Department of Mathematics and Computer Science
University of Southern Denmark, Odense

September 11, 2008
Marco Chiarandini

DM811 - Heuristics for Combinatorial Optimization

Laboratory Assignment 2, Fall 2008

Exercise 1

We defined iterative improvement such that a neighbor is only accepted if it has strictly better cost. Give a reason to not accept a neighbor that has equal cost.

Exercise 2

The Steiner tree problem is a generalization of the minimum spanning tree problem in that it asks for a spanning tree covering the vertices of a set U. Extra intermediate vertices and edges may be added to the graph in order to reduce the length of the spanning tree. These new vertices introduced to decrease the total length of the connection are called Steiner[1] vertices.

Definition 1 Steiner Tree Problem [2]
Input: *A graph $G = (V, E)$, a weight function $\omega : E \mapsto \mathbf{N}$, and a subset $U \subseteq V$.*
Task: *Find a Steiner tree, that is, a subtree $T = (V_T, E_T)$ of G that includes all the vertices of U and such that the sum of the weights of the edges in the subtree is minimal.*

The example in Figure 1 is an instance of the Euclidean Steiner problem showing that the use of Steiner vertices may help to obtain cheaper subtrees including all vertices from U.

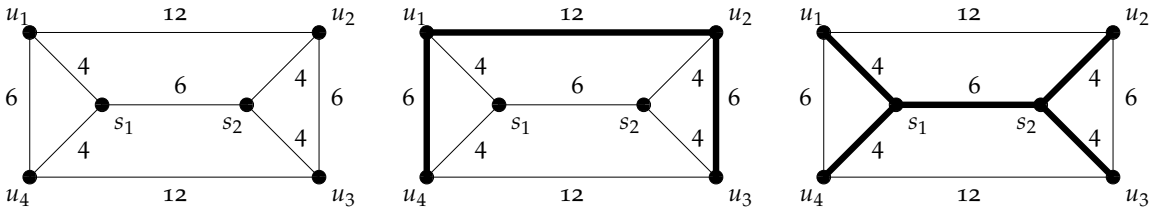

Figure 1: Vertices u_1, u_2, u_3, u_4 belong to the set U of special vertices to be covered and vertices s_1, s_2 belong to the set S of Steiner vertices. The Steiner tree in the second graph has cost 24 while the one in the third graph has cost 22.

1. Design one or more local search algorithms for the Steiner tree problem. In particular, define the solution representation and the neighborhood function.

[1] Jakob Steiner (18 March 1796 April 1, 1863) was a Swiss mathematician.

[2] It is recommendable to search information on the problems posed, above all about the proof of their hardness. However, to maximize the positive effect of the exercises, it should be preferable to search information after you understood the problem and answered the questions.

2. Provide an analysis of the computational cost of the basic operations in the local search algorithms designed at the previous point. In particular, consider the size of the neighborhood, and the cost of evaluating a neighbor.

Exercise 3

Definition 2 Graph Partitioning Problem
Input: *A graph $G = (V, E)$, weights $w(v) \in Z^+$ for each $v \in V$ and $l(e) \in Z^+$ for each $e \in E$.*
Task: *Find a partition of V into disjoint sets $V_1, V_2, \ldots, V_m$ such that $\sum_{v \in V_i} w(v) \leq K$ for $1 \leq i \leq m$ and such that if $E' \subseteq E$ is the set of edges that have their two endpoints in two different sets V_i, then $\sum_{e \in E'} l(e)$ is minimal.*

Consider the specific case of graph bipartitioning, that is, the case $|V| = 2n$ and $K = n$ and $w(v) = 1, \forall v \in V$.

1. Design a local search algorithm by defining the solution representation and the neighborhood function.
2. Determine the size of the search space, the size of the neighborhood and the computational cost of a step in the local search algorithm.
3. Show that, by maintaining appropriate data, it is possible to calculate the cost of a swap of vertices between the two partitions in constant time and that the update of the auxiliary data can also be made in constant time.

Exercise 4

Definition 3 Total Weighted Completion Time on Unrelated Parallel Machines Problem
Input: *A set of jobs J to be processed on a set of parallel machines M. Each job $j \in J$ has a weight w_j and processing time p_{ij} that depends on the machine $i \in M$ on which it is processed.*
Task: *Find a schedule of the jobs on the machines such that the sum of weighted completion time of the jobs is minimal.*

1. Gather information about the hardness of the problem.
2. Design a local search algorithm by defining the solution representation and the neighborhood function.

Exercise 5

Definition 4 Bin Packing Problem
Input: *A finite set U of items, a size $s(u) \in Z^+$ for each $u \in U$, and a positive integer bin capacity B.*
Task: *Find the minimal number of bins K for which there exits a partition of U into disjoint sets $U_1, U_2, \ldots, U_k$ and the sum of the sizes of the items in each U_i is B or less.*

Definition 5 Two-dimensional bin packing
Input: *A finite set U of rectangular items, each with a width $w_u \in Z^+$ and a height $h_u \in Z^+$, $u \in U$, and an unlimited number of identical rectangular bins of width $W \in Z^+$ and height $H \in Z^+$.*
Task: *Allocate all the items into a minimum number of bins, such that the bin widths and heights are not exceeded and the original orientation is respected (no rotation of the items is allowed).*

1. Compare the Bin packing problem to the Knapsack problem.
2. Design construction heuristics for both the two problems.
3. Design local search algorithms for the two problems focusing on solution representation and neighborhood function.