Implementation exercises for the course Heuristic Optimization

Dr. Manuel López-Ibáñez
manuel.lopez-ibanez@ulb.ac.be

IRIDIA, CoDE, ULB

February 28, 2011

Exercise 1.1: Iterative Improvement for the SMTWTP

Implement iterative local search algorithms for the SMTWTP

1. The Single Machine Total Weighted Tardiness Problem
2. First-improvement and Best-Improvement Iterative Local Search
3. Transpose, exchange and insert neighborhoods
4. Random initialization vs. Earliest due date heuristic
5. Statistical Empirical Analysis

The Single Machine Total Weighted Tardiness Problem

Given

- Single machine, continuously available
- n jobs, for each job j is given its processing time p_j, its due date d_j and its importance w_j

The Single Machine Total Weighted Tardiness Problem

Given

- Single machine, continuously available
- n jobs, for each job j is given its processing time p_j, its due date d_j and its importance w_j

The Single Machine Total Weighted Tardiness Problem (SMTWTP)

- Completion time of job j: C_j (when it finished)
- Tardiness: $T_j = \max\{C_j - d_j, 0\}$ (how late after it should)
- Find a permutation π of the jobs that minimizes the sum of the weighted tardiness:

$$\min_{\pi \in \Phi^n} F(\pi) = \sum_{i=1}^{n} w_{\pi_i} \cdot T_{\pi_i}$$

Exercise 1.1: Iterative Improvement for the SMTWTP

Implement 12 iterative improvements algorithms for the SMTWTP

Exercise 1.1: Iterative Improvement for the SMTWTP

Implement 12 iterative improvements algorithms for the SMTWTP

- Pivoting rule:
  1. first-improvement
  2. best-improvement
- Neighborhood:
  1. Transpose
  2. Exchange
  3. Insert
- Initial solution:
  1. Random permutation
  2. Earliest due date heuristic

Exercise 1.1: Iterative Improvement for the SMTWTP

Implement 12 iterative improvements algorithms for the SMTWTP

- Pivoting rule:
  1. first-improvement
  2. best-improvement
- Neighborhood:
  1. Transpose
  2. Exchange
  3. Insert
- Initial solution:
  1. Random permutation
  2. Earliest due date heuristic

2 pivoting rules $\times$ 3 neighborhoods $\times$ 2 initialization methods $=$
12 combinations

Exercise 1.1: Iterative Improvement for the SMTWTP

Implement 12 iterative improvements algorithms for the SMTWTP

Don't implement 12 programs!
Reuse code and use command-line parameters

```
smptwtp-ii --first --transpose --earliest-due-date
smptwtp-ii --best --exchange --random-init
...
```

Exercise 1.1: Iterative Improvement for the SMTWTP

Iterative Improvement

$\pi :=$ `GenerateInitialSolution()`
while π is not a local optimum **do**
  choose a neighbour $\pi' \in \mathcal{N}(\pi)$ such that $F(\pi') < F(\pi)$
  $\pi := \pi'$

Exercise 1.1: Iterative Improvement for the SMTWTP

Iterative Improvement

$\pi :=$ `GenerateInitialSolution()`
while π is not a local optimum **do**
  choose a neighbour $\pi' \in \mathcal{N}(\pi)$ such that $F(\pi') < F(\pi)$
  $\pi := \pi'$

Which neighbour to choose? Pivoting rule

- **Best Improvement:** choose best from all neighbours of s
- **First improvement:** evaluate neighbours in fixed order and choose first improving neighbour.

Exercise 1.1: Iterative Improvement for the SMTWTP

Iterative Improvement

$\pi :=$ `GenerateInitialSolution()`
while π is not a local optimum **do**
  choose a neighbour $\pi' \in \mathcal{N}(\pi)$ such that $F(\pi') < F(\pi)$
  $\pi := \pi'$

Which neighbour to choose? Pivoting rule

- **Best Improvement:** choose best from all neighbours of s
  - ✔ Better quality
  - ✘ Requires evaluation of all neighbours in each step
- **First improvement:** evaluate neighbours in fixed order and choose first improving neighbour.

Exercise 1.1: Iterative Improvement for the SMTWTP

Iterative Improvement

$\pi := \texttt{GenerateInitialSolution()}$
while π is not a local optimum **do**
  choose a neighbour $\pi' \in \mathcal{N}(\pi)$ such that $F(\pi') < F(\pi)$
  $\pi := \pi'$

Which neighbour to choose? Pivoting rule

- **Best Improvement:** choose best from all neighbours of s
  - ✔ Better quality
  - ✘ Requires evaluation of all neighbours in each step
- **First improvement:** evaluate neighbours in fixed order and choose first improving neighbour.
  - ✔ More efficient
  - ✘ Order of evaluation may impact quality / performance

Exercise 1.1: Iterative Improvement for the SMTWTP

Iterative Improvement

$\pi :=$ `GenerateInitialSolution()`
while π is not a local optimum **do**
  choose a neighbour $\pi' \in \mathcal{N}(\pi)$ such that $F(\pi') < F(\pi)$
  $\pi := \pi'$

Initial solution

- Random permutation
- Earliest due date heuristic (sort jobs by increasing due date)

Exercise 1.1: Iterative Improvement for the SMTWTP

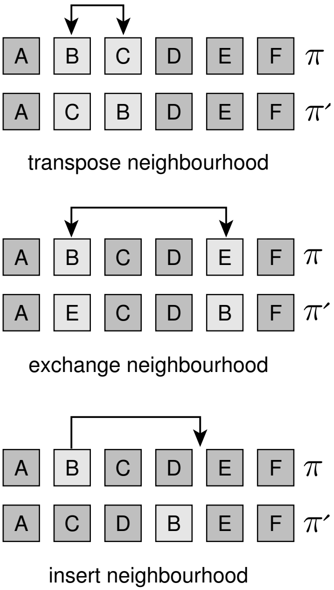
Iterative Improvement

$\pi :=$ `GenerateInitialSolution()`
while π is not a local optimum **do**
  choose a neighbour $\pi' \in \mathcal{N}(\pi)$ such that $F(\pi') < F(\pi)$
  $\pi := \pi'$

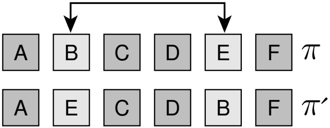
Which neighborhood $\mathcal{N}(\pi)$?

- Transpose
- Exchange
- Insertion

Exercise 1.1: Iterative Improvement for the SMTWTP



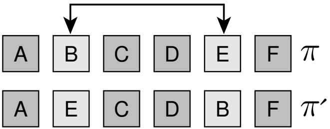
Exercise 1.1: Iterative Improvement for the SMTWTP



exchange neighbourhood

Example: Exchange π_i and π_j ($i < j$), $\pi' = \mathtt{Exchange}(\pi, i, j)$

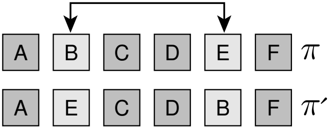
Exercise 1.1: Iterative Improvement for the SMTWTP



exchange neighbourhood

Example: Exchange π_i and π_j ($i < j$), $\pi' = \mathtt{Exchange}(\pi, i, j)$

- $k < i$: Same C_{π_k}, same T_{π_k}

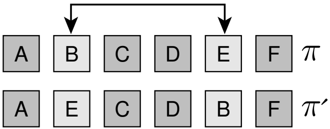


exchange neighbourhood

Example: Exchange π_i and π_j ($i < j$), $\pi' = \texttt{Exchange}(\pi, i, j)$

- $k < i$: Same C_{π_k}, same T_{π_k}
- $i \leq k < j$: different order, everything changes!

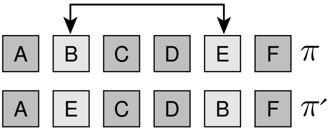
Exercise 1.1: Iterative Improvement for the SMTWTP



exchange neighbourhood

Example: Exchange π_i and π_j ($i < j$), $\pi' = \mathtt{Exchange}(\pi, i, j)$

- $k < i$: Same C_{π_k}, same T_{π_k}
- $i \leq k < j$: different order, everything changes!
- $C_{\pi'_j} = C_{\pi_j}$ but $d_{\pi'_j} \neq d_{\pi_j}$

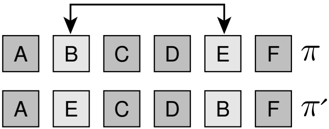


exchange neighbourhood

Example: Exchange π_i and π_j ($i < j$), $\pi' = \mathtt{Exchange}(\pi, i, j)$

- $k < i$: Same C_{π_k}, same T_{π_k}
- $i \leq k < j$: different order, everything changes!
- $C_{\pi'_j} = C_{\pi_j}$ but $d_{\pi'_j} \neq d_{\pi_j}$
- $k > j$: Same C_{π_k}, same T_{π_k}

Exercise 1.1: Iterative Improvement for the SMTWTP



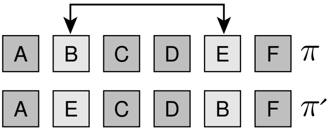
exchange neighbourhood

Example: Exchange π_i and π_j ($i < j$), $\pi' = \texttt{Exchange}(\pi, i, j)$

- $k < i$: Same C_{π_k}, same T_{π_k}
- $i \leq k < j$: different order, everything changes!
- $C_{\pi'_j} = C_{\pi_j}$ but $d_{\pi'_j} \neq d_{\pi_j}$
- $k > j$: Same C_{π_k}, same T_{π_k}

Only jobs between i and j are affected!

Exercise 1.1: Iterative Improvement for the SMTWTP



exchange neighbourhood

Example: Exchange π_i and π_j ($i < j$), $\pi' = \texttt{Exchange}(\pi, i, j)$

- $k < i$: Same C_{π_k}, same T_{π_k}
- $i \leq k < j$: different order, everything changes!
- $C_{\pi'_j} = C_{\pi_j}$ but $d_{\pi'_j} \neq d_{\pi_j}$
- $k > j$: Same C_{π_k}, same T_{π_k}

Only jobs between i and j are affected!
Equivalent speed-ups with Transpose and Insertion

Exercise 1.1: Iterative Improvement for the SMTWTP

Instances

- 125 SMTWTP instances in a single file: `wt100.txt`
- More info: http://iridia.ulb.ac.be/~stuetzle/Teaching/HO/

Exercise 1.1: Iterative Improvement for the SMTWTP

Instances

- 125 SMTWTP instances in a single file: `wt100.txt`
- More info: http://iridia.ulb.ac.be/~stuetzle/Teaching/HO/

Experiments

Apply each algorithm k once to each instance i and compute:

1. Relative percentage deviation $\Delta_{ki} = 100 \cdot \dfrac{\text{cost}_{ki} - \text{best-known}_i}{\text{best-known}_i}$
2. Computation time (t_{ki})

Exercise 1.1: Iterative Improvement for the SMTWTP

Instances

- 125 SMTWTP instances in a single file: `wt100.txt`
- More info: http://iridia.ulb.ac.be/~stuetzle/Teaching/HO/

Experiments

Apply each algorithm *k* once to each instance *i* and compute:

1. Relative percentage deviation $\Delta_{ki} = 100 \cdot \dfrac{\text{cost}_{ki} - \text{best-known}_i}{\text{best-known}_i}$
2. Computation time (t_{ki})

Report for each algorithm *k*

- Average relative percentage deviation
- Sum of computation time

Exercise 1.1: Iterative Improvement for the SMTWTP

Is there a statistically significant difference between the solution quality generated by the different algorithms?

Statistical test

- Paired t-test
- Wilcoxon signed-rank test

Exercise 1.1: Iterative Improvement for the SMTWTP

Is there a statistically significant difference between the solution quality generated by the different algorithms?

Background: Statistical hypothesis tests (1)

- *Statistical hypothesis tests* are used to assess the validity of statements about properties of or relations between sets of statistical data.

Is there a statistically significant difference between the solution quality generated by the different algorithms?

Background: Statistical hypothesis tests (1)

- *Statistical hypothesis tests* are used to assess the validity of statements about properties of or relations between sets of statistical data.
- The statement to be tested (or its negation) is called the *null hypothesis* (H_0) of the test.

Exercise 1.1: Iterative Improvement for the SMTWTP

Is there a statistically significant difference between the solution quality generated by the different algorithms?

Background: Statistical hypothesis tests (1)

- *Statistical hypothesis tests* are used to assess the validity of statements about properties of or relations between sets of statistical data.
- The statement to be tested (or its negation) is called the *null hypothesis* (H_0) of the test.
  Example: For the Wilcoxon signed-rank test, the null hypothesis is that 'the median of the differences is zero'.

Is there a statistically significant difference between the solution quality generated by the different algorithms?

Background: Statistical hypothesis tests (1)

- *Statistical hypothesis tests* are used to assess the validity of statements about properties of or relations between sets of statistical data.
- The statement to be tested (or its negation) is called the *null hypothesis* (H_0) of the test.
  Example: For the Wilcoxon signed-rank test, the null hypothesis is that ‘the median of the differences is zero’.
- The *significance level* (α) determines the maximum allowable probability of incorrectly rejecting the null hypothesis.

Is there a statistically significant difference between the solution quality generated by the different algorithms?

Background: Statistical hypothesis tests (1)

- *Statistical hypothesis tests* are used to assess the validity of statements about properties of or relations between sets of statistical data.
- The statement to be tested (or its negation) is called the *null hypothesis* (H_0) of the test.
  Example: For the Wilcoxon signed-rank test, the null hypothesis is that 'the median of the differences is zero'.
- The *significance level* (α) determines the maximum allowable probability of incorrectly rejecting the null hypothesis.
  Typical values of α are 0.05 or 0.01.

Exercise 1.1: Iterative Improvement for the SMTWTP

Is there a statistically significant difference between the solution quality generated by the different algorithms?

Background: Statistical hypothesis tests (2)

- The application of a test to a given data set results in a *p-value*, which represents the probability that the null hypothesis is incorrectly rejected.

Is there a statistically significant difference between the solution quality generated by the different algorithms?

Background: Statistical hypothesis tests (2)

- The application of a test to a given data set results in a *p-value*, which represents the probability that the null hypothesis is incorrectly rejected.
- The null hypothesis is rejected iff this p-value is smaller than the previously chosen significance level.

Is there a statistically significant difference between the solution quality generated by the different algorithms?

Background: Statistical hypothesis tests (2)

- The application of a test to a given data set results in a *p-value*, which represents the probability that the null hypothesis is incorrectly rejected.
- The null hypothesis is rejected iff this p-value is smaller than the previously chosen significance level.
- Most common statistical hypothesis tests are already implemented in statistical software such as the *R software environment* (http://www.r-project.org/).

Is there a statistically significant difference between the solution quality generated by the different algorithms?

Example in R

```
best.known <- read.table ("best-known.dat")
```

Exercise 1.1: Iterative Improvement for the SMTWTP

Is there a statistically significant difference between the solution quality generated by the different algorithms?

Example in R

```
best.known <- read.table ("best-known.dat")
a.cost <- read.table("ii-best-ex-rand.dat")$V1
```

Is there a statistically significant difference between the solution quality generated by the different algorithms?

Example in R

```
best.known <- read.table ("best-known.dat")
a.cost <- read.table("ii-best-ex-rand.dat")$V1
a.cost <- 100 * (a.cost - best.known) / best.known
```

Is there a statistically significant difference between the solution quality generated by the different algorithms?

Example in R

```
best.known <- read.table ("best-known.dat")
a.cost <- read.table("ii-best-ex-rand.dat")$V1
a.cost <- 100 * (a.cost - best.known) / best.known
b.cost <- read.table("ii-best-ins-rand.dat")$V1
b.cost <- 100 * (b.cost - best.known) / best.known
```

Is there a statistically significant difference between the solution quality generated by the different algorithms?

Example in R

```
best.known <- read.table ("best-known.dat")
a.cost <- read.table("ii-best-ex-rand.dat")$V1
a.cost <- 100 * (a.cost - best.known) / best.known
b.cost <- read.table("ii-best-ins-rand.dat")$V1
b.cost <- 100 * (b.cost - best.known) / best.known
t.test (a.cost, b.cost, paired=T)$p.value
[1] 0.8819112
```

Exercise 1.1: Iterative Improvement for the SMTWTP

Is there a statistically significant difference between the solution quality generated by the different algorithms?

Example in R

```
best.known <- read.table ("best-known.dat")
a.cost <- read.table("ii-best-ex-rand.dat")$V1
a.cost <- 100 * (a.cost - best.known) / best.known
b.cost <- read.table("ii-best-ins-rand.dat")$V1
b.cost <- 100 * (b.cost - best.known) / best.known
t.test (a.cost, b.cost, paired=T)$p.value
[1] 0.8819112
wilcox.test (a.cost, b.cost, paired=T)$p.value
[1] 0.0019212
```

Implement 8 VND algorithms for the SMTWTP

- Type:
  1. standard VND
  2. piped VND
- Pivoting rule: first-improvement
- Neighborhood order:
  1. transpose $\rightarrow$ exchange $\rightarrow$ insert
  2. transpose $\rightarrow$ insert $\rightarrow$ exchange
- Initial solution:
  1. Random permutation
  2. Earliest due date heuristic

Variable Neighbourhood Descent (VND)

k neighborhoods $\mathcal{N}_1, \ldots, \mathcal{N}_k$
$\pi := \texttt{GenerateInitialSolution()}$
$i := 1$
repeat
    choose the first improving neighbor $\pi' \in \mathcal{N}_i(\pi)$
    if $\nexists \pi'$ **then**
        $i := i + 1$
    else
        $\pi := \pi'$
        $i := 1$
until $i > k$

Exercise 1.2 VND algorithms for the SMTWTP

Variable Neighbourhood Descent (VND)

k neighborhoods $\mathcal{N}_1, \ldots, \mathcal{N}_k$
$\pi :=$ `GenerateInitialSolution()`
$i := 1$
repeat
  choose the first improving neighbor $\pi' \in \mathcal{N}_i(\pi)$
  if $\nexists \pi'$ **then**
    $i := i + 1$
  else
    $\pi := \pi'$
    $i := 1$
until $i > k$

Piped VND

- Simply chain different II algorithms one after the other
- Use final solution of one II as initial solution of the next

Exercise 1.2 VND algorithms for the SMTWTP

Implement 8 VND algorithms for the SMTWTP

- Instances: Same as 1.1
- Experiments: one run of each algorithm per instance
- Report: Same as 1.1
- Statistical tests: Same as 1.1