Implementation exercises for the course Heuristic Optimization

Dr. Franco Mascia[1]
fmascia@ulb.ac.be

IRIDIA, CoDE, ULB

February 26, 2014

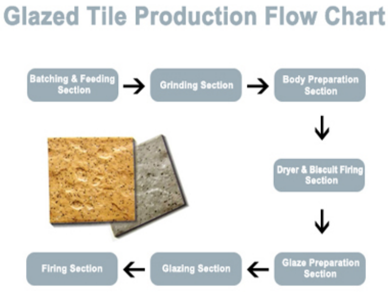
[1] Slides based on last year's excersises by Dr. Manuel López-Ibáñez.

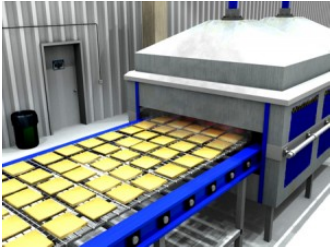
Exercise 1.1: Iterative Improvement for the PFSP

Implement perturbative local search algorithms for the PFSP

1. Permutation Flow Shop Scheduling Problem (PFSP)
2. First-improvement and Best-Improvement
3. Transpose, exchange and insert neighborhoods
4. Random initialization vs. SLACK heuristic
5. Statistical Empirical Analysis

The Permutation Flow Shop Scheduling Problem (1/4)

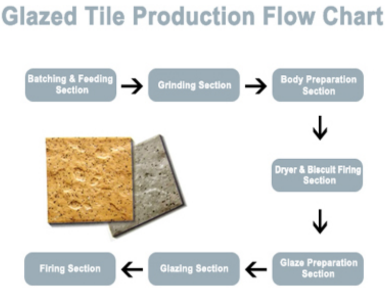


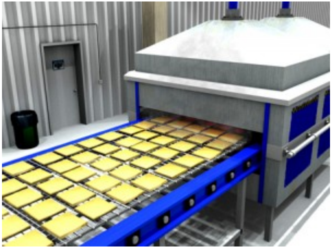


Example in ceramic tile production

- Tiles need several processing steps with different machines
- Tiles of different type require specific processing times for each machine
- Goal: find a schedule of the jobs that minimizes an objective function (makespan or total completion time)

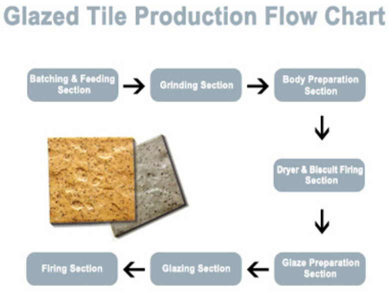
The Permutation Flow Shop Scheduling Problem (1/4)

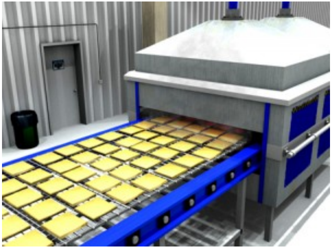




Example in ceramic tile production

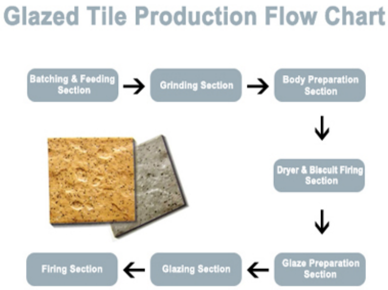
- Tiles need several processing steps with different machines
- Tiles of different type require specific processing times for each machine
- Goal: find a schedule of the jobs that minimizes an objective function (makespan or total completion time)

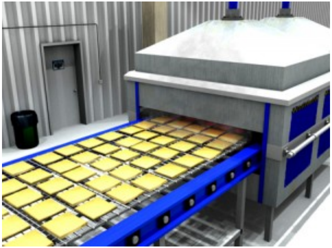




Example in ceramic tile production

- Tiles need several processing steps with different machines
- Tiles of different type require specific processing times for each machine
- Goal: find a schedule of the jobs that minimizes an objective function (makespan or total completion time)





Example in ceramic tile production

- Tiles need several processing steps with different machines
- Tiles of different type require specific processing times for each machine
- Goal: find a schedule of the jobs that minimizes an objective function (makespan or total completion time)

The Permutation Flow Shop Scheduling Problem (2/4)

Flow Shop Scheduling

- Several scheduling problems have been proposed with different formulations and constraints.
- In permutation flow shop problems:
  - jobs composed by operations to be executed on several machines
  - all jobs pass through the machines in the same order
  - all jobs available at time zero
  - pre-emption not allowed
  - each operation has to be performed on a specific machine
  - each job at most on one machine at a time
  - each machine at most one job at a time

The Permutation Flow Shop Scheduling Problem (PFSP)

- Jobs pass trough all machines in the same order (FCFS queues)
- No constraints: infinite buffers between machines, no blocking, no no-wait requirements (steel production)

Flow Shop Scheduling

- Several scheduling problems have been proposed with different formulations and constraints.
- In permutation flow shop problems:
  - jobs composed by operations to be executed on several machines
  - all jobs pass through the machines in the same order
  - all jobs available at time zero
  - pre-emption not allowed
  - each operation has to be performed on a specific machine
  - each job at most on one machine at a time
  - each machine at most one job at a time

The Permutation Flow Shop Scheduling Problem (PFSP)

- Jobs pass trough all machines in the same order (FCFS queues)
- No constraints: infinite buffers between machines, no blocking, no no-wait requirements (steel production)

Flow Shop Scheduling

- Several scheduling problems have been proposed with different formulations and constraints.
- In permutation flow shop problems:
  - jobs composed by operations to be executed on several machines
  - all jobs pass through the machines in the same order
  - all jobs available at time zero
  - pre-emption not allowed
  - each operation has to be performed on a specific machine
  - each job at most on one machine at a time
  - each machine at most one job at a time

The Permutation Flow Shop Scheduling Problem (PFSP)

- Jobs pass trough all machines in the same order (FCFS queues)
- No constraints: infinite buffers between machines, no blocking, no no-wait requirements (steel production)

The Permutation Flow Shop Scheduling Problem (2/4)

Flow Shop Scheduling

- Several scheduling problems have been proposed with different formulations and constraints.
- In permutation flow shop problems:
  - jobs composed by operations to be executed on several machines
  - all jobs pass through the machines in the same order
  - all jobs available at time zero
  - pre-emption not allowed
  - each operation has to be performed on a specific machine
  - each job at most on one machine at a time
  - each machine at most one job at a time

The Permutation Flow Shop Scheduling Problem (PFSP)

- Jobs pass trough all machines in the same order (FCFS queues)
- No constraints: infinite buffers between machines, no blocking, no no-wait requirements (steel production)

The Permutation Flow Shop Scheduling Problem (2/4)

Flow Shop Scheduling

- Several scheduling problems have been proposed with different formulations and constraints.
- In permutation flow shop problems:
  - jobs composed by operations to be executed on several machines
  - all jobs pass through the machines in the same order
  - all jobs available at time zero
  - pre-emption not allowed
  - each operation has to be performed on a specific machine
  - each job at most on one machine at a time
  - each machine at most one job at a time

The Permutation Flow Shop Scheduling Problem (PFSP)

- Jobs pass trough all machines in the same order (FCFS queues)
- No constraints: infinite buffers between machines, no blocking, no no-wait requirements (steel production)

The Permutation Flow Shop Scheduling Problem (2/4)

Flow Shop Scheduling

- Several scheduling problems have been proposed with different formulations and constraints.
- In permutation flow shop problems:
  - jobs composed by operations to be executed on several machines
  - all jobs pass through the machines in the same order
  - all jobs available at time zero
  - pre-emption not allowed
  - each operation has to be performed on a specific machine
  - each job at most on one machine at a time
  - each machine at most one job at a time

The Permutation Flow Shop Scheduling Problem (PFSP)

- Jobs pass trough all machines in the same order (FCFS queues)
- No constraints: infinite buffers between machines, no blocking, no no-wait requirements (steel production)

Flow Shop Scheduling

- Several scheduling problems have been proposed with different formulations and constraints.
- In permutation flow shop problems:
  - jobs composed by operations to be executed on several machines
  - all jobs pass through the machines in the same order
  - all jobs available at time zero
  - pre-emption not allowed
  - each operation has to be performed on a specific machine
  - each job at most on one machine at a time
  - each machine at most one job at a time

The Permutation Flow Shop Scheduling Problem (PFSP)

- Jobs pass trough all machines in the same order (FCFS queues)
- No constraints: infinite buffers between machines, no blocking, no no-wait requirements (steel production)

The Permutation Flow Shop Scheduling Problem (2/4)

Flow Shop Scheduling

- Several scheduling problems have been proposed with different formulations and constraints.
- In permutation flow shop problems:
  - jobs composed by operations to be executed on several machines
  - all jobs pass through the machines in the same order
  - all jobs available at time zero
  - pre-emption not allowed
  - each operation has to be performed on a specific machine
  - each job at most on one machine at a time
  - each machine at most one job at a time

The Permutation Flow Shop Scheduling Problem (PFSP)

- Jobs pass trough all machines in the same order (FCFS queues)
- No constraints: infinite buffers between machines, no blocking, no no-wait requirements (steel production)

The Permutation Flow Shop Scheduling Problem (2/4)

Flow Shop Scheduling

- Several scheduling problems have been proposed with different formulations and constraints.
- In permutation flow shop problems:
  - jobs composed by operations to be executed on several machines
  - all jobs pass through the machines in the same order
  - all jobs available at time zero
  - pre-emption not allowed
  - each operation has to be performed on a specific machine
  - each job at most on one machine at a time
  - each machine at most one job at a time

The Permutation Flow Shop Scheduling Problem (PFSP)

- Jobs pass trough all machines in the same order (FCFS queues)
- No constraints: infinite buffers between machines, no blocking, no no-wait requirements (steel production)

Given

A set of n jobs $J_1, \ldots, J_n$ jobs, where each job J_i consists of m operations $o_{i1}, \ldots, o_{im}$ performed on $M_1, \ldots, M_m$ machines in that order, with processing time p_{ij} for operation o_{ij}.

Due dates

Each job J_i has a *due date* d_i and a priority w_i. Let C_{ij} be the completion time of job J_i on machine M_j, and C_i the completion time of job J_i on the last machine. The tardiness of such a job J_i is $T_i = \max\{C_i - d_i, 0\}$.

Objective

Find a permutation π that minimizes the sum of the total weighted tardiness $\sum_{i=1}^{n} w_i \cdot T_i$.

Given

A set of n jobs $J_1, \ldots, J_n$ jobs, where each job J_i consists of m operations $o_{i1}, \ldots, o_{im}$ performed on $M_1, \ldots, M_m$ machines in that order, with processing time p_{ij} for operation o_{ij}.

Due dates

Each job J_i has a *due date* d_i and a priority w_i. Let C_{ij} be the completion time of job J_i on machine M_j, and C_i the completion time of job J_i on the last machine. The tardiness of such a job J_i is $T_i = \max\{C_i - d_i, 0\}$.

Objective

Find a permutation π that minimizes the sum of the total weighted tardiness $\sum_{i=1}^{n} w_i \cdot T_i$.

The Permutation Flow Shop Scheduling Problem (3/4)

Given

A set of n jobs $J_1, \ldots, J_n$ jobs, where each job J_i consists of m operations $o_{i1}, \ldots, o_{im}$ performed on $M_1, \ldots, M_m$ machines in that order, with processing time p_{ij} for operation o_{ij}.

Due dates

Each job J_i has a *due date* d_i and a priority w_i. Let C_{ij} be the completion time of job J_i on machine M_j, and C_i the completion time of job J_i on the last machine. The tardiness of such a job J_i is $T_i = \max\{C_i - d_i, 0\}$.

Objective

Find a permutation π that minimizes the sum of the total weighted tardiness $\sum_{i=1}^{n} w_i \cdot T_i$.

Given

A set of n jobs $J_1, \ldots, J_n$ jobs, where each job J_i consists of m operations $o_{i1}, \ldots, o_{im}$ performed on $M_1, \ldots, M_m$ machines in that order, with processing time p_{ij} for operation o_{ij}.

Due dates

Each job J_i has a *due date* d_i and a priority w_i. Let C_{ij} be the completion time of job J_i on machine M_j, and C_i the completion time of job J_i on the last machine. The tardiness of such a job J_i is $T_i = \max\{C_i - d_i, 0\}$.

Objective

Find a permutation π that minimizes the sum of the total weighted tardiness $\sum_{i=1}^{n} w_i \cdot T_i$.

The Permutation Flow Shop Scheduling Problem (4/4)

Computing completion times

$$C_{\pi(1)j} = \sum_{h=1}^{j} p_{\pi(1)h} \qquad j = 1, \ldots\ m$$

$$C_{\pi(k)1} = \sum_{h=1}^{k} p_{\pi(h)1} \qquad k = 1, \ldots\ n$$

$$C_{\pi(k)j} = max\{C_{\pi(k-1)j}, C_{\pi(k)(j-1)}\} + p_{\pi(k)j} \qquad k = 2, \ldots\ n \quad j = 2, \ldots\ m$$

Job	J_1	J_2	J_3	J_4	J_5
p_{i1}	3	3	4	2	3
p_{i2}	2	1	3	3	1
p_{i3}	4	2	1	2	3
d_i	11	9	10	11	30
w_i	1	2	4	2	3

	3	6	10	12	15
	5	7	13	16	17
	9	11	14	18	21
T_i	0	2	4	7	0

Makespan = 21

Sum of Completion times = 73

Total Weighted Tardiness = 34

Computing completion times

$$C_{\pi(1)j} = \sum_{h=1}^{j} p_{\pi(1)h} \qquad j = 1, \ldots\ m$$
$$C_{\pi(k)1} = \sum_{h=1}^{k} p_{\pi(h)1} \qquad k = 1, \ldots\ n$$
$$C_{\pi(k)j} = max\{C_{\pi(k-1)j}, C_{\pi(k)(j-1)}\} + p_{\pi(k)j} \qquad k = 2, \ldots\ n,\ j = 2, \ldots\ m$$

Job	J_1	J_2	J_3	J_4	J_5
p_{i1}	3	3	4	2	3
p_{i2}	2	1	3	3	1
p_{i3}	4	2	1	2	3
d_i	11	9	10	11	30
w_i	1	2	4	2	3

	3	6	10	12	15
	5	7	13	16	17
	9	11	14	18	21
T_i	0	2	4	7	0

Makespan = 21

Sum of Completion times = 73

Total Weighted Tardiness = 34

Computing completion times

$$C_{\pi(1)j} = \sum_{h=1}^{j} p_{\pi(1)h} \qquad j = 1, \ldots m$$
$$C_{\pi(k)1} = \sum_{h=1}^{k} p_{\pi(h)1} \qquad k = 1, \ldots n$$
$$C_{\pi(k)j} = max\{C_{\pi(k-1)j}, C_{\pi(k)(j-1)}\} + p_{\pi(k)j} \qquad k = 2, \ldots n \quad j = 2, \ldots m$$

Job	J_1	J_2	J_3	J_4	J_5
p_{i1}	3	3	4	2	3
p_{i2}	2	1	3	3	1
p_{i3}	4	2	1	2	3
d_i	11	9	10	11	30
w_i	1	2	4	2	3

	3	6	10	12	15
	5	7	13	16	17
	9	11	14	18	21
T_i	0	2	4	7	0

Makespan = 21

Sum of Completion times = 73

Total Weighted Tardiness = 34

The Permutation Flow Shop Scheduling Problem (4/4)

Computing completion times

$$
\begin{aligned}
C_{\pi(1)j} &= \textstyle\sum_{h=1}^{j} p_{\pi(1)h} && j = 1, \dots\ m \\
C_{\pi(k)1} &= \textstyle\sum_{h=1}^{k} p_{\pi(h)1} && k = 1, \dots\ n \\
C_{\pi(k)j} &= max\{C_{\pi(k-1)j}, C_{\pi(k)(j-1)}\} + p_{\pi(k)j} && k = 2, \dots\ n \\
& && j = 2, \dots\ m
\end{aligned}
$$

Job	J_1	J_2	J_3	J_4	J_5
p_{i1}	3	3	4	2	3
p_{i2}	2	1	3	3	1
p_{i3}	4	2	1	2	3
d_i	11	9	10	11	30
w_i	1	2	4	2	3

	3	6	10	12	15
	5	7	13	16	17
	9	11	14	18	21
T_i	0	2	4	7	0

Makespan = 21

Sum of Completion times = 73

Total Weighted Tardiness = 34

Computing completion times

$$C_{\pi(1)j} = \sum_{h=1}^{j} p_{\pi(1)h} \qquad j = 1, \ldots m$$
$$C_{\pi(k)1} = \sum_{h=1}^{k} p_{\pi(h)1} \qquad k = 1, \ldots n$$
$$C_{\pi(k)j} = max\{C_{\pi(k-1)j}, C_{\pi(k)(j-1)}\} + p_{\pi(k)j} \qquad k = 2, \ldots n \quad j = 2, \ldots m$$

Job	J_1	J_2	J_3	J_4	J_5
p_{i1}	3	3	4	2	3
p_{i2}	2	1	3	3	1
p_{i3}	4	2	1	2	3

d_i	11	9	10	11	30
w_i	1	2	4	2	3

	3	6	10	12	15
	5	7	13	16	17
	9	11	14	18	21
T_i	0	2	4	7	0

Makespan = 21

Sum of Completion times = 73

Total Weighted Tardiness = 34

Computing completion times

$$
\begin{aligned}
C_{\pi(1)j} &= \textstyle\sum_{h=1}^{j} p_{\pi(1)h} & j &= 1,\dots\ m\\
C_{\pi(k)1} &= \textstyle\sum_{h=1}^{k} p_{\pi(h)1} & k &= 1,\dots\ n\\
C_{\pi(k)j} &= max\{C_{\pi(k-1)j}, C_{\pi(k)(j-1)}\} + p_{\pi(k)j} & k &= 2,\dots\ n\\
& & j &= 2,\dots\ m
\end{aligned}
$$

Job	J_1	J_2	J_3	J_4	J_5
p_{i1}	3	3	4	2	3
p_{i2}	2	1	3	3	1
p_{i3}	4	2	1	2	3
d_i	11	9	10	11	30
w_i	1	2	4	2	3

	3	6	10	12	15
	5	7	13	16	17
	9	11	14	18	21
T_i	0	2	4	7	0

Makespan = 21

Sum of Completion times = 73

Total Weighted Tardiness = 34

Computing completion times

$$C_{\pi(1)j} = \sum_{h=1}^{j} p_{\pi(1)h} \qquad j = 1, \ldots\ m$$
$$C_{\pi(k)1} = \sum_{h=1}^{k} p_{\pi(h)1} \qquad k = 1, \ldots\ n$$
$$C_{\pi(k)j} = max\{C_{\pi(k-1)j}, C_{\pi(k)(j-1)}\} + p_{\pi(k)j} \qquad k = 2, \ldots\ n \quad j = 2, \ldots\ m$$

Job	J_1	J_2	J_3	J_4	J_5
p_{i1}	3	3	4	2	3
p_{i2}	2	1	3	3	1
p_{i3}	4	2	1	2	3
d_i	11	9	10	11	30
w_i	1	2	4	2	3

	3	6	10	12	15
	5	7	13	16	17
	9	11	14	18	21
T_i	0	2	4	7	0

Makespan = 21

Sum of Completion times = 73

Total Weighted Tardiness = 34

Computing completion times

$$C_{\pi(1)j} = \sum_{h=1}^{j} p_{\pi(1)h} \qquad j = 1, \ldots\ m$$
$$C_{\pi(k)1} = \sum_{h=1}^{k} p_{\pi(h)1} \qquad k = 1, \ldots\ n$$
$$C_{\pi(k)j} = max\{C_{\pi(k-1)j}, C_{\pi(k)(j-1)}\} + p_{\pi(k)j} \qquad k = 2, \ldots\ n \quad j = 2, \ldots\ m$$

Job	J_1	J_2	J_3	J_4	J_5
p_{i1}	3	3	4	2	3
p_{i2}	2	1	3	3	1
p_{i3}	4	2	1	2	3
d_i	11	9	10	11	30
w_i	1	2	4	2	3

	3	6	10	12	15
	5	7	13	16	17
	9	11	14	18	21
T_i	0	2	4	7	0

Makespan = 21

Sum of Completion times = 73

Total Weighted Tardiness = 34

Computing completion times

$$
\begin{aligned}
C_{\pi(1)j} &= \sum_{h=1}^{j} p_{\pi(1)h} & j &= 1, \ldots\ m \\
C_{\pi(k)1} &= \sum_{h=1}^{k} p_{\pi(h)1} & k &= 1, \ldots\ n \\
C_{\pi(k)j} &= max\{C_{\pi(k-1)j}, C_{\pi(k)(j-1)}\} + p_{\pi(k)j} & k &= 2, \ldots\ n \\
& & j &= 2, \ldots\ m
\end{aligned}
$$

Job	J_1	J_2	J_3	J_4	J_5
p_{i1}	3	3	4	2	3
p_{i2}	2	1	3	3	1
p_{i3}	4	2	1	2	3
d_i	11	9	10	11	30
w_i	1	2	4	2	3

	3	6	10	12	15
	5	7	13	16	17
	9	11	14	18	21
T_i	0	2	4	7	0

Makespan = 21

Sum of Completion times = 73

Total Weighted Tardiness = 34

Computing completion times

$$C_{\pi(1)j} = \sum_{h=1}^{j} p_{\pi(1)h} \qquad j = 1, \ldots m$$
$$C_{\pi(k)1} = \sum_{h=1}^{k} p_{\pi(h)1} \qquad k = 1, \ldots n$$
$$C_{\pi(k)j} = max\{C_{\pi(k-1)j}, C_{\pi(k)(j-1)}\} + p_{\pi(k)j} \qquad k = 2, \ldots n \quad j = 2, \ldots m$$

Job	J_1	J_2	J_3	J_4	J_5
p_{i1}	3	3	4	2	3
p_{i2}	2	1	3	3	1
p_{i3}	4	2	1	2	3

d_i	11	9	10	11	30
w_i	1	2	4	2	3

	3	6	10	12	15
	5	7	13	16	17
	9	11	14	18	21
T_i	0	2	4	7	0

Makespan = 21

Sum of Completion times = 73

Total Weighted Tardiness = 34

Computing completion times

$$C_{\pi(1)j} = \sum_{h=1}^{j} p_{\pi(1)h} \qquad j = 1, \ldots\ m$$
$$C_{\pi(k)1} = \sum_{h=1}^{k} p_{\pi(h)1} \qquad k = 1, \ldots\ n$$
$$C_{\pi(k)j} = max\{C_{\pi(k-1)j}, C_{\pi(k)(j-1)}\} + p_{\pi(k)j} \qquad k = 2, \ldots\ n \quad j = 2, \ldots\ m$$

Job	J_1	J_2	J_3	J_4	J_5
p_{i1}	3	3	4	2	3
p_{i2}	2	1	3	3	1
p_{i3}	4	2	1	2	3
d_i	11	9	10	11	30
w_i	1	2	4	2	3

	3	6	10	12	15
	5	7	13	16	17
	9	11	14	18	21
T_i	0	2	4	7	0

Makespan = 21

Sum of Completion times = 73

Total Weighted Tardiness = 34

Computing completion times

$$C_{\pi(1)j} = \sum_{h=1}^{j} p_{\pi(1)h} \qquad j = 1, \ldots m$$
$$C_{\pi(k)1} = \sum_{h=1}^{k} p_{\pi(h)1} \qquad k = 1, \ldots n$$
$$C_{\pi(k)j} = max\{C_{\pi(k-1)j}, C_{\pi(k)(j-1)}\} + p_{\pi(k)j} \qquad k = 2, \ldots n \quad j = 2, \ldots m$$

Job	J_1	J_2	J_3	J_4	J_5
p_{i1}	3	3	4	2	3
p_{i2}	2	1	3	3	1
p_{i3}	4	2	1	2	3
d_i	11	9	10	11	30
w_i	1	2	4	2	3

	3	6	10	12	15
	5	7	13	16	17
	9	11	14	18	21
T_i	0	2	4	7	0

Makespan = 21

Sum of Completion times = 73

Total Weighted Tardiness = 34

Computing completion times

$$C_{\pi(1)j} = \sum_{h=1}^{j} p_{\pi(1)h} \qquad j = 1, \ldots m$$
$$C_{\pi(k)1} = \sum_{h=1}^{k} p_{\pi(h)1} \qquad k = 1, \ldots n$$
$$C_{\pi(k)j} = max\{C_{\pi(k-1)j}, C_{\pi(k)(j-1)}\} + p_{\pi(k)j} \qquad k = 2, \ldots n \quad j = 2, \ldots m$$

Job	J_1	J_2	J_3	J_4	J_5
p_{i1}	3	3	4	2	3
p_{i2}	2	1	3	3	1
p_{i3}	4	2	1	2	3
d_i	11	9	10	11	30
w_i	1	2	4	2	3

	3	6	10	12	15
	5	7	13	16	17
	9	11	14	18	21
T_i	0	2	4	7	0

Makespan = 21

Sum of Completion times = 73

Total Weighted Tardiness = 34

The Permutation Flow Shop Scheduling Problem (4/4)

Computing completion times

$$C_{\pi(1)j} = \sum_{h=1}^{j} p_{\pi(1)h} \qquad j = 1, \ldots\ m$$
$$C_{\pi(k)1} = \sum_{h=1}^{k} p_{\pi(h)1} \qquad k = 1, \ldots\ n$$
$$C_{\pi(k)j} = max\{C_{\pi(k-1)j}, C_{\pi(k)(j-1)}\} + p_{\pi(k)j} \qquad k = 2, \ldots\ n \quad j = 2, \ldots\ m$$

Job	J_1	J_2	J_3	J_4	J_5
p_{i1}	3	3	4	2	3
p_{i2}	2	1	3	3	1
p_{i3}	4	2	1	2	3
d_i	11	9	10	11	30
w_i	1	2	4	2	3

	3	6	10	12	15
	5	7	13	16	17
	9	11	14	18	21
T_i	0	2	4	7	0

Makespan = 21

Sum of Completion times = 73

Total Weighted Tardiness = 34

Computing completion times

$$C_{\pi(1)j} = \sum_{h=1}^{j} p_{\pi(1)h} \qquad j = 1, \ldots m$$
$$C_{\pi(k)1} = \sum_{h=1}^{k} p_{\pi(h)1} \qquad k = 1, \ldots n$$
$$C_{\pi(k)j} = max\{C_{\pi(k-1)j}, C_{\pi(k)(j-1)}\} + p_{\pi(k)j} \qquad k = 2, \ldots n \quad j = 2, \ldots m$$

Job	J_1	J_2	J_3	J_4	J_5
p_{i1}	3	3	4	2	3
p_{i2}	2	1	3	3	1
p_{i3}	4	2	1	2	3
d_i	11	9	10	11	30
w_i	1	2	4	2	3

	3	6	10	12	15
	5	7	13	16	17
	9	11	14	18	21
T_i	0	2	4	7	0

Makespan = 21

Sum of Completion times = 73

Total Weighted Tardiness = 34

The Permutation Flow Shop Scheduling Problem (4/4)

Computing completion times

$$C_{\pi(1)j} = \sum_{h=1}^{j} p_{\pi(1)h} \qquad j = 1, \ldots\ m$$
$$C_{\pi(k)1} = \sum_{h=1}^{k} p_{\pi(h)1} \qquad k = 1, \ldots\ n$$
$$C_{\pi(k)j} = max\{C_{\pi(k-1)j}, C_{\pi(k)(j-1)}\} + p_{\pi(k)j} \qquad k = 2, \ldots\ n \quad j = 2, \ldots\ m$$

Job	J_1	J_2	J_3	J_4	J_5
p_{i1}	3	3	4	2	3
p_{i2}	2	1	3	3	1
p_{i3}	4	2	1	2	3
d_i	11	9	10	11	30
w_i	1	2	4	2	3

	3	6	10	12	15
	5	7	13	16	17
	9	11	14	18	21
T_i	0	2	4	7	0

Makespan = 21

Sum of Completion times = 73

Total Weighted Tardiness = 34

Computing completion times

$$C_{\pi(1)j} = \sum_{h=1}^{j} p_{\pi(1)h} \qquad j = 1, \dots m$$
$$C_{\pi(k)1} = \sum_{h=1}^{k} p_{\pi(h)1} \qquad k = 1, \dots n$$
$$C_{\pi(k)j} = max\{C_{\pi(k-1)j}, C_{\pi(k)(j-1)}\} + p_{\pi(k)j} \qquad k = 2, \dots n \quad j = 2, \dots m$$

Job	J_1	J_2	J_3	J_4	J_5
p_{i1}	3	3	4	2	3
p_{i2}	2	1	3	3	1
p_{i3}	4	2	1	2	3
d_i	11	9	10	11	30
w_i	1	2	4	2	3

	3	6	10	12	15
	5	7	13	16	17
	9	11	14	18	21
T_i	0	2	4	7	0

Makespan = 21

Sum of Completion times = 73

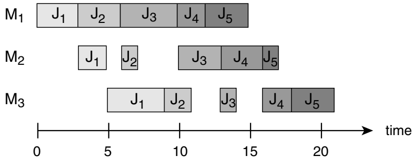
Total Weighted Tardiness = 34

The Permutation Flow Shop Scheduling Problem (4/4)

Computing completion times

$$
\begin{aligned}
C_{\pi(1)j} &= \textstyle\sum_{h=1}^{j} p_{\pi(1)h} & j &= 1, \ldots\ m \\
C_{\pi(k)1} &= \textstyle\sum_{h=1}^{k} p_{\pi(h)1} & k &= 1, \ldots\ n \\
C_{\pi(k)j} &= max\{C_{\pi(k-1)j}, C_{\pi(k)(j-1)}\} + p_{\pi(k)j} & k &= 2, \ldots\ n \\
& & j &= 2, \ldots\ m
\end{aligned}
$$

Job	J_1	J_2	J_3	J_4	J_5
p_{i1}	3	3	4	2	3
p_{i2}	2	1	3	3	1
p_{i3}	4	2	1	2	3
d_i	11	9	10	11	30
w_i	1	2	4	2	3



	3	6	10	12	15
	5	7	13	16	17
	9	11	14	18	21
T_i	0	2	4	7	0

Makespan = 21

Sum of Completion times = 73

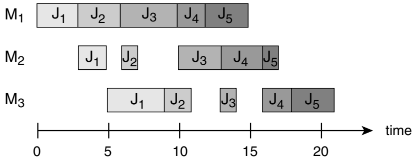
Total Weighted Tardiness = 34

The Permutation Flow Shop Scheduling Problem (4/4)

Computing completion times

$$
\begin{aligned}
C_{\pi(1)j} &= \textstyle\sum_{h=1}^{j} p_{\pi(1)h} && j = 1, \ldots\ m \\
C_{\pi(k)1} &= \textstyle\sum_{h=1}^{k} p_{\pi(h)1} && k = 1, \ldots\ n \\
C_{\pi(k)j} &= max\{C_{\pi(k-1)j}, C_{\pi(k)(j-1)}\} + p_{\pi(k)j} && k = 2, \ldots\ n \\
& && j = 2, \ldots\ m
\end{aligned}
$$

Job	J_1	J_2	J_3	J_4	J_5
p_{i1}	3	3	4	2	3
p_{i2}	2	1	3	3	1
p_{i3}	4	2	1	2	3
d_i	11	9	10	11	30
w_i	1	2	4	2	3



	3	6	10	12	15
	5	7	13	16	17
	9	11	14	18	21
T_i	0	2	4	7	0

Makespan = 21

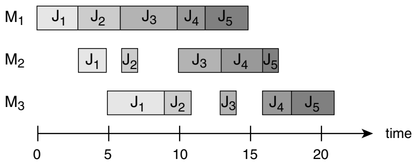
Sum of Completion times = 73

Total Weighted Tardiness = 34

Computing completion times

$$
\begin{aligned}
C_{\pi(1)j} &= \sum_{h=1}^{j} p_{\pi(1)h} && j = 1, \ldots\ m \\
C_{\pi(k)1} &= \sum_{h=1}^{k} p_{\pi(h)1} && k = 1, \ldots\ n \\
C_{\pi(k)j} &= max\{C_{\pi(k-1)j}, C_{\pi(k)(j-1)}\} + p_{\pi(k)j} && k = 2, \ldots\ n \\
& && j = 2, \ldots\ m
\end{aligned}
$$

Job	J_1	J_2	J_3	J_4	J_5
p_{i1}	3	3	4	2	3
p_{i2}	2	1	3	3	1
p_{i3}	4	2	1	2	3
d_i	11	9	10	11	30
w_i	1	2	4	2	3



	3	6	10	12	15
	5	7	13	16	17
	9	11	14	18	21
T_i	0	2	4	7	0

Makespan = 21

Sum of Completion times = 73

Total Weighted Tardiness = 34

The Permutation Flow Shop Scheduling Problem (4/4)

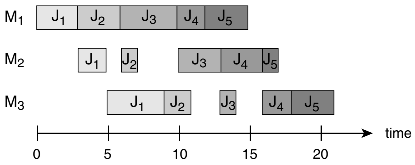
Computing completion times

$$C_{\pi(1)j} = \sum_{h=1}^{j} p_{\pi(1)h} \qquad j = 1, \ldots m$$

$$C_{\pi(k)1} = \sum_{h=1}^{k} p_{\pi(h)1} \qquad k = 1, \ldots n$$

$$C_{\pi(k)j} = max\{C_{\pi(k-1)j}, C_{\pi(k)(j-1)}\} + p_{\pi(k)j} \qquad k = 2, \ldots n \quad j = 2, \ldots m$$

Job	J_1	J_2	J_3	J_4	J_5
p_{i1}	3	3	4	2	3
p_{i2}	2	1	3	3	1
p_{i3}	4	2	1	2	3
d_i	11	9	10	11	30
w_i	1	2	4	2	3



	3	6	10	12	15
	5	7	13	16	17
	9	11	14	18	21
T_i	0	2	4	7	0

Makespan = 21

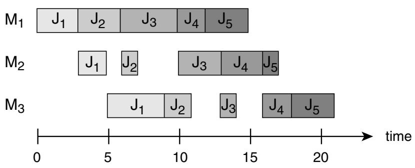
Sum of Completion times = 73

Total Weighted Tardiness = 34

Computing completion times

$$C_{\pi(1)j} = \sum_{h=1}^{j} p_{\pi(1)h} \qquad j = 1, \ldots m$$
$$C_{\pi(k)1} = \sum_{h=1}^{k} p_{\pi(h)1} \qquad k = 1, \ldots n$$
$$C_{\pi(k)j} = max\{C_{\pi(k-1)j}, C_{\pi(k)(j-1)}\} + p_{\pi(k)j} \qquad k = 2, \ldots n \quad j = 2, \ldots m$$

Job	J_1	J_2	J_3	J_4	J_5
p_{i1}	3	3	4	2	3
p_{i2}	2	1	3	3	1
p_{i3}	4	2	1	2	3
d_i	11	9	10	11	30
w_i	1	2	4	2	3



	3	6	10	12	15
	5	7	13	16	17
	9	11	14	18	21
T_i	0	2	4	7	0

Makespan = 21

Sum of Completion times = 73

Total Weighted Tardiness = 34

The Permutation Flow Shop Scheduling Problem (4/4)

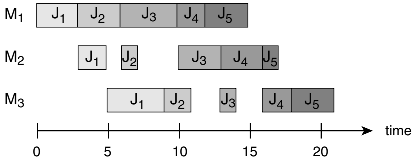
Computing completion times

$$C_{\pi(1)j} = \sum_{h=1}^{j} p_{\pi(1)h} \qquad j = 1, \ldots\ m$$

$$C_{\pi(k)1} = \sum_{h=1}^{k} p_{\pi(h)1} \qquad k = 1, \ldots\ n$$

$$C_{\pi(k)j} = max\{C_{\pi(k-1)j}, C_{\pi(k)(j-1)}\} + p_{\pi(k)j} \qquad k = 2, \ldots\ n \quad j = 2, \ldots\ m$$

Job	J_1	J_2	J_3	J_4	J_5
p_{i1}	3	3	4	2	3
p_{i2}	2	1	3	3	1
p_{i3}	4	2	1	2	3
d_i	11	9	10	11	30
w_i	1	2	4	2	3



	3	6	10	12	15
	5	7	13	16	17
	9	11	14	18	21
T_i	0	2	4	7	0

Makespan = 21

Sum of Completion times = 73

Total Weighted Tardiness = 34

Exercise 1.1: Iterative Improvement for the PFSP

Implement 12 iterative improvements algorithms for the PFSP

Exercise 1.1: Iterative Improvement for the PFSP

Implement 12 iterative improvements algorithms for the PFSP

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

Exercise 1.1: Iterative Improvement for the PFSP

Implement 12 iterative improvements algorithms for the PFSP

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

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

Exercise 1.1: Iterative Improvement for the PFSP

Implement 12 iterative improvements algorithms for the PFSP

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

```
pfsp-ii --first --transpose --neh
pfsp-ii --best --exchange --random-init
...
```

Exercise 1.1: Iterative Improvement for the PFSP

Iterative Improvement

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

Exercise 1.1: Iterative Improvement for the PFSP

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.
  - ✔ More efficient
  - ✘ Order of evaluation may impact quality / performance

Exercise 1.1: Iterative Improvement for the PFSP

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 PFSP

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 PFSP

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
- SLACK heuristic

Exercise 1.1: Iterative Improvement for the PFSP

Iterative Improvement

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

SLACK heuristic

Construct the solution inserting **one job at a time**, by always selecting the one that minimizes the weighted earlyness.

The weighted earlyness of job J_i is computed as $w_i \cdot (d_i - C_i)$.

Note: the solution is constructed incrementally, and at each iteration C_i corresponds to the makespan of the partial solution.

Exercise 1.1: Iterative Improvement for the PFSP

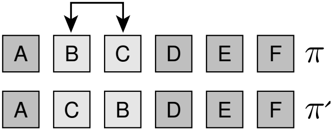
Iterative Improvement

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

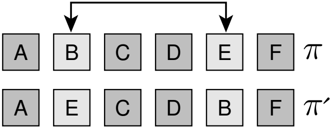
Which neighborhood $\mathcal{N}(\pi)$?

- Transpose
- Exchange
- Insertion

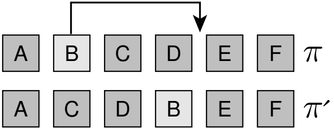
Exercise 1.1: Iterative Improvement for the PFSP



transpose neighbourhood

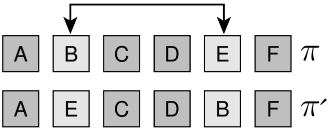


exchange neighbourhood



insert neighbourhood

Exercise 1.1: Iterative Improvement for the PFSP



exchange neighbourhood

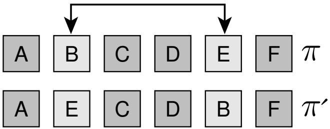
Example: Exchange π_i and π_j ($i < j$), $\pi' = \texttt{Exchange}(\pi, i, j)$

Only jobs after i are affected!

Do not recompute the evaluation function from scratch!

Equivalent speed-ups with Transpose and Insertion

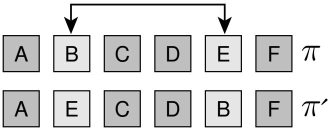
Exercise 1.1: Iterative Improvement for the PFSP



exchange neighbourhood

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

Only jobs after i are affected!
Do not recompute the evaluation function from scratch!
Equivalent speed-ups with Transpose and Insertion



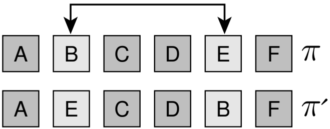
exchange neighbourhood

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

Only jobs after i are affected!

Do not recompute the evaluation function from scratch!

Equivalent speed-ups with Transpose and Insertion



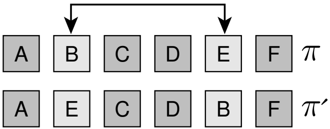
exchange neighbourhood

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

Only jobs after i are affected!

Do not recompute the evaluation function from scratch!

Equivalent speed-ups with Transpose and Insertion



exchange neighbourhood

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

Only jobs after i are affected!

Do not recompute the evaluation function from scratch!

Equivalent speed-ups with Transpose and Insertion

Exercise 1.1: Iterative Improvement for the PFSP

Instances

- PFSP instances with 50, 60, 70, 80, 90, and 100 jobs, and 20 machines.
- 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 PFSP

Instances

- PFSP instances with 50, 60, 70, 80, 90, and 100 jobs, and 20 machines.
- 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 PFSP

Instances

- PFSP instances with 50, 60, 70, 80, 90, and 100 jobs, and 20 machines.
- 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 PFSP

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

Statistical test

- Paired t-test
- Wilcoxon signed-rank test

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 PFSP

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.

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.

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 PFSP

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.

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/).

Exercise 1.1: Iterative Improvement for the PFSP

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/).

Exercise 1.1: Iterative Improvement for the PFSP

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/).

Exercise 1.1: Iterative Improvement for the PFSP

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
```

Exercise 1.1: Iterative Improvement for the PFSP

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
```

Exercise 1.1: Iterative Improvement for the PFSP

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
```

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
```

Exercise 1.1: Iterative Improvement for the PFSP

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
```

Exercise 1.1: Iterative Improvement for the PFSP

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
```

Exercise 1.2 VND algorithms for the PFSP

Implement 4 VND algorithms for the PFSP

- Pivoting rule: first-improvement
- Neighborhood order:
  1. transpose $\rightarrow$ exchange $\rightarrow$ insert
  2. transpose $\rightarrow$ insert $\rightarrow$ exchange
- Initial solution:
  1. Random permutation
  2. SLACK heuristic

Exercise 1.2 VND algorithms for the PFSP

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 $i > k$

Implement 4 VND algorithms for the PFSP

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