com algumas entradas usando o algoritmo de branch-and-bound.

Após alguns ajustes nos códigos, os algoritmos passaram a resultar em soluções não válidas segundo o programa avaliador. Entretanto, por falta de tempo, não foi possível avaliar a causa. Há também um problema de loop infinito no branch-and-bound usando a entrada gr_berlin52.

Isso ocorreu pois foram gastas várias horas durante o período até o prazo inicial tentando configurar o ambiente na máquina pessoal e tentando entender o algoritmo guloso para familiarização com o LEMON.

Efetivamente, a implementação foi feita em dois dias. No primeiro, foram gastas 7 horas. No segundo, foram gastas 9 horas incluindo a elaboração deste relatório. A inexperiência com o LEMON e a impossibilidade de uso de recursão com métodos auxiliares foram complicadores relevantes.

Entretanto, com mais tempo, seria possível procurar as causas dos problemas encontrados na execução dos algoritmos.

Os testes foram realizados no laboratório 303 do IC-3, no Instituto de Computação da Universidade Estadual de Campinas (Unicamp). A máquina utilizada era dotada de processador Intel Core i5-4590 de 3,30GHz com 8GB de memória RAM, sistema operacional Fedora (versão 23).

Lamentando os problemas relacionados à distribuição pessoal de tempo encerra-se este relatório.