DM841 - Heuristics and Constraint Programming for Discrete Optimization

Contents

Solution

Included.

Part I

Exercises

Chapter 1

Local Search Modeling

Gather in groups of two or three and carry out one of the following exercises per group. Make sure that all groups work on a different problem by first chooses, first decides. It is all right if a problem remains without group. Use the first part of the work to make sure that you have understood the problem, make eventually a small example or a drawing. Then address the points listed in the exercise.
After 20 minutes of work, create new groups in such a way that all members come from different problems. In turn, each member has to present the problem and report the answer to the other members.

1.1

Definition 1 Single Machine Total Weighted Tardiness Problem
Input: *A set J of jobs $\{1, \ldots, n\}$ to be processed on a single machine and for each job $j \in J$ a processing time p_j, a weight w_j and a due date d_j.*

Task: *Find a schedule that minimizes the total weighted tardiness $\sum_{j=1}^{n} w_j \cdot T_j$, where $T_j = \max\{C_j - d_j, 0\}$ (C_j completion time of job j).*

1. Design a local search algorithm for the problem, ie, specify:
    - the variables and the solution representation
    - the constraints:
        - the implicit constraints
        - the invariants and the one-way cosntraints
        - the soft constraints
    - the evaluation function
    - the neighborhood(s)
    - the calculation of the delta function for the evaluation function given one of the neighborhoods defined
    - the implementation of the step function
    - the initial solution
    - the termination criterion
2. Classify the local search you designed above as one of those studied in class.
3. Provide a computational analysis, ie, give:
    - the neighborhood size

- the computational cost of evaluating a neighbor
- the total cost of a step in the local search designed

4. consider improvements in the efficiency (ie, in computational cost) of the operations for performing a first improvement or a best improvement local search with the neighborhood(s) you have put forward. Such improvements can be obtained by:

   A. fast incremental evaluation, ie, fast delta evaluation

   B. neighborhood pruning, ie, avoiding to examine moves that are certainly not leading to an improvement

   C. use of smart data structures

Solution

Point 1:

Variables and the solution representation We use linear permutations of jobs in J to define solutions: the permutation represents the order in which the jobs are processed on the machine. For a permutation π, π_i indicates the ith job to be processed. Hence, the search space has size $n!$. In CP, terms the variables are:

$$X_i \in J \quad \forall i = 1..n$$

and they represent the job in the sequence that is processed on the machine as the ith job. Hence, we must impose the constraint:

$$\text{alldiff}(X)$$

Implicit constraints The alldiff constraint is implicitly satisfied if we restrict the search to only permutations of the jobs J.

Invariants and one-way constraints In the objective function we will need to calculate the tardiness of each job, therefore we define the following invariants and one-way constaints:

- $Y_j \leftarrow \{i \mid X_i = j\} \quad \forall j \in J$, the position of the job j in the sequence.
- $C_j^X \leftarrow \sum_{i=1}^{Y_j} p_{X_i}$, the completion time of the job j given the sequence X
- $T_j^X \leftarrow \max\{(C_j - d_j), 0\}$, the tardiness of job j given the sequence X.

We used a left arrow to indicate that we defined invariants and one-way (right to left) constraints.

Soft constraints There are none, because all constraints (for example that only one job is processed at a time in the machine) are already satisfied by the solution representation chosen. Hence, there is no constraint to relax in the objective function.

Evaluation function Since there is no constraint to relax in the objective function, the evaluation function corresponds to the objective function and it is given by:

$$f(X) = \sum_{j=1}^{n} w_j \cdot T_j^X = \sum_{i=1}^{n} w_{X_i} \cdot T_{X_i}^X$$

where we use the index i to denote positions and j to denote jobs. For the calculation we use the invariants defined above.

Neighborhood(s) We can define the three standard neighborhoods for linear permutations: swap, interchange and insert. See the slides for a precise defintion.

Delta calculations For a solution X a neighboring solution X' obtained by a swap δ^i has the jobs in the positions i and $i+1$ exchanged. We can calculate $f(X')$ by calculating only the contribution to the objective function of the jobs in positions i and $i+1$. Thus, we seek Δ such that:

$$f(X') = f(X) + \Delta$$

Since the function f is separable we can caluclate Δ in constant time:

$$\begin{aligned} f(X') &= \sum_{j=1}^{n} w_j \cdot T_j^X \\ &= \sum_{j=1}^{n} w_j \cdot T_j^X - w_{X_i} T_{X_i}^X - w_{X_{i+1}} T_{X_{i+1}}^X + w_{X'_i} T_{X'_i}^{X'} + w_{X'_{i+1}} T_{X'_{i+1}}^{X'} = \\ &= f(X) - w_{X_i} T_{X_i}^{X'} - w_{X_{i+1}} T_{X_{i+1}}^{X'} + w_{X'_i} T_{X'_i}^{X'} + w_{X'_{i+1}} T_{X'_{i+1}}^{X'} \end{aligned}$$

and

$$\Delta = -w_{X_i} T_{X_i}^X - w_{X_{i+1}} T_{X_{i+1}}^X + w_{X'_i} T_{X'_i}^{X'} + w_{X'_{i+1}} T_{X'_{i+1}}^{X'}$$

The tardiness of jobs in positions i and $i+1$ of X' can be calculated by looking at the invariants Y_j, C_j, T_j. Let $j_1 = X_i = X'_{i+1}$ and $j_2 = X_{i+1} = X'_i$ then

$$\begin{aligned} C_{j_2}^{X'} &= \textstyle\sum_{l=1}^{i-1} w_{X_l} T_{X_l}^X + p_{j_2} = C_{X_{i-1}}^X + p_{j_2} \\ C_{j_1}^{X'} &= \textstyle\sum_{l=1}^{i-1} w_{X_l} T_{X_l}^X + p_{j_2} + p_{j_1} = C_{X_{i-1}}^X + p_{j_2} + p_{j_1} \end{aligned}$$

Hence, the changes to the invariants can be calculated in constant time and consequently the delta can also be calculated in constant time.

Implementation of the step function We can use with each neighborhood a first improvement pivot rule. Alternatively, we can implement a stochastic local search like variable neighborhood descent (see the slides for an algorithmic sketch).

Initial solution We can construct an initial solution by for example the heuristic: shortest weighted processing time. It generates a permutation of jobs sorted in decreasing order of the ration p_j/w_j.

Termination criterion We can let the algorithm terminate naturally when a local optimum in all neighborhoods is reached.

Point 2: varible neighborhood descent

Point 3:
As seen above, for the swap neighborhood:

- size: $O(n)$
- delta calculation: $O(1)$
- a first improvement implies that in the worst case the whole neighborhood has to be examined, hence $O(n) * O(1) = O(n)$

Insert neighborhood: TODO
Interchange neighborhood: TODO

Point 4: TODO

1.2

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 2 Steiner Tree Problem [2]
Input: *A graph $G = (V, E)$, a weight function $\omega : E \to \mathbb{N}$, 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 1.1 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. You may have a look also at this animation.

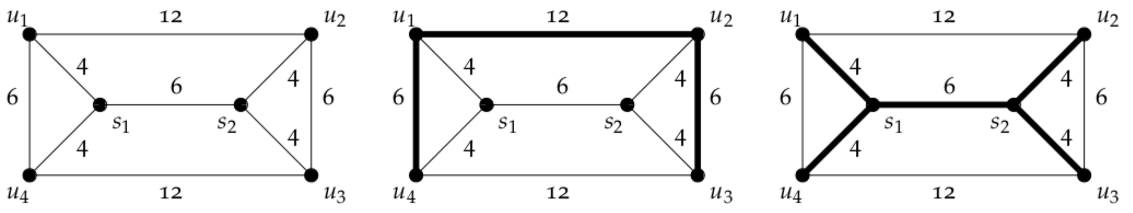


Figure 1.1: Vertices u_1, u_2, u_3, u_4 belong to the set U of special vertices to be covered and vertices s_1, s_2 belong to the set S of Steiner vertices. The Steiner tree in the second graph has cost 24 while the one in the third graph has cost 22.

Address for this problem the same points of exercise 1.1.

1.3

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.*

Address for this problem the same points of exercise 1.1.

1.4

Definition 4 p-Median Problem
Input: *A set U of locations for n* users, *a set F of locations for m* facilities *and a distance matrix $D = [d_{ij}] \in \mathbb{R}^{n\times m}$.*
Task: *Select a set $J \subseteq F$ of p locations where to install facilities such that the sum of the distances of each user to its closest installed facility is minimized, i.e.,*

$$\min\left\{\sum_{i\in U}\min_{j\in J} d_{ij} \;\middle|\; J \subseteq F \textit{ and } |J| = p\right\}$$

Address for this problem the same points of exercise 1.1.

[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.

1.5

Definition 5 Quadratic Assignment Problem
Input: *A set of n* locations *with a matrix $D = [d_{ij}] \in \mathbb{R}^{n\times n}$ of distances and a set of n* units *with a matrix $F = [f_{kl}] \in \mathbb{R}^{n\times n}$ of flows between them*

Task: *Find the assignment σ of units to locations that minimizes the sum of products between flows and distances, i.e.,*

$$\min_{\sigma\in\Sigma} \sum_{i,j} f_{ij} d_{\sigma(i)\sigma(j)}$$

Address for this problem the same points of exercise 1.1.

1.6

Definition 6 ***Job Shop Scheduling****. Given are m machines and a set of n jobs $J = \{1, 2, \ldots, n\}$, where each job $j \in J$ consists of a set $\{o_{1j}, o_{2j}, \ldots, o_{m_j j}\}$, of m_j operations. Furthermore, for each operation o_{ij} we are given a machine μ_{ij} on which it has to be processed and a processing requirement p_{ij}, where we have $\mu_{ij} \neq \mu_{i+1,j}$ for all $j \in J$ and $1 \le i \le m_j$. The problem is to find a schedule with minimum makespan. This means that we have to find a starting time σ_{ij} for each operation o_{ij}, such that for all $j \in J$ and $1 \le i \le m_j$ we have*

$$\sigma_{ij} + p_{ij} \le \sigma_{i+1,j}$$

such that for all $j, j' \in J$, $1 \le i \le m_j$, and $1 \le i' \le m_{j'}$ with $o_{ij} \neq o_{i'j'}$ we have

$$\mu_{ij} = \mu_{i'j'} \wedge \sigma_{ij} \le \sigma_{i'j'} \implies \sigma_{ij} + p_{ij} \le \sigma_{i'j'}$$

and such that

$$\max_{j\in J}(\sigma_{m_j,j} + p_{m_j,j})$$

is minimal.

Address for this problem the same points of exercise 1.1.

1.7

Definition 7 Traveling Salesman Problem
Input: *A graph $G = (V, E)$ and a cost function $\omega : V \times V \to \mathbb{R}$.*
Task: *Find an Hamiltonian cycle of minimum cost.*

Design construction heuristics for this problem.

1.8

Definition 8 (Maximum) K-Satisfiability (SAT)
Input: *A set U of variables, a collection C of disjunctive clauses of at most k literals, where a literal is a variable or a negated variable in U. k is a constant, $k \ge 2$.*
Task: *A truth assignment for U or an truth assignment that maximizes the number of clauses satisfied.*

1. show how the decision version of the graph coloring problem (GCP) can be encoded in a SAT problem
2. show how the constraint satisfaction problem (CSP) can be encoded in a SAT problem

3. are the results of the two previous points proves of the NP-completeness of the CSP and GCP?
4. devise preprocessing rules, ie, polynomial time simplification rules
5. design one or more construction heuristics for the problem

1.9

Definition 9 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 10 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. Discuss differences between the Bin packing problem and the Knapsack problem.
2. Design construction heuristics for the 1D and 2D Bin Packing problems
3. Design local search algorithms for the two problems focusing on solution representation and neighborhood function.
4. Consider the extension of the Bin Packing problem in which bins have variable size. More precisely, each bin in the set B is of a different type k and for each type we are given a cost c_k and a capacity W_k. We are asked to minimize the total cost of used bins. Modify the heuristics at the previous point to handle this version of the problem.

1.10

Asymmetric TSP into Symmetric TSP
How to encode an asymmetric TSP into a symmetric TSP?

1.11

Definition 11 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.
4. Design an (efficient!) variable depth local search algorithm that uses λ-exchanges where λ is not fixed a priori.

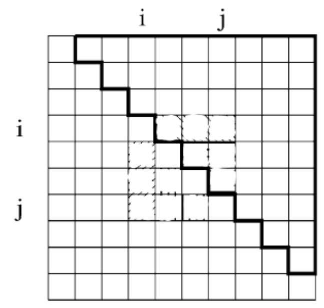
1.12

Definition 12 Linear Ordering Problem
The two following problems are equivalent.
Input*: an $n \times n$ matrix C*

Task*: Find a permutation π of the column and row indices $\{1, \ldots, n\}$ such that the value*

$$f(\pi) = \sum_{i=1}^{n} \sum_{j=i+1}^{n} c_{\pi_i \pi_j}$$

is maximized. In other terms, find a permutation of the columns and rows of C such that the elements in the upper triangle is maximized.



Definition 13 Feedback arc set problem (FASP)
Input*: A directed graph $D = (V, A)$, where $V = \{1, 2, \ldots, n\}$, and arc weights c_{ij} for all $[ij] \in A$*

Task*: Find a permutation $\pi_1, \pi_2, \ldots \pi_n$ of V (that is, a linear ordering of V) such that the total costs of those arcs $[\pi_j \pi_i]$ where $j > i$ (that is, the arcs that point backwards in the ordering)*

$$f(\pi) = \sum_{i=1}^{n} \sum_{j=i+1}^{n} c_{\pi_j \pi_i}$$

is minimized.

Design a simple construction heuristic and a simple local search algorithm.

1.13

Definition 14 Single Machine Total Weighted Tardiness Problem
Input: *A set J of jobs $\{1, \ldots, n\}$ to be processed on a single machine and for each job $j \in J$ a processing time p_j, a weight w_j and a due date d_j.*

Task: *Find a schedule that minimizes the total weighted tardiness $\sum_{j=1}^{n} w_j \cdot T_j$, where $T_j = \{C_j - d_j, 0\}$ (C_j completion time of job j).*

Is this problem NP-hard? If the problem is NP-hard, design at least one construction heuristic and at least one local search (solution representation + neighborhood + evaluation function).
Prepare for class discussion an answer to the following exercises. Work possibly in group. Due date: September 22.

1.14

In a 3-opt local search algorithm for the TSP how many possible ways are there to add three new edges once three edges have been removed in order to re-obtain an Hamiltonian tour? Justify your answer.

Solution
After the removal of three edges we are left with three segments of the tour. Let's call them ab, cd, ef. We must consider all possible ways to rearrange these segments in the original or in the reversed direction.This corresponds to calculate in how many different ways we can permute ab, cd, ef and for each permutation consider all possible directions of the segments. For each segment there are two directions which have to be multiplied for the directions of the other. Taking fixed ab we can calculate all ways to reinsert the others which are $2^2 \cdot 2! = 4 \cdot 2$. Thus, there are 8 ways to reobtain a tour. However, one of these ways will lead to the same exact tour. Other three will be two-exchanges rather than three exchanges. These are all those in which the second segment stays exactly where it is, hence two, plus the ones that have the edge de, that is one configuration $(ab - fe - dc)$.
In [1] page 535 there is the answer generalized to $k > 3$.

1.15

A possible application of local search to the GCP defines the following:

- solution representation: k-variable, complete improper coloring;
- neighborhood: one-exchange;
- evaluation function: $-\sum_{i=1}^{k} |C_i|^2 + \sum_{i=1}^{k} 2|C_i||E_i|$

Show that any local optimum of this function corresponds to a feasible coloring. Does a global optimum corresponds to a coloring that use the minimal possible number of colors?

1.16

In an iterative improvement algorithm for GCP that defines:

- solution representation: k-fixed, complete, improper coloring;
- neighborhood: one-exchange;
- evaluation function: number of edges causing a violation;

provide a computational analysis for the cost of examining the neighborhood and select the best neighbor. (Hint: it can be done in $O(nk)$. How?)

1.17

Make a Venn diagram (set diagram) representing the relation between the following classes of algorithms: construction heuristics (CH), metaheuristics (MH), local search (LS), stochastic local search (SLS), best improvement (BI), iterative improvement (II), first improvement (FI), uninformed random walk (RW), beam search (BS), rollout (RO), iterated greedy (IG), iterated local search (ILS).

1.18

Compare the pivot rules in tabu search and in simulated annealing. Think about computational issues and devise a guideline about the situations in which one or the other metaheuristic may be more appropriate.

1.19

Design a tabu search algorithm for the single machine total weighted tardiness problem using all three neighborhoods: insert, swap and interchange.

1.20

Definition 15 THE MAX WEIGHTED INDEPENDENT SET PROBLEM
Given: *an undirected graph $G(V, E)$ and a non-negative weight function ω on V $(\omega : V \to \mathbb{R})$*
Task: *A largest weight* independent set *(aka, stable set) of vertices, i.e., a subset $V' \subseteq V$ such that no two vertices in V' are joined by an edge in E.*

Related problems:

Definition 16 VERTEX COVER
Given: *an undirected graph $G(V, E)$ and a non-negative weight function ω on V $(\omega : V \to \mathbb{R})$*
Task: *A smallest weight* vertex cover*, i.e., a subset $V' \subseteq V$ such that each edge of G has at least one endpoint in V'.*

Definition 17 MAXIMUM CLIQUE
Given: *an undirected graph $G(V, E)$*
Task: *A maximum cardinality* clique*, i.e., a subset $V' \subseteq V$ such that every two vertices in V' are joined by an edge in E*

Note also that the independent set problem is equivalent to the *set packing* problem and the vertex cover problem is a strict special case of *set covering* problem.

1.21

Definition 18 SET COVERING
Input: *Collection C of subsets of a finite set S and a weight function $\omega : C \to \mathbf{R}$.*
Task: *Find a set cover for S, i.e., a subset $C' \subseteq C$ such that every element in S belongs to at least one member of C' and the sum of the costs associated with the subsets in C' is minimal.*

Is this problem NP-hard? If the problem is NP-hard, design at least one construction heuristic and at least one local search (solution representation + neighborhood + evaluation function).
Make a randomized version of the construction heuristic designed.

1.22

Definition 19 SET PROBLEMS

Set Covering

$$\begin{aligned} \min \quad & \sum_{j=1}^{n} c_j x_j \\ & \sum_{j=1}^{n} a_{ij} x_j \geq 1 \quad \forall i \\ & x_j \in \{0,1\} \end{aligned}$$

Set Partitioning

$$\begin{aligned} \min \quad & \sum_{j=1}^{n} c_j x_j \\ & \sum_{j=1}^{n} a_{ij} x_j = 1 \quad \forall i \\ & x_j \in \{0,1\} \end{aligned}$$

Set Packing

$$\begin{aligned} \max \quad & \sum_{j=1}^{n} c_j x_j \\ & \sum_{j=1}^{n} a_{ij} x_j \leq 1 \quad \forall i \\ & x_j \in \{0,1\} \end{aligned}$$

Design a simple construction heuristic and a simple local search algorithm for these problems.

1.23

Discuss how you would approach the following problem by local search: Dispose on a 8×8 table numbers from 1 to 256 in such a way that each number is placed only once and that the binary representation of the numbers have the minimum hamming distance in the neighborhood, that is, the largest distance from any of the numbers in the 8 neighboring cells.

Bibliography

[1] David L. Applegate, Robert E. Bixby, Vasek Chvátal, and William J. Cook. *The Traveling Salesman Problem: A Computational Study*. Princeton University Press, 2006.