


SAPIENZA
UNIVERSITÀ DI ROMA

# Heuristic Algorithms

## Worked exercises

Worked exercises on the chapters of the lecture notes, heuristic algorithms: each exercise is followed by its complete solution.

**Fabio Furini**

fabiofurini.github.io

Department of Computer, Control and Management Engineering — Sapienza University of Rome

Online version and downloadable material:
fabiofurini.github.io/heuristic-algorithms/
github.com/fabiofurini/heuristic-algorithms

Academic year 2026–2027

# Contents


**1 Heuristic algorithms: definitions** **2**

**2 Constructive heuristics** **10**

2.1 Knapsack: orderings of the items . . . 10
2.2 Bin packing: decreasing heuristics . . . 16
2.3 $P||C_{\max}$: LIST and LPT . . . 22
2.4 TSP: constructive heuristics . . . 26
2.5 Vertex coloring: greedy and DSATUR . . . 31

**3 Local search** **35**

3.1 Knapsack . . . 35
3.2 Bin packing . . . 44
3.3 TSP: 2-opt . . . 51
3.4 $P||C_{\max}$ . . . 59

**4 Metaheuristics** **68**

4.1 Simulated annealing . . . 68
4.2 Tabu search . . . 74
4.3 Iterated local search and VNS . . . 77
4.4 GRASP . . . 80
4.5 Large neighbourhood search . . . 84
4.6 Genetic algorithms . . . 87

**5 MIP heuristics for the classical problems** **90**

5.1 Rounding and diving . . . 90
5.2 RINS, local branching and proximity search . . . 92

Chapter 1

# Heuristic algorithms: definitions

**Exercise 1.1 — Gap and relative error (1)**

We consider a knapsack of capacity $c = 12$ and $n = 5$ items with the following profits and weights:

| | $j=1$ | $j=2$ | $j=3$ | $j=4$ | $j=5$ |
|---|---|---|---|---|---|
| $p_j$ | 20 | 18 | 14 | 9 | 6 |
| $w_j$ | 6 | 5 | 4 | 3 | 2 |

A heuristic algorithm returns the items $\{1, 2\}$.

**Questions**

1. Verify that the heuristic solution is feasible, and give its value $\mathscr{A}(I)$. Is it a primal or a dual bound, an upper or a lower bound?
2. Compute the dual bound $\mathrm{UB}(I)$ given by the LP relaxation, rounded down.
3. Compute the gap of the heuristic solution.
4. Find the optimal value $z(I)$ by enumeration, compute the relative error, and check that it is at most the gap.

## Solution

For a heuristic value $\mathscr{A}(I)$ and a dual bound, the gap is the difference of the two bounds divided by the lower one:

$$gap(I) = \frac{\mathscr{A}(I) - \mathrm{LB}(I)}{\mathrm{LB}(I)} \quad \text{(minimization)},$$
$$gap(I) = \frac{\mathrm{UB}(I) - \mathscr{A}(I)}{\mathscr{A}(I)} \quad \text{(maximization)},$$

and the relative error is $\frac{\mathscr{A}(I)-z(I)}{z(I)}$ (minimization) or $\frac{z(I)-\mathscr{A}(I)}{z(I)}$ (maximization). The relative error is at most the gap.

1. The weight is $6+5=11 \leq 12$, so the solution is feasible, with $\mathscr{A}(I) = 20+18 = 38$. The KP is a maximization problem: $\mathscr{A}(I)$ is a primal bound, and a lower bound on $z(I)$.

2. The efficiencies $\frac{p_j}{w_j}$ are $\frac{10}{3}$, $\frac{18}{5}$, $\frac{7}{2}$, 3, 3, so the items sorted by non-increasing efficiency are $(2,3,1,4,5)$. The relaxation packs the items 2, 3 (weight 9) and a fraction $\frac{1}{2}$ of item 1:
$$z(I) \leq 18 + 14 + 20 \cdot \frac{1}{2} = 42.$$
The profits are integer, so $\mathrm{UB}(I) = \lfloor 42 \rfloor = 42$.

3. $$gap(I) = \frac{\mathrm{UB}(I) - \mathscr{A}(I)}{\mathscr{A}(I)} = \frac{42-38}{38} = \frac{2}{19} \approx 10.5\%.$$

4. Enumerating the $2^5 = 32$ subsets gives $z(I) = 41$, attained by $\{2,3,4\}$ (weight 12). The relative error is
$$\frac{z(I) - \mathscr{A}(I)}{z(I)} = \frac{41-38}{41} = \frac{3}{41} \approx 7.3\% \ \leq \ 10.5\%.$$
The gap overestimates the error because $\mathrm{UB}(I) > z(I)$, and because it divides by $\mathscr{A}(I) < z(I)$.

## Exercise 1.2 — Gaps for minimization problems (1)

Consider the following three instances and heuristic solutions.

(a) BPP with $c = 10$ and weights $(6,5,4,4,3,3,2,2)$; heuristic packing patterns $\{1\}$, $\{2,3\}$, $\{4,5\}$, $\{6,7,8\}$.

(b) BPP with $c = 10$ and weights $(6,6,6,5,2)$; heuristic packing patterns $\{1\}$, $\{2\}$, $\{3\}$, $\{4\}$, $\{5\}$.

(c) $P\|C_{\max}$ with $m = 3$ and processing times $(7,6,5,4,3,3,2)$; heuristic schedule $J_1 = \{1,4\}$, $J_2 = \{2,5,6\}$, $J_3 = \{3,7\}$.

**Questions**

1. Verify that the three heuristic solutions are feasible, and give their values $\mathscr{A}(I)$.
2. Compute the dual bounds $\mathrm{LB}(I) = \left\lceil \frac{\sum_j w_j}{c} \right\rceil$ for the BPP and $\mathrm{LB}(I) = \max\left\{\max_j t_j, \left\lceil \frac{\sum_j t_j}{m} \right\rceil\right\}$ for $P\|C_{\max}$, and the gaps.
3. Compute the optimal values (Gurobi for the BPP, enumeration for $P\|C_{\max}$) and the relative errors. When are the gap and the error equal?

## Solution

For a heuristic value $\mathscr{A}(I)$ and a dual bound, the gap is the difference of the two bounds divided by the lower one:

$$gap(I) = \frac{\mathscr{A}(I) - \mathrm{LB}(I)}{\mathrm{LB}(I)} \quad \text{(minimization)},$$
$$gap(I) = \frac{\mathrm{UB}(I) - \mathscr{A}(I)}{\mathscr{A}(I)} \quad \text{(maximization)},$$

and the relative error is $\frac{\mathscr{A}(I)-z(I)}{z(I)}$ (minimization) or $\frac{z(I)-\mathscr{A}(I)}{z(I)}$ (maximization). The relative error is at most the gap.

1. (a) The pattern weights are 6, 9, 7, 7, all at most 10: $\mathscr{A}(I) = 4$.
   (b) Every item is alone in its bin: $\mathscr{A}(I) = 5$.
   (c) The loads are 11, 12, 7: $\mathscr{A}(I) = 12$.

   All three problems are minimization problems, so these values are upper bounds on $z(I)$.

2. (a) $\mathrm{LB}(I) = \lceil \frac{29}{10} \rceil = 3$ and $gap(I) = \frac{4-3}{3} \approx 33.3\%$.
   (b) $\mathrm{LB}(I) = \lceil \frac{5}{2} \rceil = 3$ and $gap(I) = \frac{5-3}{3} \approx 66.7\%$.
   (c) $\mathrm{LB}(I) = \max\{7, \lceil 30/3 \rceil\} = 10$ and $gap(I) = \frac{12-10}{10} = \frac{1}{5} \approx 20.0\%$.

3. (a) $z(I) = 3$, for instance with $\{1,3\}$, $\{2,5,7\}$, $\{4,6,8\}$; the error is $\frac{4-3}{3} \approx 33.3\%$.
   (b) $z(I) = 4$: the three items of weight 6 need three bins, and the item of weight 5 fits with none of them; the error is $\frac{5-4}{4} = \frac{1}{4} = 25.0\%$, much smaller than the gap.
   (c) $z(I) = 10$, for instance with $\{1,5\}$, $\{2,4\}$, $\{3,6,7\}$; the error is $\frac{1}{5} \approx 20.0\%$.

   The gap equals the relative error exactly when the dual bound equals $z(I)$, as in (a) and (c). In (b) the bound is weak, and the gap overestimates the error.

## Exercise 1.3 — A dual bound for the TSP (1)

We consider a TSP instance with $n = 5$ cities and the symmetric distances

| $d_{ij}$ | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | – | 4 | 10 | 2 | 8 |
| 2 | 4 | – | 11 | 2 | 5 |
| 3 | 10 | 11 | – | 4 | 2 |
| 4 | 2 | 2 | 4 | – | 2 |
| 5 | 8 | 5 | 2 | 2 | – |

Two heuristic algorithms return the tours $\sigma_1 = (1, 2, 3, 4, 5)$ and $\sigma_2 = (1, 4, 2, 5, 3)$.

**Questions**

1. Compute the lengths $L(\sigma_1)$ and $L(\sigma_2)$.
2. Compute the dual bound of the chapter based on the two smallest distances from each city.
3. Compute the gaps of the two tours.
4. Find the optimal tour by enumerating the distinct tours, and compute the relative errors. Compare them with the gaps.

## Solution

For a heuristic value $\mathscr{A}(I)$ and a dual bound, the gap is the difference of the two bounds divided by the lower one:

$$gap(I) = \frac{\mathscr{A}(I) - \mathrm{LB}(I)}{\mathrm{LB}(I)} \quad \text{(minimization)},$$
$$gap(I) = \frac{\mathrm{UB}(I) - \mathscr{A}(I)}{\mathscr{A}(I)} \quad \text{(maximization)},$$

and the relative error is $\frac{\mathscr{A}(I)-z(I)}{z(I)}$ (minimization) or $\frac{z(I)-\mathscr{A}(I)}{z(I)}$ (maximization). The relative error is at most the gap.

1. $L(\sigma_1) = 4 + 11 + 4 + 2 + 8 = 29$ and $L(\sigma_2) = 2 + 2 + 5 + 2 + 10 = 21$.
2. The two smallest distances from each city are $(2, 4)$ for city 1, $(2, 4)$ for city 2, $(2, 4)$ for city 3, $(2, 2)$ for city 4, $(2, 2)$ for city 5. Hence
$$\mathrm{LB}(I) = \left\lceil \frac{1}{2}(6 + 6 + 6 + 4 + 4) \right\rceil = \lceil 13 \rceil = 13.$$
3. $gap_1 = \frac{29-13}{13} = \frac{16}{13} \approx 123.1\%$ and $gap_2 = \frac{21-13}{13} = \frac{8}{13} \approx 61.5\%$.
4. There are $\frac{(n-1)!}{2} = 12$ distinct tours, and the shortest is $(1, 2, 5, 3, 4)$, of length $z(I) = 17$. The relative errors are $\frac{29-17}{17} \approx 70.6\%$ and $\frac{21-17}{17} \approx 23.5\%$, smaller than the gaps, since $\mathrm{LB}(I) = 13 < z(I) = 17$. Both measures rank $\sigma_2$ better than $\sigma_1$.

## Exercise 1.4 — Neighbourhoods and local optima (1)

We consider a knapsack of capacity $c = 11$ and $n = 5$ items:

| | $j = 1$ | $j = 2$ | $j = 3$ | $j = 4$ | $j = 5$ |
|---|---|---|---|---|---|
| $p_j$ | 12 | 10 | 9 | 7 | 4 |
| $w_j$ | 6 | 5 | 4 | 3 | 2 |

The solutions are represented by binary vectors $\boldsymbol{x} \in \{0,1\}^n$. The **flip** neighbourhood adds or removes one item; the **swap** neighbourhood removes one selected item and adds one unselected item. Only feasible neighbours count.

**Questions**

1. Give the size of the search space, and the number of its elements that are feasible solutions.
2. Consider the solution $\{1,3\}$. List its flip neighbours and its swap neighbours, and decide whether it is a local optimum for each neighbourhood.
3. Do the same for the solution $\{1,2\}$.
4. Find the global optimum by enumeration. Is either neighbourhood exact on this instance?

## Solution

1. The search space is $\{0,1\}^5$, with $2^5 = 32$ vectors; 20 of them respect the capacity. The others exceed it: the representation can describe infeasible objects.
2. The solution has weight 10 and profit 21.

   Flip neighbours:

| neighbour | move | weight | profit |
|---|---|---|---|
| $\{3\}$ | out 1 | 4 | 9 |
| $\{1,2,3\}$ | in 2 | 15 | — (infeasible) |
| $\{1\}$ | out 3 | 6 | 12 |
| $\{1,3,4\}$ | in 4 | 13 | — (infeasible) |
| $\{1,3,5\}$ | in 5 | 12 | — (infeasible) |

   No feasible flip neighbour has profit larger than 21: the solution is a local optimum for the flip neighbourhood.

   Swap neighbours:

| neighbour | move | weight | profit |
|---|---|---|---|
| $\{2,3\}$ | out 1, in 2 | 9 | 19 |
| $\{3,4\}$ | out 1, in 4 | 7 | 16 |
| $\{3,5\}$ | out 1, in 5 | 6 | 13 |
| $\{1,2\}$ | out 3, in 2 | 11 | 22 |
| $\{1,4\}$ | out 3, in 4 | 9 | 19 |
| $\{1,5\}$ | out 3, in 5 | 8 | 16 |

The neighbour $\{1,2\}$ has profit $22 > 21$: the solution is not a local optimum for the swap neighbourhood.

3. The solution has weight 11 and profit 22.

   Flip neighbours:

| neighbour | move | weight | profit |
|---|---|---|---|
| $\{2\}$ | out 1 | 5 | 10 |
| $\{1\}$ | out 2 | 6 | 12 |
| $\{1,2,3\}$ | in 3 | 15 | — (infeasible) |
| $\{1,2,4\}$ | in 4 | 14 | — (infeasible) |
| $\{1,2,5\}$ | in 5 | 13 | — (infeasible) |

   Swap neighbours:

| neighbour | move | weight | profit |
|---|---|---|---|
| $\{2,3\}$ | out 1, in 3 | 9 | 19 |
| $\{2,4\}$ | out 1, in 4 | 8 | 17 |
| $\{2,5\}$ | out 1, in 5 | 7 | 14 |
| $\{1,3\}$ | out 2, in 3 | 10 | 21 |
| $\{1,4\}$ | out 2, in 4 | 9 | 19 |
| $\{1,5\}$ | out 2, in 5 | 8 | 16 |

   No feasible neighbour is better: $\{1,2\}$ is a local optimum for both neighbourhoods.

4. The global optimum is $\{1,4,5\}$, of weight 11 and profit $z(I) = 23$. The solution $\{1,2\}$ is a local optimum for both neighbourhoods with profit $22 < 23$, so neither neighbourhood is exact: reaching the optimum from $\{1,2\}$ needs to remove one item and add two.

**Exercise 1.5 — Constructive heuristics, local search and metaheuristics (1)**

We consider $P||C_{\max}$ with $m = 2$ machines and $n = 6$ jobs:

| | $j=1$ | $j=2$ | $j=3$ | $j=4$ | $j=5$ | $j=6$ |
|---|---|---|---|---|---|---|
| $t_j$ | 8 | 4 | 4 | 6 | 3 | 7 |

A solution is represented by the vector $\boldsymbol{mach}$, where $mach_j$ is the machine of job $j$. Three procedures are proposed:

(P1) consider the jobs in the order $1, \dots, n$ and assign each job to the machine with the smallest current load (ties: smallest machine index); never change an assignment;

(P2) starting from a solution, move one job to the other machine, choosing the move with the smallest makespan (ties: smallest job index); repeat while the makespan strictly decreases;

(P3) repeat (P2) many times, each time from a solution obtained by assigning every job to a random machine, and keep the best solution found.

**Questions**

1. Give the size of the search space of this representation. Are all its elements feasible?
2. Classify the three procedures using the classes of the chapter.
3. Apply (P1), and then apply (P2) starting from its solution. Show the neighbours examined at each step.
4. Find the optimal value by enumeration. Is the solution returned by (P2) a global optimum? Would it be a local optimum for the neighbourhood that exchanges two jobs on different machines?
5. Explain the roles of intensification and diversification in (P2) and (P3).

**Solution**

1. The search space is $\{1,2\}^6$, with $m^n = 2^6 = 64$ vectors. Every vector is a feasible schedule, since the machines have no capacity.
2. (P1) is a constructive heuristic: it extends a partial solution one job at a time and never revises a choice. (P2) is a local search with the neighbourhood that moves one job. (P3) is a metaheuristic: it guides the local search beyond the first local optimum by restarting it from other solutions.
3. (P1) gives $\boldsymbol{mach} = (1,2,2,1,2,2)$, with loads $(14, 18)$ and makespan 18.

   **Iteration 1**

   The neighbours of the solution of (P1) are:

| move | loads | makespan |
|---|---|---|
| job 1 to machine 2 | $(6, 26)$ | 26 |
| job 2 to machine 1 | $(18, 14)$ | 18 |
| job 3 to machine 1 | $(18, 14)$ | 18 |
| job 4 to machine 2 | $(8, 24)$ | 24 |
| job 5 to machine 1 | $(17, 15)$ | 17 |
| job 6 to machine 1 | $(21, 11)$ | 21 |

The best one has makespan $17 < 18$: the new solution is $\boldsymbol{mach} = (1, 2, 2, 1, 1, 2)$.

**Iteration 2**

Its neighbours are:

| move | loads | makespan |
|---|---|---|
| job 1 to machine 2 | $(9, 23)$ | 23 |
| job 2 to machine 1 | $(21, 11)$ | 21 |
| job 3 to machine 1 | $(21, 11)$ | 21 |
| job 4 to machine 2 | $(11, 21)$ | 21 |
| job 5 to machine 2 | $(14, 18)$ | 18 |
| job 6 to machine 1 | $(24, 8)$ | 24 |

No neighbour has makespan smaller than 17: (P2) stops at a local optimum.

4. Enumeration gives $z(I) = 16$, for instance with $J_1 = \{1, 2, 3\}$ and $J_2 = \{4, 5, 6\}$. The local optimum of (P2), of makespan 17, is not a global optimum. Exchanging jobs 1 and 6 gives loads $(16, 16)$ and makespan 16: for the exchange neighbourhood the solution is not a local optimum. Being a local optimum depends on the neighbourhood.

5. (P2) is pure intensification: it moves downhill and stops at the bottom of the first valley it meets. The random starting solutions of (P3) are diversification: they bring the search to other regions of the search space, where another local search can find a better valley. Keeping the best solution makes the repeated runs useful.

# Constructive heuristics

## 2.1 Knapsack: orderings of the items

**Exercise 2.1 — Greedy orders for the KP 1**

We consider a knapsack of capacity $c = 9$ and a set $\mathscr{I} = \{1, \dots, 5\}$ of 5 items ($n = 5$), with the following profits and weights:

| | $j=1$ | $j=2$ | $j=3$ | $j=4$ | $j=5$ |
|---|---|---|---|---|---|
| $p_j$ | 4 | 15 | 11 | 8 | 2 |
| $w_j$ | 3 | 8 | 7 | 2 | 5 |

**Questions**

1. Sort the items by non-increasing profit, by non-decreasing weight and by non-increasing efficiency $\frac{p_j}{w_j}$ (ties by index).
2. Apply the greedy algorithm with each of the three orders ($GR_p$, $GR_w$, $GR$), showing the residual capacity $rc$ after each step, and give the three solutions and their profits.
3. Compute the upper bound UB given by the LP relaxation (items by non-increasing efficiency, the split item taken fractionally) and the gap $\frac{\lfloor \mathrm{UB} \rfloor - \mathscr{A}(I)}{\mathscr{A}(I)}$ of the best of the three solutions.
4. Knowing that the optimal value is $z(I) = 19$, say which orders find an optimal solution, and compute the relative error of the others. Explain what goes wrong.

**Solution**

1. The efficiencies are

$$\frac{p_1}{w_1} = \frac{4}{3} \approx 1.33, \quad \frac{p_2}{w_2} = \frac{15}{8} \approx 1.88, \quad \frac{p_3}{w_3} = \frac{11}{7} \approx 1.57,$$

$$\frac{p_4}{w_4} = \frac{8}{2} \approx 4.00, \quad \frac{p_5}{w_5} = \frac{2}{5} \approx 0.40$$

and the three orders are

| criterion | order |
|---|---|
| profit | $(2, 3, 4, 1, 5)$ |
| weight | $(4, 1, 5, 3, 2)$ |
| efficiency | $(4, 2, 3, 1, 5)$ |

2. $GR_p$, order $(2, 3, 4, 1, 5)$, starting from $rc = 9$:

   - item 2 ($w_2 = 8 \leq rc = 9$): selected, $rc \leftarrow 1$;
   - item 3 ($w_3 = 7 > rc = 1$): not selected;
   - item 4 ($w_4 = 2 > rc = 1$): not selected;
   - item 1 ($w_1 = 3 > rc = 1$): not selected;
   - item 5 ($w_5 = 5 > rc = 1$): not selected;

   Solution $\{2\}$, $GR_p(I) = 15$.

   $GR_w$, order $(4, 1, 5, 3, 2)$, starting from $rc = 9$:

   - item 4 ($w_4 = 2 \leq rc = 9$): selected, $rc \leftarrow 7$;
   - item 1 ($w_1 = 3 \leq rc = 7$): selected, $rc \leftarrow 4$;
   - item 5 ($w_5 = 5 > rc = 4$): not selected;
   - item 3 ($w_3 = 7 > rc = 4$): not selected;
   - item 2 ($w_2 = 8 > rc = 4$): not selected;

   Solution $\{1, 4\}$, $GR_w(I) = 4 + 8 = 12$.

   $GR$, order $(4, 2, 3, 1, 5)$, starting from $rc = 9$:

   - item 4 ($w_4 = 2 \leq rc = 9$): selected, $rc \leftarrow 7$;
   - item 2 ($w_2 = 8 > rc = 7$): not selected;
   - item 3 ($w_3 = 7 \leq rc = 7$): selected, $rc \leftarrow 0$;
   - item 1 ($w_1 = 3 > rc = 0$): not selected;
   - item 5 ($w_5 = 5 > rc = 0$): not selected;

   Solution $\{3, 4\}$, $GR(I) = 11 + 8 = 19$.

3. In the efficiency order $(4, 2, 3, 1, 5)$ the items $\{4\}$ fit, with total weight 2, and the split item is 2, with residual capacity $rc = 7$. Hence

$$\mathrm{UB} = 8 + 15 \cdot \frac{7}{8} = \frac{169}{8} \approx 21.12, \qquad \lfloor \mathrm{UB} \rfloor = 21.$$

   The best of the three solutions has profit 19, and

$$gap(I) = \frac{21 - 19}{19} \approx 10.5\%.$$

4. The optimal solution is $\{3, 4\}$, with $z(I) = 19$ (the bound $\lfloor \mathrm{UB} \rfloor = 21$ is not attained: here the gap overestimates the error). Only $GR$ finds it. $GR_p$ takes item 2, the most valuable but also the heaviest, which leaves room for nothing else of value: $GR_p(I) = 15$, relative error 21.1%. $GR_w$ fills the knapsack with the two lightest items, 4 and 1, which have little value: $GR_w(I) = 12$, relative error 36.8%. On this instance efficiency is the right criterion.

## Exercise 2.2 — Greedy orders for the KP 2

We consider a knapsack of capacity $c = 13$ and a set $\mathscr{I} = \{1, \ldots, 5\}$ of 5 items ($n = 5$), with the following profits and weights:

| | $j=1$ | $j=2$ | $j=3$ | $j=4$ | $j=5$ |
|---|---|---|---|---|---|
| $p_j$ | 9 | 4 | 8 | 6 | 7 |
| $w_j$ | 7 | 2 | 9 | 5 | 6 |

**Questions**

1. Sort the items by non-increasing profit, by non-decreasing weight and by non-increasing efficiency $\frac{p_j}{w_j}$ (ties by index).
2. Apply the greedy algorithm with each of the three orders ($GR_p$, $GR_w$, $GR$), showing the residual capacity $rc$ after each step, and give the three solutions and their profits.
3. Compute the upper bound UB given by the LP relaxation (items by non-increasing efficiency, the split item taken fractionally) and the gap $\frac{\lfloor \mathrm{UB} \rfloor - \mathscr{A}(I)}{\mathscr{A}(I)}$ of the best of the three solutions.
4. Is any of the three solutions optimal? Compute the relative error of the others, and explain why the order by efficiency fails on this instance.

## Solution

1. The efficiencies are

$$\frac{p_1}{w_1} = \frac{9}{7} \approx 1.29, \quad \frac{p_2}{w_2} = \frac{4}{2} \approx 2.00, \quad \frac{p_3}{w_3} = \frac{8}{9} \approx 0.89,$$
$$\frac{p_4}{w_4} = \frac{6}{5} \approx 1.20, \quad \frac{p_5}{w_5} = \frac{7}{6} \approx 1.17$$

and the three orders are

| criterion | order |
|---|---|
| profit | $(1, 3, 5, 4, 2)$ |
| weight | $(2, 4, 5, 1, 3)$ |
| efficiency | $(2, 1, 4, 5, 3)$ |

2. $GR_p$, order $(1, 3, 5, 4, 2)$, starting from $rc = 13$:

   - item 1 ($w_1 = 7 \leq rc = 13$): selected, $rc \leftarrow 6$;
   - item 3 ($w_3 = 9 > rc = 6$): not selected;
   - item 5 ($w_5 = 6 \leq rc = 6$): selected, $rc \leftarrow 0$;
   - item 4 ($w_4 = 5 > rc = 0$): not selected;
   - item 2 ($w_2 = 2 > rc = 0$): not selected;

   Solution $\{1, 5\}$, $GR_p(I) = 9 + 7 = 16$.

   $GR_w$, order $(2, 4, 5, 1, 3)$, starting from $rc = 13$:

   - item 2 ($w_2 = 2 \leq rc = 13$): selected, $rc \leftarrow 11$;
   - item 4 ($w_4 = 5 \leq rc = 11$): selected, $rc \leftarrow 6$;
   - item 5 ($w_5 = 6 \leq rc = 6$): selected, $rc \leftarrow 0$;
   - item 1 ($w_1 = 7 > rc = 0$): not selected;
   - item 3 ($w_3 = 9 > rc = 0$): not selected;

   Solution $\{2, 4, 5\}$, $GR_w(I) = 4 + 6 + 7 = 17$.

   $GR$, order $(2, 1, 4, 5, 3)$, starting from $rc = 13$:

   - item 2 ($w_2 = 2 \leq rc = 13$): selected, $rc \leftarrow 11$;
   - item 1 ($w_1 = 7 \leq rc = 11$): selected, $rc \leftarrow 4$;
   - item 4 ($w_4 = 5 > rc = 4$): not selected;
   - item 5 ($w_5 = 6 > rc = 4$): not selected;
   - item 3 ($w_3 = 9 > rc = 4$): not selected;

   Solution $\{1, 2\}$, $GR(I) = 9 + 4 = 13$.

3. In the efficiency order $(2, 1, 4, 5, 3)$ the items $\{2, 1\}$ fit, with total weight 9, and the split item is 4, with residual capacity $rc = 4$. Hence

$$\mathrm{UB} = 4 + 9 + 6 \cdot \frac{4}{5} = \frac{89}{5} \approx 17.80, \qquad \lfloor \mathrm{UB} \rfloor = 17.$$

   The best of the three solutions has profit 17, and

$$gap(I) = \frac{17 - 17}{17} = 0\%.$$

4. $GR_w(I) = 17 = \lfloor \mathrm{UB} \rfloor$, so the solution $\{2, 4, 5\}$ of $GR_w$ is optimal and $z(I) = 17$. $GR$ fails: after the two most efficient items 2 and 1 the residual capacity is $13 - 9 = 4$, too small for any other item, and $GR(I) = 13$, relative error 23.5%. $GR_p(I) = 16$ is also worse than $GR_w$, with relative error $\frac{17-16}{17} \approx 5.9\%$. Three light items fill the knapsack exactly $(2 + 5 + 6 = 13)$: here what matters is how well the items fill the capacity, which the efficiency ignores.

## Exercise 2.3 — Greedy orders for the KP 3

We consider a knapsack of capacity $c = 10$ and a set $\mathscr{I} = \{1, \ldots, 5\}$ of 5 items ($n = 5$), with the following profits and weights:

| | $j=1$ | $j=2$ | $j=3$ | $j=4$ | $j=5$ |
|---|---|---|---|---|---|
| $p_j$ | 8 | 13 | 5 | 12 | 7 |
| $w_j$ | 5 | 9 | 4 | 8 | 2 |

**Questions**

1. Sort the items by non-increasing profit, by non-decreasing weight and by non-increasing efficiency $\frac{p_j}{w_j}$ (ties by index).
2. Apply the greedy algorithm with each of the three orders ($GR_p$, $GR_w$, $GR$), showing the residual capacity $rc$ after each step, and give the three solutions and their profits.
3. Compute the upper bound UB given by the LP relaxation (items by non-increasing efficiency, the split item taken fractionally) and the gap $\frac{\lfloor \mathrm{UB} \rfloor - \mathscr{A}(I)}{\mathscr{A}(I)}$ of the best of the three solutions.
4. Find a feasible solution of profit $\lfloor \mathrm{UB} \rfloor$, and compute the relative error of each of the three greedy solutions. Which choice prevents the greedy by efficiency from finding it?

## Solution

1. The efficiencies are

$$\frac{p_1}{w_1} = \frac{8}{5} \approx 1.60, \quad \frac{p_2}{w_2} = \frac{13}{9} \approx 1.44, \quad \frac{p_3}{w_3} = \frac{5}{4} \approx 1.25,$$
$$\frac{p_4}{w_4} = \frac{12}{8} \approx 1.50, \quad \frac{p_5}{w_5} = \frac{7}{2} \approx 3.50$$

and the three orders are

| criterion | order |
|---|---|
| profit | $(2, 4, 1, 5, 3)$ |
| weight | $(5, 3, 1, 4, 2)$ |
| efficiency | $(5, 1, 4, 2, 3)$ |

2. $GR_p$, order $(2, 4, 1, 5, 3)$, starting from $rc = 10$:
   - item 2 ($w_2 = 9 \leq rc = 10$): selected, $rc \leftarrow 1$;
   - item 4 ($w_4 = 8 > rc = 1$): not selected;
   - item 1 ($w_1 = 5 > rc = 1$): not selected;

- item 5 ($w_5 = 2 > rc = 1$): not selected;
- item 3 ($w_3 = 4 > rc = 1$): not selected;

Solution $\{2\}$, $GR_p(I) = 13$.

$GR_w$, order $(5, 3, 1, 4, 2)$, starting from $rc = 10$:

- item 5 ($w_5 = 2 \leq rc = 10$): selected, $rc \leftarrow 8$;
- item 3 ($w_3 = 4 \leq rc = 8$): selected, $rc \leftarrow 4$;
- item 1 ($w_1 = 5 > rc = 4$): not selected;
- item 4 ($w_4 = 8 > rc = 4$): not selected;
- item 2 ($w_2 = 9 > rc = 4$): not selected;

Solution $\{3, 5\}$, $GR_w(I) = 5 + 7 = 12$.

$GR$, order $(5, 1, 4, 2, 3)$, starting from $rc = 10$:

- item 5 ($w_5 = 2 \leq rc = 10$): selected, $rc \leftarrow 8$;
- item 1 ($w_1 = 5 \leq rc = 8$): selected, $rc \leftarrow 3$;
- item 4 ($w_4 = 8 > rc = 3$): not selected;
- item 2 ($w_2 = 9 > rc = 3$): not selected;
- item 3 ($w_3 = 4 > rc = 3$): not selected;

Solution $\{1, 5\}$, $GR(I) = 8 + 7 = 15$.

3. In the efficiency order $(5, 1, 4, 2, 3)$ the items $\{5, 1\}$ fit, with total weight 7, and the split item is 4, with residual capacity $rc = 3$. Hence

$$\text{UB} = 7 + 8 + 12 \cdot \frac{3}{8} = \frac{39}{2} \approx 19.50, \qquad \lfloor \text{UB} \rfloor = 19.$$

The best of the three solutions has profit 15, and

$$gap(I) = \frac{19 - 15}{15} \approx 26.7\%.$$

4. Since $GR(I) = 15 < \lfloor \text{UB} \rfloor$, the bound does not certify optimality. Indeed the solution $\{4, 5\}$ has weight $8 + 2 = 10 \leq 10$ and profit $19 = \lfloor \text{UB} \rfloor$, so it is optimal, $z(I) = 19$. The relative errors are 31.6% for $GR_p$, 36.8% for $GR_w$ and 21.1% for $GR$: no order finds the optimum. The optimal solution combines the most efficient item 5 with item 4, which comes third in the efficiency order: after item 1 is selected, item 4 no longer fits, and the choice of item 1 is never revised.

## 2.2 Bin packing: decreasing heuristics

**Exercise 2.4 — Decreasing bin packing 1**

We consider bins of capacity $c = 10$ and a set $\mathscr{I} = \{1, \dots, 7\}$ of 7 items ($n = 7$), with the following weights:

| | $j=1$ | $j=2$ | $j=3$ | $j=4$ | $j=5$ | $j=6$ | $j=7$ |
|---|---|---|---|---|---|---|---|
| $w_j$ | 3 | 3 | 5 | 2 | 6 | 2 | 6 |

**Questions**

1. Apply First Fit (FF) to the items in the input order, step by step, and give the packing patterns.
2. Sort the items by non-increasing weight (ties by index) and apply First Fit Decreasing (FFD).
3. Apply Best Fit Decreasing (BFD), and compare its packing patterns with those of FFD.
4. Compute the lower bound $\left\lceil \frac{\sum_j w_j}{c} \right\rceil$. Which solutions are optimal? Explain why sorting helps here.

**Solution**

1. (a) Item 1 (weight 3): open bin 1, $rc[1] = 7$.
   (b) Item 2 (weight 3): first bin that fits ($rc[1] = 7$): bin 1, $rc[1] = 4$.
   (c) Item 3 (weight 5): fits into no bin ($rc[1] = 4$): open bin 2, $rc[2] = 5$.
   (d) Item 4 (weight 2): first bin that fits ($rc[1] = 4$, $rc[2] = 5$): bin 1, $rc[1] = 2$.
   (e) Item 5 (weight 6): fits into no bin ($rc[1] = 2$, $rc[2] = 5$): open bin 3, $rc[3] = 4$.
   (f) Item 6 (weight 2): first bin that fits ($rc[1] = 2$, $rc[2] = 5$, $rc[3] = 4$): bin 1, $rc[1] = 0$.
   (g) Item 7 (weight 6): fits into no bin ($rc[1] = 0$, $rc[2] = 5$, $rc[3] = 4$): open bin 4, $rc[4] = 4$.

   The packing patterns are

   $$P_1 = \{1, 2, 4, 6\}, \quad P_2 = \{3\}, \quad P_3 = \{5\}, \quad P_4 = \{7\}$$

   and $FF(I) = 4$.
2. The order is $\sigma = (5, 7, 3, 1, 2, 4, 6)$ (weights $6, 6, 5, 3, 3, 2, 2$).
   (a) Item 5 (weight 6): open bin 1, $rc[1] = 4$.

(b) Item 7 (weight 6): fits into no bin ($rc[1] = 4$): open bin 2, $rc[2] = 4$.

(c) Item 3 (weight 5): fits into no bin ($rc[1] = 4$, $rc[2] = 4$): open bin 3, $rc[3] = 5$.

(d) Item 1 (weight 3): first bin that fits ($rc[1] = 4$, $rc[2] = 4$, $rc[3] = 5$): bin 1, $rc[1] = 1$.

(e) Item 2 (weight 3): first bin that fits ($rc[1] = 1$, $rc[2] = 4$, $rc[3] = 5$): bin 2, $rc[2] = 1$.

(f) Item 4 (weight 2): first bin that fits ($rc[1] = 1$, $rc[2] = 1$, $rc[3] = 5$): bin 3, $rc[3] = 3$.

(g) Item 6 (weight 2): first bin that fits ($rc[1] = 1$, $rc[2] = 1$, $rc[3] = 3$): bin 3, $rc[3] = 1$.

The packing patterns are

$$P_1 = \{1, 5\}, \quad P_2 = \{2, 7\}, \quad P_3 = \{3, 4, 6\}$$

and $FFD(I) = 3$.

3. The order is $\sigma = (5, 7, 3, 1, 2, 4, 6)$ (weights $6, 6, 5, 3, 3, 2, 2$).

(a) Item 5 (weight 6): open bin 1, $rc[1] = 4$.

(b) Item 7 (weight 6): fits into no bin ($rc[1] = 4$): open bin 2, $rc[2] = 4$.

(c) Item 3 (weight 5): fits into no bin ($rc[1] = 4$, $rc[2] = 4$): open bin 3, $rc[3] = 5$.

(d) Item 1 (weight 3): fits into bins $1, 2, 3$; the fullest is bin 1 ($rc[1] = 4$, $rc[2] = 4$, $rc[3] = 5$): bin 1, $rc[1] = 1$.

(e) Item 2 (weight 3): fits into bins $2, 3$; the fullest is bin 2 ($rc[1] = 1$, $rc[2] = 4$, $rc[3] = 5$): bin 2, $rc[2] = 1$.

(f) Item 4 (weight 2): fits only into bin 3 ($rc[1] = 1$, $rc[2] = 1$, $rc[3] = 5$): bin 3, $rc[3] = 3$.

(g) Item 6 (weight 2): fits only into bin 3 ($rc[1] = 1$, $rc[2] = 1$, $rc[3] = 3$): bin 3, $rc[3] = 1$.

The packing patterns are

$$P_1 = \{1, 5\}, \quad P_2 = \{2, 7\}, \quad P_3 = \{3, 4, 6\}$$

and $BFD(I) = 3$.

4. The total weight is 27, so $LB = \lceil \frac{27}{10} \rceil = 3$. FFD and BFD use 3 bins and are therefore optimal, $z(I) = 3$, while FF uses 4. In the input order the four small items (1, 2, 4, 6, of weights $3, 3, 2, 2$) all end up together in bin 1, and the large items 3, 5, 7 get a bin each, with gaps 5, 4, 4 that no item is left to fill. Sorting packs the large items first, and the small items fill their gaps: the right order wins.

## Exercise 2.5 — Decreasing bin packing 2

We consider bins of capacity $c = 10$ and a set $\mathscr{I} = \{1, \dots, 8\}$ of 8 items ($n = 8$), with the following weights:

| | $j=1$ | $j=2$ | $j=3$ | $j=4$ | $j=5$ | $j=6$ | $j=7$ | $j=8$ |
|---|---|---|---|---|---|---|---|---|
| $w_j$ | 2 | 6 | 3 | 5 | 4 | 2 | 6 | 2 |

**Questions**

1. Apply First Fit (FF) to the items in the input order, step by step, and give the packing patterns.
2. Sort the items by non-increasing weight (ties by index) and apply First Fit Decreasing (FFD).
3. Apply Best Fit Decreasing (BFD), and compare its packing patterns with those of FFD.
4. Compute the lower bound $\left\lceil \frac{\sum_j w_j}{c} \right\rceil$. Which solutions are optimal? Explain why sorting does not help here.

## Solution

1. (a) Item 1 (weight 2): open bin 1, $rc[1] = 8$.
   (b) Item 2 (weight 6): first bin that fits ($rc[1] = 8$): bin 1, $rc[1] = 2$.
   (c) Item 3 (weight 3): fits into no bin ($rc[1] = 2$): open bin 2, $rc[2] = 7$.
   (d) Item 4 (weight 5): first bin that fits ($rc[1] = 2$, $rc[2] = 7$): bin 2, $rc[2] = 2$.
   (e) Item 5 (weight 4): fits into no bin ($rc[1] = 2$, $rc[2] = 2$): open bin 3, $rc[3] = 6$.
   (f) Item 6 (weight 2): first bin that fits ($rc[1] = 2$, $rc[2] = 2$, $rc[3] = 6$): bin 1, $rc[1] = 0$.
   (g) Item 7 (weight 6): first bin that fits ($rc[1] = 0$, $rc[2] = 2$, $rc[3] = 6$): bin 3, $rc[3] = 0$.
   (h) Item 8 (weight 2): first bin that fits ($rc[1] = 0$, $rc[2] = 2$, $rc[3] = 0$): bin 2, $rc[2] = 0$.

   The packing patterns are

   $$P_1 = \{1, 2, 6\}, \quad P_2 = \{3, 4, 8\}, \quad P_3 = \{5, 7\}$$

   and $FF(I) = 3$.

2. The order is $\sigma = (2, 7, 4, 5, 3, 1, 6, 8)$ (weights $6, 6, 5, 4, 3, 2, 2, 2$).
   (a) Item 2 (weight 6): open bin 1, $rc[1] = 4$.

(b) Item 7 (weight 6): fits into no bin ($rc[1] = 4$): open bin 2, $rc[2] = 4$.

(c) Item 4 (weight 5): fits into no bin ($rc[1] = 4$, $rc[2] = 4$): open bin 3, $rc[3] = 5$.

(d) Item 5 (weight 4): first bin that fits ($rc[1] = 4$, $rc[2] = 4$, $rc[3] = 5$): bin 1, $rc[1] = 0$.

(e) Item 3 (weight 3): first bin that fits ($rc[1] = 0$, $rc[2] = 4$, $rc[3] = 5$): bin 2, $rc[2] = 1$.

(f) Item 1 (weight 2): first bin that fits ($rc[1] = 0$, $rc[2] = 1$, $rc[3] = 5$): bin 3, $rc[3] = 3$.

(g) Item 6 (weight 2): first bin that fits ($rc[1] = 0$, $rc[2] = 1$, $rc[3] = 3$): bin 3, $rc[3] = 1$.

(h) Item 8 (weight 2): fits into no bin ($rc[1] = 0$, $rc[2] = 1$, $rc[3] = 1$): open bin 4, $rc[4] = 8$.

The packing patterns are

$$P_1 = \{2, 5\}, \quad P_2 = \{3, 7\}, \quad P_3 = \{1, 4, 6\}, \quad P_4 = \{8\}$$

and $FFD(I) = 4$.

3. The order is $\sigma = (2, 7, 4, 5, 3, 1, 6, 8)$ (weights $6, 6, 5, 4, 3, 2, 2, 2$).

(a) Item 2 (weight 6): open bin 1, $rc[1] = 4$.

(b) Item 7 (weight 6): fits into no bin ($rc[1] = 4$): open bin 2, $rc[2] = 4$.

(c) Item 4 (weight 5): fits into no bin ($rc[1] = 4$, $rc[2] = 4$): open bin 3, $rc[3] = 5$.

(d) Item 5 (weight 4): fits into bins $1, 2, 3$; the fullest is bin 1 ($rc[1] = 4$, $rc[2] = 4$, $rc[3] = 5$): bin 1, $rc[1] = 0$.

(e) Item 3 (weight 3): fits into bins $2, 3$; the fullest is bin 2 ($rc[1] = 0$, $rc[2] = 4$, $rc[3] = 5$): bin 2, $rc[2] = 1$.

(f) Item 1 (weight 2): fits only into bin 3 ($rc[1] = 0$, $rc[2] = 1$, $rc[3] = 5$): bin 3, $rc[3] = 3$.

(g) Item 6 (weight 2): fits only into bin 3 ($rc[1] = 0$, $rc[2] = 1$, $rc[3] = 3$): bin 3, $rc[3] = 1$.

(h) Item 8 (weight 2): fits into no bin ($rc[1] = 0$, $rc[2] = 1$, $rc[3] = 1$): open bin 4, $rc[4] = 8$.

The packing patterns are

$$P_1 = \{2, 5\}, \quad P_2 = \{3, 7\}, \quad P_3 = \{1, 4, 6\}, \quad P_4 = \{8\}$$

and $BFD(I) = 4$.

4. The total weight is 30, so $LB = \lceil \frac{30}{10} \rceil = 3$: FF, with 3 bins, is optimal, $z(I) = 3$, while FFD and BFD use 4 bins. In the sorted order the two items of weight 6 are paired with the items of weight 4 and 3, which leaves a gap of 1 in bin 2; bin 3 also ends with a gap of 1, and the last item of weight 2 fits nowhere. The input order happens to produce three full bins ($2+6+2$, $3+5+2$, $4+6$). A good guarantee in the worst case ($\frac{11}{9}$ for FFD) does not mean a better solution on every instance: here sorting hurts.

## Exercise 2.6 — Decreasing bin packing 3

We consider bins of capacity $c = 10$ and a set $\mathscr{I} = \{1, \dots, 7\}$ of 7 items ($n = 7$), with the following weights:

| | $j=1$ | $j=2$ | $j=3$ | $j=4$ | $j=5$ | $j=6$ | $j=7$ |
|---|---|---|---|---|---|---|---|
| $w_j$ | 4 | 4 | 5 | 7 | 2 | 2 | 4 |

**Questions**

1. Apply First Fit (FF) to the items in the input order, step by step, and give the packing patterns.
2. Sort the items by non-increasing weight (ties by index) and apply First Fit Decreasing (FFD).
3. Apply Best Fit Decreasing (BFD), and compare its packing patterns with those of FFD.
4. Compute the lower bound $\left\lceil \frac{\sum_j w_j}{c} \right\rceil$ and compare the four rules FF, BF (both in the input order), FFD and BFD.

## Solution

1. (a) Item 1 (weight 4): open bin 1, $rc[1] = 6$.
   (b) Item 2 (weight 4): first bin that fits ($rc[1] = 6$): bin 1, $rc[1] = 2$.
   (c) Item 3 (weight 5): fits into no bin ($rc[1] = 2$): open bin 2, $rc[2] = 5$.
   (d) Item 4 (weight 7): fits into no bin ($rc[1] = 2$, $rc[2] = 5$): open bin 3, $rc[3] = 3$.
   (e) Item 5 (weight 2): first bin that fits ($rc[1] = 2$, $rc[2] = 5$, $rc[3] = 3$): bin 1, $rc[1] = 0$.
   (f) Item 6 (weight 2): first bin that fits ($rc[1] = 0$, $rc[2] = 5$, $rc[3] = 3$): bin 2, $rc[2] = 3$.
   (g) Item 7 (weight 4): fits into no bin ($rc[1] = 0$, $rc[2] = 3$, $rc[3] = 3$): open bin 4, $rc[4] = 6$.

   The packing patterns are

   $$P_1 = \{1, 2, 5\}, \quad P_2 = \{3, 6\}, \quad P_3 = \{4\}, \quad P_4 = \{7\}$$

   and $FF(I) = 4$.

2. The order is $\sigma = (4, 3, 1, 2, 7, 5, 6)$ (weights $7, 5, 4, 4, 4, 2, 2$).
   (a) Item 4 (weight 7): open bin 1, $rc[1] = 3$.
   (b) Item 3 (weight 5): fits into no bin ($rc[1] = 3$): open bin 2, $rc[2] = 5$.

(c) Item 1 (weight 4): first bin that fits ($rc[1] = 3$, $rc[2] = 5$): bin 2, $rc[2] = 1$.

(d) Item 2 (weight 4): fits into no bin ($rc[1] = 3$, $rc[2] = 1$): open bin 3, $rc[3] = 6$.

(e) Item 7 (weight 4): first bin that fits ($rc[1] = 3$, $rc[2] = 1$, $rc[3] = 6$): bin 3, $rc[3] = 2$.

(f) Item 5 (weight 2): first bin that fits ($rc[1] = 3$, $rc[2] = 1$, $rc[3] = 2$): bin 1, $rc[1] = 1$.

(g) Item 6 (weight 2): first bin that fits ($rc[1] = 1$, $rc[2] = 1$, $rc[3] = 2$): bin 3, $rc[3] = 0$.

The packing patterns are

$$P_1 = \{4, 5\}, \quad P_2 = \{1, 3\}, \quad P_3 = \{2, 6, 7\}$$

and $FFD(I) = 3$.

3. The order is $\sigma = (4, 3, 1, 2, 7, 5, 6)$ (weights $7, 5, 4, 4, 4, 2, 2$).

(a) Item 4 (weight 7): open bin 1, $rc[1] = 3$.

(b) Item 3 (weight 5): fits into no bin ($rc[1] = 3$): open bin 2, $rc[2] = 5$.

(c) Item 1 (weight 4): fits only into bin 2 ($rc[1] = 3$, $rc[2] = 5$): bin 2, $rc[2] = 1$.

(d) Item 2 (weight 4): fits into no bin ($rc[1] = 3$, $rc[2] = 1$): open bin 3, $rc[3] = 6$.

(e) Item 7 (weight 4): fits only into bin 3 ($rc[1] = 3$, $rc[2] = 1$, $rc[3] = 6$): bin 3, $rc[3] = 2$.

(f) Item 5 (weight 2): fits into bins $1, 3$; the fullest is bin 3 ($rc[1] = 3$, $rc[2] = 1$, $rc[3] = 2$): bin 3, $rc[3] = 0$.

(g) Item 6 (weight 2): fits only into bin 1 ($rc[1] = 3$, $rc[2] = 1$, $rc[3] = 0$): bin 1, $rc[1] = 1$.

The packing patterns are

$$P_1 = \{4, 6\}, \quad P_2 = \{1, 3\}, \quad P_3 = \{2, 5, 7\}$$

and $BFD(I) = 3$.

4. The total weight is 28, so $LB = \lceil \frac{28}{10} \rceil = 3$, and FFD and BFD are both optimal, $z(I) = 3$, with different packing patterns: item 5, of weight 2, examined right after item 7, goes into bin 1 in FFD (the first bin that fits) and into bin 3 in BFD (the fullest one), and the two rules end with different, equally good, solutions. In the input order, FF needs 4 bins while BF (Best Fit without sorting) finds 3, with patterns $P_1 = \{1, 2, 5\}$, $P_2 = \{3, 7\}$, $P_3 = \{4, 6\}$. On this instance the choice of the bin matters for the unsorted rules, while after sorting both choices give the optimal number of bins.

## 2.3 $P\|C_{\max}$: LIST and LPT

**Exercise 2.7 — List scheduling and LPT 1**

We consider $m = 3$ identical machines and a set $\mathcal{J} = \{1, \dots, 8\}$ of 8 jobs ($n = 8$), with the following processing times:

| | $j=1$ | $j=2$ | $j=3$ | $j=4$ | $j=5$ | $j=6$ | $j=7$ | $j=8$ |
|---|---|---|---|---|---|---|---|---|
| $t_j$ | 2 | 8 | 7 | 6 | 8 | 3 | 4 | 9 |

**Questions**

1. Apply list scheduling ($LIST$) with the jobs in the input order, showing the loads before each assignment.
2. Apply $LPT$: sort the jobs by non-increasing processing time (ties by index) and apply list scheduling.
3. Compute the lower bound $\max\left\{\max_j t_j, \left\lceil \frac{\sum_j t_j}{m} \right\rceil\right\}$ and the gaps of the two solutions.
4. Is the $LPT$ solution optimal? Explain why the input order performs worse.

**Solution**

1.

| job $j$ | $t_j$ | loads before | machine |
|---|---|---|---|
| 1 | 2 | $(0,0,0)$ | 1 |
| 2 | 8 | $(2,0,0)$ | 2 |
| 3 | 7 | $(2,8,0)$ | 3 |
| 4 | 6 | $(2,8,7)$ | 1 |
| 5 | 8 | $(8,8,7)$ | 3 |
| 6 | 3 | $(8,8,15)$ | 1 |
| 7 | 4 | $(11,8,15)$ | 2 |
| 8 | 9 | $(11,12,15)$ | 1 |

The sets are $J_1 = \{1,4,6,8\}$, $J_2 = \{2,7\}$, $J_3 = \{3,5\}$, with loads $(20,12,15)$. Hence $LIST(I) = 20$.

2. The list is $\sigma = (8,2,5,3,4,7,6,1)$.

| job $j$ | $t_j$ | loads before | machine |
|---|---|---|---|
| 8 | 9 | $(0,0,0)$ | 1 |
| 2 | 8 | $(9,0,0)$ | 2 |
| 5 | 8 | $(9,8,0)$ | 3 |
| 3 | 7 | $(9,8,8)$ | 2 |
| 4 | 6 | $(9,15,8)$ | 3 |
| 7 | 4 | $(9,15,14)$ | 1 |
| 6 | 3 | $(13,15,14)$ | 1 |
| 1 | 2 | $(16,15,14)$ | 3 |

The sets are $J_1 = \{6,7,8\}$, $J_2 = \{2,3\}$, $J_3 = \{1,4,5\}$, with loads $(16,15,16)$. Hence $LPT(I) = 16$.

3. $\sum_j t_j = 47$ and $\max_j t_j = 9$, so

$$LB = \max\left\{9, \left\lceil \frac{47}{3} \right\rceil\right\} = 16,$$

and the gaps are $\frac{20-16}{16} = 25\%$ for $LIST$ and $\frac{16-16}{16} = 0\%$ for $LPT$.

4. $LPT(I) = 16 = LB$, so the $LPT$ solution is optimal and $z(I) = 16$; $LIST$ in the input order is 25.0% above the optimum. In the input order the longest job, job 8 of length 9, is the last one: it arrives when the least loaded machine already has load 11, and the makespan jumps to 20. With the longest jobs first, the last jobs are short and level the loads.

## Exercise 2.8 — List scheduling and LPT 2

We consider $m = 3$ identical machines and a set $\mathcal{J} = \{1, \ldots, 7\}$ of 7 jobs ($n = 7$), with the following processing times:

| | $j=1$ | $j=2$ | $j=3$ | $j=4$ | $j=5$ | $j=6$ | $j=7$ |
|---|---|---|---|---|---|---|---|
| $t_j$ | 3 | 6 | 5 | 4 | 4 | 9 | 7 |

**Questions**

1. Apply list scheduling ($LIST$) with the jobs in the input order, showing the loads before each assignment.
2. Apply $LPT$: sort the jobs by non-increasing processing time (ties by index) and apply list scheduling.
3. Compute the lower bound $\max\left\{\max_j t_j, \left\lceil \frac{\sum_j t_j}{m} \right\rceil\right\}$ and the gaps of the two solutions.
4. Find an assignment with makespan equal to the lower bound. Check Graham's bounds for $LIST$ and $LPT$ on this instance.

**Solution**

1.

| job $j$ | $t_j$ | loads before | machine |
|---|---|---|---|
| 1 | 3 | $(0,0,0)$ | 1 |
| 2 | 6 | $(3,0,0)$ | 2 |
| 3 | 5 | $(3,6,0)$ | 3 |
| 4 | 4 | $(3,6,5)$ | 1 |
| 5 | 4 | $(7,6,5)$ | 3 |
| 6 | 9 | $(7,6,9)$ | 2 |
| 7 | 7 | $(7,15,9)$ | 1 |

The sets are $J_1 = \{1,4,7\}$, $J_2 = \{2,6\}$, $J_3 = \{3,5\}$, with loads $(14,15,9)$. Hence $LIST(I) = 15$.

2. The list is $\sigma = (6,7,2,3,4,5,1)$.

| job $j$ | $t_j$ | loads before | machine |
|---|---|---|---|
| 6 | 9 | $(0,0,0)$ | 1 |
| 7 | 7 | $(9,0,0)$ | 2 |
| 2 | 6 | $(9,7,0)$ | 3 |
| 3 | 5 | $(9,7,6)$ | 3 |
| 4 | 4 | $(9,7,11)$ | 2 |
| 5 | 4 | $(9,11,11)$ | 1 |
| 1 | 3 | $(13,11,11)$ | 2 |

The sets are $J_1 = \{5,6\}$, $J_2 = \{1,4,7\}$, $J_3 = \{2,3\}$, with loads $(13,14,11)$. Hence $LPT(I) = 14$.

3. $\sum_j t_j = 38$ and $\max_j t_j = 9$, so

$$LB = \max\left\{9, \left\lceil \frac{38}{3} \right\rceil\right\} = 13,$$

and the gaps are $\frac{15-13}{13} \approx 15.4\%$ for $LIST$ and $\frac{14-13}{13} \approx 7.7\%$ for $LPT$.

4. The assignment $J_1 = \{1,2,4\}$, $J_2 = \{3,7\}$, $J_3 = \{5,6\}$ has loads $(13,12,13)$ and makespan $13 = LB$: it is optimal, $z(I) = 13$, and $LPT$ is not. Graham's bound gives $LPT(I) \le \left(\frac{4}{3} - \frac{1}{9}\right) z(I) = \frac{11}{9} \cdot 13 \approx 15.89$, and indeed $14 \le 15.89$; for $LIST$, $\left(2 - \frac{1}{3}\right) z(I) = \frac{5}{3} \cdot 13 \approx 21.67 \ge 15$. Sorting improves the makespan from 15 to 14, but does not reach the optimum.

**Exercise 2.9 — List scheduling and LPT 3**

We consider $m = 3$ identical machines and a set $\mathscr{J} = \{1,\dots,7\}$ of 7 jobs ($n = 7$), with the following processing times:

| | $j=1$ | $j=2$ | $j=3$ | $j=4$ | $j=5$ | $j=6$ | $j=7$ |
|---|---|---|---|---|---|---|---|
| $t_j$ | 5 | 8 | 8 | 4 | 5 | 7 | 4 |

**Questions**

1. Apply list scheduling ($LIST$) with the jobs in the input order, showing the loads before each assignment.
2. Apply $LPT$: sort the jobs by non-increasing processing time (ties by index) and apply list scheduling.
3. Compute the lower bound $\max\left\{\max_j t_j, \left\lceil \frac{\sum_j t_j}{m} \right\rceil\right\}$ and the gaps of the two solutions.
4. Compare the two solutions. Knowing that the optimal makespan is 15, which one is optimal? What does this say about sorting?

**Solution**

1.

| job $j$ | $t_j$ | loads before | machine |
|---|---|---|---|
| 1 | 5 | $(0,0,0)$ | 1 |
| 2 | 8 | $(5,0,0)$ | 2 |
| 3 | 8 | $(5,8,0)$ | 3 |
| 4 | 4 | $(5,8,8)$ | 1 |
| 5 | 5 | $(9,8,8)$ | 2 |
| 6 | 7 | $(9,13,8)$ | 3 |
| 7 | 4 | $(9,13,15)$ | 1 |

The sets are $J_1=\{1,4,7\}$, $J_2=\{2,5\}$, $J_3=\{3,6\}$, with loads $(13,13,15)$. Hence $LIST(I)=15$.

2. The list is $\sigma=(2,3,6,1,5,4,7)$.

| job $j$ | $t_j$ | loads before | machine |
|---|---|---|---|
| 2 | 8 | $(0,0,0)$ | 1 |
| 3 | 8 | $(8,0,0)$ | 2 |
| 6 | 7 | $(8,8,0)$ | 3 |
| 1 | 5 | $(8,8,7)$ | 3 |
| 5 | 5 | $(8,8,12)$ | 1 |
| 4 | 4 | $(13,8,12)$ | 2 |
| 7 | 4 | $(13,12,12)$ | 2 |

The sets are $J_1=\{2,5\}$, $J_2=\{3,4,7\}$, $J_3=\{1,6\}$, with loads $(13,16,12)$. Hence $LPT(I)=16$.

3. $\sum_j t_j = 41$ and $\max_j t_j = 8$, so

$$LB = \max\left\{8, \left\lceil \frac{41}{3} \right\rceil\right\} = 14,$$

and the gaps are $\frac{15-14}{14} \approx 7.1\%$ for $LIST$ and $\frac{16-14}{14} \approx 14.3\%$ for $LPT$.

4. Here $LIST(I) = 15 < LPT(I) = 16$: on this instance the input order is better than the sorted one. The lower bound 14 is not attained; a complete enumeration of the $3^7$ assignments shows that $z(I) = 15$ (for instance $J_1 = \{1, 2\}$, $J_2 = \{3, 5\}$, $J_3 = \{4, 6, 7\}$, loads $(13, 13, 15)$), so $LIST$ in the input order is optimal, while $LPT$ is 6.7% above the optimum. Sorting is a rule that works well in general, and Graham's bound for $LPT$ is better than the one for $LIST$; but a guarantee is a statement on the worst case, and on a single instance an arbitrary order can be lucky.

## 2.4 TSP: constructive heuristics

**Exercise 2.10 — TSP construction 1**

We consider a set $\mathcal{V} = \{1, \ldots, 6\}$ of 6 cities with coordinates 1: $(8, 0)$, 2: $(1, 0)$, 3: $(1, 2)$, 4: $(1, 5)$, 5: $(6, 3)$, 6: $(0, 4)$ and distances equal to the Euclidean distances rounded to the nearest integer (the instance is metric):

| $d_{ij}$ | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| 1 | — | 7 | 7 | 9 | 4 | 9 |
| 2 | 7 | — | 2 | 5 | 6 | 4 |
| 3 | 7 | 2 | — | 3 | 5 | 2 |
| 4 | 9 | 5 | 3 | — | 5 | 1 |
| 5 | 4 | 6 | 5 | 5 | — | 6 |
| 6 | 9 | 4 | 2 | 1 | 6 | — |

**Questions**

1. Apply the nearest neighbour heuristic ($NN$) from city 1, step by step (ties by index).
2. Apply $NN$ from each of the other cities, and give the result of the multi-start $NN$.
3. Compute the lower bound $\left\lceil \frac{1}{2} \sum_i (a_i + b_i) \right\rceil$, where $a_i$ and $b_i$ are the two smallest distances from city $i$, and the gap of the multi-start solution.
4. Knowing that the optimal value is $z(I) = 21$, compute the relative error of $NN$ from city 1, and explain why its tour is long.

**Solution**

1. (a) from city 1 the distances to $2, 3, 4, 5, 6$ are $7, 7, 9, 4, 9$: city 5;
   (b) from city 5 the distances to $2, 3, 4, 6$ are $6, 5, 5, 6$: city 3 (tie with 4, broken by index);
   (c) from city 3 the distances to $2, 4, 6$ are $2, 3, 2$: city 2 (tie with 6, broken by index);
   (d) from city 2 the distances to $4, 6$ are $5, 4$: city 6;
   (e) from city 6 only city 4 is left, at distance 1;
   (f) the tour closes with the edge $(4, 1)$, of length 9;

   Hence $\pi = (1, 5, 3, 2, 6, 4)$ and $NN(I) = 4 + 5 + 2 + 4 + 1 + 9 = 25$.

2. Proceeding in the same way from every city:

| start | tour | $L$ |
|---|---|---|
| 1 | $(1, 5, 3, 2, 6, 4)$ | 25 |
| 2 | $(2, 3, 6, 4, 5, 1)$ | 21 |
| 3 | $(3, 2, 6, 4, 5, 1)$ | 23 |
| 4 | $(4, 6, 3, 2, 5, 1)$ | 24 |
| 5 | $(5, 1, 2, 3, 6, 4)$ | 21 |
| 6 | $(6, 4, 3, 2, 5, 1)$ | 25 |

   The multi-start $NN$ returns the first shortest tour, $(2, 3, 6, 4, 5, 1)$, of length 21.

3. The two smallest distances are $(4, 7)$ for city 1, $(2, 4)$ for city 2, $(2, 2)$ for city 3, $(1, 3)$ for city 4, $(4, 5)$ for city 5, $(1, 2)$ for city 6, so

$$LB = \left\lceil \frac{11 + 6 + 4 + 4 + 9 + 3}{2} \right\rceil = \lceil 18.5 \rceil = 19,$$

   and the gap of the multi-start solution is $\frac{21-19}{19} \approx 10.5\%$.

4. The relative error of $NN$ from city 1 is $\frac{25-21}{21} \approx 19.0\%$, while the multi-start finds the optimal value 21 (from cities 2 and 5). From city 1 the nearest city is 5, and the path then moves to the cluster of cities $2, 3, 4, 6$; city 1 is far from all of them (its distances to $2, 3, 4, 6$ are $7, 7, 9, 9$), and the path must come back from the last city of the cluster: the closing edge $(4, 1)$ has length 9. Changing the starting city changes which long edge is forced at the end.

**Exercise 2.11 — TSP construction 2**

We consider a set $\mathcal{V} = \{1, \ldots, 6\}$ of 6 cities with coordinates 1: $(4, 3)$, 2: $(6, 1)$, 3: $(0, 5)$, 4: $(0, 1)$, 5: $(4, 5)$, 6: $(4, 0)$ and distances equal to the Euclidean distances rounded to the nearest integer (the instance is metric):

| $d_{ij}$ | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| 1 | — | 3 | 4 | 4 | 2 | 3 |
| 2 | 3 | — | 7 | 6 | 4 | 2 |
| 3 | 4 | 7 | — | 4 | 4 | 6 |
| 4 | 4 | 6 | 4 | — | 6 | 4 |
| 5 | 2 | 4 | 4 | 6 | — | 5 |
| 6 | 3 | 2 | 6 | 4 | 5 | — |

**Questions**

1. Apply nearest insertion ($NI$) from city 1: at each step give $d(q, T)$ for the cities $q$ not in the subtour $T$, the selected city and its cheapest edge.
2. Apply farthest insertion ($FI$) from city 1.
3. Apply cheapest insertion ($CI$) from city 1.
4. Knowing that the optimal value is $z(I) = 19$, compare the three tours with each other and with the tour of $NN$ from city 1.

**Solution**

1. Starting from $T = (1)$ (inserting city $q$ into the subtour $(1)$ costs $2d_{1q}$):

| subtour $T$ | $d(q, T)$ | insertion costs of the selected city | inserted |
|---|---|---|---|
| $(1)$ | 2: 3, 3: 4, 4: 4, 5: 2, 6: 3 | $(4)$ | 5 on $(1, 1)$, cost 4 |
| $(1, 5)$ | 2: 3, 3: 4, 4: 4, 6: 3 | $(5, 5)$ | 2 on $(1, 5)$, cost 5 |
| $(1, 2, 5)$ | 3: 4, 4: 4, 6: 2 | $(2, 3, 6)$ | 6 on $(1, 2)$, cost 2 |
| $(1, 6, 2, 5)$ | 3: 4, 4: 4 | $(7, 11, 7, 6)$ | 3 on $(5, 1)$, cost 6 |
| $(1, 6, 2, 5, 3)$ | 4: 4 | $(5, 8, 8, 6, 4)$ | 4 on $(3, 1)$, cost 4 |

   Hence $\pi = (1, 6, 2, 5, 3, 4)$ and $NI(I) = 3 + 2 + 4 + 4 + 4 + 4 = 21$.

2. Starting from $T = (1)$ (inserting city $q$ into the subtour $(1)$ costs $2d_{1q}$):

| subtour $T$ | $d(q, T)$ | insertion costs of the selected city | inserted |
|---|---|---|---|
| $(1)$ | 2: 3, 3: 4, 4: 4, 5: 2, 6: 3 | $(8)$ | 3 on $(1, 1)$, cost 8 |
| $(1, 3)$ | 2: 3, 4: 4, 5: 2, 6: 3 | $(4, 4)$ | 4 on $(1, 3)$, cost 4 |
| $(1, 4, 3)$ | 2: 3, 5: 2, 6: 3 | $(5, 9, 6)$ | 2 on $(1, 4)$, cost 5 |
| $(1, 2, 4, 3)$ | 5: 2, 6: 2 | $(3, 4, 6, 2)$ | 5 on $(3, 1)$, cost 2 |
| $(1, 2, 4, 3, 5)$ | 6: 2 | $(2, 0, 6, 7, 6)$ | 6 on $(2, 4)$, cost 0 |

Hence $\pi = (1, 2, 6, 4, 3, 5)$ and $FI(I) = 3 + 2 + 4 + 4 + 4 + 2 = 19$.

3. Starting from $T = (1)$ (inserting city $q$ into the subtour $(1)$ costs $2d_{1q}$):

| subtour $T$ | minimum insertion cost | insertion costs of the selected city | inserted |
|---|---|---|---|
| $(1)$ | 2: 6, 3: 8, 4: 8, 5: 4, 6: 6 | $(4)$ | 5 on $(1, 1)$, cost 4 |
| $(1, 5)$ | 2: 5, 3: 6, 4: 8, 6: 6 | $(5, 5)$ | 2 on $(1, 5)$, cost 5 |
| $(1, 2, 5)$ | 3: 6, 4: 7, 6: 2 | $(2, 3, 6)$ | 6 on $(1, 2)$, cost 2 |
| $(1, 6, 2, 5)$ | 3: 6, 4: 5 | $(5, 8, 8, 8)$ | 4 on $(1, 6)$, cost 5 |
| $(1, 4, 6, 2, 5)$ | 3: 4 | $(4, 6, 11, 7, 6)$ | 3 on $(1, 4)$, cost 4 |

Hence $\pi = (1, 3, 4, 6, 2, 5)$ and $CI(I) = 4 + 4 + 4 + 2 + 4 + 2 = 20$.

4. The lengths are $NI(I) = 21$, $FI(I) = 19$, $CI(I) = 20$ and $NN(I) = 20$ from city 1. $FI$ finds an optimal tour, $z(I) = 19$; the relative errors are 10.5% for $NI$, 5.3% for $CI$ and 5.3% for $NN$. $FI$ first builds the outline of the tour through cities 3 and 4, the farthest from city 1, and then inserts cities 2, 5 and 6 where they cost little (city 6 at cost 0); $NI$ and $CI$ start from cities 5, 2 and 6, close to city 1, and must insert the far cities 3 and 4 at the end, when the subtour no longer has a cheap place for them.

## Exercise 2.12 — TSP construction 3

We consider a set $\mathcal{V} = \{1, \dots, 5\}$ of 5 cities with coordinates 1: $(8, 1)$, 2: $(5, 0)$, 3: $(6, 4)$, 4: $(6, 3)$, 5: $(3, 5)$ and distances equal to the Euclidean distances rounded to the nearest integer (the instance is metric):

| $d_{ij}$ | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | — | 3 | 4 | 3 | 6 |
| 2 | 3 | — | 4 | 3 | 5 |
| 3 | 4 | 4 | — | 1 | 3 |
| 4 | 3 | 3 | 1 | — | 4 |
| 5 | 6 | 5 | 3 | 4 | — |

**Questions**

1. Apply the nearest neighbour heuristic ($NN$) from city 1.
2. Apply the randomized nearest neighbour $RNN_\alpha$ from city 1 with $\alpha = \frac{1}{2}$, using the random numbers $u_1 = 0.0159$, $u_2 = 0.4956$, $u_3 = 0.8564$ in this order. The RCL lists its cities by increasing index, the city in position $\lfloor u \cdot |RCL| \rfloor + 1$ is chosen, and a random number is used only when $|RCL| \geq 2$. Give the tour of the first run.

3. Continue with the remaining random numbers for a second run of $RNN_\alpha$ from city 1.

4. Knowing that the optimal value is $z(I) = 15$, compare the three tours. What happens to the RCL with $\alpha = 0$ and with $\alpha = 1$?

**Solution**

1. (a) from city 1 the distances to $2, 3, 4, 5$ are $3, 4, 3, 6$: city 2 (tie with 4, broken by index);
   (b) from city 2 the distances to $3, 4, 5$ are $4, 3, 5$: city 4;
   (c) from city 4 the distances to $3, 5$ are $1, 4$: city 3;
   (d) from city 3 only city 5 is left, at distance 3;
   (e) the tour closes with the edge $(5, 1)$, of length 6;

   Hence $\pi = (1, 2, 4, 3, 5)$ and $NN(I) = 3 + 3 + 1 + 3 + 6 = 16$.

2. (a) from city 1 the distances to $2, 3, 4, 5$ are $3, 4, 3, 6$; threshold $3 + \frac{1}{2} \cdot (6 - 3) = 4.5$, $RCL = \{2, 3, 4\}$; $u_1 = 0.0159$, position $\lfloor 0.0159 \cdot 3 \rfloor + 1 = 1$: city 2;
   (b) from city 2 the distances to $3, 4, 5$ are $4, 3, 5$; threshold $3 + \frac{1}{2} \cdot (5 - 3) = 4$, $RCL = \{3, 4\}$; $u_2 = 0.4956$, position $\lfloor 0.4956 \cdot 2 \rfloor + 1 = 1$: city 3;
   (c) from city 3 the distances to $4, 5$ are $1, 3$; threshold $1 + \frac{1}{2} \cdot (3 - 1) = 2$, $RCL = \{4\}$: city 4;
   (d) from city 4 only city 5 is left;

   Hence $\pi = (1, 2, 3, 4, 5)$ and $L = 3 + 4 + 1 + 4 + 6 = 18$.

3. (a) from city 1 the distances to $2, 3, 4, 5$ are $3, 4, 3, 6$; threshold $3 + \frac{1}{2} \cdot (6 - 3) = 4.5$, $RCL = \{2, 3, 4\}$; $u_3 = 0.8564$, position $\lfloor 0.8564 \cdot 3 \rfloor + 1 = 3$: city 4;
   (b) from city 4 the distances to $2, 3, 5$ are $3, 1, 4$; threshold $1 + \frac{1}{2} \cdot (4 - 1) = 2.5$, $RCL = \{3\}$: city 3;
   (c) from city 3 the distances to $2, 5$ are $4, 3$; threshold $3 + \frac{1}{2} \cdot (4 - 3) = 3.5$, $RCL = \{5\}$: city 5;
   (d) from city 5 only city 2 is left;

   Hence $\pi = (1, 4, 3, 5, 2)$ and $L = 3 + 1 + 3 + 5 + 3 = 15$.

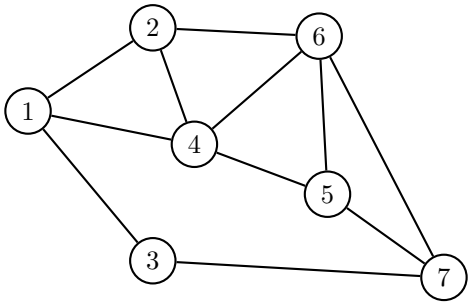
4. $NN$ from city 1 gives 16, 6.7% above the optimum; the first randomized run gives 18, worse than $NN$, and the second $15 = z(I)$, an optimal tour. Randomization can give worse or better tours than the greedy choice, and repeating the construction and keeping the best tour is what makes it useful. With $\alpha = 0$ the RCL contains only the nearest cities, and $RNN_0$ is $NN$ with random tie-breaking; with $\alpha = 1$ the RCL contains all the unvisited cities, and the tour is completely random.

## 2.5 Vertex coloring: greedy and DSATUR

**Exercise 2.13 — Greedy coloring and DSATUR 1**

We consider the graph $G = (V, E)$ with $V = \{1, \dots, 7\}$ ($n = 7$) and $|E| = 11$ edges:

$$E = \big\{\{1,2\},\{1,3\},\{1,4\},\{2,4\},\{2,6\},\{3,7\},\{4,5\},\{4,6\},\{5,6\},\{5,7\},\{6,7\}\big\}.$$




**Questions**

1. Compute the degrees and $\Delta(G)$, and the bound $\Delta(G)+1$ on the number of colors of $GC$.
2. Apply $GC$ with the list $\lambda = (1, 2, \dots, 7)$: for each vertex give the set $U$ of colors of its colored neighbours and the color assigned.
3. Apply $DSATUR$: for each step give the vertex chosen, its saturation degree $sat$, its degree among the uncolored vertices $du$, and its color.
4. Find a clique of maximum size and determine $\chi(G)$. Which of the two colorings is optimal?

**Solution**

1. The neighbourhoods are $N(1) = \{2,3,4\}$, $N(2) = \{1,4,6\}$, $N(3) = \{1,7\}$, $N(4) = \{1,2,5,6\}$, $N(5) = \{4,6,7\}$, $N(6) = \{2,4,5,7\}$, $N(7) = \{3,5,6\}$. The degrees are $(3,3,2,4,3,4,3)$, so $\Delta(G) = 4$ and $GC$ uses at most $\Delta(G)+1 = 5$ colors, with any list.

2.

| $v$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| $U$ | $\emptyset$ | $\{1\}$ | $\{1\}$ | $\{1,2\}$ | $\{3\}$ | $\{1,2,3\}$ | $\{1,2,4\}$ |
| $col[v]$ | 1 | 2 | 2 | 3 | 1 | 4 | 3 |

   $GC$ uses 4 colors: $Q_1 = \{1,5\}$, $Q_2 = \{2,3\}$, $Q_3 = \{4,7\}$, $Q_4 = \{6\}$. The trouble starts at vertex 5: it takes color 1, and vertex 6, adjacent to 2, 4 and 5, then sees the colors $1, 2, 3$.

3. At step 1 all the saturation degrees are 0, and the vertices of largest degree are 4 and 6: vertex 4 is chosen.

| step | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| $v$ | 4 | 6 | 2 | 1 | 5 | 7 | 3 |
| $sat[v]$ | 0 | 1 | 2 | 2 | 2 | 2 | 2 |
| $du[v]$ | 4 | 3 | 1 | 1 | 1 | 1 | 0 |
| $col[v]$ | 1 | 2 | 3 | 2 | 3 | 1 | 3 |

At step 2 the neighbours of 4 have saturation degree 1, and vertex 6 has the most uncolored neighbours (5, 7 and 2). At step 3 the vertices 2 and 5 (both adjacent to 4 and 6) have saturation degree 2 and one uncolored neighbour each, and 2 is chosen. $DSATUR$ uses 3 colors: $Q_1 = \{4, 7\}$, $Q_2 = \{1, 6\}$, $Q_3 = \{2, 3, 5\}$.

4. $\{1, 2, 4\}$ is a clique of size 3 ($\{1, 2\}, \{1, 4\}, \{2, 4\} \in E$), so $\chi(G) \geq \omega(G) \geq 3$; there are no cliques of size 4, but this is not needed: the $DSATUR$ coloring uses 3 colors, hence $\chi(G) = 3$. The $DSATUR$ coloring is optimal, the one of $GC$ with the list $(1, \ldots, 7)$ uses one color more:

$$\frac{GC(I)}{\chi(G)} = \frac{4}{3}, \qquad \frac{DSATUR(I)}{\chi(G)} = 1.$$

## Exercise 2.14 — Greedy coloring and DSATUR 2

We consider the crown graph $G_5$ with $n = 10$ vertices: the vertices $1, \ldots, 5$ on one side, the vertices $6, \ldots, 10$ on the other, and vertex $i \in \{1, \ldots, 5\}$ adjacent to vertex $j + 5$, $j \in \{1, \ldots, 5\}$, if and only if $i \neq j$:

$$E = \big\{\{i, j + 5\} : i, j \in \{1, \ldots, 5\},\ i \neq j\big\}, \qquad |E| = 20.$$

**Questions**

1. Compute $\Delta(G_5)$ and $\chi(G_5)$.
2. Apply $GC$ with the list $\lambda = (1, 6, 2, 7, 3, 8, 4, 9, 5, 10)$ and give $GC(I)$. Compare it with $\Delta(G_5) + 1$ and with $\chi(G_5)$.
3. Give a list for which $GC$ is optimal.
4. Apply $DSATUR$ and give $DSATUR(I)$.

## Solution

1. Each vertex $i \leq 5$ is adjacent to the 4 vertices $j + 5$ with $j \neq i$, and symmetrically: every degree is 4 and $\Delta(G_5) = 4$. The graph is bipartite (all the edges join $\{1, \ldots, 5\}$ and $\{6, \ldots, 10\}$) and has edges, so $\chi(G_5) = 2$.
2. With $u_i = i$ and $v_i = i + 5$ the list is $(u_1, v_1, u_2, v_2, \ldots, u_5, v_5)$, the bad list of the notes:

| $v$ | 1 | 6 | 2 | 7 | 3 | 8 | 4 | 9 | 5 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| $U$ | $\emptyset$ | $\emptyset$ | $\{1\}$ | $\{1\}$ | $\{1,2\}$ | $\{1,2\}$ | $\{1,2,3\}$ | $\{1,2,3\}$ | $\{1,2,3,4\}$ | $\{1,2,3,4\}$ |
| $col[v]$ | 1 | 1 | 2 | 2 | 3 | 3 | 4 | 4 | 5 | 5 |

Vertices $i$ and $i+5$ are not adjacent and both get color $i$, so $GC(I) = 5 = \frac{n}{2}$. This is exactly the bound $\Delta(G_5) + 1 = 5$, and

$$\frac{GC(I)}{\chi(G_5)} = \frac{5}{2}.$$

3. The list $(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)$, one side first: the vertices $1, \ldots, 5$ are pairwise non-adjacent and all get color 1; each vertex $6, \ldots, 10$ has only neighbours of color 1 and gets color 2. Hence $GC(I) = 2 = \chi(G_5)$.

4. At step 1 all the vertices have degree 4 and vertex 1 is chosen (color 1). Then:

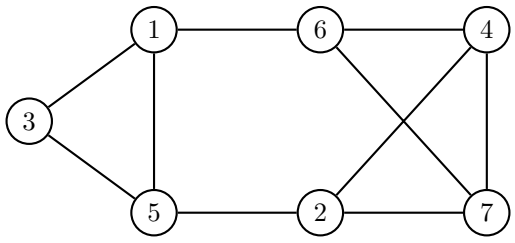
| step | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| $v$ | 1 | 7 | 3 | 4 | 5 | 6 | 2 | 8 | 9 | 10 |
| $sat[v]$ | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| $du[v]$ | 4 | 3 | 3 | 3 | 3 | 1 | 3 | 0 | 0 | 0 |
| $col[v]$ | 1 | 2 | 1 | 1 | 1 | 2 | 1 | 2 | 2 | 2 |

At step 2 the neighbours $7, 8, 9, 10$ of vertex 1 have saturation degree 1 and 3 uncolored neighbours each: vertex 7 is chosen and gets color 2. From then on, every vertex adjacent to a colored vertex has saturation degree 1 (its colored neighbours are all on the other side, with the same color); the vertices of the side $\{1, \ldots, 5\}$ get color 1 and those of the side $\{6, \ldots, 10\}$ color 2. Hence $DSATUR(I) = 2 = \chi(G_5)$, as guaranteed on bipartite graphs.

## Exercise 2.15 — Greedy coloring and DSATUR 3

We consider the graph $G = (V, E)$ with $V = \{1, \ldots, 7\}$ ($n = 7$) and $|E| = 10$ edges:

$$E = \big\{\{1,3\}, \{1,5\}, \{1,6\}, \{2,4\}, \{2,5\}, \{2,7\}, \{3,5\}, \{4,6\}, \{4,7\}, \{6,7\}\big\}.$$




**Questions**

1. Apply $DSATUR$ (ties: largest degree among the uncolored vertices, then smallest index) and give $DSATUR(I)$.

2. Show that $\chi(G) = 3$.
3. Is the result of Question 1 in contradiction with the bound $\Delta(G) + 1$?

**Solution**

1. The neighbourhoods are $N(1) = \{3, 5, 6\}$, $N(2) = \{4, 5, 7\}$, $N(3) = \{1, 5\}$, $N(4) = \{2, 6, 7\}$, $N(5) = \{1, 2, 3\}$, $N(6) = \{1, 4, 7\}$, $N(7) = \{2, 4, 6\}$.

| step | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| $v$ | 1 | 5 | 3 | 2 | 4 | 6 | 7 |
| $sat[v]$ | 0 | 1 | 2 | 1 | 1 | 2 | 3 |
| $du[v]$ | 3 | 2 | 0 | 2 | 2 | 1 | 0 |
| $col[v]$ | 1 | 2 | 3 | 1 | 2 | 3 | 4 |

   - Step 1: vertex 1 (degree 3, smallest index among the vertices of degree 3), color 1.
   - Step 2: the neighbours $3, 5, 6$ of vertex 1 have saturation degree 1; their uncolored neighbours are 1, 2, 2 respectively, and vertex 5 is chosen: color 2.
   - Step 3: vertex 3 sees colors 1 and 2 (saturation degree 2): color 3.
   - Step 4: the vertices 2 (neighbour 5) and 6 (neighbour 1) have saturation degree 1 and two uncolored neighbours each; vertex 2 is chosen and gets color 1.
   - Step 5: the vertices 4, 6, 7 have saturation degree 1 and two uncolored neighbours each; vertex 4 gets color 2.
   - Step 6: vertex 6 sees colors 1 (vertex 1) and 2 (vertex 4), vertex 7 sees 1 (vertex 2) and 2 (vertex 4); both have saturation degree 2 and one uncolored neighbour, and vertex 6 gets color 3.
   - Step 7: vertex 7 sees colors $1, 2, 3$ and gets color 4.

   Hence $DSATUR(I) = 4$.

2. $\{1, 3, 5\}$ is a triangle, so $\chi(G) \geq 3$. The coloring $Q_1 = \{1, 4\}$, $Q_2 = \{2, 3, 6\}$, $Q_3 = \{5, 7\}$ is feasible (no edge inside a class: $\{1, 4\}$, $\{2, 3\}$, $\{2, 6\}$, $\{3, 6\}$, $\{5, 7\} \notin E$), so $\chi(G) = 3$ and
$$\frac{DSATUR(I)}{\chi(G)} = \frac{4}{3}.$$
   The damage is done by step 5: vertices 6 and 7 already see color 1 (from vertices 1 and 2), and vertex 4, their common neighbour, takes color 2. The two adjacent vertices 6 and 7 are left with color 3 only, and one of them needs a fourth color. In the optimal coloring vertex 4 has the same color as vertex 1.

3. No: the degrees are $(3, 3, 2, 3, 3, 3, 3)$, so $\Delta(G) + 1 = 4$ and $DSATUR(I) = 4$ attains the bound. The bound $\Delta(G) + 1$ limits the number of colors, it does not compare it with $\chi(G)$.

Chapter 3

# Local search

## 3.1 Knapsack

**Exercise 3.1 — Local search KP 1**

We consider a knapsack of capacity $c = 18$ and a set $\mathscr{I} = \{1, 2, 3, 4, 5\}$ of 5 items ($n = 5$), with the following profits and weights:

| | $j = 1$ | $j = 2$ | $j = 3$ | $j = 4$ | $j = 5$ |
|---|---|---|---|---|---|
| $p_j$ | 10 | 7 | 13 | 14 | 8 |
| $w_j$ | 2 | 3 | 8 | 9 | 7 |

The items are sorted by non-increasing efficiency. The neighbourhood flip+swap contains the feasible vectors obtained by a flip (add or remove one item) or by a swap (remove a selected item $j$, add an unselected item $k$); the flips are examined first, for $j = 1, \ldots, n$, then the swaps, for increasing $j$ and, for each $j$, increasing $k$. Ties in best improvement are broken in favour of the move examined first.

**Questions**

1. Compute the solution of the greedy algorithm $GR$ (by efficiency) and its profit.
2. Apply the local search with the neighbourhood flip+swap and the best-improvement rule, starting from the solution of $GR$. At every iteration list the neighbours, with their weight and profit, and state the move accepted.
3. Compute the solution of the greedy by profit $GR_p$, and apply the same local search starting from it.
4. The optimal value is $z(I) = 32$. Which of the two local searches finds an optimal solution? Explain why the other one stops, by looking at the moves needed to reach the optimal solution.

**Solution**

1. The efficiencies are $\frac{10}{2}$, $\frac{7}{3}$, $\frac{13}{8}$, $\frac{14}{9}$, $\frac{8}{7}$, non-increasing. With $rc = 18$:
   - item 1 ($w_1 = 2$) is selected, $rc = 16$;
   - item 2 ($w_2 = 3$) is selected, $rc = 13$;
   - item 3 ($w_3 = 8$) is selected, $rc = 5$;
   - item 4 ($w_4 = 9 > rc = 5$) is not selected;
   - item 5 ($w_5 = 7 > rc = 5$) is not selected;

   $GR$ returns $x = (1, 1, 1, 0, 0)$, the items $\{1, 2, 3\}$, of weight 13 and profit $GR(I) = 30$.

2. Iteration 1: current solution $x = (1, 1, 1, 0, 0)$, weight 13, profit 30.

| move | neighbour | weight | profit | |
|---|---|---|---|---|
| flip 1 | $(0, 1, 1, 0, 0)$ | 11 | 20 | |
| flip 2 | $(1, 0, 1, 0, 0)$ | 10 | 23 | |
| flip 3 | $(1, 1, 0, 0, 0)$ | 5 | 17 | |
| flip 4 | $(1, 1, 1, 1, 0)$ | 22 | — | infeasible |
| flip 5 | $(1, 1, 1, 0, 1)$ | 20 | — | infeasible |
| 1 out, 4 in | $(0, 1, 1, 1, 0)$ | 20 | — | infeasible |
| 1 out, 5 in | $(0, 1, 1, 0, 1)$ | 18 | 28 | |
| 2 out, 4 in | $(1, 0, 1, 1, 0)$ | 19 | — | infeasible |
| 2 out, 5 in | $(1, 0, 1, 0, 1)$ | 17 | 31 | ← accepted |
| 3 out, 4 in | $(1, 1, 0, 1, 0)$ | 14 | 31 | |
| 3 out, 5 in | $(1, 1, 0, 0, 1)$ | 12 | 25 | |

   The move is accepted after examining 7 feasible neighbours.

   Iteration 2: current solution $x = (1, 0, 1, 0, 1)$, weight 17, profit 31.

| move | neighbour | weight | profit | |
|---|---|---|---|---|
| flip 1 | $(0, 0, 1, 0, 1)$ | 15 | 21 | |
| flip 2 | $(1, 1, 1, 0, 1)$ | 20 | — | infeasible |
| flip 3 | $(1, 0, 0, 0, 1)$ | 9 | 18 | |
| flip 4 | $(1, 0, 1, 1, 1)$ | 26 | — | infeasible |
| flip 5 | $(1, 0, 1, 0, 0)$ | 10 | 23 | |
| 1 out, 2 in | $(0, 1, 1, 0, 1)$ | 18 | 28 | |
| 1 out, 4 in | $(0, 0, 1, 1, 1)$ | 24 | — | infeasible |
| 3 out, 2 in | $(1, 1, 0, 0, 1)$ | 12 | 25 | |
| 3 out, 4 in | $(1, 0, 0, 1, 1)$ | 18 | 32 | ← accepted |
| 5 out, 2 in | $(1, 1, 1, 0, 0)$ | 13 | 30 | |
| 5 out, 4 in | $(1, 0, 1, 1, 0)$ | 19 | — | infeasible |

   The move is accepted after examining 7 feasible neighbours.

   Iteration 3: current solution $x = (1, 0, 0, 1, 1)$, weight 18, profit 32.

| move | neighbour | weight | profit | |
|---|---|---|---|---|
| flip 1 | $(0,0,0,1,1)$ | 16 | 22 | |
| flip 2 | $(1,1,0,1,1)$ | 21 | — | infeasible |
| flip 3 | $(1,0,1,1,1)$ | 26 | — | infeasible |
| flip 4 | $(1,0,0,0,1)$ | 9 | 18 | |
| flip 5 | $(1,0,0,1,0)$ | 11 | 24 | |
| 1 out, 2 in | $(0,1,0,1,1)$ | 19 | — | infeasible |
| 1 out, 3 in | $(0,0,1,1,1)$ | 24 | — | infeasible |
| 4 out, 2 in | $(1,1,0,0,1)$ | 12 | 25 | |
| 4 out, 3 in | $(1,0,1,0,1)$ | 17 | 31 | |
| 5 out, 2 in | $(1,1,0,1,0)$ | 14 | 31 | |
| 5 out, 3 in | $(1,0,1,1,0)$ | 19 | — | infeasible |

No feasible neighbour has a larger profit: $x = (1,0,0,1,1)$ is a **local optimum**, of profit 32 (6 feasible neighbours examined).

The local search makes 2 moves, $30 \to 31 \to 32$, and stops in $x = (1,0,0,1,1)$, of profit 32.

3. The items by non-increasing profit are $(4,3,1,5,2)$. With $rc = 18$:

   - item 4 ($w_4 = 9$) is selected, $rc = 9$;
   - item 3 ($w_3 = 8$) is selected, $rc = 1$;
   - item 1 ($w_1 = 2 > rc = 1$) is not selected;
   - item 5 ($w_5 = 7 > rc = 1$) is not selected;
   - item 2 ($w_2 = 3 > rc = 1$) is not selected;

   The greedy algorithm by profit returns $x = (0,0,1,1,0)$, of weight 17 and profit 27.

   Iteration 1: current solution $x = (0,0,1,1,0)$, weight 17, profit 27.

| move | neighbour | weight | profit | |
|---|---|---|---|---|
| flip 1 | $(1,0,1,1,0)$ | 19 | — | infeasible |
| flip 2 | $(0,1,1,1,0)$ | 20 | — | infeasible |
| flip 3 | $(0,0,0,1,0)$ | 9 | 14 | |
| flip 4 | $(0,0,1,0,0)$ | 8 | 13 | |
| flip 5 | $(0,0,1,1,1)$ | 24 | — | infeasible |
| 3 out, 1 in | $(1,0,0,1,0)$ | 11 | 24 | |
| 3 out, 2 in | $(0,1,0,1,0)$ | 12 | 21 | |
| 3 out, 5 in | $(0,0,0,1,1)$ | 16 | 22 | |
| 4 out, 1 in | $(1,0,1,0,0)$ | 10 | 23 | |
| 4 out, 2 in | $(0,1,1,0,0)$ | 11 | 20 | |
| 4 out, 5 in | $(0,0,1,0,1)$ | 15 | 21 | |

No feasible neighbour has a larger profit: $x = (0,0,1,1,0)$ is a **local optimum**, of profit 27 (8 feasible neighbours examined).

4. The local search from $GR$ reaches $x = (1, 0, 0, 1, 1)$, of profit $32 = z(I)$: an **optimal** solution. The local search from the greedy solution by profit stops immediately in $x = (0, 0, 1, 1, 0)$, of profit 27, a local optimum that is not optimal. To go from $\{3, 4\}$ to the optimal set $\{1, 4, 5\}$ one must remove item 3 and add the items 1 and 5: three flips, or one swap and one flip, and every intermediate vector is either infeasible or worse. The local optimum reached depends on the **starting solution**.

## Exercise 3.2 — Local search KP 2

We consider a knapsack of capacity $c = 13$ and a set $\mathscr{I} = \{1, 2, 3, 4, 5\}$ of 5 items ($n = 5$), with the following profits and weights:

| | $j=1$ | $j=2$ | $j=3$ | $j=4$ | $j=5$ |
|---|---|---|---|---|---|
| $p_j$ | 8 | 9 | 14 | 11 | 9 |
| $w_j$ | 3 | 5 | 9 | 9 | 8 |

The items are sorted by non-increasing efficiency. The neighbourhood flip+swap contains the feasible vectors obtained by a flip (add or remove one item) or by a swap (remove a selected item $j$, add an unselected item $k$); the flips are examined first, for $j = 1, \ldots, n$, then the swaps, for increasing $j$ and, for each $j$, increasing $k$. Ties in best improvement are broken in favour of the move examined first.

**Questions**

1. Compute the solution of the greedy algorithm $GR$ (by efficiency) and its profit.
2. Apply the local search with the neighbourhood flip+swap and the **first-improvement** rule, starting from the solution of $GR$.
3. Apply the same local search with the **best-improvement** rule, starting from the solution of $GR$.
4. Compare the two local optima, and the number of feasible neighbours examined by the two rules. Is one of the two local optima optimal? (Hint: the LP relaxation bound is not enough here; split the solutions according to whether item 3 is selected.)

## Solution

1. The efficiencies are $\frac{8}{3}$, $\frac{9}{5}$, $\frac{14}{9}$, $\frac{11}{9}$, $\frac{9}{8}$, non-increasing. With $rc = 13$:
   - item 1 ($w_1 = 3$) is selected, $rc = 10$;
   - item 2 ($w_2 = 5$) is selected, $rc = 5$;
   - item 3 ($w_3 = 9 > rc = 5$) is not selected;
   - item 4 ($w_4 = 9 > rc = 5$) is not selected;
   - item 5 ($w_5 = 8 > rc = 5$) is not selected;

$GR$ returns $x = (1, 1, 0, 0, 0)$, of weight 8 and profit $GR(I) = 17$.

2. Iteration 1: current solution $x = (1, 1, 0, 0, 0)$, weight 8, profit 17.

| move | neighbour | weight | profit | |
|---|---|---|---|---|
| flip 1 | $(0, 1, 0, 0, 0)$ | 5 | 9 | |
| flip 2 | $(1, 0, 0, 0, 0)$ | 3 | 8 | |
| flip 3 | $(1, 1, 1, 0, 0)$ | 17 | — | infeasible |
| flip 4 | $(1, 1, 0, 1, 0)$ | 17 | — | infeasible |
| flip 5 | $(1, 1, 0, 0, 1)$ | 16 | — | infeasible |
| 1 out, 3 in | $(0, 1, 1, 0, 0)$ | 14 | — | infeasible |
| 1 out, 4 in | $(0, 1, 0, 1, 0)$ | 14 | — | infeasible |
| 1 out, 5 in | $(0, 1, 0, 0, 1)$ | 13 | 18 | ← accepted |
| 2 out, 3 in | $(1, 0, 1, 0, 0)$ | 12 | 22 | |
| 2 out, 4 in | $(1, 0, 0, 1, 0)$ | 12 | 19 | |
| 2 out, 5 in | $(1, 0, 0, 0, 1)$ | 11 | 17 | |

The move is accepted after examining 3 feasible neighbours.

Iteration 2: current solution $x = (0, 1, 0, 0, 1)$, weight 13, profit 18.

| move | neighbour | weight | profit | |
|---|---|---|---|---|
| flip 1 | $(1, 1, 0, 0, 1)$ | 16 | — | infeasible |
| flip 2 | $(0, 0, 0, 0, 1)$ | 8 | 9 | |
| flip 3 | $(0, 1, 1, 0, 1)$ | 22 | — | infeasible |
| flip 4 | $(0, 1, 0, 1, 1)$ | 22 | — | infeasible |
| flip 5 | $(0, 1, 0, 0, 0)$ | 5 | 9 | |
| 2 out, 1 in | $(1, 0, 0, 0, 1)$ | 11 | 17 | |
| 2 out, 3 in | $(0, 0, 1, 0, 1)$ | 17 | — | infeasible |
| 2 out, 4 in | $(0, 0, 0, 1, 1)$ | 17 | — | infeasible |
| 5 out, 1 in | $(1, 1, 0, 0, 0)$ | 8 | 17 | |
| 5 out, 3 in | $(0, 1, 1, 0, 0)$ | 14 | — | infeasible |
| 5 out, 4 in | $(0, 1, 0, 1, 0)$ | 14 | — | infeasible |

No feasible neighbour has a larger profit: $x = (0, 1, 0, 0, 1)$ is a **local optimum**, of profit 18 (4 feasible neighbours examined).

First improvement stops in $x = (0, 1, 0, 0, 1)$, of profit 18, after examining 7 feasible neighbours in total.

3. Iteration 1: current solution $x = (1, 1, 0, 0, 0)$, weight 8, profit 17.

| move | neighbour | weight | profit | |
|---|---|---|---|---|
| flip 1 | $(0,1,0,0,0)$ | 5 | 9 | |
| flip 2 | $(1,0,0,0,0)$ | 3 | 8 | |
| flip 3 | $(1,1,1,0,0)$ | 17 | — | infeasible |
| flip 4 | $(1,1,0,1,0)$ | 17 | — | infeasible |
| flip 5 | $(1,1,0,0,1)$ | 16 | — | infeasible |
| 1 out, 3 in | $(0,1,1,0,0)$ | 14 | — | infeasible |
| 1 out, 4 in | $(0,1,0,1,0)$ | 14 | — | infeasible |
| 1 out, 5 in | $(0,1,0,0,1)$ | 13 | 18 | |
| 2 out, 3 in | $(1,0,1,0,0)$ | 12 | 22 | ← accepted |
| 2 out, 4 in | $(1,0,0,1,0)$ | 12 | 19 | |
| 2 out, 5 in | $(1,0,0,0,1)$ | 11 | 17 | |

The move is accepted after examining 6 feasible neighbours.

Iteration 2: current solution $x = (1,0,1,0,0)$, weight 12, profit 22.

| move | neighbour | weight | profit | |
|---|---|---|---|---|
| flip 1 | $(0,0,1,0,0)$ | 9 | 14 | |
| flip 2 | $(1,1,1,0,0)$ | 17 | — | infeasible |
| flip 3 | $(1,0,0,0,0)$ | 3 | 8 | |
| flip 4 | $(1,0,1,1,0)$ | 21 | — | infeasible |
| flip 5 | $(1,0,1,0,1)$ | 20 | — | infeasible |
| 1 out, 2 in | $(0,1,1,0,0)$ | 14 | — | infeasible |
| 1 out, 4 in | $(0,0,1,1,0)$ | 18 | — | infeasible |
| 1 out, 5 in | $(0,0,1,0,1)$ | 17 | — | infeasible |
| 3 out, 2 in | $(1,1,0,0,0)$ | 8 | 17 | |
| 3 out, 4 in | $(1,0,0,1,0)$ | 12 | 19 | |
| 3 out, 5 in | $(1,0,0,0,1)$ | 11 | 17 | |

No feasible neighbour has a larger profit: $x = (1,0,1,0,0)$ is a **local optimum**, of profit 22 (5 feasible neighbours examined).

Best improvement stops in $x = (1,0,1,0,0)$, of profit 22, after examining 11 feasible neighbours in total.

4. First improvement accepts the first improving swap it finds, which gains only 1, and stops in a local optimum of profit 18; best improvement takes the largest gain and reaches profit 22. They examine 7 and 11 feasible neighbours. The LP relaxation gives $z(I) \leq 8 + 9 + 14 \cdot \frac{5}{9} = \frac{223}{9}$, hence $z(I) \leq 24$, which does not prove that 22 is optimal. We argue directly. If item 3 is selected, the residual capacity is 4 and only item 1 fits, for a profit of 22. If item 3 is not selected, the best solution is $\{1,4\}$, of profit 19. Hence $z(I) = 22$ and the solution of best improvement, $x = (1,0,1,0,0)$, is **optimal**. From the same starting solution the two rules reach **different local optima**.

## Exercise 3.3 — Local search KP 3

We consider a knapsack of capacity $c = 11$ and a set $\mathscr{I} = \{1, 2, 3, 4, 5\}$ of 5 items ($n = 5$), with the following profits and weights:

| | $j=1$ | $j=2$ | $j=3$ | $j=4$ | $j=5$ |
|---|---|---|---|---|---|
| $p_j$ | 14 | 7 | 10 | 8 | 6 |
| $w_j$ | 4 | 4 | 7 | 7 | 9 |

The items are sorted by non-increasing efficiency. With penalties, the local search maximizes $g_\alpha(x) = \sum_j p_j x_j - \alpha \max\{0, \sum_j w_j x_j - c\}$ over all the binary vectors, with the flip neighbourhood (flips examined for $j = 1, \ldots, n$) and the best-improvement rule.

**Questions**

1. Compute the solution of $GR$, and show that it is a local optimum for the flip neighbourhood restricted to feasible vectors.
2. Apply the penalty flip local search with $\alpha = 1$, starting from the solution of $GR$. Is the local optimum feasible? Which solution should the search return?
3. Apply the penalty flip local search with $\alpha = 2$, starting from the solution of $GR$.
4. Apply it with $\alpha = 3$. The optimal value is $z(I) = 24$: discuss the role of $\alpha$, and state for which values of $\alpha$ the local optima are surely feasible.

## Solution

1. With $rc = 11$:
   - item 1 ($w_1 = 4$) is selected, $rc = 7$;
   - item 2 ($w_2 = 4$) is selected, $rc = 3$;
   - item 3 ($w_3 = 7 > rc = 3$) is not selected;
   - item 4 ($w_4 = 7 > rc = 3$) is not selected;
   - item 5 ($w_5 = 9 > rc = 3$) is not selected;

   $GR$ returns $x = (1, 1, 0, 0, 0)$, of weight 8 and profit 21. Its flips are

| move | neighbour | weight | profit | |
|---|---|---|---|---|
| flip 1 | $(0, 1, 0, 0, 0)$ | 4 | 7 | |
| flip 2 | $(1, 0, 0, 0, 0)$ | 4 | 14 | |
| flip 3 | $(1, 1, 1, 0, 0)$ | 15 | — | infeasible |
| flip 4 | $(1, 1, 0, 1, 0)$ | 15 | — | infeasible |
| flip 5 | $(1, 1, 0, 0, 1)$ | 17 | — | infeasible |

The feasible neighbours remove an item and have a smaller profit: $x$ is a local optimum for the feasible flip neighbourhood.

2. Iteration 1: $x = (1, 1, 0, 0, 0)$, weight 8, $g_1(x) = 21$.

| move | neighbour | weight | excess | $g_1$ | |
|---|---|---|---|---|---|
| flip 1 | $(0, 1, 0, 0, 0)$ | 4 | 0 | $7 - 1 \cdot 0 = 7$ | |
| flip 2 | $(1, 0, 0, 0, 0)$ | 4 | 0 | $14 - 1 \cdot 0 = 14$ | |
| flip 3 | $(1, 1, 1, 0, 0)$ | 15 | 4 | $31 - 1 \cdot 4 = 27$ | $\leftarrow$ |
| flip 4 | $(1, 1, 0, 1, 0)$ | 15 | 4 | $29 - 1 \cdot 4 = 25$ | |
| flip 5 | $(1, 1, 0, 0, 1)$ | 17 | 6 | $27 - 1 \cdot 6 = 21$ | |

Iteration 2: $x = (1, 1, 1, 0, 0)$, weight 15, $g_1(x) = 27$.

| move | neighbour | weight | excess | $g_1$ | |
|---|---|---|---|---|---|
| flip 1 | $(0, 1, 1, 0, 0)$ | 11 | 0 | $17 - 1 \cdot 0 = 17$ | |
| flip 2 | $(1, 0, 1, 0, 0)$ | 11 | 0 | $24 - 1 \cdot 0 = 24$ | |
| flip 3 | $(1, 1, 0, 0, 0)$ | 8 | 0 | $21 - 1 \cdot 0 = 21$ | |
| flip 4 | $(1, 1, 1, 1, 0)$ | 22 | 11 | $39 - 1 \cdot 11 = 28$ | $\leftarrow$ |
| flip 5 | $(1, 1, 1, 0, 1)$ | 24 | 13 | $37 - 1 \cdot 13 = 24$ | |

Iteration 3: $x = (1, 1, 1, 1, 0)$, weight 22, $g_1(x) = 28$.

| move | neighbour | weight | excess | $g_1$ |
|---|---|---|---|---|
| flip 1 | $(0, 1, 1, 1, 0)$ | 18 | 7 | $25 - 1 \cdot 7 = 18$ |
| flip 2 | $(1, 0, 1, 1, 0)$ | 18 | 7 | $32 - 1 \cdot 7 = 25$ |
| flip 3 | $(1, 1, 0, 1, 0)$ | 15 | 4 | $29 - 1 \cdot 4 = 25$ |
| flip 4 | $(1, 1, 1, 0, 0)$ | 15 | 4 | $31 - 1 \cdot 4 = 27$ |
| flip 5 | $(1, 1, 1, 1, 1)$ | 31 | 20 | $45 - 1 \cdot 20 = 25$ |

No flip increases $g_1$: the search stops in $x = (1, 1, 1, 1, 0)$, of weight 22 and profit 39.

The local optimum is **infeasible** (weight $22 > 11$): the penalty is too small. The search must return the best feasible vector visited, $(1, 1, 0, 0, 0)$, of profit 21, i.e., the starting solution.

3. Iteration 1: $x = (1, 1, 0, 0, 0)$, weight 8, $g_2(x) = 21$.

| move | neighbour | weight | excess | $g_2$ | |
|---|---|---|---|---|---|
| flip 1 | $(0, 1, 0, 0, 0)$ | 4 | 0 | $7 - 2 \cdot 0 = 7$ | |
| flip 2 | $(1, 0, 0, 0, 0)$ | 4 | 0 | $14 - 2 \cdot 0 = 14$ | |
| flip 3 | $(1, 1, 1, 0, 0)$ | 15 | 4 | $31 - 2 \cdot 4 = 23$ | $\leftarrow$ |
| flip 4 | $(1, 1, 0, 1, 0)$ | 15 | 4 | $29 - 2 \cdot 4 = 21$ | |
| flip 5 | $(1, 1, 0, 0, 1)$ | 17 | 6 | $27 - 2 \cdot 6 = 15$ | |

Iteration 2: $x = (1, 1, 1, 0, 0)$, weight 15, $g_2(x) = 23$.

| move | neighbour | weight | excess | $g_2$ | |
|---|---|---|---|---|---|
| flip 1 | $(0, 1, 1, 0, 0)$ | 11 | 0 | $17 - 2 \cdot 0 = 17$ | |
| flip 2 | $(1, 0, 1, 0, 0)$ | 11 | 0 | $24 - 2 \cdot 0 = 24$ | $\leftarrow$ |
| flip 3 | $(1, 1, 0, 0, 0)$ | 8 | 0 | $21 - 2 \cdot 0 = 21$ | |
| flip 4 | $(1, 1, 1, 1, 0)$ | 22 | 11 | $39 - 2 \cdot 11 = 17$ | |
| flip 5 | $(1, 1, 1, 0, 1)$ | 24 | 13 | $37 - 2 \cdot 13 = 11$ | |

Iteration 3: $x = (1, 0, 1, 0, 0)$, weight 11, $g_2(x) = 24$.

| move | neighbour | weight | excess | $g_2$ |
|---|---|---|---|---|
| flip 1 | $(0, 0, 1, 0, 0)$ | 7 | 0 | $10 - 2 \cdot 0 = 10$ |
| flip 2 | $(1, 1, 1, 0, 0)$ | 15 | 4 | $31 - 2 \cdot 4 = 23$ |
| flip 3 | $(1, 0, 0, 0, 0)$ | 4 | 0 | $14 - 2 \cdot 0 = 14$ |
| flip 4 | $(1, 0, 1, 1, 0)$ | 18 | 7 | $32 - 2 \cdot 7 = 18$ |
| flip 5 | $(1, 0, 1, 0, 1)$ | 20 | 9 | $30 - 2 \cdot 9 = 12$ |

No flip increases $g_2$: the search stops in $x = (1, 0, 1, 0, 0)$, of weight 11 and profit 24.

The local optimum $(1, 0, 1, 0, 0)$ is feasible, of profit $24 = z(I)$: it is **optimal**. The search reached it through the infeasible vector $(1, 1, 1, 0, 0)$.

4. Iteration 1: $x = (1, 1, 0, 0, 0)$, weight 8, $g_3(x) = 21$.

| move | neighbour | weight | excess | $g_3$ |
|---|---|---|---|---|
| flip 1 | $(0, 1, 0, 0, 0)$ | 4 | 0 | $7 - 3 \cdot 0 = 7$ |
| flip 2 | $(1, 0, 0, 0, 0)$ | 4 | 0 | $14 - 3 \cdot 0 = 14$ |
| flip 3 | $(1, 1, 1, 0, 0)$ | 15 | 4 | $31 - 3 \cdot 4 = 19$ |
| flip 4 | $(1, 1, 0, 1, 0)$ | 15 | 4 | $29 - 3 \cdot 4 = 17$ |
| flip 5 | $(1, 1, 0, 0, 1)$ | 17 | 6 | $27 - 3 \cdot 6 = 9$ |

No flip increases $g_3$: the search stops in $x = (1, 1, 0, 0, 0)$, of weight 8 and profit 21.

With $\alpha = 3$ the search does not move: it behaves as the feasible search and stops in the solution of $GR$, of profit 21. A small $\alpha$ (1) lets the search stop among infeasible vectors; a large $\alpha$ (3) forbids it to cross the infeasible region; here only $\alpha = 2$ reaches the optimum. If $\alpha > \max_j p_j = 14$, every infeasible vector has an improving flip (removing any selected item decreases the excess by at least 1 and gains at least $\alpha - p_j > 0$), so all the local optima are feasible; but then, from a feasible vector, no flip that creates an excess is improving, and the search never leaves the feasible region.

## 3.2 Bin packing

**Exercise 3.4 — Local search BPP 1**

Bins of capacity $c = 10$ and a set $\mathscr{I} = \{1, 2, 3, 4, 5, 6, 7\}$ of 7 items with weights

| | $j=1$ | $j=2$ | $j=3$ | $j=4$ | $j=5$ | $j=6$ | $j=7$ |
|---|---|---|---|---|---|---|---|
| $w_j$ | 2 | 3 | 2 | 7 | 4 | 7 | 4 |

The bins are labelled $1, 2, \ldots$ in the order in which $FF$ opens them, and keep their label. A *move* puts one item into another non-empty bin where it fits (examined for $j = 1, \ldots, n$ and, for each $j$, for increasing bin label); a *swap* exchanges two items $j < k$ of different weight in different bins, if both bins stay feasible. The auxiliary objective $SQ = \sum_b L_b^2$ is maximized; the gain of a transfer of $\delta$ from bin $a$ to bin $b$ is $2\delta(L_b - L_a + \delta)$ (a move of item $j$ transfers $\delta = w_j$, a swap of item $j$ in bin $a$ and item $k$ in bin $b$ transfers $\delta = w_j - w_k$). Best improvement breaks ties in favour of the move examined first.

**Questions**

1. Compute the solution of First Fit ($FF$), its number of bins and the lower bound $\lceil \sum_j w_j / c \rceil$.
2. Apply the move local search with the objective $nb$ (number of bins), starting from the solution of $FF$. Why does it stop?
3. Apply the move local search with the auxiliary objective $SQ = \sum_b L_b^2$ and the best-improvement rule, starting from the solution of $FF$.
4. Is the final solution optimal? Explain the different behaviour of the two objectives.

**Solution**

1. 
   - item 1 ($w_1 = 2$) goes into bin 1, opened for it, load 2;
   - item 2 ($w_2 = 3$) goes into bin 1, load 5;
   - item 3 ($w_3 = 2$) goes into bin 1, load 7;
   - item 4 ($w_4 = 7$) fits in no open bin: bin 2 is opened, load 7;
   - item 5 ($w_5 = 4$) fits in no open bin: bin 3 is opened, load 4;
   - item 6 ($w_6 = 7$) fits in no open bin: bin 4 is opened, load 7;
   - item 7 ($w_7 = 4$) goes into bin 3, load 8;

   $FF$ uses $nb = 4$ bins, with packing patterns $\{1, 2, 3\}, \{4\}, \{5, 7\}, \{6\}$ and loads $(7, 7, 8, 7)$. The lower bound is $\lceil 29/10 \rceil = 3$.
2. Iteration 1 (move): packing patterns $\{1, 2, 3\}, \{4\}, \{5, 7\}, \{6\}$, loads $(7, 7, 8, 7)$, $nb = 4$, $SQ = 211$.

| move | loads | bin emptied |
|---|---|---|
| item 1: bin $1 \to 2$ | $(5, 9, 8, 7)$ | no |
| item 1: bin $1 \to 3$ | $(5, 7, 10, 7)$ | no |
| item 1: bin $1 \to 4$ | $(5, 7, 8, 9)$ | no |
| item 2: bin $1 \to 2$ | $(4, 10, 8, 7)$ | no |
| item 2: bin $1 \to 4$ | $(4, 7, 8, 10)$ | no |
| item 3: bin $1 \to 2$ | $(5, 9, 8, 7)$ | no |
| item 3: bin $1 \to 3$ | $(5, 7, 10, 7)$ | no |
| item 3: bin $1 \to 4$ | $(5, 7, 8, 9)$ | no |

No move is improving.

No move empties a bin, so no move decreases $nb$: the solution of $FF$ is a local optimum with 4 bins. All the moves are neutral for $nb$: the search is on a **plateau**.

3. Iteration 1 (move): packing patterns $\{1, 2, 3\}, \{4\}, \{5, 7\}, \{6\}$, loads $(7, 7, 8, 7)$, $nb = 4$, $SQ = 211$.

| move | loads | gain | |
|---|---|---|---|
| item 1: bin $1 \to 2$ | $(5, 9, 8, 7)$ | $2 \cdot 2 \cdot (7 - 7 + 2) = 8$ | |
| item 1: bin $1 \to 3$ | $(5, 7, 10, 7)$ | $2 \cdot 2 \cdot (8 - 7 + 2) = 12$ | |
| item 1: bin $1 \to 4$ | $(5, 7, 8, 9)$ | $2 \cdot 2 \cdot (7 - 7 + 2) = 8$ | |
| item 2: bin $1 \to 2$ | $(4, 10, 8, 7)$ | $2 \cdot 3 \cdot (7 - 7 + 3) = 18$ | $\leftarrow$ |
| item 2: bin $1 \to 4$ | $(4, 7, 8, 10)$ | $2 \cdot 3 \cdot (7 - 7 + 3) = 18$ | |
| item 3: bin $1 \to 2$ | $(5, 9, 8, 7)$ | $2 \cdot 2 \cdot (7 - 7 + 2) = 8$ | |
| item 3: bin $1 \to 3$ | $(5, 7, 10, 7)$ | $2 \cdot 2 \cdot (8 - 7 + 2) = 12$ | |
| item 3: bin $1 \to 4$ | $(5, 7, 8, 9)$ | $2 \cdot 2 \cdot (7 - 7 + 2) = 8$ | |

The move marked is accepted.

Iteration 2 (move): packing patterns $\{1, 3\}, \{2, 4\}, \{5, 7\}, \{6\}$, loads $(4, 10, 8, 7)$, $nb = 4$, $SQ = 229$.

| move | loads | gain | |
|---|---|---|---|
| item 1: bin $1 \to 3$ | $(2, 10, 10, 7)$ | $2 \cdot 2 \cdot (8 - 4 + 2) = 24$ | $\leftarrow$ |
| item 1: bin $1 \to 4$ | $(2, 10, 8, 9)$ | $2 \cdot 2 \cdot (7 - 4 + 2) = 20$ | |
| item 2: bin $2 \to 1$ | $(7, 7, 8, 7)$ | $2 \cdot 3 \cdot (4 - 10 + 3) = -18$ | |
| item 2: bin $2 \to 4$ | $(4, 7, 8, 10)$ | $2 \cdot 3 \cdot (7 - 10 + 3) = 0$ | |
| item 3: bin $1 \to 3$ | $(2, 10, 10, 7)$ | $2 \cdot 2 \cdot (8 - 4 + 2) = 24$ | |
| item 3: bin $1 \to 4$ | $(2, 10, 8, 9)$ | $2 \cdot 2 \cdot (7 - 4 + 2) = 20$ | |
| item 5: bin $3 \to 1$ | $(8, 10, 4, 7)$ | $2 \cdot 4 \cdot (4 - 8 + 4) = 0$ | |
| item 7: bin $3 \to 1$ | $(8, 10, 4, 7)$ | $2 \cdot 4 \cdot (4 - 8 + 4) = 0$ | |

The move marked is accepted.

Iteration 3 (move): packing patterns $\{3\}, \{2, 4\}, \{1, 5, 7\}, \{6\}$, loads $(2, 10, 10, 7)$, $nb = 4$, $SQ = 253$.

| move | loads | gain | |
|---|---|---|---|
| item 1: bin $3 \to 1$ | $(4, 10, 8, 7)$ | $2 \cdot 2 \cdot (2 - 10 + 2) = -24$ | |
| item 1: bin $3 \to 4$ | $(2, 10, 8, 9)$ | $2 \cdot 2 \cdot (7 - 10 + 2) = -4$ | |
| item 2: bin $2 \to 1$ | $(5, 7, 10, 7)$ | $2 \cdot 3 \cdot (2 - 10 + 3) = -30$ | |
| item 2: bin $2 \to 4$ | $(2, 7, 10, 10)$ | $2 \cdot 3 \cdot (7 - 10 + 3) = 0$ | |
| item 3: bin $1 \to 4$ | $(10, 10, 9)$ | $2 \cdot 2 \cdot (7 - 2 + 2) = 28$ | $\leftarrow$ |
| item 4: bin $2 \to 1$ | $(9, 3, 10, 7)$ | $2 \cdot 7 \cdot (2 - 10 + 7) = -14$ | |
| item 5: bin $3 \to 1$ | $(6, 10, 6, 7)$ | $2 \cdot 4 \cdot (2 - 10 + 4) = -32$ | |
| item 6: bin $4 \to 1$ | $(9, 10, 10)$ | $2 \cdot 7 \cdot (2 - 7 + 7) = 28$ | |
| item 7: bin $3 \to 1$ | $(6, 10, 6, 7)$ | $2 \cdot 4 \cdot (2 - 10 + 4) = -32$ | |

The move marked is accepted.

Iteration 4 (move): packing patterns $\{2, 4\}, \{1, 5, 7\}, \{3, 6\}$, loads $(10, 10, 9)$, $nb = 3$, $SQ = 281$. There is no feasible move.

The search makes 3 moves, with $SQ$: $211 \to 229 \to 253 \to 281$, and stops with packing patterns $\{2, 4\}, \{1, 5, 7\}, \{3, 6\}$ and $nb = 3$.

4. The final solution uses $3 = \lceil 29/10 \rceil$ bins: it is **optimal**. With $nb$ the search sees no difference between the neighbours, since only a move that empties a bin changes $nb$; with $SQ$ the moves that make a heavy bin heavier and a light bin lighter are improving, and they prepare the move that empties a bin (which always increases $SQ$).

## Exercise 3.5 — Local search BPP 2

Bins of capacity $c = 10$ and a set $\mathscr{I} = \{1, 2, 3, 4, 5, 6\}$ of 6 items with weights

| | $j = 1$ | $j = 2$ | $j = 3$ | $j = 4$ | $j = 5$ | $j = 6$ |
|---|---|---|---|---|---|---|
| $w_j$ | 2 | 3 | 4 | 5 | 4 | 2 |

The bins are labelled $1, 2, \ldots$ in the order in which $FF$ opens them, and keep their label. A *move* puts one item into another non-empty bin where it fits (examined for $j = 1, \ldots, n$ and, for each $j$, for increasing bin label); a *swap* exchanges two items $j < k$ of different weight in different bins, if both bins stay feasible. The auxiliary objective $SQ = \sum_b L_b^2$ is maximized; the gain of a transfer of $\delta$ from bin $a$ to bin $b$ is $2\delta(L_b - L_a + \delta)$ (a move of item $j$ transfers $\delta = w_j$, a swap of item $j$ in bin $a$ and item $k$ in bin $b$ transfers $\delta = w_j - w_k$). Best improvement breaks ties in favour of the move examined first.

**Questions**

1. Compute the solution of First Fit ($FF$) and the lower bound $\lceil \sum_j w_j / c \rceil$.
2. Apply the move local search with the auxiliary objective $SQ$ and the best-improvement rule, starting from the solution of $FF$.

3. Apply the swap local search with the objective $SQ$ and the best-improvement rule, starting from the solution of $FF$.

4. Apply the variable neighbourhood descent with $N_1 =$ move and $N_2 =$ swap (objective $SQ$, best improvement), starting from the solution of $FF$. Is the result optimal? What do the three local searches show?

**Solution**

1. 
   - item 1 ($w_1 = 2$) goes into bin 1, opened for it, load 2;
   - item 2 ($w_2 = 3$) goes into bin 1, load 5;
   - item 3 ($w_3 = 4$) goes into bin 1, load 9;
   - item 4 ($w_4 = 5$) fits in no open bin: bin 2 is opened, load 5;
   - item 5 ($w_5 = 4$) goes into bin 2, load 9;
   - item 6 ($w_6 = 2$) fits in no open bin: bin 3 is opened, load 2;

   $FF$ uses $nb = 3$ bins, with packing patterns $\{1,2,3\},\{4,5\},\{6\}$ and loads $(9,9,2)$. The lower bound is $\lceil 20/10 \rceil = 2$.

2. Iteration 1 (move): packing patterns $\{1,2,3\},\{4,5\},\{6\}$, loads $(9,9,2)$, $nb = 3$, $SQ = 166$.

| move | loads | gain |
|---|---|---|
| item 1: bin $1 \to 3$ | $(7,9,4)$ | $2 \cdot 2 \cdot (2-9+2) = -20$ |
| item 2: bin $1 \to 3$ | $(6,9,5)$ | $2 \cdot 3 \cdot (2-9+3) = -24$ |
| item 3: bin $1 \to 3$ | $(5,9,6)$ | $2 \cdot 4 \cdot (2-9+4) = -24$ |
| item 4: bin $2 \to 3$ | $(9,4,7)$ | $2 \cdot 5 \cdot (2-9+5) = -20$ |
| item 5: bin $2 \to 3$ | $(9,5,6)$ | $2 \cdot 4 \cdot (2-9+4) = -24$ |

   No move is improving.

   The solution of $FF$ is a local optimum for move, with 3 bins.

3. Iteration 1 (swap): packing patterns $\{1,2,3\},\{4,5\},\{6\}$, loads $(9,9,2)$, $nb = 3$, $SQ = 166$.

| swap | loads | gain | |
|---|---|---|---|
| items 2, 5 (bins 1, 2) | $(10,8,2)$ | $2 \cdot (-1) \cdot (9-9-1) = 2$ | $\leftarrow$ |
| items 2, 6 (bins 1, 3) | $(8,9,3)$ | $2 \cdot (1) \cdot (2-9+1) = -12$ | |
| items 3, 4 (bins 1, 2) | $(10,8,2)$ | $2 \cdot (-1) \cdot (9-9-1) = 2$ | |
| items 3, 6 (bins 1, 3) | $(7,9,4)$ | $2 \cdot (2) \cdot (2-9+2) = -20$ | |
| items 4, 6 (bins 2, 3) | $(9,6,5)$ | $2 \cdot (3) \cdot (2-9+3) = -24$ | |
| items 5, 6 (bins 2, 3) | $(9,7,4)$ | $2 \cdot (2) \cdot (2-9+2) = -20$ | |

   The swap marked is accepted.

   Iteration 2 (swap): packing patterns $\{1,3,5\},\{2,4\},\{6\}$, loads $(10,8,2)$, $nb = 3$, $SQ = 168$.

| swap | loads | gain |
|---|---|---|
| items $2,3$ (bins $2,1$) | $(9,9,2)$ | $2\cdot(-1)\cdot(10-8-1)=-2$ |
| items $2,5$ (bins $2,1$) | $(9,9,2)$ | $2\cdot(-1)\cdot(10-8-1)=-2$ |
| items $2,6$ (bins $2,3$) | $(10,7,3)$ | $2\cdot(1)\cdot(2-8+1)=-10$ |
| items $3,6$ (bins $1,3$) | $(8,8,4)$ | $2\cdot(2)\cdot(2-10+2)=-24$ |
| items $4,6$ (bins $2,3$) | $(10,5,5)$ | $2\cdot(3)\cdot(2-8+3)=-18$ |
| items $5,6$ (bins $1,3$) | $(8,8,4)$ | $2\cdot(2)\cdot(2-10+2)=-24$ |

No swap is improving.

The swap local search stops with packing patterns $\{1,3,5\},\{2,4\},\{6\}$, $SQ=168$, still 3 bins (a swap never changes $nb$).

4. Iteration 1 (move): packing patterns $\{1,2,3\},\{4,5\},\{6\}$, loads $(9,9,2)$, $nb=3$, $SQ=166$.

| move | loads | gain |
|---|---|---|
| item 1: bin $1\to3$ | $(7,9,4)$ | $2\cdot2\cdot(2-9+2)=-20$ |
| item 2: bin $1\to3$ | $(6,9,5)$ | $2\cdot3\cdot(2-9+3)=-24$ |
| item 3: bin $1\to3$ | $(5,9,6)$ | $2\cdot4\cdot(2-9+4)=-24$ |
| item 4: bin $2\to3$ | $(9,4,7)$ | $2\cdot5\cdot(2-9+5)=-20$ |
| item 5: bin $2\to3$ | $(9,5,6)$ | $2\cdot4\cdot(2-9+4)=-24$ |

No move is improving.

Iteration 2 (swap): packing patterns $\{1,2,3\},\{4,5\},\{6\}$, loads $(9,9,2)$, $nb=3$, $SQ=166$.

| swap | loads | gain | |
|---|---|---|---|
| items $2,5$ (bins $1,2$) | $(10,8,2)$ | $2\cdot(-1)\cdot(9-9-1)=2$ | $\leftarrow$ |
| items $2,6$ (bins $1,3$) | $(8,9,3)$ | $2\cdot(1)\cdot(2-9+1)=-12$ | |
| items $3,4$ (bins $1,2$) | $(10,8,2)$ | $2\cdot(-1)\cdot(9-9-1)=2$ | |
| items $3,6$ (bins $1,3$) | $(7,9,4)$ | $2\cdot(2)\cdot(2-9+2)=-20$ | |
| items $4,6$ (bins $2,3$) | $(9,6,5)$ | $2\cdot(3)\cdot(2-9+3)=-24$ | |
| items $5,6$ (bins $2,3$) | $(9,7,4)$ | $2\cdot(2)\cdot(2-9+2)=-20$ | |

The swap marked is accepted, and the search goes back to the move neighbourhood.

Iteration 3 (move): packing patterns $\{1,3,5\},\{2,4\},\{6\}$, loads $(10,8,2)$, $nb=3$, $SQ=168$.

| move | loads | gain | |
|---|---|---|---|
| item 1: bin $1 \to 2$ | $(8, 10, 2)$ | $2 \cdot 2 \cdot (8 - 10 + 2) = 0$ | |
| item 1: bin $1 \to 3$ | $(8, 8, 4)$ | $2 \cdot 2 \cdot (2 - 10 + 2) = -24$ | |
| item 2: bin $2 \to 3$ | $(10, 5, 5)$ | $2 \cdot 3 \cdot (2 - 8 + 3) = -18$ | |
| item 3: bin $1 \to 3$ | $(6, 8, 6)$ | $2 \cdot 4 \cdot (2 - 10 + 4) = -32$ | |
| item 4: bin $2 \to 3$ | $(10, 3, 7)$ | $2 \cdot 5 \cdot (2 - 8 + 5) = -10$ | |
| item 5: bin $1 \to 3$ | $(6, 8, 6)$ | $2 \cdot 4 \cdot (2 - 10 + 4) = -32$ | |
| item 6: bin $3 \to 2$ | $(10, 10)$ | $2 \cdot 2 \cdot (8 - 2 + 2) = 32$ | $\leftarrow$ |

The move marked is accepted.

Iteration 4 (move): packing patterns $\{1, 3, 5\}, \{2, 4, 6\}$, loads $(10, 10)$, $nb = 2$, $SQ = 200$. There is no feasible move.

Iteration 5 (swap): packing patterns $\{1, 3, 5\}, \{2, 4, 6\}$, loads $(10, 10)$, $nb = 2$, $SQ = 200$. There is no feasible swap.

$VND$ stops with packing patterns $\{1, 3, 5\}, \{2, 4, 6\}$: $2 = \lceil 20/10 \rceil$ bins, an **optimal** solution. The solution of $FF$ is a local optimum for move, the swap alone cannot change the number of bins, but the swap prepares a move that empties a bin: the local optimum **depends on the neighbourhood**, and combining the two neighbourhoods reaches the optimum.

## Exercise 3.6 — Local search BPP 3

Bins of capacity $c = 10$ and a set $\mathscr{I} = \{1, 2, 3, 4, 5, 6, 7\}$ of 7 items with weights

| | $j = 1$ | $j = 2$ | $j = 3$ | $j = 4$ | $j = 5$ | $j = 6$ | $j = 7$ |
|---|---|---|---|---|---|---|---|
| $w_j$ | 5 | 3 | 3 | 2 | 8 | 4 | 4 |

The bins are labelled $1, 2, \ldots$ in the order in which $FF$ opens them, and keep their label. A *move* puts one item into another non-empty bin where it fits (examined for $j = 1, \ldots, n$ and, for each $j$, for increasing bin label); a *swap* exchanges two items $j < k$ of different weight in different bins, if both bins stay feasible. The auxiliary objective $SQ = \sum_b L_b^2$ is maximized; the gain of a transfer of $\delta$ from bin $a$ to bin $b$ is $2\delta(L_b - L_a + \delta)$ (a move of item $j$ transfers $\delta = w_j$, a swap of item $j$ in bin $a$ and item $k$ in bin $b$ transfers $\delta = w_j - w_k$). Best improvement breaks ties in favour of the move examined first.

**Questions**

1. Compute the solution of First Fit ($FF$) and the lower bound $\lceil \sum_j w_j / c \rceil$.
2. Apply the variable neighbourhood descent with $N_1 =$ move and $N_2 =$ swap (objective $SQ$, best improvement), starting from the solution of $FF$.
3. Find a packing that uses $\lceil \sum_j w_j / c \rceil$ bins.
4. Explain why the local search cannot reach that packing from its local optimum.

**Solution**

1. 
   - item 1 ($w_1 = 5$) goes into bin 1, opened for it, load 5;
   - item 2 ($w_2 = 3$) goes into bin 1, load 8;
   - item 3 ($w_3 = 3$) fits in no open bin: bin 2 is opened, load 3;
   - item 4 ($w_4 = 2$) goes into bin 1, load 10;
   - item 5 ($w_5 = 8$) fits in no open bin: bin 3 is opened, load 8;
   - item 6 ($w_6 = 4$) goes into bin 2, load 7;
   - item 7 ($w_7 = 4$) fits in no open bin: bin 4 is opened, load 4;

   $FF$ uses $nb = 4$ bins, with packing patterns $\{1,2,4\},\{3,6\},\{5\},\{7\}$ and loads $(10,7,8,4)$. The lower bound is $\lceil 29/10 \rceil = 3$.

2. Iteration 1 (move): packing patterns $\{1,2,4\},\{3,6\},\{5\},\{7\}$, loads $(10,7,8,4)$, $nb = 4$, $SQ = 229$.

| move | loads | gain | |
|---|---|---|---|
| item 1: bin $1 \to 4$ | $(5,7,8,9)$ | $2 \cdot 5 \cdot (4 - 10 + 5) = -10$ | |
| item 2: bin $1 \to 2$ | $(7,10,8,4)$ | $2 \cdot 3 \cdot (7 - 10 + 3) = 0$ | |
| item 2: bin $1 \to 4$ | $(7,7,8,7)$ | $2 \cdot 3 \cdot (4 - 10 + 3) = -18$ | |
| item 3: bin $2 \to 4$ | $(10,4,8,7)$ | $2 \cdot 3 \cdot (4 - 7 + 3) = 0$ | |
| item 4: bin $1 \to 2$ | $(8,9,8,4)$ | $2 \cdot 2 \cdot (7 - 10 + 2) = -4$ | |
| item 4: bin $1 \to 3$ | $(8,7,10,4)$ | $2 \cdot 2 \cdot (8 - 10 + 2) = 0$ | |
| item 4: bin $1 \to 4$ | $(8,7,8,6)$ | $2 \cdot 2 \cdot (4 - 10 + 2) = -16$ | |
| item 6: bin $2 \to 4$ | $(10,3,8,8)$ | $2 \cdot 4 \cdot (4 - 7 + 4) = 8$ | $\leftarrow$ |

   The move marked is accepted.

   Iteration 2 (move): packing patterns $\{1,2,4\},\{3\},\{5\},\{6,7\}$, loads $(10,3,8,8)$, $nb = 4$, $SQ = 237$.

| move | loads | gain |
|---|---|---|
| item 1: bin $1 \to 2$ | $(5,8,8,8)$ | $2 \cdot 5 \cdot (3 - 10 + 5) = -20$ |
| item 2: bin $1 \to 2$ | $(7,6,8,8)$ | $2 \cdot 3 \cdot (3 - 10 + 3) = -24$ |
| item 4: bin $1 \to 2$ | $(8,5,8,8)$ | $2 \cdot 2 \cdot (3 - 10 + 2) = -20$ |
| item 4: bin $1 \to 3$ | $(8,3,10,8)$ | $2 \cdot 2 \cdot (8 - 10 + 2) = 0$ |
| item 4: bin $1 \to 4$ | $(8,3,8,10)$ | $2 \cdot 2 \cdot (8 - 10 + 2) = 0$ |
| item 6: bin $4 \to 2$ | $(10,7,8,4)$ | $2 \cdot 4 \cdot (3 - 8 + 4) = -8$ |
| item 7: bin $4 \to 2$ | $(10,7,8,4)$ | $2 \cdot 4 \cdot (3 - 8 + 4) = -8$ |

   No move is improving.

   Iteration 3 (swap): packing patterns $\{1,2,4\},\{3\},\{5\},\{6,7\}$, loads $(10,3,8,8)$, $nb = 4$, $SQ = 237$.

| swap | loads | gain |
|---|---|---|
| items $1,3$ (bins $1,2$) | $(8,5,8,8)$ | $2\cdot(2)\cdot(3-10+2)=-20$ |
| items $1,6$ (bins $1,4$) | $(9,3,8,9)$ | $2\cdot(1)\cdot(8-10+1)=-2$ |
| items $1,7$ (bins $1,4$) | $(9,3,8,9)$ | $2\cdot(1)\cdot(8-10+1)=-2$ |
| items $3,5$ (bins $2,3$) | $(10,8,3,8)$ | $2\cdot(-5)\cdot(8-3-5)=0$ |
| items $3,6$ (bins $2,4$) | $(10,4,8,7)$ | $2\cdot(-1)\cdot(8-3-1)=-8$ |
| items $3,7$ (bins $2,4$) | $(10,4,8,7)$ | $2\cdot(-1)\cdot(8-3-1)=-8$ |

No swap is improving.

$VND$ stops with packing patterns $\{1,2,4\},\{3\},\{5\},\{6,7\}$, loads $(10,3,8,8)$, $SQ=237$ and $nb=4$ bins.

3. The packing patterns $\{1,6\},\{4,5\},\{2,3,7\}$, of loads $(9,10,10)$, use 3 bins: since $3=\lceil 29/10\rceil$, this packing is **optimal**, and the local optimum is not.

4. In the optimal packing items 1 and 2 are in different bins, while in the local optimum they are together in the full bin $\{1,2,4\}$. To separate them, an item must leave a full bin (load 10) for a lighter one: by the gain formula $2\delta(L_b-L_a+\delta)$ such a move is improving only if $L_b+\delta>10$, which is impossible. The auxiliary objective protects full bins, which is usually good, but here the full bin of $FF$ is the wrong one: the search is trapped in a local optimum.

## 3.3 TSP: 2-opt

**Exercise 3.7 — Local search TSP 1**

6 cities with the symmetric distances

| $d_{ij}$ | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| 1 | — | 7 | 6 | 3 | 4 | 6 |
| 2 | 7 | — | 6 | 5 | 7 | 4 |
| 3 | 6 | 6 | — | 5 | 5 | 2 |
| 4 | 3 | 5 | 5 | — | 5 | 3 |
| 5 | 4 | 7 | 5 | 5 | — | 4 |
| 6 | 6 | 4 | 2 | 3 | 4 | — |

A tour $\sigma=(\sigma_1,\dots,\sigma_n)$ keeps city 1 in the first position. The 2-opt move $(i,j)$, for $i=1,\dots,n-2$ and $j=i+2,\dots,n$ (except $(1,n)$), examined in this order, removes the edges $\{\sigma_i,\sigma_{i+1}\}$, $\{\sigma_j,\sigma_{j+1}\}$ (with $\sigma_{n+1}=\sigma_1$), reverses $\sigma_{i+1},\dots,\sigma_j$, and changes the length by $\Delta=d_{\sigma_i\sigma_j}+d_{\sigma_{i+1}\sigma_{j+1}}-d_{\sigma_i\sigma_{i+1}}-d_{\sigma_j\sigma_{j+1}}$. Best improvement breaks ties in favour of the move examined first. The nearest neighbour heuristic $NN$ breaks ties in favour of the city of smallest index.

**Questions**

1. Compute the tour of $NN$ from city 1 and its length.
2. Compute $\Delta$ for all the 2-opt moves of the tour of $NN$, and check one of them by computing the length of the new tour directly.
3. Apply the 2-opt local search with the best-improvement rule, starting from the tour of $NN$.
4. The optimal value is $z(I) = 23$. Is the final tour optimal? How many moves were made, compared with the bound $L(\sigma^0) - z(I)$ on the number of moves?

**Solution**

1. 
   - from city 1 the nearest unvisited city is 4 ($d_{1,4} = 3$);
   - from city 4 the nearest unvisited city is 6 ($d_{4,6} = 3$);
   - from city 6 the nearest unvisited city is 3 ($d_{6,3} = 2$);
   - from city 3 the nearest unvisited city is 5 ($d_{3,5} = 5$);
   - from city 5 the nearest unvisited city is 2 ($d_{5,2} = 7$);
   - the tour goes back to city 1 ($d_{2,1} = 7$).

   The tour is $\sigma = (1, 4, 6, 3, 5, 2)$, of length 27.

2. The 9 moves of the tour of $NN$ are

| $(i, j)$ | removed edges | $\Delta$ | new tour | length |
|---|---|---|---|---|
| $(1, 3)$ | $\{1, 4\}, \{6, 3\}$ | $6 + 5 - 3 - 2 = +6$ | $(1, 6, 4, 3, 5, 2)$ | 33 |
| $(1, 4)$ | $\{1, 4\}, \{3, 5\}$ | $6 + 5 - 3 - 5 = +3$ | $(1, 3, 6, 4, 5, 2)$ | 30 |
| $(1, 5)$ | $\{1, 4\}, \{5, 2\}$ | $4 + 5 - 3 - 7 = -1$ | $(1, 5, 3, 6, 4, 2)$ | 26 |
| $(2, 4)$ | $\{4, 6\}, \{3, 5\}$ | $5 + 4 - 3 - 5 = +1$ | $(1, 4, 3, 6, 5, 2)$ | 28 |
| $(2, 5)$ | $\{4, 6\}, \{5, 2\}$ | $5 + 4 - 3 - 7 = -1$ | $(1, 4, 5, 3, 6, 2)$ | 26 |
| $(2, 6)$ | $\{4, 6\}, \{2, 1\}$ | $5 + 6 - 3 - 7 = +1$ | $(1, 4, 2, 5, 3, 6)$ | 28 |
| $(3, 5)$ | $\{6, 3\}, \{5, 2\}$ | $4 + 6 - 2 - 7 = +1$ | $(1, 4, 6, 5, 3, 2)$ | 28 |
| $(3, 6)$ | $\{6, 3\}, \{2, 1\}$ | $4 + 6 - 2 - 7 = +1$ | $(1, 4, 6, 2, 5, 3)$ | 28 |
| $(4, 6)$ | $\{3, 5\}, \{2, 1\}$ | $6 + 4 - 5 - 7 = -2$ | $(1, 4, 6, 3, 2, 5)$ | 25 |

   For example, the move $(1, 3)$ gives the tour $(1, 6, 4, 3, 5, 2)$, of length $6+3+5+5+7+7 = 33 = 27 + 6$.

3. Iteration 1: $\sigma = (1, 4, 6, 3, 5, 2)$, length 27.

| $(i,j)$ | removed edges | $\Delta$ | new tour | length | |
|---|---|---|---|---|---|
| $(1,3)$ | $\{1,4\},\{6,3\}$ | $6+5-3-2=+6$ | $(1,6,4,3,5,2)$ | 33 | |
| $(1,4)$ | $\{1,4\},\{3,5\}$ | $6+5-3-5=+3$ | $(1,3,6,4,5,2)$ | 30 | |
| $(1,5)$ | $\{1,4\},\{5,2\}$ | $4+5-3-7=-1$ | $(1,5,3,6,4,2)$ | 26 | |
| $(2,4)$ | $\{4,6\},\{3,5\}$ | $5+4-3-5=+1$ | $(1,4,3,6,5,2)$ | 28 | |
| $(2,5)$ | $\{4,6\},\{5,2\}$ | $5+4-3-7=-1$ | $(1,4,5,3,6,2)$ | 26 | |
| $(2,6)$ | $\{4,6\},\{2,1\}$ | $5+6-3-7=+1$ | $(1,4,2,5,3,6)$ | 28 | |
| $(3,5)$ | $\{6,3\},\{5,2\}$ | $4+6-2-7=+1$ | $(1,4,6,5,3,2)$ | 28 | |
| $(3,6)$ | $\{6,3\},\{2,1\}$ | $4+6-2-7=+1$ | $(1,4,6,2,5,3)$ | 28 | |
| $(4,6)$ | $\{3,5\},\{2,1\}$ | $6+4-5-7=-2$ | $(1,4,6,3,2,5)$ | 25 | $\leftarrow$ |

The move $(4,6)$ is accepted after 9 evaluations.

Iteration 2: $\sigma=(1,4,6,3,2,5)$, length 25.

| $(i,j)$ | removed edges | $\Delta$ | new tour | length | |
|---|---|---|---|---|---|
| $(1,3)$ | $\{1,4\},\{6,3\}$ | $6+5-3-2=+6$ | $(1,6,4,3,2,5)$ | 31 | |
| $(1,4)$ | $\{1,4\},\{3,2\}$ | $6+5-3-6=+2$ | $(1,3,6,4,2,5)$ | 27 | |
| $(1,5)$ | $\{1,4\},\{2,5\}$ | $7+5-3-7=+2$ | $(1,2,3,6,4,5)$ | 27 | |
| $(2,4)$ | $\{4,6\},\{3,2\}$ | $5+4-3-6=0$ | $(1,4,3,6,2,5)$ | 25 | |
| $(2,5)$ | $\{4,6\},\{2,5\}$ | $5+4-3-7=-1$ | $(1,4,2,3,6,5)$ | 24 | $\leftarrow$ |
| $(2,6)$ | $\{4,6\},\{5,1\}$ | $5+6-3-4=+4$ | $(1,4,5,2,3,6)$ | 29 | |
| $(3,5)$ | $\{6,3\},\{2,5\}$ | $4+5-2-7=0$ | $(1,4,6,2,3,5)$ | 25 | |
| $(3,6)$ | $\{6,3\},\{5,1\}$ | $4+6-2-4=+4$ | $(1,4,6,5,2,3)$ | 29 | |
| $(4,6)$ | $\{3,2\},\{5,1\}$ | $5+7-6-4=+2$ | $(1,4,6,3,5,2)$ | 27 | |

The move $(2,5)$ is accepted after 9 evaluations.

Iteration 3: $\sigma=(1,4,2,3,6,5)$, length 24.

| $(i,j)$ | removed edges | $\Delta$ | new tour | length | |
|---|---|---|---|---|---|
| $(1,3)$ | $\{1,4\},\{2,3\}$ | $7+5-3-6=+3$ | $(1,2,4,3,6,5)$ | 27 | |
| $(1,4)$ | $\{1,4\},\{3,6\}$ | $6+3-3-2=+4$ | $(1,3,2,4,6,5)$ | 28 | |
| $(1,5)$ | $\{1,4\},\{6,5\}$ | $6+5-3-4=+4$ | $(1,6,3,2,4,5)$ | 28 | |
| $(2,4)$ | $\{4,2\},\{3,6\}$ | $5+4-5-2=+2$ | $(1,4,3,2,6,5)$ | 26 | |
| $(2,5)$ | $\{4,2\},\{6,5\}$ | $3+7-5-4=+1$ | $(1,4,6,3,2,5)$ | 25 | |
| $(2,6)$ | $\{4,2\},\{5,1\}$ | $5+7-5-4=+3$ | $(1,4,5,6,3,2)$ | 27 | |
| $(3,5)$ | $\{2,3\},\{6,5\}$ | $4+5-6-4=-1$ | $(1,4,2,6,3,5)$ | 23 | $\leftarrow$ |
| $(3,6)$ | $\{2,3\},\{5,1\}$ | $7+6-6-4=+3$ | $(1,4,2,5,6,3)$ | 27 | |
| $(4,6)$ | $\{3,6\},\{5,1\}$ | $5+6-2-4=+5$ | $(1,4,2,3,5,6)$ | 29 | |

The move $(3,5)$ is accepted after 9 evaluations.

Iteration 4: $\sigma=(1,4,2,6,3,5)$, length 23.

| $(i,j)$ | removed edges | $\Delta$ | new tour | length |
|---|---|---|---|---|
| $(1,3)$ | $\{1,4\},\{2,6\}$ | $7+3-3-4=+3$ | $(1,2,4,6,3,5)$ | 26 |
| $(1,4)$ | $\{1,4\},\{6,3\}$ | $6+5-3-2=+6$ | $(1,6,2,4,3,5)$ | 29 |
| $(1,5)$ | $\{1,4\},\{3,5\}$ | $6+5-3-5=+3$ | $(1,3,6,2,4,5)$ | 26 |
| $(2,4)$ | $\{4,2\},\{6,3\}$ | $3+6-5-2=+2$ | $(1,4,6,2,3,5)$ | 25 |
| $(2,5)$ | $\{4,2\},\{3,5\}$ | $5+7-5-5=+2$ | $(1,4,3,6,2,5)$ | 25 |
| $(2,6)$ | $\{4,2\},\{5,1\}$ | $5+7-5-4=+3$ | $(1,4,5,3,6,2)$ | 26 |
| $(3,5)$ | $\{2,6\},\{3,5\}$ | $6+4-4-5=+1$ | $(1,4,2,3,6,5)$ | 24 |
| $(3,6)$ | $\{2,6\},\{5,1\}$ | $7+6-4-4=+5$ | $(1,4,2,5,3,6)$ | 28 |
| $(4,6)$ | $\{6,3\},\{5,1\}$ | $4+6-2-4=+4$ | $(1,4,2,6,5,3)$ | 27 |

No move has $\Delta < 0$: $\sigma = (1,4,2,6,3,5)$ is a 2-opt **local optimum** (9 evaluations).

The search makes 3 moves: $27 \to 25 \to 24 \to 23$.

4. The final tour has length $23 = z(I)$: it is **optimal**. The search made 3 moves, while the termination property of the notes allows at most $27 - 23 = 4$.

## Exercise 3.8 — Local search TSP 2

6 cities with the symmetric distances

| $d_{ij}$ | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| 1 | — | 8 | 5 | 6 | 5 | 6 |
| 2 | 8 | — | 6 | 7 | 7 | 6 |
| 3 | 5 | 6 | — | 4 | 8 | 2 |
| 4 | 6 | 7 | 4 | — | 6 | 5 |
| 5 | 5 | 7 | 8 | 6 | — | 7 |
| 6 | 6 | 6 | 2 | 5 | 7 | — |

A tour $\sigma = (\sigma_1, \ldots, \sigma_n)$ keeps city 1 in the first position. The 2-opt move $(i,j)$, for $i = 1, \ldots, n-2$ and $j = i+2, \ldots, n$ (except $(1,n)$), examined in this order, removes the edges $\{\sigma_i, \sigma_{i+1}\}$, $\{\sigma_j, \sigma_{j+1}\}$ (with $\sigma_{n+1} = \sigma_1$), reverses $\sigma_{i+1}, \ldots, \sigma_j$, and changes the length by $\Delta = d_{\sigma_i\sigma_j} + d_{\sigma_{i+1}\sigma_{j+1}} - d_{\sigma_i\sigma_{i+1}} - d_{\sigma_j\sigma_{j+1}}$. Best improvement breaks ties in favour of the move examined first. The nearest neighbour heuristic $NN$ breaks ties in favour of the city of smallest index.

**Questions**

1. Compute the tour of $NN$ from city 1 and its length.
2. Apply the 2-opt local search with the **first-improvement** rule, starting from the tour of $NN$.
3. Apply the 2-opt local search with the **best-improvement** rule, starting from the tour of $NN$.

4. The optimal value is $z(I) = 30$. Compare the two local optima and the number of evaluations of the two rules.

**Solution**

1. 
   - from city 1 the nearest unvisited city is 3 ($d_{1,3} = 5$);
   - from city 3 the nearest unvisited city is 6 ($d_{3,6} = 2$);
   - from city 6 the nearest unvisited city is 4 ($d_{6,4} = 5$);
   - from city 4 the nearest unvisited city is 5 ($d_{4,5} = 6$);
   - from city 5 the nearest unvisited city is 2 ($d_{5,2} = 7$);
   - the tour goes back to city 1 ($d_{2,1} = 8$).

   The tour is $\sigma = (1, 3, 6, 4, 5, 2)$, of length 33.

2. Iteration 1: $\sigma = (1, 3, 6, 4, 5, 2)$, length 33.

| $(i, j)$ | removed edges | $\Delta$ | new tour | length | |
|---|---|---|---|---|---|
| $(1, 3)$ | $\{1, 3\}, \{6, 4\}$ | $6 + 4 - 5 - 5 = 0$ | $(1, 6, 3, 4, 5, 2)$ | 33 | |
| $(1, 4)$ | $\{1, 3\}, \{4, 5\}$ | $6 + 8 - 5 - 6 = +3$ | $(1, 4, 6, 3, 5, 2)$ | 36 | |
| $(1, 5)$ | $\{1, 3\}, \{5, 2\}$ | $5 + 6 - 5 - 7 = -1$ | $(1, 5, 4, 6, 3, 2)$ | 32 | $\leftarrow$ |
| $(2, 4)$ | $\{3, 6\}, \{4, 5\}$ | $4 + 7 - 2 - 6 = +3$ | $(1, 3, 4, 6, 5, 2)$ | 36 | |
| $(2, 5)$ | $\{3, 6\}, \{5, 2\}$ | $8 + 6 - 2 - 7 = +5$ | $(1, 3, 5, 4, 6, 2)$ | 38 | |
| $(2, 6)$ | $\{3, 6\}, \{2, 1\}$ | $6 + 6 - 2 - 8 = +2$ | $(1, 3, 2, 5, 4, 6)$ | 35 | |
| $(3, 5)$ | $\{6, 4\}, \{5, 2\}$ | $7 + 7 - 5 - 7 = +2$ | $(1, 3, 6, 5, 4, 2)$ | 35 | |
| $(3, 6)$ | $\{6, 4\}, \{2, 1\}$ | $6 + 6 - 5 - 8 = -1$ | $(1, 3, 6, 2, 5, 4)$ | 32 | |
| $(4, 6)$ | $\{4, 5\}, \{2, 1\}$ | $7 + 5 - 6 - 8 = -2$ | $(1, 3, 6, 4, 2, 5)$ | 31 | |

   The move $(1, 5)$ is accepted after 3 evaluations.

   Iteration 2: $\sigma = (1, 5, 4, 6, 3, 2)$, length 32.

| $(i, j)$ | removed edges | $\Delta$ | new tour | length | |
|---|---|---|---|---|---|
| $(1, 3)$ | $\{1, 5\}, \{4, 6\}$ | $6 + 7 - 5 - 5 = +3$ | $(1, 4, 5, 6, 3, 2)$ | 35 | |
| $(1, 4)$ | $\{1, 5\}, \{6, 3\}$ | $6 + 8 - 5 - 2 = +7$ | $(1, 6, 4, 5, 3, 2)$ | 39 | |
| $(1, 5)$ | $\{1, 5\}, \{3, 2\}$ | $5 + 7 - 5 - 6 = +1$ | $(1, 3, 6, 4, 5, 2)$ | 33 | |
| $(2, 4)$ | $\{5, 4\}, \{6, 3\}$ | $7 + 4 - 6 - 2 = +3$ | $(1, 5, 6, 4, 3, 2)$ | 35 | |
| $(2, 5)$ | $\{5, 4\}, \{3, 2\}$ | $8 + 7 - 6 - 6 = +3$ | $(1, 5, 3, 6, 4, 2)$ | 35 | |
| $(2, 6)$ | $\{5, 4\}, \{2, 1\}$ | $7 + 6 - 6 - 8 = -1$ | $(1, 5, 2, 3, 6, 4)$ | 31 | $\leftarrow$ |
| $(3, 5)$ | $\{4, 6\}, \{3, 2\}$ | $4 + 6 - 5 - 6 = -1$ | $(1, 5, 4, 3, 6, 2)$ | 31 | |
| $(3, 6)$ | $\{4, 6\}, \{2, 1\}$ | $7 + 6 - 5 - 8 = 0$ | $(1, 5, 4, 2, 3, 6)$ | 32 | |
| $(4, 6)$ | $\{6, 3\}, \{2, 1\}$ | $6 + 5 - 2 - 8 = +1$ | $(1, 5, 4, 6, 2, 3)$ | 33 | |

   The move $(2, 6)$ is accepted after 6 evaluations.

   Iteration 3: $\sigma = (1, 5, 2, 3, 6, 4)$, length 31.

| $(i,j)$ | removed edges | $\Delta$ | new tour | length | |
|---|---|---|---|---|---|
| $(1,3)$ | $\{1,5\},\{2,3\}$ | $8+8-5-6=+5$ | $(1,2,5,3,6,4)$ | 36 | |
| $(1,4)$ | $\{1,5\},\{3,6\}$ | $5+7-5-2=+5$ | $(1,3,2,5,6,4)$ | 36 | |
| $(1,5)$ | $\{1,5\},\{6,4\}$ | $6+6-5-5=+2$ | $(1,6,3,2,5,4)$ | 33 | |
| $(2,4)$ | $\{5,2\},\{3,6\}$ | $8+6-7-2=+5$ | $(1,5,3,2,6,4)$ | 36 | |
| $(2,5)$ | $\{5,2\},\{6,4\}$ | $7+7-7-5=+2$ | $(1,5,6,3,2,4)$ | 33 | |
| $(2,6)$ | $\{5,2\},\{4,1\}$ | $6+8-7-6=+1$ | $(1,5,4,6,3,2)$ | 32 | |
| $(3,5)$ | $\{2,3\},\{6,4\}$ | $6+4-6-5=-1$ | $(1,5,2,6,3,4)$ | 30 | $\leftarrow$ |
| $(3,6)$ | $\{2,3\},\{4,1\}$ | $7+5-6-6=0$ | $(1,5,2,4,6,3)$ | 31 | |
| $(4,6)$ | $\{3,6\},\{4,1\}$ | $4+6-2-6=+2$ | $(1,5,2,3,4,6)$ | 33 | |

The move $(3,5)$ is accepted after 7 evaluations.

Iteration 4: $\sigma=(1,5,2,6,3,4)$, length 30.

| $(i,j)$ | removed edges | $\Delta$ | new tour | length |
|---|---|---|---|---|
| $(1,3)$ | $\{1,5\},\{2,6\}$ | $8+7-5-6=+4$ | $(1,2,5,6,3,4)$ | 34 |
| $(1,4)$ | $\{1,5\},\{6,3\}$ | $6+8-5-2=+7$ | $(1,6,2,5,3,4)$ | 37 |
| $(1,5)$ | $\{1,5\},\{3,4\}$ | $5+6-5-4=+2$ | $(1,3,6,2,5,4)$ | 32 |
| $(2,4)$ | $\{5,2\},\{6,3\}$ | $7+6-7-2=+4$ | $(1,5,6,2,3,4)$ | 34 |
| $(2,5)$ | $\{5,2\},\{3,4\}$ | $8+7-7-4=+4$ | $(1,5,3,6,2,4)$ | 34 |
| $(2,6)$ | $\{5,2\},\{4,1\}$ | $6+8-7-6=+1$ | $(1,5,4,3,6,2)$ | 31 |
| $(3,5)$ | $\{2,6\},\{3,4\}$ | $6+5-6-4=+1$ | $(1,5,2,3,6,4)$ | 31 |
| $(3,6)$ | $\{2,6\},\{4,1\}$ | $7+6-6-6=+1$ | $(1,5,2,4,3,6)$ | 31 |
| $(4,6)$ | $\{6,3\},\{4,1\}$ | $5+5-2-6=+2$ | $(1,5,2,6,4,3)$ | 32 |

No move has $\Delta<0$: $\sigma=(1,5,2,6,3,4)$ is a 2-opt **local optimum** (9 evaluations).

First improvement makes 3 moves, $33\to32\to31\to30$, with 25 evaluations in total.

3. Iteration 1: $\sigma=(1,3,6,4,5,2)$, length 33.

| $(i,j)$ | removed edges | $\Delta$ | new tour | length | |
|---|---|---|---|---|---|
| $(1,3)$ | $\{1,3\},\{6,4\}$ | $6+4-5-5=0$ | $(1,6,3,4,5,2)$ | 33 | |
| $(1,4)$ | $\{1,3\},\{4,5\}$ | $6+8-5-6=+3$ | $(1,4,6,3,5,2)$ | 36 | |
| $(1,5)$ | $\{1,3\},\{5,2\}$ | $5+6-5-7=-1$ | $(1,5,4,6,3,2)$ | 32 | |
| $(2,4)$ | $\{3,6\},\{4,5\}$ | $4+7-2-6=+3$ | $(1,3,4,6,5,2)$ | 36 | |
| $(2,5)$ | $\{3,6\},\{5,2\}$ | $8+6-2-7=+5$ | $(1,3,5,4,6,2)$ | 38 | |
| $(2,6)$ | $\{3,6\},\{2,1\}$ | $6+6-2-8=+2$ | $(1,3,2,5,4,6)$ | 35 | |
| $(3,5)$ | $\{6,4\},\{5,2\}$ | $7+7-5-7=+2$ | $(1,3,6,5,4,2)$ | 35 | |
| $(3,6)$ | $\{6,4\},\{2,1\}$ | $6+6-5-8=-1$ | $(1,3,6,2,5,4)$ | 32 | |
| $(4,6)$ | $\{4,5\},\{2,1\}$ | $7+5-6-8=-2$ | $(1,3,6,4,2,5)$ | 31 | $\leftarrow$ |

The move $(4,6)$ is accepted after 9 evaluations.

Iteration 2: $\sigma=(1,3,6,4,2,5)$, length 31.

| $(i,j)$ | removed edges | $\Delta$ | new tour | length |
|---|---|---|---|---|
| $(1,3)$ | $\{1,3\},\{6,4\}$ | $6+4-5-5=0$ | $(1,6,3,4,2,5)$ | 31 |
| $(1,4)$ | $\{1,3\},\{4,2\}$ | $6+6-5-7=0$ | $(1,4,6,3,2,5)$ | 31 |
| $(1,5)$ | $\{1,3\},\{2,5\}$ | $8+8-5-7=+4$ | $(1,2,4,6,3,5)$ | 35 |
| $(2,4)$ | $\{3,6\},\{4,2\}$ | $4+6-2-7=+1$ | $(1,3,4,6,2,5)$ | 32 |
| $(2,5)$ | $\{3,6\},\{2,5\}$ | $6+7-2-7=+4$ | $(1,3,2,4,6,5)$ | 35 |
| $(2,6)$ | $\{3,6\},\{5,1\}$ | $8+6-2-5=+7$ | $(1,3,5,2,4,6)$ | 38 |
| $(3,5)$ | $\{6,4\},\{2,5\}$ | $6+6-5-7=0$ | $(1,3,6,2,4,5)$ | 31 |
| $(3,6)$ | $\{6,4\},\{5,1\}$ | $7+6-5-5=+3$ | $(1,3,6,5,2,4)$ | 34 |
| $(4,6)$ | $\{4,2\},\{5,1\}$ | $6+8-7-5=+2$ | $(1,3,6,4,5,2)$ | 33 |

No move has $\Delta < 0$: $\sigma = (1,3,6,4,2,5)$ is a 2-opt **local optimum** (9 evaluations).

Best improvement makes 1 move, $33 \to 31$, with 18 evaluations in total.

4. First improvement reaches a tour of length $30 = z(I)$, an **optimal** tour, while best improvement stops in a local optimum of length 31, 3.3% above the optimum. The steepest first move ($\Delta = -2$) leads into a worse valley: best improvement is not always better than first improvement. First improvement examines 25 moves in total, best improvement 18: the better local optimum costs more evaluations here, because first improvement makes more moves.

## Exercise 3.9 — Local search TSP 3

6 cities with the symmetric distances

| $d_{ij}$ | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| 1 | — | 7 | 8 | 8 | 9 | 9 |
| 2 | 7 | — | 7 | 4 | 9 | 8 |
| 3 | 8 | 7 | — | 7 | 3 | 9 |
| 4 | 8 | 4 | 7 | — | 5 | 6 |
| 5 | 9 | 9 | 3 | 5 | — | 7 |
| 6 | 9 | 8 | 9 | 6 | 7 | — |

A tour $\sigma = (\sigma_1, \ldots, \sigma_n)$ keeps city 1 in the first position. The 2-opt move $(i,j)$, for $i = 1, \ldots, n-2$ and $j = i+2, \ldots, n$ (except $(1,n)$), examined in this order, removes the edges $\{\sigma_i, \sigma_{i+1}\}$, $\{\sigma_j, \sigma_{j+1}\}$ (with $\sigma_{n+1} = \sigma_1$), reverses $\sigma_{i+1}, \ldots, \sigma_j$, and changes the length by $\Delta = d_{\sigma_i\sigma_j} + d_{\sigma_{i+1}\sigma_{j+1}} - d_{\sigma_i\sigma_{i+1}} - d_{\sigma_j\sigma_{j+1}}$. Best improvement breaks ties in favour of the move examined first. The nearest neighbour heuristic $NN$ breaks ties in favour of the city of smallest index. An Or-opt move with $\ell = 1$ takes one city $\sigma_i$ ($i \geq 2$) and inserts it between two consecutive cities $\sigma_k, \sigma_{k+1}$ not adjacent to it ($k \neq i-1, i$); with $p = \sigma_{i-1}$, $q = \sigma_{i+1}$, $s = \sigma_i$, $a = \sigma_k$, $b = \sigma_{k+1}$ it changes the length by $\Delta = d_{pq} + d_{as} + d_{sb} - d_{ps} - d_{sq} - d_{ab}$. The moves are examined for $i = 2, \ldots, n$ and $k = 1, \ldots, n$.

**Questions**

1. Compute the tour of $NN$ from city 1 and its length.
2. Show that the tour of $NN$ is a 2-opt local optimum.
3. Apply one iteration of best improvement with the Or-opt moves with $\ell = 1$ to the tour of $NN$.
4. The optimal value is $z(I) = 35$. Is the new tour optimal? Is it a 2-opt local optimum? What does this show about combining the two neighbourhoods?

**Solution**

1. 
   - from city 1 the nearest unvisited city is 2 ($d_{1,2} = 7$);
   - from city 2 the nearest unvisited city is 4 ($d_{2,4} = 4$);
   - from city 4 the nearest unvisited city is 5 ($d_{4,5} = 5$);
   - from city 5 the nearest unvisited city is 3 ($d_{5,3} = 3$);
   - from city 3 the nearest unvisited city is 6 ($d_{3,6} = 9$);
   - the tour goes back to city 1 ($d_{6,1} = 9$).

   The tour is $\sigma = (1, 2, 4, 5, 3, 6)$, of length 37.

2. The 2-opt moves of the tour of $NN$ are

| $(i, j)$ | removed edges | $\Delta$ | new tour | length |
|---|---|---|---|---|
| $(1, 3)$ | $\{1, 2\}, \{4, 5\}$ | $8 + 9 - 7 - 5 = +5$ | $(1, 4, 2, 5, 3, 6)$ | 42 |
| $(1, 4)$ | $\{1, 2\}, \{5, 3\}$ | $9 + 7 - 7 - 3 = +6$ | $(1, 5, 4, 2, 3, 6)$ | 43 |
| $(1, 5)$ | $\{1, 2\}, \{3, 6\}$ | $8 + 8 - 7 - 9 = 0$ | $(1, 3, 5, 4, 2, 6)$ | 37 |
| $(2, 4)$ | $\{2, 4\}, \{5, 3\}$ | $9 + 7 - 4 - 3 = +9$ | $(1, 2, 5, 4, 3, 6)$ | 46 |
| $(2, 5)$ | $\{2, 4\}, \{3, 6\}$ | $7 + 6 - 4 - 9 = 0$ | $(1, 2, 3, 5, 4, 6)$ | 37 |
| $(2, 6)$ | $\{2, 4\}, \{6, 1\}$ | $8 + 8 - 4 - 9 = +3$ | $(1, 2, 6, 3, 5, 4)$ | 40 |
| $(3, 5)$ | $\{4, 5\}, \{3, 6\}$ | $7 + 7 - 5 - 9 = 0$ | $(1, 2, 4, 3, 5, 6)$ | 37 |
| $(3, 6)$ | $\{4, 5\}, \{6, 1\}$ | $6 + 9 - 5 - 9 = +1$ | $(1, 2, 4, 6, 3, 5)$ | 38 |
| $(4, 6)$ | $\{5, 3\}, \{6, 1\}$ | $7 + 8 - 3 - 9 = +3$ | $(1, 2, 4, 5, 6, 3)$ | 40 |

   All the moves have $\Delta \geq 0$: the tour of $NN$ is a 2-opt **local optimum**.

3. The Or-opt moves with $\ell = 1$ of the tour of $NN$ are

| city $s$ | between | $\Delta$ | length | |
|---|---|---|---|---|
| 2 | $4,5$ | $8+4+9-7-4-5=+5$ | 42 | |
| 2 | $5,3$ | $8+9+7-7-4-3=+10$ | 47 | |
| 2 | $3,6$ | $8+7+8-7-4-9=+3$ | 40 | |
| 2 | $6,1$ | $8+8+7-7-4-9=+3$ | 40 | |
| 4 | $1,2$ | $9+8+4-4-5-7=+5$ | 42 | |
| 4 | $5,3$ | $9+5+7-4-5-3=+9$ | 46 | |
| 4 | $3,6$ | $9+7+6-4-5-9=+4$ | 41 | |
| 4 | $6,1$ | $9+6+8-4-5-9=+5$ | 42 | |
| 5 | $1,2$ | $7+9+9-5-3-7=+10$ | 47 | |
| 5 | $2,4$ | $7+9+5-5-3-4=+9$ | 46 | |
| 5 | $3,6$ | $7+3+7-5-3-9=0$ | 37 | |
| 5 | $6,1$ | $7+7+9-5-3-9=+6$ | 43 | |
| 3 | $1,2$ | $7+8+7-3-9-7=+3$ | 40 | |
| 3 | $2,4$ | $7+7+7-3-9-4=+5$ | 42 | |
| 3 | $4,5$ | $7+7+3-3-9-5=0$ | 37 | |
| 3 | $6,1$ | $7+9+8-3-9-9=+3$ | 40 | |
| 6 | $1,2$ | $8+9+8-9-9-7=0$ | 37 | |
| 6 | $2,4$ | $8+8+6-9-9-4=0$ | 37 | |
| 6 | $4,5$ | $8+6+7-9-9-5=-2$ | 35 | $\leftarrow$ |
| 6 | $5,3$ | $8+7+9-9-9-3=+3$ | 40 | |

The best move inserts city 6 between cities 4 and 5, with $\Delta=-2$: the new tour is $(1,2,4,6,5,3)$, of length 35.

4. The new tour has length $35=z(I)$: it is **optimal**, hence also a 2-opt local optimum. The tour of $NN$ is a local optimum for 2-opt but not for Or-opt: a variable neighbourhood descent with $N_1=$ 2-opt and $N_2=$ Or-opt would stop only in a tour that is a local optimum for both, and here it finds the optimum.

## 3.4 $P\|C_{\max}$

**Exercise 3.10 — Local search $P\,||\,C_{\max}$ 1**

$m=3$ identical machines and 6 jobs with processing times

| | $j=1$ | $j=2$ | $j=3$ | $j=4$ | $j=5$ | $j=6$ |
|---|---|---|---|---|---|---|
| $t_j$ | 5 | 7 | 3 | 4 | 4 | 9 |

A *move* reassigns one job $j$ to another machine $h$ (examined for $j=1,\dots,n$ and $h=1,\dots,m$); a *swap* exchanges the machines of two jobs $j<k$ on different machines with $t_j\neq t_k$. The local searches use the best-improvement rule, with ties broken in favour of the move examined first.

**Questions**

1. Compute the schedule of list scheduling ($LIST$) with the jobs in the order $1, 2, \ldots, n$, and its makespan.
2. Apply the move local search with the makespan as objective, starting from the schedule of $LIST$.
3. Compute the lower bound $\max\{\max_j t_j, \lceil \sum_j t_j/m \rceil\}$.
4. Prove that the final schedule is optimal.

**Solution**

1. 
   - job 1 ($t_1 = 5$) goes to machine 1, the least loaded (load 0), which reaches 5;
   - job 2 ($t_2 = 7$) goes to machine 2, the least loaded (load 0), which reaches 7;
   - job 3 ($t_3 = 3$) goes to machine 3, the least loaded (load 0), which reaches 3;
   - job 4 ($t_4 = 4$) goes to machine 3, the least loaded (load 3), which reaches 7;
   - job 5 ($t_5 = 4$) goes to machine 1, the least loaded (load 5), which reaches 9;
   - job 6 ($t_6 = 9$) goes to machine 2, the least loaded (load 7), which reaches 16;

   The schedule is $J_1 = \{1, 5\}, J_2 = \{2, 6\}, J_3 = \{3, 4\}$, with loads $(9, 16, 7)$ and $C_{\max} = 16$.

2. Iteration 1: $J_1 = \{1, 5\}, J_2 = \{2, 6\}, J_3 = \{3, 4\}$, loads $(9, 16, 7)$, $C_{\max} = 16$.

| move | loads | $C_{\max}$ | |
|---|---|---|---|
| job 1: $M_1 \to M_2$ | $(4, 21, 7)$ | 21 | |
| job 1: $M_1 \to M_3$ | $(4, 16, 12)$ | 16 | |
| job 2: $M_2 \to M_1$ | $(16, 9, 7)$ | 16 | |
| job 2: $M_2 \to M_3$ | $(9, 9, 14)$ | 14 | $\leftarrow$ |
| job 3: $M_3 \to M_1$ | $(12, 16, 4)$ | 16 | |
| job 3: $M_3 \to M_2$ | $(9, 19, 4)$ | 19 | |
| job 4: $M_3 \to M_1$ | $(13, 16, 3)$ | 16 | |
| job 4: $M_3 \to M_2$ | $(9, 20, 3)$ | 20 | |
| job 5: $M_1 \to M_2$ | $(5, 20, 7)$ | 20 | |
| job 5: $M_1 \to M_3$ | $(5, 16, 11)$ | 16 | |
| job 6: $M_2 \to M_1$ | $(18, 7, 7)$ | 18 | |
| job 6: $M_2 \to M_3$ | $(9, 7, 16)$ | 16 | |

   The move marked is accepted.

   Iteration 2: $J_1 = \{1, 5\}, J_2 = \{6\}, J_3 = \{2, 3, 4\}$, loads $(9, 9, 14)$, $C_{\max} = 14$.

| move | loads | $C_{\max}$ | |
|---|---|---|---|
| job 1: $M_1 \to M_2$ | $(4, 14, 14)$ | 14 | |
| job 1: $M_1 \to M_3$ | $(4, 9, 19)$ | 19 | |
| job 2: $M_3 \to M_1$ | $(16, 9, 7)$ | 16 | |
| job 2: $M_3 \to M_2$ | $(9, 16, 7)$ | 16 | |
| job 3: $M_3 \to M_1$ | $(12, 9, 11)$ | 12 | $\leftarrow$ |
| job 3: $M_3 \to M_2$ | $(9, 12, 11)$ | 12 | |
| job 4: $M_3 \to M_1$ | $(13, 9, 10)$ | 13 | |
| job 4: $M_3 \to M_2$ | $(9, 13, 10)$ | 13 | |
| job 5: $M_1 \to M_2$ | $(5, 13, 14)$ | 14 | |
| job 5: $M_1 \to M_3$ | $(5, 9, 18)$ | 18 | |
| job 6: $M_2 \to M_1$ | $(18, 0, 14)$ | 18 | |
| job 6: $M_2 \to M_3$ | $(9, 0, 23)$ | 23 | |

The move marked is accepted.

Iteration 3: $J_1 = \{1, 3, 5\}, J_2 = \{6\}, J_3 = \{2, 4\}$, loads $(12, 9, 11)$, $C_{\max} = 12$.

| move | loads | $C_{\max}$ |
|---|---|---|
| job 1: $M_1 \to M_2$ | $(7, 14, 11)$ | 14 |
| job 1: $M_1 \to M_3$ | $(7, 9, 16)$ | 16 |
| job 2: $M_3 \to M_1$ | $(19, 9, 4)$ | 19 |
| job 2: $M_3 \to M_2$ | $(12, 16, 4)$ | 16 |
| job 3: $M_1 \to M_2$ | $(9, 12, 11)$ | 12 |
| job 3: $M_1 \to M_3$ | $(9, 9, 14)$ | 14 |
| job 4: $M_3 \to M_1$ | $(16, 9, 7)$ | 16 |
| job 4: $M_3 \to M_2$ | $(12, 13, 7)$ | 13 |
| job 5: $M_1 \to M_2$ | $(8, 13, 11)$ | 13 |
| job 5: $M_1 \to M_3$ | $(8, 9, 15)$ | 15 |
| job 6: $M_2 \to M_1$ | $(21, 0, 11)$ | 21 |
| job 6: $M_2 \to M_3$ | $(12, 0, 20)$ | 20 |

No move is improving: **local optimum**.

The search makes 2 moves, $16 \to 14 \to 12$, and stops in $J_1 = \{1, 3, 5\}, J_2 = \{6\}, J_3 = \{2, 4\}$.

3. The bound is $\max\{9, \lceil 32/3 \rceil\} = 11$.

4. The bound 11 is not attained, but a makespan of 11 is impossible. Job 6 ($t_6 = 9$) can share its machine only with jobs of total time at most $11 - 9 = 2$, and every other job has $t_j \geq 3$: its machine contains job 6 only. The other 5 jobs, of total time $32 - 9 = 23$, should fit on two machines of load at most 11, i.e., at most 22: impossible. Hence $z(I) \geq 12$, and the final schedule, with $C_{\max} = 12$, is **optimal**.

## Exercise 3.11 — Local search $P||C_{\max}$ 2

$m = 3$ identical machines and 7 jobs with processing times

| | $j=1$ | $j=2$ | $j=3$ | $j=4$ | $j=5$ | $j=6$ | $j=7$ |
|---|---|---|---|---|---|---|---|
| $t_j$ | 8 | 4 | 6 | 7 | 5 | 9 | 8 |

A *move* reassigns one job $j$ to another machine $h$ (examined for $j = 1, \dots, n$ and $h = 1, \dots, m$); a *swap* exchanges the machines of two jobs $j < k$ on different machines with $t_j \neq t_k$. The local searches use the best-improvement rule, with ties broken in favour of the move examined first.

**Questions**

1. Compute the schedule of $LPT$ (ties broken by the smallest index) and its makespan.
2. Show that the schedule of $LPT$ is a local optimum for the move neighbourhood.
3. Apply the swap local search with the makespan as objective, starting from the schedule of $LPT$.
4. Compute the lower bound $\lceil \sum_j t_j/m \rceil$ and conclude. What does the exercise show?

## Solution

1. The jobs by non-increasing time are $(6, 1, 7, 4, 3, 5, 2)$.
   - job 6 ($t_6 = 9$) goes to machine 1, the least loaded (load 0), which reaches 9;
   - job 1 ($t_1 = 8$) goes to machine 2, the least loaded (load 0), which reaches 8;
   - job 7 ($t_7 = 8$) goes to machine 3, the least loaded (load 0), which reaches 8;
   - job 4 ($t_4 = 7$) goes to machine 2, the least loaded (load 8), which reaches 15;
   - job 3 ($t_3 = 6$) goes to machine 3, the least loaded (load 8), which reaches 14;
   - job 5 ($t_5 = 5$) goes to machine 1, the least loaded (load 9), which reaches 14;
   - job 2 ($t_2 = 4$) goes to machine 1, the least loaded (load 14), which reaches 18;

   The schedule is $J_1 = \{2, 5, 6\}, J_2 = \{1, 4\}, J_3 = \{3, 7\}$, with loads $(18, 15, 14)$ and $C_{\max} = 18$.
2. Iteration 1: $J_1 = \{2, 5, 6\}, J_2 = \{1, 4\}, J_3 = \{3, 7\}$, loads $(18, 15, 14)$, $C_{\max} = 18$.

| move | loads | $C_{\max}$ |
|---|---|---|
| job 1: $M_2 \to M_1$ | $(26, 7, 14)$ | 26 |
| job 1: $M_2 \to M_3$ | $(18, 7, 22)$ | 22 |
| job 2: $M_1 \to M_2$ | $(14, 19, 14)$ | 19 |
| job 2: $M_1 \to M_3$ | $(14, 15, 18)$ | 18 |
| job 3: $M_3 \to M_1$ | $(24, 15, 8)$ | 24 |
| job 3: $M_3 \to M_2$ | $(18, 21, 8)$ | 21 |
| job 4: $M_2 \to M_1$ | $(25, 8, 14)$ | 25 |
| job 4: $M_2 \to M_3$ | $(18, 8, 21)$ | 21 |
| job 5: $M_1 \to M_2$ | $(13, 20, 14)$ | 20 |
| job 5: $M_1 \to M_3$ | $(13, 15, 19)$ | 19 |
| job 6: $M_1 \to M_2$ | $(9, 24, 14)$ | 24 |
| job 6: $M_1 \to M_3$ | $(9, 15, 23)$ | 23 |
| job 7: $M_3 \to M_1$ | $(26, 15, 6)$ | 26 |
| job 7: $M_3 \to M_2$ | $(18, 23, 6)$ | 23 |

No move is improving: **local optimum**.

The schedule of $LPT$ is a **local optimum for move**, with $C_{\max} = 18$.

3. Iteration 1: $J_1 = \{2, 5, 6\}, J_2 = \{1, 4\}, J_3 = \{3, 7\}$, loads $(18, 15, 14)$, $C_{\max} = 18$.

| swap | loads | $C_{\max}$ | |
|---|---|---|---|
| jobs $1, 2$ $(M_2, M_1)$ | $(22, 11, 14)$ | 22 | |
| jobs $1, 3$ $(M_2, M_3)$ | $(18, 13, 16)$ | 18 | |
| jobs $1, 5$ $(M_2, M_1)$ | $(21, 12, 14)$ | 21 | |
| jobs $1, 6$ $(M_2, M_1)$ | $(17, 16, 14)$ | 17 | $\leftarrow$ |
| jobs $2, 3$ $(M_1, M_3)$ | $(20, 15, 12)$ | 20 | |
| jobs $2, 4$ $(M_1, M_2)$ | $(21, 12, 14)$ | 21 | |
| jobs $2, 7$ $(M_1, M_3)$ | $(22, 15, 10)$ | 22 | |
| jobs $3, 4$ $(M_3, M_2)$ | $(18, 14, 15)$ | 18 | |
| jobs $3, 5$ $(M_3, M_1)$ | $(19, 15, 13)$ | 19 | |
| jobs $3, 6$ $(M_3, M_1)$ | $(15, 15, 17)$ | 17 | |
| jobs $4, 5$ $(M_2, M_1)$ | $(20, 13, 14)$ | 20 | |
| jobs $4, 6$ $(M_2, M_1)$ | $(16, 17, 14)$ | 17 | |
| jobs $4, 7$ $(M_2, M_3)$ | $(18, 16, 13)$ | 18 | |
| jobs $5, 7$ $(M_1, M_3)$ | $(21, 15, 11)$ | 21 | |
| jobs $6, 7$ $(M_1, M_3)$ | $(17, 15, 15)$ | 17 | |

The swap marked is accepted.

Iteration 2: $J_1 = \{1, 2, 5\}, J_2 = \{4, 6\}, J_3 = \{3, 7\}$, loads $(17, 16, 14)$, $C_{\max} = 17$.

| swap | loads | $C_{\max}$ | |
|---|---|---|---|
| jobs $1,3$ $(M_1, M_3)$ | $(15, 16, 16)$ | 16 | $\leftarrow$ |
| jobs $1,4$ $(M_1, M_2)$ | $(16, 17, 14)$ | 17 | |
| jobs $1,6$ $(M_1, M_2)$ | $(18, 15, 14)$ | 18 | |
| jobs $2,3$ $(M_1, M_3)$ | $(19, 16, 12)$ | 19 | |
| jobs $2,4$ $(M_1, M_2)$ | $(20, 13, 14)$ | 20 | |
| jobs $2,6$ $(M_1, M_2)$ | $(22, 11, 14)$ | 22 | |
| jobs $2,7$ $(M_1, M_3)$ | $(21, 16, 10)$ | 21 | |
| jobs $3,4$ $(M_3, M_2)$ | $(17, 15, 15)$ | 17 | |
| jobs $3,5$ $(M_3, M_1)$ | $(18, 16, 13)$ | 18 | |
| jobs $3,6$ $(M_3, M_2)$ | $(17, 13, 17)$ | 17 | |
| jobs $4,5$ $(M_2, M_1)$ | $(19, 14, 14)$ | 19 | |
| jobs $4,7$ $(M_2, M_3)$ | $(17, 17, 13)$ | 17 | |
| jobs $5,6$ $(M_1, M_2)$ | $(21, 12, 14)$ | 21 | |
| jobs $5,7$ $(M_1, M_3)$ | $(20, 16, 11)$ | 20 | |
| jobs $6,7$ $(M_2, M_3)$ | $(17, 15, 15)$ | 17 | |

The swap marked is accepted.

Iteration 3: $J_1 = \{2, 3, 5\}, J_2 = \{4, 6\}, J_3 = \{1, 7\}$, loads $(15, 16, 16)$, $C_{\max} = 16$.

| swap | loads | $C_{\max}$ |
|---|---|---|
| jobs $1,2$ $(M_3, M_1)$ | $(19, 16, 12)$ | 19 |
| jobs $1,3$ $(M_3, M_1)$ | $(17, 16, 14)$ | 17 |
| jobs $1,4$ $(M_3, M_2)$ | $(15, 17, 15)$ | 17 |
| jobs $1,5$ $(M_3, M_1)$ | $(18, 16, 13)$ | 18 |
| jobs $1,6$ $(M_3, M_2)$ | $(15, 15, 17)$ | 17 |
| jobs $2,4$ $(M_1, M_2)$ | $(18, 13, 16)$ | 18 |
| jobs $2,6$ $(M_1, M_2)$ | $(20, 11, 16)$ | 20 |
| jobs $2,7$ $(M_1, M_3)$ | $(19, 16, 12)$ | 19 |
| jobs $3,4$ $(M_1, M_2)$ | $(16, 15, 16)$ | 16 |
| jobs $3,6$ $(M_1, M_2)$ | $(18, 13, 16)$ | 18 |
| jobs $3,7$ $(M_1, M_3)$ | $(17, 16, 14)$ | 17 |
| jobs $4,5$ $(M_2, M_1)$ | $(17, 14, 16)$ | 17 |
| jobs $4,7$ $(M_2, M_3)$ | $(15, 17, 15)$ | 17 |
| jobs $5,6$ $(M_1, M_2)$ | $(19, 12, 16)$ | 19 |
| jobs $5,7$ $(M_1, M_3)$ | $(18, 16, 13)$ | 18 |
| jobs $6,7$ $(M_2, M_3)$ | $(15, 15, 17)$ | 17 |

No swap is improving: **local optimum**.

The swap local search makes 2 moves, $18 \to 17 \to 16$, and stops in $J_1 = \{2, 3, 5\}, J_2 = \{4, 6\}, J_3 = \{1, 7\}$.

4. The bound is $\lceil 47/3 \rceil = 16$, attained by the final schedule: it is **optimal**. The schedule of $LPT$ is a local optimum for move and not for swap: the local optimum **depends on the neighbourhood**.

## Exercise 3.12 — Local search $P\,||\,C_{\max}$ 3

$m = 3$ identical machines and 6 jobs with processing times

| | $j=1$ | $j=2$ | $j=3$ | $j=4$ | $j=5$ | $j=6$ |
|---|---|---|---|---|---|---|
| $t_j$ | 5 | 5 | 7 | 9 | 2 | 7 |

A *move* reassigns one job $j$ to another machine $h$ (examined for $j = 1, \dots, n$ and $h = 1, \dots, m$); a *swap* exchanges the machines of two jobs $j < k$ on different machines with $t_j \neq t_k$. The local searches use the best-improvement rule, with ties broken in favour of the move examined first.

**Questions**

1. Compute the schedule of list scheduling ($LIST$) with the jobs in the order $1, 2, \dots, n$, and its makespan. How many machines are critical?
2. Apply the move local search with the makespan as objective, starting from the schedule of $LIST$. Why does it stop?
3. Apply the move local search with the auxiliary objective $\sum_h L_h^2$ (minimized), starting from the schedule of $LIST$.
4. Compute the lower bound $\max\{\max_j t_j, \lceil \sum_j t_j / m \rceil\}$ and conclude.

## Solution

1. 
   - job 1 ($t_1 = 5$) goes to machine 1, the least loaded (load 0), which reaches 5;
   - job 2 ($t_2 = 5$) goes to machine 2, the least loaded (load 0), which reaches 5;
   - job 3 ($t_3 = 7$) goes to machine 3, the least loaded (load 0), which reaches 7;
   - job 4 ($t_4 = 9$) goes to machine 1, the least loaded (load 5), which reaches 14;
   - job 5 ($t_5 = 2$) goes to machine 2, the least loaded (load 5), which reaches 7;
   - job 6 ($t_6 = 7$) goes to machine 2, the least loaded (load 7), which reaches 14;

   The schedule is $J_1 = \{1, 4\}, J_2 = \{2, 5, 6\}, J_3 = \{3\}$, with loads $(14, 14, 7)$ and $C_{\max} = 14$: 2 machines are critical.
2. Iteration 1: $J_1 = \{1, 4\}, J_2 = \{2, 5, 6\}, J_3 = \{3\}$, loads $(14, 14, 7)$, $C_{\max} = 14$.

| move | loads | $C_{\max}$ |
|---|---|---|
| job 1: $M_1 \to M_2$ | $(9, 19, 7)$ | 19 |
| job 1: $M_1 \to M_3$ | $(9, 14, 12)$ | 14 |
| job 2: $M_2 \to M_1$ | $(19, 9, 7)$ | 19 |
| job 2: $M_2 \to M_3$ | $(14, 9, 12)$ | 14 |
| job 3: $M_3 \to M_1$ | $(21, 14, 0)$ | 21 |
| job 3: $M_3 \to M_2$ | $(14, 21, 0)$ | 21 |
| job 4: $M_1 \to M_2$ | $(5, 23, 7)$ | 23 |
| job 4: $M_1 \to M_3$ | $(5, 14, 16)$ | 16 |
| job 5: $M_2 \to M_1$ | $(16, 12, 7)$ | 16 |
| job 5: $M_2 \to M_3$ | $(14, 12, 9)$ | 14 |
| job 6: $M_2 \to M_1$ | $(21, 7, 7)$ | 21 |
| job 6: $M_2 \to M_3$ | $(14, 7, 14)$ | 14 |

No move is improving: **local optimum**.

Two machines have load 14. A move changes only two loads, and lowers the makespan only if it starts from the unique critical machine: here no move can decrease both critical loads, so the schedule of $LIST$ is a local optimum. Several moves keep $C_{\max} = 14$: the search is on a **plateau**.

3. Iteration 1: $J_1 = \{1, 4\}, J_2 = \{2, 5, 6\}, J_3 = \{3\}$, loads $(14, 14, 7)$, $C_{\max} = 14$, $\sum_h L_h^2 = 441$.

| move | loads | $C_{\max}$ | $\sum_h L_h^2$ | |
|---|---|---|---|---|
| job 1: $M_1 \to M_2$ | $(9, 19, 7)$ | 19 | 491 | |
| job 1: $M_1 \to M_3$ | $(9, 14, 12)$ | 14 | 421 | $\leftarrow$ |
| job 2: $M_2 \to M_1$ | $(19, 9, 7)$ | 19 | 491 | |
| job 2: $M_2 \to M_3$ | $(14, 9, 12)$ | 14 | 421 | |
| job 3: $M_3 \to M_1$ | $(21, 14, 0)$ | 21 | 637 | |
| job 3: $M_3 \to M_2$ | $(14, 21, 0)$ | 21 | 637 | |
| job 4: $M_1 \to M_2$ | $(5, 23, 7)$ | 23 | 603 | |
| job 4: $M_1 \to M_3$ | $(5, 14, 16)$ | 16 | 477 | |
| job 5: $M_2 \to M_1$ | $(16, 12, 7)$ | 16 | 449 | |
| job 5: $M_2 \to M_3$ | $(14, 12, 9)$ | 14 | 421 | |
| job 6: $M_2 \to M_1$ | $(21, 7, 7)$ | 21 | 539 | |
| job 6: $M_2 \to M_3$ | $(14, 7, 14)$ | 14 | 441 | |

The move marked is accepted.

Iteration 2: $J_1 = \{4\}, J_2 = \{2, 5, 6\}, J_3 = \{1, 3\}$, loads $(9, 14, 12)$, $C_{\max} = 14$, $\sum_h L_h^2 = 421$.

| move | loads | $C_{\max}$ | $\sum_h L_h^2$ | |
|---|---|---|---|---|
| job 1: $M_3 \to M_1$ | $(14, 14, 7)$ | 14 | 441 | |
| job 1: $M_3 \to M_2$ | $(9, 19, 7)$ | 19 | 491 | |
| job 2: $M_2 \to M_1$ | $(14, 9, 12)$ | 14 | 421 | |
| job 2: $M_2 \to M_3$ | $(9, 9, 17)$ | 17 | 451 | |
| job 3: $M_3 \to M_1$ | $(16, 14, 5)$ | 16 | 477 | |
| job 3: $M_3 \to M_2$ | $(9, 21, 5)$ | 21 | 547 | |
| job 4: $M_1 \to M_2$ | $(0, 23, 12)$ | 23 | 673 | |
| job 4: $M_1 \to M_3$ | $(0, 14, 21)$ | 21 | 637 | |
| job 5: $M_2 \to M_1$ | $(11, 12, 12)$ | 12 | 409 | $\leftarrow$ |
| job 5: $M_2 \to M_3$ | $(9, 12, 14)$ | 14 | 421 | |
| job 6: $M_2 \to M_1$ | $(16, 7, 12)$ | 16 | 449 | |
| job 6: $M_2 \to M_3$ | $(9, 7, 19)$ | 19 | 491 | |

The move marked is accepted.

Iteration 3: $J_1 = \{4, 5\}, J_2 = \{2, 6\}, J_3 = \{1, 3\}$, loads $(11, 12, 12)$, $C_{\max} = 12$, $\sum_h L_h^2 = 409$.

| move | loads | $C_{\max}$ | $\sum_h L_h^2$ |
|---|---|---|---|
| job 1: $M_3 \to M_1$ | $(16, 12, 7)$ | 16 | 449 |
| job 1: $M_3 \to M_2$ | $(11, 17, 7)$ | 17 | 459 |
| job 2: $M_2 \to M_1$ | $(16, 7, 12)$ | 16 | 449 |
| job 2: $M_2 \to M_3$ | $(11, 7, 17)$ | 17 | 459 |
| job 3: $M_3 \to M_1$ | $(18, 12, 5)$ | 18 | 493 |
| job 3: $M_3 \to M_2$ | $(11, 19, 5)$ | 19 | 507 |
| job 4: $M_1 \to M_2$ | $(2, 21, 12)$ | 21 | 589 |
| job 4: $M_1 \to M_3$ | $(2, 12, 21)$ | 21 | 589 |
| job 5: $M_1 \to M_2$ | $(9, 14, 12)$ | 14 | 421 |
| job 5: $M_1 \to M_3$ | $(9, 12, 14)$ | 14 | 421 |
| job 6: $M_2 \to M_1$ | $(18, 5, 12)$ | 18 | 493 |
| job 6: $M_2 \to M_3$ | $(11, 5, 19)$ | 19 | 507 |

No move is improving: **local optimum**.

The search makes 2 moves, with makespan $14 \to 14 \to 12$ and $\sum_h L_h^2$: $441 \to 421 \to 409$. The first move does not change the makespan, but it prepares the second one.

4. The bound is $\max\{9, \lceil 35/3 \rceil\} = 12$, attained by the final schedule: it is **optimal**. The auxiliary objective gives a slope to the plateau of the makespan.

Chapter 4

# Metaheuristics

## 4.1 Simulated annealing

## Exercise 4.1 — Simulated annealing 1

A TSP instance with 6 cities and the symmetric distances

| $d_{ij}$ | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| 1 | — | 5 | 1 | 7 | 7 | 4 |
| 2 | 5 | — | 4 | 5 | 2 | 1 |
| 3 | 1 | 4 | — | 6 | 6 | 4 |
| 4 | 7 | 5 | 6 | — | 7 | 6 |
| 5 | 7 | 2 | 6 | 7 | — | 2 |
| 6 | 4 | 1 | 4 | 6 | 2 | — |

The 2-opt moves of a tour of 6 cities are numbered

$$1:(1,3), 2:(1,4), 3:(1,5), 4:(2,4), 5:(2,5), 6:(2,6), 7:(3,5), 8:(3,6), 9:(4,6)$$

and the random numbers to use, in this order, are 0.13, 0.76, 0.46, 0.53, 0.22, 0.05, 0.68, 0.68, 0.93, 0.38, 0.52, 0.83.

**Questions**

1. Compute the nearest neighbour tour from city 1 (ties: smallest index) and its length, and show that it is a local optimum for the 2-opt neighbourhood.
2. Apply simulated annealing with the 2-opt neighbourhood from this tour, with $T_0 = 5$, $\alpha = 0.8$, $L = 2$ and $T_{\min} = 3$. How many iterations are performed?
3. At each iteration the first random number $u$ chooses the move $1 + \lfloor 9u \rfloor$ and the second one, $v$, is used in the Metropolis test. Show every iteration: move, $\delta$, acceptance test, current and best tour.
4. Compare the best tour found with the optimal one, and explain which accepted move made it possible to escape from the starting local optimum.

## Solution

1. The nearest neighbour tour is $(1, 3, 2, 6, 5, 4)$, of length 22. Its 9 neighbours have $\delta$: $(1,3)$: $+7$, $(1,4)$: $+7$, $(1,5)$: $+5$, $(2,4)$: $+0$, $(2,5)$: $+0$, $(2,6)$: $+0$, $(3,5)$: $+0$, $(3,6)$: $+1$, $(4,6)$: $+4$. No $\delta$ is negative: the tour is a 2-opt local optimum.
2. The temperatures are 5, 4 and 3.2; then $T = 2.56 \leq 3$ stops the search: $3 \cdot 2 = 6$ iterations, which use 12 random numbers.
3. - Iteration 1 ($T = 5$): $u = 0.13$ gives the move $1 + \lfloor 9 \cdot 0.13 \rfloor = 2$, i.e. $(1,4)$, with $\delta = 7$; $v = 0.76 \geq e^{-7/5} = 0.2466$: rejected. Current tour $(1, 3, 2, 6, 5, 4)$, length 22, best 22.
   - Iteration 2 ($T = 5$): $u = 0.46$ gives the move $1 + \lfloor 9 \cdot 0.46 \rfloor = 5$, i.e. $(2,5)$, with $\delta = 0$; $\delta \leq 0$: accepted. Current tour $(1, 3, 5, 6, 2, 4)$, length 22, best 22.

- Iteration 3 ($T = 4$): $u = 0.22$ gives the move $1 + \lfloor 9 \cdot 0.22 \rfloor = 2$, i.e. $(1, 4)$, with $\delta = 6$; $v = 0.05 < e^{-6/4} = 0.2231$: accepted. Current tour $(1, 6, 5, 3, 2, 4)$, length 28, best 22.
- Iteration 4 ($T = 4$): $u = 0.68$ gives the move $1 + \lfloor 9 \cdot 0.68 \rfloor = 7$, i.e. $(3, 5)$, with $\delta = -3$; $\delta \leq 0$: accepted. Current tour $(1, 6, 5, 2, 3, 4)$, length 25, best 22.
- Iteration 5 ($T = 3.2$): $u = 0.93$ gives the move $1 + \lfloor 9 \cdot 0.93 \rfloor = 9$, i.e. $(4, 6)$, with $\delta = -5$; $\delta \leq 0$: accepted. Current tour $(1, 6, 5, 2, 4, 3)$, length 20, best 20.
- Iteration 6 ($T = 3.2$): $u = 0.52$ gives the move $1 + \lfloor 9 \cdot 0.52 \rfloor = 5$, i.e. $(2, 5)$, with $\delta = 4$; $v = 0.83 \geq e^{-4/3.2} = 0.2865$: rejected. Current tour $(1, 6, 5, 2, 4, 3)$, length 20, best 20.

4. The best tour found is $(1, 6, 5, 2, 4, 3)$ of length 20; by enumeration of the 60 tours the optimal tour is $(1, 3, 4, 2, 5, 6)$, of length 20: simulated annealing finds the optimum. The escape is due to the accepted moves with $\delta \geq 0$ (iterations 2, 3), which lead the search away from the local optimum 22, from where improving moves reach 20.

## Exercise 4.2 — Simulated annealing 2

A knapsack of capacity $c = 16$ and 5 items:

| | $j = 1$ | $j = 2$ | $j = 3$ | $j = 4$ | $j = 5$ |
|---|---|---|---|---|---|
| $p_j$ | 11 | 5 | 18 | 8 | 8 |
| $w_j$ | 5 | 3 | 11 | 6 | 7 |

Simulated annealing with the flip neighbourhood maximizes

$$f_\beta(x) = \sum_j p_j x_j - \beta \max\Big\{0, \sum_j w_j x_j - c\Big\},$$

with $T_0 = 4$, $\alpha = 0.8$, $L = 2$, $T_{\min} = 1.7$. At each iteration $u$ chooses the item $1 + \lfloor 5u \rfloor$ to flip and $v$ is used in the acceptance test. The random numbers are 0.63, 0.11, 0.17, 0.66, 0.38, 0.94, 0.58, 0.59, 0.69, 0.67, 0.39, 0.62, 0.69, 0.07, 0.59, 0.42.

**Questions**

1. Compute the solution $x$ of the greedy algorithm by efficiency, and show that it is a local optimum for the flip neighbourhood among the feasible solutions.
2. Apply simulated annealing from $x$ with $\beta = 1$. Which is the best feasible solution found, and where is the search at the end?
3. Apply simulated annealing from $x$ with $\beta = 3$, with the same random numbers. Which is the best feasible solution found?
4. Explain the different behaviour with the two values of $\beta$, and compare with the optimal value.

**Solution**

1. The efficiencies are $\frac{11}{5}$, $\frac{5}{3}$, $\frac{18}{11}$, $\frac{8}{6}$, $\frac{8}{7}$, already non-increasing. Greedy selects items 1, 2 ($rc = 8$), skips item 3 ($w_3 = 11$), selects item 4 ($rc = 2$) and skips item 5: $x = (1, 1, 0, 1, 0)$, weight 14, profit 24. Feasible flips only remove items (adding item 3 or 5 exceeds the capacity), so every feasible neighbour has smaller profit: $x$ is a local optimum.

2. With $\beta = 1$ the 8 iterations ($T = 4, 4, 3.2, 3.2, 2.56, 2.56, 2.048, 2.048$) are:

   - Iteration 1 ($T = 4$): $u = 0.63$ flips item 4: $y = (1, 1, 0, 0, 0)$, weight 8, profit 16, $f(y) = 16$, $\delta = 8$; $v = 0.11 < e^{-8/4} = 0.1353$: accepted. Best feasible profit: 24.
   - Iteration 2 ($T = 4$): $u = 0.17$ flips item 1: $y = (0, 1, 0, 0, 0)$, weight 3, profit 5, $f(y) = 5$, $\delta = 11$; $v = 0.66 \geq e^{-11/4} = 0.0639$: rejected. Best feasible profit: 24.
   - Iteration 3 ($T = 3.2$): $u = 0.38$ flips item 2: $y = (1, 0, 0, 0, 0)$, weight 5, profit 11, $f(y) = 11$, $\delta = 5$; $v = 0.94 \geq e^{-5/3.2} = 0.2096$: rejected. Best feasible profit: 24.
   - Iteration 4 ($T = 3.2$): $u = 0.58$ flips item 3: $y = (1, 1, 1, 0, 0)$, weight 19, profit 34, $f(y) = 31$, $\delta = -15$; $\delta \leq 0$: accepted. Best feasible profit: 24.
   - Iteration 5 ($T = 2.56$): $u = 0.69$ flips item 4: $y = (1, 1, 1, 1, 0)$, weight 25, profit 42, $f(y) = 33$, $\delta = -2$; $\delta \leq 0$: accepted. Best feasible profit: 24.
   - Iteration 6 ($T = 2.56$): $u = 0.39$ flips item 2: $y = (1, 0, 1, 1, 0)$, weight 22, profit 37, $f(y) = 31$, $\delta = 2$; $v = 0.62 \geq e^{-2/2.56} = 0.4578$: rejected. Best feasible profit: 24.
   - Iteration 7 ($T = 2.048$): $u = 0.69$ flips item 4: $y = (1, 1, 1, 0, 0)$, weight 19, profit 34, $f(y) = 31$, $\delta = 2$; $v = 0.07 < e^{-2/2.048} = 0.3766$: accepted. Best feasible profit: 24.
   - Iteration 8 ($T = 2.048$): $u = 0.59$ flips item 3: $y = (1, 1, 0, 0, 0)$, weight 8, profit 16, $f(y) = 16$, $\delta = 15$; $v = 0.42 \geq e^{-15/2.048} = 0.0007$: rejected. Best feasible profit: 24.

   The best feasible profit remains 24, and at the end the current solution is $(1, 1, 1, 0, 0)$, of weight $19 > 16$: the search ends in the infeasible region.

3. With $\beta = 3$:

   - Iteration 1 ($T = 4$): $u = 0.63$ flips item 4: $y = (1, 1, 0, 0, 0)$, weight 8, profit 16, $f(y) = 16$, $\delta = 8$; $v = 0.11 < e^{-8/4} = 0.1353$: accepted. Best feasible profit: 24.
   - Iteration 2 ($T = 4$): $u = 0.17$ flips item 1: $y = (0, 1, 0, 0, 0)$, weight 3, profit 5, $f(y) = 5$, $\delta = 11$; $v = 0.66 \geq e^{-11/4} = 0.0639$: rejected. Best feasible profit: 24.
   - Iteration 3 ($T = 3.2$): $u = 0.38$ flips item 2: $y = (1, 0, 0, 0, 0)$, weight 5, profit 11, $f(y) = 11$, $\delta = 5$; $v = 0.94 \geq e^{-5/3.2} = 0.2096$: rejected. Best feasible profit: 24.
   - Iteration 4 ($T = 3.2$): $u = 0.58$ flips item 3: $y = (1, 1, 1, 0, 0)$, weight 19, profit 34, $f(y) = 25$, $\delta = -9$; $\delta \leq 0$: accepted. Best feasible profit: 24.

- Iteration 5 ($T = 2.56$): $u = 0.69$ flips item 4: $y = (1, 1, 1, 1, 0)$, weight 25, profit 42, $f(y) = 15$, $\delta = 10$; $v = 0.67 \geq e^{-10/2.56} = 0.0201$: rejected. Best feasible profit: 24.
- Iteration 6 ($T = 2.56$): $u = 0.39$ flips item 2: $y = (1, 0, 1, 0, 0)$, weight 16, profit 29, $f(y) = 29$, $\delta = -4$; $\delta \leq 0$: accepted. Best feasible profit: 29.
- Iteration 7 ($T = 2.048$): $u = 0.69$ flips item 4: $y = (1, 0, 1, 1, 0)$, weight 22, profit 37, $f(y) = 19$, $\delta = 10$; $v = 0.07 \geq e^{-10/2.048} = 0.0076$: rejected. Best feasible profit: 29.
- Iteration 8 ($T = 2.048$): $u = 0.59$ flips item 3: $y = (1, 0, 0, 0, 0)$, weight 5, profit 11, $f(y) = 11$, $\delta = 18$; $v = 0.42 \geq e^{-18/2.048} = 0.0002$: rejected. Best feasible profit: 29.

The best feasible solution is $(1, 0, 1, 0, 0)$, of profit 29.

4. With $\beta = 1$ an excess weight costs only 1 per unit, and the infeasible solutions containing item 3 have a large penalized value (for instance $f_1(1, 1, 1, 1, 0) = 42 - 9 = 33$): the search stays in the infeasible region and never comes back to a better feasible solution. With $\beta = 3$ the infeasible solutions are only a bridge: the search crosses them and reaches $\{1, 3\}$, of profit 29, which is optimal (by enumeration). The penalty coefficient must be large enough to make the infeasible solutions less attractive than the good feasible ones.

## Exercise 4.3 — Simulated annealing 3

A TSP instance with 6 cities and the symmetric distances

| $d_{ij}$ | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| 1 | — | 5 | 1 | 7 | 7 | 4 |
| 2 | 5 | — | 4 | 5 | 2 | 1 |
| 3 | 1 | 4 | — | 6 | 6 | 4 |
| 4 | 7 | 5 | 6 | — | 7 | 6 |
| 5 | 7 | 2 | 6 | 7 | — | 2 |
| 6 | 4 | 1 | 4 | 6 | 2 | — |

The starting tour is the nearest neighbour tour $(1, 3, 2, 6, 5, 4)$, of length 22, a 2-opt local optimum. The 2-opt moves are numbered

$$1 : (1, 3), 2 : (1, 4), 3 : (1, 5), 4 : (2, 4), 5 : (2, 5), 6 : (2, 6), 7 : (3, 5), 8 : (3, 6), 9 : (4, 6)$$

and the random numbers to use are 0.13, 0.76, 0.46, 0.53, 0.22, 0.05, 0.68, 0.68, 0.93, 0.38, 0.52, 0.83.

**Questions**

1. Compute the acceptance probability $e^{-\delta/T}$ of a worsening $\delta = 6$ at $T = 5$ and at $T = 0.5$.

2. With the geometric schedule, how many temperatures are used with $T_0 = 0.5$, $\alpha = 0.8$, $T_{\min} = 0.3$? And with $T_0 = 5$, $\alpha = 0.95$, $T_{\min} = 0.5$?
3. Apply simulated annealing with $T_0 = 0.5$, $\alpha = 0.8$, $L = 2$, $T_{\min} = 0.3$ (move $1 + \lfloor 9u \rfloor$, Metropolis test with $v$). Show every iteration.
4. The same random numbers with $T_0 = 5$, $T_{\min} = 3$ lead to the optimal tour of length 20. Explain why the run at low temperature does not.

**Solution**

1. $e^{-6/5} = 0.3012$ and $e^{-6/0.5} = e^{-12} = 0.000006$: at low temperature a worsening of 6 is practically never accepted.
2. With $T_0 = 0.5$: 0.5, 0.4, 0.32 are larger than 0.3, and 0.256 is not: 3 temperatures, 6 iterations with $L = 2$. With $T_0 = 5$, $\alpha = 0.95$: the smallest $K$ with $5 \cdot 0.95^K \leq 0.5$ is $K = \lceil \ln(0.1)/\ln(0.95) \rceil = 45$ temperatures.
3. - Iteration 1 ($T = 0.5$): $u = 0.13$ gives the move $1 + \lfloor 9 \cdot 0.13 \rfloor = 2$, i.e. $(1, 4)$, with $\delta = 7$; $v = 0.76 \geq e^{-7/0.5} = 0.0000$: rejected. Current tour $(1, 3, 2, 6, 5, 4)$, length 22, best 22.
   - Iteration 2 ($T = 0.5$): $u = 0.46$ gives the move $1 + \lfloor 9 \cdot 0.46 \rfloor = 5$, i.e. $(2, 5)$, with $\delta = 0$; $\delta \leq 0$: accepted. Current tour $(1, 3, 5, 6, 2, 4)$, length 22, best 22.
   - Iteration 3 ($T = 0.4$): $u = 0.22$ gives the move $1 + \lfloor 9 \cdot 0.22 \rfloor = 2$, i.e. $(1, 4)$, with $\delta = 6$; $v = 0.05 \geq e^{-6/0.4} = 0.0000$: rejected. Current tour $(1, 3, 5, 6, 2, 4)$, length 22, best 22.
   - Iteration 4 ($T = 0.4$): $u = 0.68$ gives the move $1 + \lfloor 9 \cdot 0.68 \rfloor = 7$, i.e. $(3, 5)$, with $\delta = 1$; $v = 0.68 \geq e^{-1/0.4} = 0.0821$: rejected. Current tour $(1, 3, 5, 6, 2, 4)$, length 22, best 22.
   - Iteration 5 ($T = 0.32$): $u = 0.93$ gives the move $1 + \lfloor 9 \cdot 0.93 \rfloor = 9$, i.e. $(4, 6)$, with $\delta = 3$; $v = 0.38 \geq e^{-3/0.32} = 0.0001$: rejected. Current tour $(1, 3, 5, 6, 2, 4)$, length 22, best 22.
   - Iteration 6 ($T = 0.32$): $u = 0.52$ gives the move $1 + \lfloor 9 \cdot 0.52 \rfloor = 5$, i.e. $(2, 5)$, with $\delta = 0$; $\delta \leq 0$: accepted. Current tour $(1, 3, 2, 6, 5, 4)$, length 22, best 22.
4. At $T \leq 0.5$ the worsening moves of the starting tour ($\delta \geq 1$) are accepted with probability at most $e^{-2} \approx 0.14$, and with these random numbers they are all rejected: only the moves with $\delta = 0$ are accepted, and the search stays on tours of length 22. Simulated annealing at low temperature is a local search: it cannot leave the valley of the starting local optimum. At $T_0 = 5$ the worsening move of iteration 3 ($\delta = 6$) is accepted, and the search reaches the optimum.

## 4.2 Tabu search

**Exercise 4.4 — Tabu search 1**

$m = 2$ identical machines and 5 jobs with processing times $t = (3, 3, 2, 2, 2)$. Tabu search uses the move neighbourhood, compares the solutions lexicographically by $(C_{\max}, \sum_h L_h^2)$, forbids moving a job back to the machine it has just left for *tenure* iterations (aspiration: a tabu move is allowed if it gives a makespan smaller than the best one), and stops when $C_{\max}$ equals the lower bound. Ties between moves: smallest job, then smallest machine.

**Questions**

1. Compute the LPT solution and the lower bound $LB = \max\{\max_j t_j, \lceil \sum_j t_j/m \rceil\}$.
2. Show that the LPT solution is a local optimum for the move neighbourhood.
3. Apply tabu search with *tenure* $= 3$ from the LPT solution, showing at each iteration the evaluation of the neighbours, the move performed and the tabu list.
4. Is the final solution optimal? At which iteration does the search accept a worse solution, and why?

**Solution**

1. LPT assigns the jobs $1, 2$ (time 3) to $M_1$ and $M_2$, then job 3 to $M_1$ (5), job 4 to $M_2$ (5), job 5 to $M_1$ (7): $J_1 = \{1, 3, 5\}$, $J_2 = \{2, 4\}$, loads $(7, 5)$, $C_{\max} = 7$. $LB = \max\{3, \lceil 12/2 \rceil\} = 6$.
2. The neighbours are: job $1 \to M_2$: $(8, 80)$, job $2 \to M_1$: $(10, 104)$, job $3 \to M_2$: $(7, 74)$, job $4 \to M_1$: $(9, 90)$, job $5 \to M_2$: $(7, 74)$. No neighbour has makespan smaller than 7, nor makespan 7 and $\sum_h L_h^2 < 74$: the LPT solution is a local optimum.
3. - Iteration 1. Neighbours $(C_{\max}, \sum_h L_h^2)$: $1 \to M_2$: $(8, 80)$, $2 \to M_1$: $(10, 104)$, $3 \to M_2$: $(7, 74)$, $4 \to M_1$: $(9, 90)$, $5 \to M_2$: $(7, 74)$. Move job 3 from $M_1$ to $M_2$: loads $(5, 7)$, $C_{\max} = 7$; the attribute $(3, M_1)$ is tabu until iteration 4; best makespan 7.
   - Iteration 2. Neighbours $(C_{\max}, \sum_h L_h^2)$: $1 \to M_2$: $(10, 104)$, $2 \to M_1$: $(8, 80)$, $3 \to M_1$: $(7, 74)$ tabu, $4 \to M_1$: $(7, 74)$, $5 \to M_2$: $(9, 90)$. Move job 4 from $M_2$ to $M_1$: loads $(7, 5)$, $C_{\max} = 7$; the attribute $(4, M_2)$ is tabu until iteration 5; best makespan 7.
   - Iteration 3. Neighbours $(C_{\max}, \sum_h L_h^2)$: $1 \to M_2$: $(8, 80)$, $2 \to M_1$: $(10, 104)$, $3 \to M_1$: $(9, 90)$ tabu, $4 \to M_2$: $(7, 74)$ tabu, $5 \to M_2$: $(7, 74)$. Move job 5 from $M_1$ to $M_2$: loads $(5, 7)$, $C_{\max} = 7$; the attribute $(5, M_1)$ is tabu until iteration 6; best makespan 7.
   - Iteration 4. Neighbours $(C_{\max}, \sum_h L_h^2)$: $1 \to M_2$: $(10, 104)$, $2 \to M_1$: $(8, 80)$, $3 \to M_1$: $(7, 74)$ tabu, $4 \to M_2$: $(9, 90)$ tabu, $5 \to M_1$: $(7, 74)$ tabu. Move job

2 from $M_2$ to $M_1$: loads $(8, 4)$, $C_{\max} = 8$; the attribute $(2, M_2)$ is tabu until iteration 7; best makespan 7.

- Iteration 5. Neighbours $(C_{\max}, \sum_h L_h^2)$: $1 \to M_2$: $(7, 74)$, $2 \to M_2$: $(7, 74)$ tabu, $3 \to M_1$: $(10, 104)$, $4 \to M_2$: $(6, 72)$ tabu, $5 \to M_1$: $(10, 104)$ tabu. Move job 4 from $M_1$ to $M_2$: loads $(6, 6)$, $C_{\max} = 6$; the attribute $(4, M_1)$ is tabu until iteration 8; best makespan 6.

4. At iteration 5 the makespan reaches $LB = 6$: the solution $J_1 = \{1, 2\}$, $J_2 = \{3, 4, 5\}$ is optimal and the search stops. The search accepts worse solutions (in the lexicographic order) at iteration 4, because the best non-tabu neighbour is worse than the current solution: this is how tabu search leaves the plateau of makespan 7.

## Exercise 4.5 — Tabu search 2

$m = 2$ identical machines and 5 jobs with processing times $t = (3, 3, 2, 2, 2)$. The starting solution is the LPT solution $J_1 = \{1, 3, 5\}$, $J_2 = \{2, 4\}$. Tabu search uses the move neighbourhood and the lexicographic comparison by $(C_{\max}, \sum_h L_h^2)$; after moving job $j$ away from machine $h$, the attribute $(j, h)$ is tabu for *tenure* iterations. Ties: smallest job, then smallest machine.

**Questions**

1. Apply 4 iterations of tabu search with *tenure* $= 0$ (no memory). What happens?
2. Apply 6 iterations with *tenure* $= 1$. What happens?
3. Explain why short tabu lists do not prevent the search from cycling on this instance.
4. The same search with *tenure* $= 3$ reaches the optimum 6 in 5 iterations. Which is the role of the tabu list, and what can go wrong with a tabu list that is too long?

## Solution

1. 
   - Iteration 1: job 3 from $M_1$ to $M_2$, loads $(5, 7)$.
   - Iteration 2: job 3 from $M_2$ to $M_1$, loads $(7, 5)$.
   - Iteration 3: job 3 from $M_1$ to $M_2$, loads $(5, 7)$.
   - Iteration 4: job 3 from $M_2$ to $M_1$, loads $(7, 5)$.

   The search moves job 3 back and forth: it cycles between two solutions of makespan 7.

2. 
   - Iteration 1: job 3 from $M_1$ to $M_2$, loads $(5, 7)$.
   - Iteration 2: job 4 from $M_2$ to $M_1$, loads $(7, 5)$.
   - Iteration 3: job 5 from $M_1$ to $M_2$, loads $(5, 7)$.
   - Iteration 4: job 3 from $M_2$ to $M_1$, loads $(7, 5)$.
   - Iteration 5: job 4 from $M_1$ to $M_2$, loads $(5, 7)$.

- Iteration 6: job 5 from $M_2$ to $M_1$, loads $(7, 5)$.

The search still visits only solutions of makespan 7, and it repeats the same sequence of moves: a longer cycle.

3. All the neighbours with makespan 7 move a job of time 2 between the machines. With $tenure \leq 2$ the attribute just created expires before the search has moved far enough, and the best non-tabu move brings it back to a solution already visited: the cycle is longer than the memory.

4. The tabu list forbids undoing the recent moves, so that the search is forced to make worse moves and leave the plateau (with $tenure = 3$ the worsening moves lead to the optimum). A tabu list that is too long forbids too many moves, possibly the good ones; in the extreme, all moves are tabu and the search is driven only by the aspiration criterion. The tenure must be tuned.

## Exercise 4.6 — Tabu search 3

A TSP instance with 6 cities and the symmetric distances

| $d_{ij}$ | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| 1 | — | 5 | 1 | 7 | 7 | 4 |
| 2 | 5 | — | 4 | 5 | 2 | 1 |
| 3 | 1 | 4 | — | 6 | 6 | 4 |
| 4 | 7 | 5 | 6 | — | 7 | 6 |
| 5 | 7 | 2 | 6 | 7 | — | 2 |
| 6 | 4 | 1 | 4 | 6 | 2 | — |

Tabu search with the 2-opt neighbourhood: at every iteration all the moves $(i, j)$ are evaluated, and the best non-tabu move is performed even if it is not improving (ties: first move in lexicographic order). The two edges removed by a move may not be added back for $tenure = 2$ iterations, unless the new tour is shorter than the best one.

**Questions**

1. Compute the nearest neighbour tour $\sigma$ from city 1 and show that it is a local optimum for 2-opt.

2. Perform the first iteration of tabu search from $\sigma$: which move is chosen, and which edges become tabu?

3. Perform the second and the third iteration.

4. Compare the best tour found with the local optimum of the first question.

**Solution**

1. $\sigma = (1, 3, 2, 6, 5, 4)$, of length 22; the deltas of its neighbours are $(1, 3)$: $+7$, $(1, 4)$: $+7$, $(1, 5)$: $+5$, $(2, 4)$: $+0$, $(2, 5)$: $+0$, $(2, 6)$: $+0$, $(3, 5)$: $+0$, $(3, 6)$: $+1$, $(4, 6)$: $+4$: none is negative, so $\sigma$ is a 2-opt local optimum.

2. From $(1, 3, 2, 6, 5, 4)$ the neighbours have $\delta$: $(1, 3)$: $+7$, $(1, 4)$: $+7$, $(1, 5)$: $+5$, $(2, 4)$: $+0$, $(2, 5)$: $+0$, $(2, 6)$: $+0$, $(3, 5)$: $+0$, $(3, 6)$: $+1$, $(4, 6)$: $+4$. The best non-tabu move is $(2, 4)$ with $\delta = 0$: it removes the edges $(3, 2)$, $(6, 5)$, which become tabu until iteration 3; new tour $(1, 3, 6, 2, 5, 4)$ of length 22. The search moves although no move improves: it leaves the local optimum through a tour of the same length.

3. - From $(1, 3, 6, 2, 5, 4)$ the neighbours have $\delta$: $(1, 3)$: $+6$, $(1, 4)$: $+8$, $(1, 5)$: $+5$, $(2, 4)$: $+0$, $(2, 5)$: $+1$, $(2, 6)$: $-1$, $(3, 5)$: $-1$, $(3, 6)$: $+3$, $(4, 6)$: $+3$. The best non-tabu move is $(2, 6)$ with $\delta = -1$: it removes the edges $(3, 6)$, $(4, 1)$, which become tabu until iteration 4; new tour $(1, 3, 4, 5, 2, 6)$ of length 21.
   - From $(1, 3, 4, 5, 2, 6)$ the neighbours have $\delta$: $(1, 3)$: $+5$, $(1, 4)$: $+8$, $(1, 5)$: $+7$, $(2, 4)$: $+3$, $(2, 5)$: $+3$, $(2, 6)$: $+1$, $(3, 5)$: $-1$, $(3, 6)$: $+2$, $(4, 6)$: $+1$. The best non-tabu move is $(3, 5)$ with $\delta = -1$: it removes the edges $(4, 5)$, $(2, 6)$, which become tabu until iteration 5; new tour $(1, 3, 4, 2, 5, 6)$ of length 20.

4. After 3 iterations the best tour is $(1, 3, 4, 2, 5, 6)$, of length 20: it is the optimal tour (by enumeration of the 60 tours), better than the local optimum 22 where the 2-opt local search stops.

## 4.3 Iterated local search and VNS

**Exercise 4.7 — Iterated local search and VNS 1**

A TSP instance with 6 cities and the symmetric distances

| $d_{ij}$ | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| 1 | — | 5 | 1 | 7 | 7 | 4 |
| 2 | 5 | — | 4 | 5 | 2 | 1 |
| 3 | 1 | 4 | — | 6 | 6 | 4 |
| 4 | 7 | 5 | 6 | — | 7 | 6 |
| 5 | 7 | 2 | 6 | 7 | — | 2 |
| 6 | 4 | 1 | 4 | 6 | 2 | — |

The 2-opt local optimum is the nearest neighbour tour $(1, 3, 2, 6, 5, 4)$ of length 22. Iterated local search uses the double-bridge kick with cut positions $p_1 = 1 + \lfloor u_1(n-3) \rfloor$, $p_2 = p_1 + 1 + \lfloor u_2(n-2-p_1) \rfloor$, $p_3 = p_2 + 1 + \lfloor u_3(n-1-p_2) \rfloor$ ($A = \sigma_1..\sigma_{p_1}$, $B = \sigma_{p_1+1}..\sigma_{p_2}$, $C = \sigma_{p_2+1}..\sigma_{p_3}$, $D = \sigma_{p_3+1}..\sigma_n$, new tour $A\,C\,B\,D$), the 2-opt local search with best improvement, and accepts a new local optimum if it is shorter. The random numbers are 0.13, 0.76, 0.46.

**Questions**

1. Compute the cut positions $p_1, p_2, p_3$ and the four segments of the tour.
2. Build the perturbed tour, list the edges removed and added, and compute its length.
3. Apply the 2-opt local search (best improvement) to the perturbed tour.
4. Is the new local optimum accepted? Compare it with the optimal tour, of length 20.

**Solution**

1. $p_1 = 1 + \lfloor 0.13 \cdot 3 \rfloor = 1$, $p_2 = 1 + 1 + \lfloor 0.76 \cdot 3 \rfloor = 4$, $p_3 = 4 + 1 + \lfloor 0.46 \cdot 1 \rfloor = 5$. Segments $A = (1)$, $B = (3, 2, 6)$, $C = (5)$, $D = (4)$.
2. The perturbed tour $A\,C\,B\,D$ is $(1, 5, 3, 2, 6, 4)$; it removes the edges $(4, 5)$, $(5, 6)$, $(1, 3)$ and adds $(1, 5)$, $(4, 6)$, $(3, 5)$; its length is 31.
3. From $(1, 5, 3, 2, 6, 4)$: the best move is $(1, 3)$ with $\delta = -8$, giving $(1, 3, 5, 2, 6, 4)$ of length 23; the best move is $(2, 4)$ with $\delta = -1$, giving $(1, 3, 2, 5, 6, 4)$ of length 22; the best move is $(2, 5)$ with $\delta = -1$, giving $(1, 3, 6, 5, 2, 4)$ of length 21; the best move is $(2, 6)$ with $\delta = -1$, giving $(1, 3, 4, 2, 5, 6)$ of length 20; then no move improves the tour. The new local optimum is $(1, 3, 4, 2, 5, 6)$, of length 20.
4. Yes: $20 < 22$, the new local optimum replaces the current one. Its length is 20: it is the optimal tour. One kick and one local search have escaped from the local optimum where the 2-opt local search alone stops.

**Exercise 4.8 — Iterated local search and VNS 2**

A TSP instance with 6 cities and the symmetric distances

| $d_{ij}$ | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| 1 | — | 5 | 1 | 7 | 7 | 4 |
| 2 | 5 | — | 4 | 5 | 2 | 1 |
| 3 | 1 | 4 | — | 6 | 6 | 4 |
| 4 | 7 | 5 | 6 | — | 7 | 6 |
| 5 | 7 | 2 | 6 | 7 | — | 2 |
| 6 | 4 | 1 | 4 | 6 | 2 | — |

The 2-opt local optimum is the nearest neighbour tour $(1, 3, 2, 6, 5, 4)$ of length 22. Iterated local search uses the double-bridge kick with cut positions $p_1 = 1 + \lfloor u_1(n-3) \rfloor$, $p_2 = p_1 + 1 + \lfloor u_2(n-2-p_1) \rfloor$, $p_3 = p_2 + 1 + \lfloor u_3(n-1-p_2) \rfloor$ ($A = \sigma_1..\sigma_{p_1}$, $B = \sigma_{p_1+1}..\sigma_{p_2}$, $C = \sigma_{p_2+1}..\sigma_{p_3}$, $D = \sigma_{p_3+1}..\sigma_n$, new tour $A\,C\,B\,D$), the 2-opt local search with best improvement, and accepts a new local optimum if it is shorter. The random numbers are 0.16, 0.13, 0.01.

**Questions**

1. Compute the cut positions $p_1, p_2, p_3$ and the four segments of the tour.

2. Build the perturbed tour, list the edges removed and added, and compute its length.
3. Apply the 2-opt local search (best improvement) to the perturbed tour.
4. Is the new local optimum accepted? What would happen with the acceptance criterion "always accept"?

**Solution**

1. $p_1 = 1 + \lfloor 0.16 \cdot 3 \rfloor = 1$, $p_2 = 1 + 1 + \lfloor 0.13 \cdot 3 \rfloor = 2$, $p_3 = 2 + 1 + \lfloor 0.01 \cdot 3 \rfloor = 3$. Segments $A = (1)$, $B = (3)$, $C = (2)$, $D = (6, 5, 4)$.
2. The perturbed tour $A\,C\,B\,D$ is $(1, 2, 3, 6, 5, 4)$; it removes the edges $(1, 3)$, $(2, 6)$ and adds $(3, 6)$, $(1, 2)$; its length is 29.
3. From $(1, 2, 3, 6, 5, 4)$: the best move is $(1, 3)$ with $\delta = -7$, giving $(1, 3, 2, 6, 5, 4)$ of length 22; then no move improves the tour. The new local optimum is $(1, 3, 2, 6, 5, 4)$, of length 22.
4. The new local optimum is the starting tour itself, of length 22: it is not shorter, and the current solution does not change. The kick was too weak: with segments $B$ and $C$ of a single city it changes only two edges, which is a 2-opt move, and the 2-opt local search undoes it at once. With "always accept" nothing would change either; the strength of the kick, not the acceptance criterion, is the problem here.

**Exercise 4.9 — Iterated local search and VNS 3**

$m = 3$ identical machines and 7 jobs with processing times $t = (4, 4, 3, 3, 2, 2, 2)$. Variable neighbourhood search starts from the LPT solution, uses $k_{\max} = 3$, shakes with $k$ random moves (each move: job $j = 1 + \lfloor 7u \rfloor$ goes to the $g$-th machine different from its own, $g = 1 + \lfloor 2u \rfloor$, with two consecutive random numbers), uses as local search the move descent with best improvement in the lexicographic order $(C_{\max}, \sum_h L_h^2)$, and stops when $C_{\max} = LB$. The random numbers are 0.26, 0.51, 0.92, 0.07, 0.44, 0.09, 0.36, 0.36, 0.87, 0.77, 0.04, 0.66.

**Questions**

1. Compute the LPT solution and the lower bound $LB$, and show that LPT is a local optimum for the move neighbourhood.
2. Explain why the swap of two jobs belongs to the neighbourhood $N_2$ used by the shaking.
3. Apply VNS, showing for each iteration the shaking, the local search and the decision.
4. Is the final solution optimal? Which value of $k$ was needed?

**Solution**

1. LPT assigns the jobs by non-increasing time to the least loaded machine: loads $(8, 6, 6)$, $J_1 = \{1, 5, 7\}$, $J_2 = \{2, 6\}$, $J_3 = \{3, 4\}$, $C_{\max} = 8$; $LB = \max\{4, \lceil 20/3 \rceil\} = 7$. A job moved to a machine of load 6 brings it to at least 8, and a job moved to $M_1$ brings it above 8: no move decreases the makespan, and the moves with makespan 8 (for instance job 5 from $M_1$ to $M_2$, loads $(6, 8, 6)$) have $\sum_h L_h^2 \geq 136$, the value of LPT. LPT is a local optimum.

2. Exchanging job $j$ on $M_a$ with job $j'$ on $M_b$ is the same as two moves: $j$ to $M_b$ and $j'$ to $M_a$. So every swap neighbour is in $N_2$, and the shaking with $k = 2$ can produce it.

3. - Iteration 1, $k = 1$: shaking job 2: $M_2 \to M_3$, loads $(8, 2, 10)$; MoveDescent: job 2 from $M_3$ to $M_2$, loads $(8, 6, 6)$; result loads $(8, 6, 6)$, not better ($k := k + 1$).
   - Iteration 2, $k = 2$: shaking job 7: $M_1 \to M_2$; job 4: $M_3 \to M_1$, loads $(9, 8, 3)$; MoveDescent: job 4 from $M_1$ to $M_3$, loads $(6, 8, 6)$; result loads $(6, 8, 6)$, not better ($k := k + 1$).
   - Iteration 3, $k = 3$: shaking job 3: $M_3 \to M_1$; job 7: $M_1 \to M_3$; job 1: $M_1 \to M_3$, loads $(5, 6, 9)$; MoveDescent: job 7 from $M_3$ to $M_1$, loads $(7, 6, 7)$; result loads $(7, 6, 7)$, better, accepted ($k := 1$).

4. The final makespan is $7 = LB$: the solution is optimal. The neighbourhoods $N_1$ and $N_2$ failed, and $N_3$ (three random moves) was needed.

## 4.4 GRASP

**Exercise 4.10 — GRASP 1**

A knapsack of capacity $c = 16$ and 5 items:

| | $j = 1$ | $j = 2$ | $j = 3$ | $j = 4$ | $j = 5$ |
|---|---|---|---|---|---|
| $p_j$ | 11 | 5 | 18 | 8 | 8 |
| $w_j$ | 5 | 3 | 11 | 6 | 7 |

GRASP with $\alpha = 0.5$: the RCL contains the items that fit with efficiency at least $e_{\max} - \alpha(e_{\max} - e_{\min})$, sorted by non-increasing efficiency, and the item in position $1 + \lfloor u|RCL| \rfloor$ is chosen; the local search uses feasible flips and swaps with best improvement. The random numbers are 0.39, 0.27, 0.38, 0.60, 0.66.

**Questions**

1. Compute the greedy solution by efficiency and the solution of the local search started from it.

2. Perform the first iteration of GRASP: construction and local search.

3. Perform the second iteration.

4. Compare the best solution of GRASP with the optimum, and explain the role of the randomization.

**Solution**

1. Greedy gives $(1,1,0,1,0)$ of profit 24 (items $1,2,4$, weight 14); the local search finds no improving flip or swap: it stops at 24.

2. - Iteration 1: $rc = 16$, candidates $\{1,2,3,4,5\}$, threshold 1.6714, $RCL = (1)$, $u = 0.39$ chooses position 1: item 1; $rc = 11$, candidates $\{2,3,4,5\}$, threshold 1.4048, $RCL = (2,3)$, $u = 0.27$ chooses position 1: item 2; $rc = 8$, candidates $\{4,5\}$, threshold 1.2381, $RCL = (4)$, $u = 0.38$ chooses position 1: item 4. Construction: $(1,1,0,1,0)$, profit 24. Local search: no improving move. Result: profit 24.

3. - Iteration 2: $rc = 16$, candidates $\{1,2,3,4,5\}$, threshold 1.6714, $RCL = (1)$, $u = 0.60$ chooses position 1: item 1; $rc = 11$, candidates $\{2,3,4,5\}$, threshold 1.4048, $RCL = (2,3)$, $u = 0.66$ chooses position 2: item 3. Construction: $(1,0,1,0,0)$, profit 29. Local search: no improving move. Result: profit 29.

4. The best solution is $(1,0,1,0,0)$ of profit 29, which is optimal (by enumeration: items $\{1,3\}$ fill the capacity exactly). The deterministic greedy algorithm never selects item 3, because after items 1 and 2 it does not fit; the randomized construction sometimes chooses item 3 before item 2, and reaches a different valley.

**Exercise 4.11 — GRASP 2**

A TSP instance with 6 cities and the symmetric distances

| $d_{ij}$ | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| 1 | — | 5 | 1 | 7 | 7 | 4 |
| 2 | 5 | — | 4 | 5 | 2 | 1 |
| 3 | 1 | 4 | — | 6 | 6 | 4 |
| 4 | 7 | 5 | 6 | — | 7 | 6 |
| 5 | 7 | 2 | 6 | 7 | — | 2 |
| 6 | 4 | 1 | 4 | 6 | 2 | — |

GRASP for the TSP: randomized nearest neighbour from city 1 with $\alpha = 0.5$ (the RCL contains the unvisited cities $j$ with $d_{\ell j} \le d_{\min} + \alpha(d_{\max} - d_{\min})$, sorted by distance, ties by index; position $1 + \lfloor u|RCL| \rfloor$), followed by the 2-opt local search with best improvement. The random numbers are 0.45, 0.31, 0.05, 0.86, 0.41, 0.52, 0.47, 0.47, 0.28, 0.22.

**Questions**

1. Perform the first iteration of GRASP.

2. Perform the second iteration.

3. Which is the best tour found? Compare it with the nearest neighbour tour.
4. What would GRASP give with $\alpha = 0$, for any number of iterations?

**Solution**

1. Construction: from 1: threshold 4, $RCL = (3, 6)$, $u = 0.45$: city 3; from 3: threshold 5, $RCL = (2, 6)$, $u = 0.31$: city 2; from 2: threshold 3, $RCL = (6, 5)$, $u = 0.05$: city 6; from 6: threshold 4, $RCL = (5)$, $u = 0.86$: city 5; from 5: threshold 7, $RCL = (4)$, $u = 0.41$: city 4. Tour $(1, 3, 2, 6, 5, 4)$, length 22. Local search: no 2-opt move improves the tour: it is already a 2-opt local optimum. Result: length 22.
2. Construction: from 1: threshold 4, $RCL = (3, 6)$, $u = 0.52$: city 6; from 6: threshold 3.5, $RCL = (2, 5)$, $u = 0.47$: city 2; from 2: threshold 3.5, $RCL = (5)$, $u = 0.47$: city 5; from 5: threshold 6.5, $RCL = (3)$, $u = 0.28$: city 3; from 3: threshold 6, $RCL = (4)$, $u = 0.22$: city 4. Tour $(1, 6, 2, 5, 3, 4)$, length 26. Local search: the best move is $(4, 6)$ with $\delta = -5$, giving $(1, 6, 2, 5, 4, 3)$ of length 21; the best move is $(2, 4)$ with $\delta = -1$, giving $(1, 6, 5, 2, 4, 3)$ of length 20; then no move improves the tour. Result: length 20.
3. The best tour is $(1, 6, 5, 2, 4, 3)$, of length 20, the optimum. The nearest neighbour tour, $(1, 3, 2, 6, 5, 4)$ of length 22, is a 2-opt local optimum: the second construction, worse than nearest neighbour (26), leads the local search into a better valley.
4. With $\alpha = 0$ the RCL contains only the nearest cities: the construction is the nearest neighbour heuristic (up to ties), every iteration builds $(1, 3, 2, 6, 5, 4)$, and the local search stops at 22.

**Exercise 4.12 — GRASP 3**

A knapsack of capacity $c = 16$ and 5 items:

| | $j = 1$ | $j = 2$ | $j = 3$ | $j = 4$ | $j = 5$ |
|---|---|---|---|---|---|
| $p_j$ | 11 | 5 | 18 | 8 | 8 |
| $w_j$ | 5 | 3 | 11 | 6 | 7 |

GRASP with 2 iterations, RCL by efficiency threshold $e_{\max} - \alpha(e_{\max} - e_{\min})$ (position $1 + \lfloor u|RCL| \rfloor$), local search with feasible flips and swaps (best improvement). The random numbers are 0.92, 0.29, 0.21, 0.73, 0.53, 0.33, 0.75, 0.76, 0.54, 0.68, 0.64, 0.82.

**Questions**

1. Which items form the RCL at the first step with $\alpha = 0$, $\alpha = 0.5$ and $\alpha = 1$?
2. Apply GRASP with $\alpha = 0$.
3. Apply GRASP with $\alpha = 1$, using the random numbers from the beginning.
4. Compare the two runs: which one diversifies, and which one intensifies?

**Solution**

1. $e_{\max} = \frac{11}{5} = 2.2$, $e_{\min} = \frac{8}{7} \approx 1.14$. With $\alpha = 0$: $RCL = (1)$; with $\alpha = 0.5$ (threshold 1.6714): $RCL = (1)$; with $\alpha = 1$: all items, $RCL = (1, 2, 3, 4, 5)$.

2. - Iteration 1: $rc = 16$, candidates $\{1, 2, 3, 4, 5\}$, threshold 2.2000, $RCL = (1)$, $u = 0.92$ chooses position 1: item 1; $rc = 11$, candidates $\{2, 3, 4, 5\}$, threshold 1.6667, $RCL = (2)$, $u = 0.29$ chooses position 1: item 2; $rc = 8$, candidates $\{4, 5\}$, threshold 1.3333, $RCL = (4)$, $u = 0.21$ chooses position 1: item 4. Construction: $(1, 1, 0, 1, 0)$, profit 24. Local search: no improving move. Result: profit 24.
   - Iteration 2: $rc = 16$, candidates $\{1, 2, 3, 4, 5\}$, threshold 2.2000, $RCL = (1)$, $u = 0.73$ chooses position 1: item 1; $rc = 11$, candidates $\{2, 3, 4, 5\}$, threshold 1.6667, $RCL = (2)$, $u = 0.53$ chooses position 1: item 2; $rc = 8$, candidates $\{4, 5\}$, threshold 1.3333, $RCL = (4)$, $u = 0.33$ chooses position 1: item 4. Construction: $(1, 1, 0, 1, 0)$, profit 24. Local search: no improving move. Result: profit 24.

   With $\alpha = 0$ both iterations build the greedy solution and stop at 24: the random numbers are irrelevant (each RCL has one element).

3. - Iteration 1: $rc = 16$, candidates $\{1, 2, 3, 4, 5\}$, threshold 1.1429, $RCL = (1, 2, 3, 4, 5)$, $u = 0.92$ chooses position 5: item 5; $rc = 9$, candidates $\{1, 2, 4\}$, threshold 1.3333, $RCL = (1, 2, 4)$, $u = 0.29$ chooses position 1: item 1; $rc = 4$, candidates $\{2\}$, threshold 1.6667, $RCL = (2)$, $u = 0.21$ chooses position 1: item 2. Construction: $(1, 1, 0, 0, 1)$, profit 24. Local search: no improving move. Result: profit 24.
   - Iteration 2: $rc = 16$, candidates $\{1, 2, 3, 4, 5\}$, threshold 1.1429, $RCL = (1, 2, 3, 4, 5)$, $u = 0.73$ chooses position 4: item 4; $rc = 10$, candidates $\{1, 2, 5\}$, threshold 1.1429, $RCL = (1, 2, 5)$, $u = 0.53$ chooses position 2: item 2; $rc = 7$, candidates $\{1, 5\}$, threshold 1.1429, $RCL = (1, 5)$, $u = 0.33$ chooses position 1: item 1. Construction: $(1, 1, 0, 1, 0)$, profit 24. Local search: no improving move. Result: profit 24.

   Best profit: 24.

4. With $\alpha = 0$ GRASP is a single local search from the greedy solution: pure intensification, always 24. With $\alpha = 1$ the constructions are random and different (diversification), and the local search intensifies around each of them; here the best result is 24, while the optimum is 29.

## 4.5 Large neighbourhood search

**Exercise 4.13 — Large neighbourhood search 1**

Bins of capacity $c = 10$ and 8 items with weights $w = (2, 7, 5, 5, 3, 2, 6, 6)$. LNS empties the least loaded bin (ties: smallest index) and one other bin chosen at random (the bin in position $1 + \lfloor u \cdot (nb - 1) \rfloor$ among the others), repacks the removed items by FFD into new bins, keeps the other bins, and accepts the new packing if it uses fewer bins, or as many bins and a sum of squared loads not smaller. It stops when $nb$ reaches $\lceil \sum_j w_j / c \rceil$. The random numbers are 0.13, 0.76, 0.46.

**Questions**

1. Compute the First Fit packing and the lower bound $\lceil \sum_j w_j / c \rceil$.
2. Perform the first iteration of LNS.
3. Continue until the search stops.
4. Is the final packing optimal? Why are packings with the same number of bins accepted?

**Solution**

1. First Fit gives $P_1 = \{1, 2\}$, $P_2 = \{3, 4\}$, $P_3 = \{5, 6\}$, $P_4 = \{7\}$, $P_5 = \{8\}$: 5 bins. $\sum_j w_j = 36$, so at least $\lceil 36/10 \rceil = 4$ bins are needed.
2. - Iteration 1: loads $(9, 10, 5, 6, 6)$; emptied bins $P_3, P_1$; removed items $\{1, 2, 5, 6\}$; FFD repacking: $P_1 = \{3, 4\}$, $P_2 = \{7\}$, $P_3 = \{8\}$, $P_4 = \{2, 5\}$, $P_5 = \{1, 6\}$; $nb' = 5$, $\sum_b (\cdot)^2$: $278 \to 288$: accepted.
3. - Iteration 2: loads $(10, 6, 6, 10, 4)$; emptied bins $P_5, P_4$; removed items $\{1, 2, 5, 6\}$; FFD repacking: $P_1 = \{3, 4\}$, $P_2 = \{7\}$, $P_3 = \{8\}$, $P_4 = \{2, 5\}$, $P_5 = \{1, 6\}$; $nb' = 5$, $\sum_b (\cdot)^2$: $288 \to 288$: accepted.
   - Iteration 3: loads $(10, 6, 6, 10, 4)$; emptied bins $P_5, P_2$; removed items $\{1, 6, 7\}$; FFD repacking: $P_1 = \{3, 4\}$, $P_2 = \{8\}$, $P_3 = \{2, 5\}$, $P_4 = \{1, 6, 7\}$; $nb' = 4$, $\sum_b (\cdot)^2$: $288 \to 336$: accepted.
4. The final packing $P_1 = \{3, 4\}$, $P_2 = \{8\}$, $P_3 = \{2, 5\}$, $P_4 = \{1, 6, 7\}$ uses 4 bins, equal to the lower bound: it is optimal. Accepting packings with the same number of bins and a larger (or equal) sum of squared loads lets the search move on the plateau of the flat objective, toward packings with fuller bins and one almost empty bin, which is easier to eliminate.

## Exercise 4.14 — Large neighbourhood search 2

Bins of capacity $c = 10$ and 8 items with weights $w = (3, 3, 6, 5, 4, 4, 2, 3)$. LNS starts from the FFD packing, empties the least loaded bin and two other bins at random ($q = 3$; each one is the bin in position $1 + \lfloor uK \rfloor$ among the $K$ bins not yet chosen), and repacks the removed items into new bins, either by FFD or optimally. The random numbers are 0.66, 0.78.

**Questions**

1. Compute the FFD packing and the lower bound.
2. Perform one iteration of LNS with the FFD repair.
3. Perform the same iteration with the exact repair: find the minimum number of bins for the removed items.
4. Compare the two repairs. Can LNS with FFD repair improve an FFD packing in this way?

## Solution

1. FFD (weights $6, 5, 4, 4, 3, 3, 3, 2$) gives $P_1 = \{3, 5\}$, $P_2 = \{4, 6\}$, $P_3 = \{1, 2, 8\}$, $P_4 = \{7\}$: 4 bins; $\lceil 30/10 \rceil = 3$.
2. - Iteration 1: loads $(10, 9, 9, 2)$; emptied bins $P_4, P_2, P_3$; removed items $\{1, 2, 4, 6, 7, 8\}$; FFD repacking: $P_1 = \{3, 5\}$, $P_2 = \{4, 6\}$, $P_3 = \{1, 2, 8\}$, $P_4 = \{7\}$; $nb' = 4$, $\sum_b (\cdot)^2$: $266 \to 266$: accepted.
3. The removed items $\{1, 2, 4, 6, 7, 8\}$ have total weight $20 > 10$, so at least 2 bins are needed, and 2 bins suffice:
   - Iteration 1: loads $(10, 9, 9, 2)$; emptied bins $P_4, P_2, P_3$; removed items $\{1, 2, 4, 6, 7, 8\}$; exact repacking: $P_1 = \{3, 5\}$, $P_2 = \{1, 4, 7\}$, $P_3 = \{2, 6, 8\}$; $nb' = 3$, $\sum_b (\cdot)^2$: $266 \to 300$: accepted.
4. The FFD repair repacks the removed items as FFD does, and finds 3 bins for them; the exact repair finds 2, and the packing becomes optimal (3 bins). Since the removed items are repacked by the same rule that built the packing, LNS with FFD repair tends to rebuild the same bins: with 200 iterations ($seed = 1$) it never goes below 4. The power of a large neighbourhood depends on the method that explores it.

## Exercise 4.15 — Large neighbourhood search 3

A TSP instance with 6 cities and the symmetric distances

| $d_{ij}$ | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| 1 | — | 5 | 1 | 7 | 7 | 4 |
| 2 | 5 | — | 4 | 5 | 2 | 1 |
| 3 | 1 | 4 | — | 6 | 6 | 4 |
| 4 | 7 | 5 | 6 | — | 7 | 6 |
| 5 | 7 | 2 | 6 | 7 | — | 2 |
| 6 | 4 | 1 | 4 | 6 | 2 | — |

LNS starts from the nearest neighbour tour $(1, 3, 2, 6, 5, 4)$ (length 22), removes $q = 2$ cities at random (the city in position $2 + \lfloor u(|\sigma| - 1) \rfloor$ of the current partial tour, so that city 1 is never removed), reinserts them in the order of removal by cheapest insertion (ties: first position), and accepts the new tour if it is shorter. The random numbers are 0.68, 0.16, 0.84, 0.92.

**Questions**

1. Perform the first iteration (destroy and repair) and decide whether the new tour is accepted.
2. Perform the second iteration.
3. Compute all the insertion costs of the first removed city in the second iteration.
4. Is the final tour optimal? How many edges of the starting tour have been changed?

**Solution**

1. Removed cities: $5, 3$; partial tour $(1, 2, 6, 4)$; insertions: city 5 between 2 and 6 (cost 3); city 3 between 1 and 2 (cost 0); new tour $(1, 3, 2, 5, 6, 4)$ of length 22: rejected.
2. Removed cities: $4, 5$; partial tour $(1, 3, 2, 6)$; insertions: city 4 between 3 and 2 (cost 7); city 5 between 2 and 6 (cost 3); new tour $(1, 3, 4, 2, 5, 6)$ of length 20: accepted.
3. Inserting city 4 into $(1, 3, 2, 6)$: $(1, 3)$: 12, $(3, 2)$: 7, $(2, 6)$: 10, $(6, 1)$: 9; the minimum cost is 7, between cities 3 and 2.
4. The final tour $(1, 3, 4, 2, 5, 6)$ has length 20: it is the optimal tour. It shares 2 edges with the starting tour: one destroy-and-repair step has changed 4 edges, a move that 2-opt cannot perform in one step.

## 4.6 Genetic algorithms

**Exercise 4.16 — Genetic algorithms 1**

A knapsack of capacity $c = 16$ and 5 items:

| | $j=1$ | $j=2$ | $j=3$ | $j=4$ | $j=5$ |
|---|---|---|---|---|---|
| $p_j$ | 11 | 5 | 18 | 8 | 8 |
| $w_j$ | 5 | 3 | 11 | 6 | 7 |

Steady-state genetic algorithm with population $pop[1] = (0,0,0,1,1)$, $pop[2] = (0,1,0,0,1)$, $pop[3] = (0,1,0,1,1)$, $pop[4] = (1,1,0,1,0)$; binary tournament (individuals $1 + \lfloor 4u \rfloor$, the fitter wins, ties: the first), one-point crossover with cut $1 + \lfloor 4u \rfloor$ (genes $1..cut$ from $a$, the others from $b$), mutation of each gene if $u < p_m = 0.2$, repair by removing the selected item of minimum efficiency until feasible; the child replaces the worst individual (ties: last index) if it is better and not already present. Each iteration uses 10 random numbers; they are 0.58, 0.07, 0.50, 0.39, 0.63, 0.56, 0.15, 0.15, 0.22, 0.60, 0.23, 0.97, 0.41, 0.64, 0.36, 0.05, 0.09, 0.60, 0.80, 0.01.

**Questions**

1. Compute the fitness of the initial population. Which items never appear in it?
2. Perform the first iteration of the genetic algorithm.
3. Perform the second iteration.
4. Can crossover alone ever produce a solution containing item 3? Which operator makes it possible?

**Solution**

1. Fitness: 16, 13, 21, 24 (weights 13, 10, 16, 14). Item 3 is in no individual.
2. - Iteration 1: Tournament $(3,1)$ selects $a = 3$, tournament $(3,2)$ selects $b = 3$; cut $= 3$: child $(0,1,0,1,1)$; mutation of genes $\{2,3\}$: $(0,0,1,1,1)$; repair removes items $\{5,4\}$: $(0,0,1,0,0)$, profit 18; worst individual 2; inserted. Population fitness: $(16,18,21,24)$.
3. - Iteration 2: Tournament $(1,4)$ selects $a = 4$, tournament $(2,3)$ selects $b = 3$; cut $= 2$: child $(1,1,0,1,1)$; mutation of genes $\{1,2,5\}$: $(0,0,0,1,0)$; feasible, profit 8; worst individual 1; not inserted. Population fitness: $(16,18,21,24)$.
4. No: crossover copies genes from the parents, and gene 3 is 0 in every individual. Only mutation can set it to 1, as in iteration 1; but the child is infeasible, and the repair removes the least efficient items, not item 3, so the result is the solution $(0,0,1,0,0)$ of profit 18. The best individual after two iterations has profit $24 < 29$.

## Exercise 4.17 — Genetic algorithms 2

A knapsack of capacity $c = 16$ and 5 items:

| | $j=1$ | $j=2$ | $j=3$ | $j=4$ | $j=5$ |
|---|---|---|---|---|---|
| $p_j$ | 11 | 5 | 18 | 8 | 8 |
| $w_j$ | 5 | 3 | 11 | 6 | 7 |

Steady-state genetic algorithm with population $pop[1] = (0,0,0,1,1)$, $pop[2] = (0,1,0,0,1)$, $pop[3] = (0,1,0,1,1)$, $pop[4] = (1,1,0,1,0)$; binary tournament (individuals $1 + \lfloor 4u \rfloor$, the fitter wins, ties: the first), one-point crossover with cut $1 + \lfloor 4u \rfloor$ (genes $1..cut$ from $a$, the others from $b$), mutation of each gene if $u < p_m = 0.2$, repair by removing the selected item of minimum efficiency until feasible; the child replaces the worst individual (ties: last index) if it is better and not already present. Each iteration uses 10 random numbers; they are 0.58, 0.07, 0.50, 0.39, 0.63, 0.56, 0.15, 0.15, 0.22, 0.60, 0.23, 0.97, 0.41, 0.64, 0.36, 0.05, 0.09, 0.60, 0.80, 0.01.

**Questions**

1. Apply the local search (feasible flips and swaps, best improvement) to each individual of the initial population, as the memetic algorithm does.
2. Perform the first iteration of the memetic algorithm with the same random numbers.
3. Perform the second iteration.
4. Compare with the genetic algorithm without local search, which after the same two iterations has best profit 24.

## Solution

1. After the local search the population is $(1,1,0,0,1)$ (24), $(1,1,0,0,1)$ (24), $(1,1,0,0,1)$ (24), $(1,1,0,1,0)$ (24).
2. - Iteration 1: Tournament $(3,1)$ selects $a = 3$, tournament $(3,2)$ selects $b = 3$; cut = 3: child $(1,1,0,0,1)$; mutation of genes $\{2,3\}$: $(1,0,1,0,1)$; repair removes items $\{5\}$: $(1,0,1,0,0)$; local search: $(1,0,1,0,0)$, profit 29; worst individual 4; inserted. Population fitness: $(24,24,24,29)$.
3. - Iteration 2: Tournament $(1,4)$ selects $a = 4$, tournament $(2,3)$ selects $b = 2$; cut = 2: child $(1,0,0,0,1)$; mutation of genes $\{1,2,5\}$: $(0,1,0,0,0)$; feasible; local search: $(1,0,1,0,0)$, profit 29; worst individual 3; not inserted. Population fitness: $(24,24,24,29)$.
4. The memetic algorithm reaches the optimum 29 (items $\{1,3\}$) in the first iteration: the mutation brings item 3 into the child, and the local search turns the repaired child into a local optimum containing item 3. Without local search the same child

is only repaired, and its value is poor. The local search adds intensification to the diversification of crossover and mutation.

Chapter 5

# MIP heuristics for the classical problems

## 5.1 Rounding and diving

**Exercise 5.1 — Rounding and diving on the KP (KP1)**

We consider a knapsack of capacity $c = 16$ and a set $\mathscr{I} = \{1, 2, 3, 4, 5\}$ of 5 items, with the following profits and weights:

| | $j = 1$ | $j = 2$ | $j = 3$ | $j = 4$ | $j = 5$ |
|---|---|---|---|---|---|
| $p_j$ | 11 | 15 | 18 | 14 | 10 |
| $w_j$ | 3 | 5 | 7 | 10 | 9 |

**Questions**

1. Verify that the items are sorted by non-increasing efficiency, and compute the optimal solution $\tilde{x}$ of the LP relaxation of the KP model and its value $z(\mathrm{LP}_{\mathrm{KP}})$.
2. Apply rounding: round $\tilde{x}$ to the nearest integer point $\bar{x}$, check whether $\bar{x}$ is feasible, and give its profit.
3. Apply fractional diving: at each step solve the LP relaxation with the variables fixed so far, fix the least fractional variable to its nearest integer, and repeat until the LP solution is integer.
4. The dive is a path of the branch-and-bound tree. Compute the value of the LP relaxation of the branches that the dive does not visit, and use them to decide whether the solution of the dive is optimal.

**Solution**

1. The efficiencies are

$$\frac{11}{3} \approx 3.67, \quad \frac{15}{5} = 3, \quad \frac{18}{7} \approx 2.57, \quad \frac{14}{10} = 1.4, \quad \frac{10}{9} \approx 1.11,$$

non-increasing. The LP takes items $1, 2, 3$ (weight $3 + 5 + 7 = 15$), and the residual capacity 1 is filled by the fraction $\frac{1}{10}$ of item 4:

$$\tilde{x} = \left(1, 1, 1, \frac{1}{10}, 0\right), \qquad z(\mathrm{LP_{KP}}) = 11 + 15 + 18 + \frac{14}{10} = \frac{227}{5} = 45.4.$$

2. $\left\lfloor \frac{1}{10} \right\rfloor = 0$, so $\bar{x} = (1, 1, 1, 0, 0)$: weight $15 \leq 16$, feasible, profit 44.

3. **Step 1.** The only fractional variable is $x_4 = \frac{1}{10}$: fix $x_4 := 0$.

   **Step 2.** Without item 4, the LP takes items $1, 2, 3$ and the fraction $\frac{1}{9}$ of item 5: $\tilde{x} = \left(1, 1, 1, 0, \frac{1}{9}\right)$, of value $44 + \frac{10}{9} = \frac{406}{9} \approx 45.11$. Fix $x_5 := \left\lfloor \frac{1}{9} \right\rfloor = 0$.

   **Step 3.** The LP solution $(1, 1, 1, 0, 0)$, of value 44, is integer: the dive returns $S = \{1, 2, 3\}$, of profit 44.

4. The dive does not visit the branches $x_4 = 1$ and $(x_4 = 0, x_5 = 1)$.

   - $x_4 = 1$: item 4 uses 10 of the capacity, profit 14; the LP adds item 1 (residual 3) and the fraction $\frac{3}{5}$ of item 2: value $14 + 11 + 9 = 34$.
   - $x_4 = 0, x_5 = 1$: item 5 uses 9, profit 10; the LP adds item 1 (residual 4) and the fraction $\frac{4}{5}$ of item 2: value $10 + 11 + 12 = 33$.

   Both values are smaller than 44: no solution in the abandoned branches is better than the solution of the dive, which is therefore **optimal**, $z(I) = 44$. The dive, together with the bounds of the branches it leaves, is a complete branch-and-bound.

## Exercise 5.2 — Rounding and diving on the KP (KP2)

We consider a knapsack of capacity $c = 15$ and a set $\mathscr{I} = \{1, 2, 3, 4, 5\}$ of 5 items, with the following profits and weights:

| | $j = 1$ | $j = 2$ | $j = 3$ | $j = 4$ | $j = 5$ |
|---|---|---|---|---|---|
| $p_j$ | 13 | 12 | 14 | 6 | 5 |
| $w_j$ | 2 | 5 | 10 | 6 | 7 |

**Questions**

1. Compute the optimal solution $\tilde{x}$ of the LP relaxation and its value $z(\mathrm{LP_{KP}})$ (the items are sorted by non-increasing efficiency).
2. Round $\tilde{x}$ to the nearest integer point: is it feasible? Round it down instead, and give the profit.
3. Apply fractional diving, step by step, and give the profit of the solution it returns.
4. Repeat the dive starting with the opposite decision at the first step, and continue with the rule of fractional diving. Then use the LP bounds of the branches to prove that the best of the two solutions is optimal.

**Solution**

1. The efficiencies are 6.5, 2.4, 1.4, 1, $\frac{5}{7} \approx 0.71$. The LP takes items $1, 2$ (weight 7) and the fraction $\frac{8}{10} = \frac{4}{5}$ of item 3:

$$\tilde{x} = \left(1, 1, \frac{4}{5}, 0, 0\right), \qquad z(\mathrm{LP_{KP}}) = 13 + 12 + 14 \cdot \frac{4}{5} = \frac{181}{5} = 36.2.$$

2. $\left\lfloor \frac{4}{5} \right\rfloor = 1$ gives $(1, 1, 1, 0, 0)$, of weight $17 > 15$: **infeasible**. Rounding down gives $(1, 1, 0, 0, 0)$, feasible (by the observation on rounding down of Chapter 10 «MIP heuristics» of the module *Integer Linear Programming*), of profit 25.

3. **Step 1.** Fix $x_3 := \left\lfloor \frac{4}{5} \right\rfloor = 1$. Item 3 uses 10; the LP takes item 1 (residual 3) and the fraction $\frac{3}{5}$ of item 2: $\tilde{x} = \left(1, \frac{3}{5}, 1, 0, 0\right)$, of value $14 + 13 + \frac{36}{5} = \frac{171}{5} = 34.2$.

   **Step 2.** Fix $x_2 := \left\lfloor \frac{3}{5} \right\rfloor = 1$. Items $2, 3$ fill the knapsack ($5 + 10 = 15$): the LP solution is $(0, 1, 1, 0, 0)$, integer, of value 26. The dive returns $S = \{2, 3\}$, of profit 26.

4. **Step 1'.** Fix $x_3 := 0$. The LP takes items $1, 2, 4$ (weight 13) and the fraction $\frac{2}{7}$ of item 5: value $13 + 12 + 6 + \frac{10}{7} = \frac{227}{7} \approx 32.43$.

   **Step 2'.** Fix $x_5 := \left\lfloor \frac{2}{7} \right\rfloor = 0$: the LP solution $(1, 1, 0, 1, 0)$ is integer, of profit 31.

   The four leaves of the tree are:

| branch | LP value | |
|---|---|---|
| $x_3 = 1$, $x_2 = 1$ | 26 | integer |
| $x_3 = 1$, $x_2 = 0$ | 30 | items $3, 1$ and half of item 4; $30 < 31$ |
| $x_3 = 0$, $x_5 = 0$ | 31 | integer |
| $x_3 = 0$, $x_5 = 1$ | 31 | not better than 31 |

   No branch can contain a solution of profit larger than 31, so $\{1, 2, 4\}$ is **optimal**: $z(I) = 31$. The first dive (26) failed because its first decision, the most natural one for the LP ($x_3 = \frac{4}{5} \to 1$), was wrong.

## 5.2 RINS, local branching and proximity search

**Exercise 5.3 — RINS, local branching and proximity search on the KP (KP3)**

We consider a knapsack of capacity $c = 22$ and a set $\mathscr{I} = \{1, 2, \ldots, 6\}$ of 6 items, sorted by non-increasing efficiency, with the following profits and weights:

| | $j=1$ | $j=2$ | $j=3$ | $j=4$ | $j=5$ | $j=6$ |
|---|---|---|---|---|---|---|
| $p_j$ | 18 | 21 | 4 | 19 | 10 | 7 |
| $w_j$ | 8 | 10 | 2 | 12 | 7 | 10 |

The incumbent is the solution of the greedy by *profit* $GR_p$.

**Questions**

1. Compute the incumbent $\bar{x}$ and its profit, and the optimal solution $\tilde{x}$ of the LP relaxation $\mathrm{LP_{KP}}$ with its value.
2. Apply RINS: list the variables fixed by RINS, write the sub-MIP with the cutoff constraint $\sum_j p_j x_j \geq f(\bar{x}) + 1$, and solve it by enumeration.
3. Show that local branching with radius $k = 2$ around $\bar{x}$ finds no better solution.
4. Apply proximity search with $\theta = 1$ from $\bar{x}$ (the optimal solution of this instance has profit 43, as can be checked by enumerating the $2^6$ subsets). Compare the three methods.

**Solution**

1. By non-increasing profit the order is 2 (21), 4 (19), 1 (18), 5, 6, 3. Item 2 is taken ($rc = 12$), item 4 is taken ($rc = 0$), nothing else fits: $\bar{x} = (0, 1, 0, 1, 0, 0)$, profit 40. The LP takes items $1, 2, 3$ (weight 20) and the fraction $\frac{2}{12} = \frac{1}{6}$ of item 4:

$$\tilde{x} = \left(1, 1, 1, \frac{1}{6}, 0, 0\right), \qquad z(\mathrm{LP_{KP}}) = 43 + \frac{19}{6} = \frac{277}{6} \approx 46.17.$$

2. $\tilde{x}$ and $\bar{x}$ agree on $x_2 = 1$, $x_5 = 0$, $x_6 = 0$, which are fixed; $x_1, x_3, x_4$ are free. The sub-MIP is

$$\begin{array}{rrrrrrr} \max & 21 & +\,18\,x_1 & +\,4\,x_3 & +\,19\,x_4 & & \\ \text{subject to} & & 8\,x_1 & +\,2\,x_3 & +\,12\,x_4 & \leq & 12 \\ & 21 & +\,18\,x_1 & +\,4\,x_3 & +\,19\,x_4 & \geq & 41 \\ & & & & x_1,\ x_3,\ x_4 & \in & \{0,1\} \end{array}$$

   If $x_4 = 1$ the capacity forces $x_1 = x_3 = 0$ and the profit is $40 < 41$. If $x_4 = 0$, the best choice is $x_1 = x_3 = 1$ (weight $10 \leq 12$), of profit 43. RINS returns $\{1, 2, 3\}$, of profit 43: the **optimum**.

3. The incumbent fills the knapsack ($10 + 12 = 22$), so adding an item is infeasible, and removing items decreases the profit. The solutions at distance 2 that keep the weight feasible are the swaps (one item out, one in):

| out \ in | 1 | 3 | 5 | 6 |
|---|---|---|---|---|
| 2 | 37 | 23 | 29 | 26 |
| 4 | 39 | 25 | 31 | 28 |

   all with profit smaller than 41: the sub-MIP with $\Delta(x, \bar{x}) \leq 2$ and the cutoff is infeasible. The optimum $(1, 1, 1, 0, 0, 0)$ differs from $\bar{x}$ in $x_1$, $x_3$, $x_4$: it is at distance 3.

4. **Iteration 1.** No solution of profit at least 41 is at distance $\leq 2$ (question 3). At distance 3 there are two: $(1,0,1,1,0,0)$, of weight 22 and profit 41, and $(1,1,1,0,0,0)$, of profit 43. The proximity sub-MIP may return either.

   - If it returns $(1,1,1,0,0,0)$, the next sub-MIP asks for profit at least 44 and is infeasible: the method stops, and 43 is proved optimal.
   - If it returns $(1,0,1,1,0,0)$, the next sub-MIP asks for profit at least 42: the closest solution is $(1,1,1,0,0,0)$, at distance 2 (item 4 out, item 2 in); then profit 44 is infeasible.

   RINS (hard fixing) succeeds because the variables on which $\tilde{x}$ and $\bar{x}$ agree keep their optimal values; local branching with $k=2$ fails because the optimum is three changes away; proximity search finds it, since the size of the move is decided by the sub-MIP.

## Exercise 5.4 — RINS, local branching and proximity search on the KP (KP4)

We consider a knapsack of capacity $c = 13$ and a set $\mathscr{I} = \{1, 2, \ldots, 6\}$ of 6 items, sorted by non-increasing efficiency, with the following profits and weights:

| | $j=1$ | $j=2$ | $j=3$ | $j=4$ | $j=5$ | $j=6$ |
|---|---|---|---|---|---|---|
| $p_j$ | 16 | 13 | 8 | 16 | 10 | 9 |
| $w_j$ | 2 | 2 | 4 | 9 | 6 | 8 |

The incumbent is the solution of the greedy algorithm $GR$ (non-increasing efficiency).

**Questions**

1. Compute the incumbent $\bar{x}$ and the optimal solution $\tilde{x}$ of the LP relaxation.
2. Apply RINS from $\bar{x}$: which variables are fixed, and what does the sub-MIP return?
3. Apply local branching from $\bar{x}$ with $k=1$ and then, after adding the reverse constraint, with $k=2$. Write the local branching constraints explicitly.
4. Prove that the solution found in question 3 is optimal, using the LP relaxation.

## Solution

1. $GR$ takes items $1,2,3$ (weight 8, $rc=5$); items $4,5,6$ do not fit: $\bar{x} = (1,1,1,0,0,0)$, profit 37. The LP takes items $1,2,3$ and the fraction $\frac{5}{9}$ of item 4:

$$\tilde{x} = \left(1,1,1,\frac{5}{9},0,0\right), \qquad z(\mathrm{LP_{KP}}) = 37 + \frac{80}{9} = \frac{413}{9} \approx 45.89.$$

2. They agree on $x_1 = x_2 = x_3 = 1$ and $x_5 = x_6 = 0$: only $x_4$ is free. The sub-MIP asks $8 + 9x_4 \leq 13$ and $37 + 16x_4 \geq 38$: the first forces $x_4 = 0$, the second $x_4 = 1$. It is infeasible: **RINS fails** and returns $\bar{x}$.

3. $k = 1$. The constraint is

$$(1 - x_1) + (1 - x_2) + (1 - x_3) + x_4 + x_5 + x_6 \leq 1.$$

The neighbours are the flips: adding item 4, 5 or 6 gives weight 17, 14, $16 > 13$; removing an item decreases the profit. No solution of profit at least 38: add the reverse constraint $(1 - x_1) + (1 - x_2) + (1 - x_3) + x_4 + x_5 + x_6 \geq 2$ and set $k = 2$.

$k = 2$. Now $2 \leq \Delta(x, \bar{x}) \leq 2$. Two removals lose profit, two additions exceed the capacity; the swaps (one of $1, 2, 3$ out, one of $4, 5, 6$ in) that fit are

| out \ in | 4 | 5 | 6 |
|---|---|---|---|
| 1 | — (weight 15) | 31 | — (weight 14) |
| 2 | — (weight 15) | 34 | — (weight 14) |
| 3 | 45 | 39 | 38 |

The best is to swap item 3 with item 4: $(1, 1, 0, 1, 0, 0)$, of weight 13 and profit 45.

4. The LP relaxation gives $z(I) \leq \frac{413}{9} \approx 45.89$, and since the profits are integer, $z(I) \leq 45$. The solution of local branching attains this bound: it is **optimal**. RINS fixed $x_3 = 1$, the variable that had to change; local branching (soft fixing) lets the solver choose which variables change.

## Exercise 5.5 — Proximity search on $P\,||\,C_{\max}$ (PCmax1)

We consider an instance of $P\,||\,C_{\max}$ with $m = 2$ machines and 6 jobs with processing times

| | $j=1$ | $j=2$ | $j=3$ | $j=4$ | $j=5$ | $j=6$ |
|---|---|---|---|---|---|---|
| $t_j$ | 9 | 3 | 9 | 5 | 8 | 2 |

and the MIP model with variables $x_{jh}$ and $C_{\max}$. The incumbent is the solution of list scheduling ($LIST$) in the order $1, 2, \ldots, 6$, each job to the least loaded machine (machine 1 in case of ties).

**Questions**

1. Compute the incumbent and its makespan, and the lower bound $\max\{\max_j t_j, \lceil \sum_j t_j / m \rceil\}$.
2. Apply the first iteration of proximity search with $\theta = 1$: find the solution with $C_{\max} \leq f(\bar{x}) - 1$ at minimum distance $\Delta(x, \bar{x})$ from the incumbent. (Recall that $\Delta = 2r$, where $r$ is the number of jobs that change machine.)
3. Apply the second iteration.
4. Apply the third iteration, and conclude.

**Solution**

1. Job 1 $\to M_1$ $(9,0)$, job 2 $\to M_2$ $(9,3)$, job 3 $\to M_2$ $(9,12)$, job 4 $\to M_1$ $(14,12)$, job 5 $\to M_2$ $(14,20)$, job 6 $\to M_1$ $(16,20)$. The incumbent is $J_1 = \{1,4,6\}$, $J_2 = \{2,3,5\}$, with loads 16 and 20: makespan 20. The lower bound is $\max\{9, \lceil 36/2 \rceil\} = 18$.

2. The cutoff is $C_{\max} \leq 19$. With $r = 1$ (distance 2) a single job moves; moving a job to $M_2$ makes its load larger than 20, so the job moves from $M_2$ to $M_1$, and needs $20 - t_j \leq 19$ and $16 + t_j \leq 19$, i.e., $1 \leq t_j \leq 3$: only job 2. The new incumbent is $J_1 = \{1,2,4,6\}$, $J_2 = \{3,5\}$, with loads 19, 17: makespan 19, at distance 2.

3. The cutoff is $C_{\max} \leq 18$: since the total is 36, both loads must be exactly 18, i.e., $M_1$ must lose exactly 1. A single move would need a job with $t_j = 1$: none (distance 2 is impossible). With two jobs ($r = 2$, distance 4): two jobs leaving $M_1$ would need $t_i + t_j = 1$, impossible; a swap of $i \in J_1$ with $j \in J_2$ needs $t_i - t_j = 1$, and the only pair is job 1 (9) with job 5 (8). The new incumbent is $J_1 = \{2,4,5,6\}$, $J_2 = \{1,3\}$, with loads 18, 18: makespan 18, at distance 4.

4. The cutoff is $C_{\max} \leq 17$, but $C_{\max} \geq 36/2 = 18$ for every assignment: the proximity sub-MIP is infeasible, and the method stops. By the proposition on proximity search of Chapter 10 «MIP heuristics» of the module *Integer Linear Programming*, makespan 18 is **optimal**; it also attains the lower bound of question 1.