Iterated Local Search

Jean-Louis Bouquard

Beijing Institute of Technology

Otimization Methods

Content

1. Previously on Optmization Methods

Previously on Optmization Methods

Local search for the $F2//\overline{T}$

- These moves can be used with every problem the solution of which is a permutation
- Move 1: consecutive swap k and $k+1$
- Move 2: any swap k_1 and k_2: 2-opt
- Move 3: rotation (k_1, k_2, k_3): 3-opt
- Move 4: Extraction and Backward Shift Reinsertion: EBSR
- Move 5: Extraction and Forward Shift Reinsertion: EFSR

Previously on Optmization Methods

Local search for the $F2//\overline{T}$

- These moves can be used with every problem the solution of which is a permutation
- Move 1: consecutive swap k and $k+1$
- Move 2: any swap k_1 and k_2: 2-opt
- Move 3: rotation (k_1, k_2, k_3): 3-opt
- Move 4: Extraction and Backward Shift Reinsertion: EBSR
- Move 5: Extraction and Forward Shift Reinsertion: EFSR

Previously on Optmization Methods

Local search for the $F2//\overline{T}$

- These moves can be used with every problem the solution of which is a permutation
- Move 1: consecutive swap k and $k+1$
- Move 2: any swap k_1 and k_2: 2-opt
- Move 3: rotation (k_1, k_2, k_3): 3-opt
- Move 4: Extraction and Backward Shift Reinsertion: EBSR
- Move 5: Extraction and Forward Shift Reinsertion: EFSR

Previously on Optmization Methods

Local search for the $F2//\overline{T}$

- These moves can be used with every problem the solution of which is a permutation
- Move 1: consecutive swap k and $k+1$
- Move 2: any swap k_1 and k_2: 2-opt
- Move 3: rotation (k_1, k_2, k_3): 3-opt
- Move 4: Extraction and Backward Shift Reinsertion: EBSR
- Move 5: Extraction and Forward Shift Reinsertion: EFSR

Previously on Optmization Methods

Local search for the $F2//\overline{T}$

- These moves can be used with every problem the solution of which is a permutation
- Move 1: consecutive swap k and $k+1$
- Move 2: any swap k_1 and k_2: 2-opt
- Move 3: rotation (k_1, k_2, k_3): 3-opt
- Move 4: Extraction and Backward Shift Reinsertion: EBSR
- Move 5: Extraction and Forward Shift Reinsertion: EFSR

Previously on Optmization Methods

Local search for the $F2//\overline{T}$

- These moves can be used with every problem the solution of which is a permutation
- Move 1: consecutive swap k and $k+1$
- Move 2: any swap k_1 and k_2: 2-opt
- Move 3: rotation (k_1, k_2, k_3): 3-opt
- Move 4: Extraction and Backward Shift Reinsertion: EBSR
- Move 5: Extraction and Forward Shift Reinsertion: EFSR

Algorithm of exploring a neighborhood

Local search

function *LocalSearch*(s)

Require: s is a solution of the problem

Ensure: The neighborhood of s is explored and *bestneighbor* is returned. A boolean value *Improved* is also returned.

```
currentvalue ← f(s)
bestvalue ← currentvalue
bestneighbor ← s
for t in neighborhood(s) do
    if (f(t) < bestvalue) then
        bestvalue ← f(t)
        bestneighbor ← t
    end if
end for
Improved ← (bestvalue < currentvalue)
return (Improved, bestneighbor)
```

Algorithm of the Iterated Local Search

Iterated Local Search

function *IteratedLS*(s)

Require: s is a solution of the problem

Ensure: The neighborhood of s is explored as long as an improvement is proved. Then the current best solution is returned.

1: *Improved* $\leftarrow$ **true**
2: **while** (*Improved*) **do**
3:     (*Improved*, *bestneighbor*) $\leftarrow$ *LocalSearch*(s)
4: **end while**
5: **return** *bestneighbor*

2 The initial solution

Initial solution

- Any solution
- $\{1, 2, \ldots, n\}$ or a random one
- A constructive method:
- (EDD): Earliest Due Date first method
- Jobs are sorted in the increasing order of the d_j's
- (NEH): Nawaz, Enscore, Ham (Omega 1983) propose a Heuristic for the general flowshop problem

Initial solution

- Any solution
- $\{1, 2, \ldots, n\}$ or a random one
- A constructive method:
- (EDD): Earliest Due Date first method
- Jobs are sorted in the increasing order of the d_j's
- (NEH): Nawaz, Enscore, Ham (Omega 1983) propose a Heuristic for the general flowshop problem

Initial solution

- Any solution
- $\{1, 2, \ldots, n\}$ or a random one
- A constructive method:
- (EDD): Earliest Due Date first method
- Jobs are sorted in the increasing order of the d_j's
- (NEH): Nawaz, Enscore, Ham (Omega 1983) propose a Heuristic for the general flowshop problem

Initial solution

- Any solution
- $\{1, 2, \ldots, n\}$ or a random one
- A constructive method:
- (EDD): Earliest Due Date first method
- Jobs are sorted in the increasing order of the d_j's
- (NEH): Nawaz, Enscore, Ham (Omega 1983) propose a Heuristic for the general flowshop problem

Initial solution

- Any solution
- $\{1, 2, \ldots, n\}$ or a random one
- A constructive method:
- (EDD): Earliest Due Date first method
- Jobs are sorted in the increasing order of the d_j's
- (NEH): Nawaz, Enscore, Ham (Omega 1983) propose a Heuristic for the general flowshop problem

Initial solution

- Any solution
- $\{1, 2, \ldots, n\}$ or a random one
- A constructive method:
- (EDD): Earliest Due Date first method
- Jobs are sorted in the increasing order of the d_j's
- (NEH): Nawaz, Enscore, Ham (Omega 1983) propose a Heuristic for the general flowshop problem

Algorithm Earliest Due Date

Earliest Due Date

function $EDD(s)$

Require: The list of the d_j's

Ensure: A permutation corresponding to EDD is returned.

1: Pair the d_j's with j $\# \; [(d_1, 1), (d_2, 2), \ldots, (d_n, n)]$

2: Sort the pairs in increasing order of the first elements

3: **return** The sequence of the second elements

Remark: EDD is optimal for the $1//Lmax$ problem

Algorithm Earliest Due Date

Earliest Due Date

function *EDD*(s)

Require: The list of the d_j's

Ensure: A permutation corresponding to EDD is returned.

1: Pair the d_j's with j $\# \; [(d_1, 1), (d_2, 2), \ldots, (d_n, n)]$

2: Sort the pairs in increasing order of the first elements

3: **return** The sequence of the second elements

Remark: EDD is optimal for the $1//Lmax$ problem

Algorithm NEH for the $F2//\overline{T}$

NEH

function *NEH*(s)

Require: The list of the $(p_{1,j}, p_{2,j}, d_j)$'s

Ensure: A permutation corresponding to NEH is returned.

$s \leftarrow$ Sort the jobs in an initial order $\#$ e.g. EDD or . . . ?

$neh \leftarrow [s(1)]$

for k **from** 2 **to** n **do** $\#$ *neh* is a list of $k-1$ elements

  $(mini, nextneh) \leftarrow (+\infty, [])$

  for j **from** 1 **to** k **do**

    $\sigma \leftarrow$ Insert job $s(k)$ in position j in *neh*

    if $(\overline{T}(\sigma) < mini)$ **then**

      $(mini, nextneh) \leftarrow (\overline{T}(\sigma), \sigma)$

    end if

  end for

  $neh \leftarrow nextneh$

end for

return *neh*

3 Experimentation

Experimentation

- What can be proved by theory?
- The cardinality of the neighborhood
- Most of the times, nothing much
- Practical efficiency is the target
- Let us experiment!

Experimentation

- What can be proved by theory?
- The cardinality of the neighborhood
- Most of the times, nothing much
- Practical efficiency is the target
- Let us experiment!

Experimentation

- What can be proved by theory?
- The cardinality of the neighborhood
- Most of the times, nothing much
- Practical efficiency is the target
- Let us experiment

Experimentation

- What can be proved by theory?
- The cardinality of the neighborhood
- Most of the times, nothing much
- Practical efficiency is the target
- Let us experiment

Experimentation

- What can be proved by theory?
- The cardinality of the neighborhood
- Most of the times, nothing much
- Practical efficiency is the target
- Let us experiment

Experimentation

- What do we need?
- A problem
- Instances
- They already exist or we generate them
- How many ?
- How to generate ?
- Choice of the parameters

Experimentation

- What do we need?
- A problem
- Instances
- They already exist or we generate them
- How many ?
- How to generate ?
- Choice of the parameters

Experimentation

- What do we need?
- A problem
- Instances
- They already exist or we generate them
- How many ?
- How to generate ?
- Choice of the parameters

Experimentation

- What do we need?
- A problem
- Instances
- They already exist or we generate them
- How many ?
- How to generate ?
- Choice of the parameters

Experimentation

- What do we need?
- A problem
- Instances
- They already exist or we generate them
- How many ?
- How to generate ?
- Choice of the parameters

Experimentation

- What do we need?
- A problem
- Instances
- They already exist or we generate them
- How many ?
- How to generate ?
- Choice of the parameters

Experimentation

- What do we need?
- A problem
- Instances
- They already exist or we generate them
- How many ?
- How to generate ?
- Choice of the parameters

Raw results

See the spreadsheet.

Conclusion

What can we conclude from all these numbers?

Reflection

- Is the consec. swap nbh included in 2opt-nbh?
- Is the 2-opt nbh included in the 3-opt nbh ?
- Consec swap nbh included in 2-opt, EBSR and EFSR
- If $Nbh_1 \subset Nbh_2$, is LS_2 better than LS_1?
- If $Nbh_1 \subset Nbh_2$, is $ItLS_2$ better than $ItLS_1$?

Reflection

- Is the consec. swap nbh included in 2opt-nbh?
- Is the 2-opt nbh included in the 3-opt nbh ?
- Consec swap nbh included in 2-opt, EBSR and EFSR
- If $Nbh_1 \subset Nbh_2$, is LS_2 better than LS_1 ?
- If $Nbh_1 \subset Nbh_2$, is $ItLS_2$ better than $ItLS_1$?

Reflection

- Is the consec. swap nbh included in 2opt-nbh?
- Is the 2-opt nbh included in the 3-opt nbh ?
- Consec swap nbh included in 2-opt, EBSR and EFSR
- If $Nbh_1 \subset Nbh_2$, is LS_2 better than LS_1 ?
- If $Nbh_1 \subset Nbh_2$, is $ItLS_2$ better than $ItLS_1$?

Reflection

- Is the consec. swap nbh included in 2opt-nbh?
- Is the 2-opt nbh included in the 3-opt nbh ?
- Consec swap nbh included in 2-opt, EBSR and EFSR
- If $Nbh_1 \subset Nbh_2$, is LS_2 better than LS_1 ?
- If $Nbh_1 \subset Nbh_2$, is $ItLS_2$ better than $ItLS_1$?

Reflection

- Is the consec. swap nbh included in 2opt-nbh?
- Is the 2-opt nbh included in the 3-opt nbh ?
- Consec swap nbh included in 2-opt, EBSR and EFSR
- If $Nbh_1 \subset Nbh_2$, is LS_2 better than LS_1 ?
- If $Nbh_1 \subset Nbh_2$, is $ItLS_2$ better than $ItLS_1$?

Thinking further

Look at this case:

	$p_{1,j}$	$p_{2,j}$	d_j
1	10	1	12
2	10	1	13
3	1	10	14
4	1	10	22

1 before 2 and 3 before 4

Domination rule: if $p_{1,j_1} = p_{1,j_2}$, $p_{2,j_1} = p_{2,j_2}$ and $d_{j_1} \leq d_{j_2}$ then j_1 can be scheduled before j_2

Swapping j_2 before j_1 cannot increase the criterion.
We can restrict our research to the set of the solutions such that j_1 is before j_2.

Thinking further

Look at this case:

	$p_{1,j}$	$p_{2,j}$	d_j
1	10	1	12
2	10	1	13
3	1	10	14
4	1	10	22

1 before 2 and 3 before 4

Domination rule: if $p_{1,j_1} = p_{1,j_2}$, $p_{2,j_1} = p_{2,j_2}$ and $d_{j_1} \leq d_{j_2}$ then j_1 can be scheduled before j_2

Swapping j_2 before j_1 cannot increase the criterion.
We can restrict our research to the set of the solutions such that j_1 is before j_2.

Thinking further

Look at this case:

	$p_{1,j}$	$p_{2,j}$	d_j
1	10	1	12
2	10	1	13
3	1	10	14
4	1	10	22

1 before 2 and 3 before 4

Domination rule: if $p_{1,j_1} = p_{1,j_2}$, $p_{2,j_1} = p_{2,j_2}$ and $d_{j_1} \leq d_{j_2}$ then j_1 can be scheduled before j_2

Swapping j_2 before j_1 cannot increase the criterion.
We can restrict our research to the set of the solutions such that j_1 is before j_2.

4 Mathematical modeling

Mathematical modeling

- Define clearly the problem, the data
- Choose the variables
- Write the constraints
- and the objective function
- Do not confuse variables (unknown, decisions) and data (known)
- Do not confuse math-model-variables and algorithm-variables

Mathematical modeling

- Define clearly the problem, the data
- Choose the variables
- Write the constraints
- and the objective function
- Do not confuse variables (unknown, decisions) and data (known)
- Do not confuse math-model-variables and algorithm-variables

Mathematical modeling

- Define clearly the problem, the data
- Choose the variables
- Write the constraints
- and the objective function
- Do not confuse variables (unknown, decisions) and data (known)
- Do not confuse math-model-variables and algorithm-variables

Mathematical modeling

- Define clearly the problem, the data
- Choose the variables
- Write the constraints
- and the objective function
- Do not confuse variables (unknown, decisions) and data (known)
- Do not confuse math-model-variables and algorithm-variables

Mathematical modeling

- Define clearly the problem, the data
- Choose the variables
- Write the constraints
- and the objective function
- Do not confuse variables (unknown, decisions) and data (known)
- Do not confuse math-model-variables and algorithm-variables

Mathematical modeling

- Define clearly the problem, the data
- Choose the variables
- Write the constraints
- and the objective function
- Do not confuse variables (unknown, decisions) and data (known)
- Do not confuse math-model-variables and algorithm-variables

A first model
Variables

- For every 2 jobs j_1 and j_2 $(j_1 < j_2)$
- Z_{j_1,j_2} is a binary variable
- indicating whether j_1 is before j_2
- $Z_{j_1,j_2} = \begin{cases} 0 & \text{if } j_1 \text{ before } j_2 \\ 1 & \text{if } j_1 \text{ after } j_2 \end{cases}$
- For every job j, $C_{1,j}$ and $C_{2,j}$ are the completion times of j
- T_j is the tardiness of j
- We have $\binom{n}{2} = \frac{n(n-1)}{2}$ binary variables
- and $3n$ continuous nonnegative variables

A first model
Variables

- For every 2 jobs j_1 and j_2 ($j_1 < j_2$)
- Z_{j_1,j_2} is a binary variable
- indicating whether j_1 is before j_2
- $Z_{j_1,j_2} = \begin{cases} 0 & \textit{if } j_1 \textit{ before } j_2 \\ 1 & \textit{if } j_1 \textit{ after } j_2 \end{cases}$
- For every job j, $C_{1,j}$ and $C_{2,j}$ are the completion times of j
- T_j is the tardiness of j
- We have $\binom{n}{2} = \frac{n(n-1)}{2}$ binary variables
- and $3n$ continuous nonnegative variables

A first model
Variables

- For every 2 jobs j_1 and j_2 ($j_1 < j_2$)
- Z_{j_1,j_2} is a binary variable
- indicating whether j_1 is before j_2
- $Z_{j_1,j_2} = \begin{cases} 0 & \textit{if } j_1 \textit{ before } j_2 \\ 1 & \textit{if } j_1 \textit{ after } j_2 \end{cases}$
- For every job j, $C_{1,j}$ and $C_{2,j}$ are the completion times of j
- T_j is the tardiness of j
- We have $\begin{pmatrix} n \\ 2 \end{pmatrix} = \frac{n(n-1)}{2}$ binary variables
- and $3n$ continuous nonnegative variables

A first model
Variables

- For every 2 jobs j_1 and j_2 ($j_1 < j_2$)
- Z_{j_1,j_2} is a binary variable
- indicating whether j_1 is before j_2
- $Z_{j_1,j_2} = \begin{cases} 0 & \textit{if } j_1 \textit{ before } j_2 \\ 1 & \textit{if } j_1 \textit{ after } j_2 \end{cases}$
- For every job j, $C_{1,j}$ and $C_{2,j}$ are the completion times of j
- T_j is the tardiness of j
- We have $\binom{n}{2} = \frac{n(n-1)}{2}$ binary variables
- and $3n$ continuous nonnegative variables

A first model
Variables

- For every 2 jobs j_1 and j_2 ($j_1 < j_2$)
- Z_{j_1,j_2} is a binary variable
- indicating whether j_1 is before j_2
- $Z_{j_1,j_2} = \begin{cases} 0 & \text{if } j_1 \text{ before } j_2 \\ 1 & \text{if } j_1 \text{ after } j_2 \end{cases}$
- For every job j, $C_{1,j}$ and $C_{2,j}$ are the completion times of j
- T_j is the tardiness of j
- We have $\binom{n}{2} = \frac{n(n-1)}{2}$ binary variables
- and $3n$ continuous nonnegative variables

A first model
Variables

- For every 2 jobs j_1 and j_2 ($j_1 < j_2$)
- Z_{j_1,j_2} is a binary variable
- indicating whether j_1 is before j_2
- $Z_{j_1,j_2} = \begin{cases} 0 & \textit{if } j_1 \textit{ before } j_2 \\ 1 & \textit{if } j_1 \textit{ after } j_2 \end{cases}$
- For every job j, $C_{1,j}$ and $C_{2,j}$ are the completion times of j
- T_j is the tardiness of j
- We have $\binom{n}{2} = \frac{n(n-1)}{2}$ binary variables
- and $3n$ continuous nonnegative variables

A first model

Variables

- For every 2 jobs j_1 and j_2 ($j_1 < j_2$)
- Z_{j_1,j_2} is a binary variable
- indicating whether j_1 is before j_2
- $Z_{j_1,j_2} = \begin{cases} 0 & \textit{if } j_1 \textit{ before } j_2 \\ 1 & \textit{if } j_1 \textit{ after } j_2 \end{cases}$
- For every job j, $C_{1,j}$ and $C_{2,j}$ are the completion times of j
- T_j is the tardiness of j
- We have $\binom{n}{2} = \frac{n(n-1)}{2}$ binary variables
- and $3n$ continuous nonnegative variables

A first model
Variables

- For every 2 jobs j_1 and j_2 ($j_1 < j_2$)
- Z_{j_1,j_2} is a binary variable
- indicating whether j_1 is before j_2
- $Z_{j_1,j_2} = \begin{cases} 0 & \textit{if } j_1 \textit{ before } j_2 \\ 1 & \textit{if } j_1 \textit{ after } j_2 \end{cases}$
- For every job j, $C_{1,j}$ and $C_{2,j}$ are the completion times of j
- T_j is the tardiness of j
- We have $\left(\begin{array}{c} n \\ 2 \end{array}\right) = \frac{n(n-1)}{2}$ binary variables
- and $3n$ continuous nonnegative variables

A first model
Constraints

Constraints:

$$\forall (j_1, j_2), 1 \leq j_1 < j_2 \leq n, \; \begin{array}{r} C_{1,j_1} + p_{1,j_1} \leq C_{1,j_2} \, if \; Z_{j_1,j_2} = 0 \\ C_{1,j_2} + p_{1,j_2} \leq C_{1,j_1} \, if \; Z_{j_1,j_2} = 1 \end{array}$$

$$\begin{array}{rll} \forall (j_1, j_2), & C_{1,j_1} + p_{1,j_1} \leq C_{1,j_2} + HV \times Z_{j_1,j_2} & (1) \\ 1 \leq j_1 < j_2 \leq n, & C_{1,j_2} + p_{1,j_2} \leq C_{1,j_1} + HV \times (1 - Z_{j_1,j_2}) & (2) \end{array}$$

$$\begin{array}{rll} \forall (j_1, j_2), & C_{2,j_1} + p_{2,j_1} \leq C_{2,j_2} + HV \times Z_{j_1,j_2} & (3) \\ 1 \leq j_1 < j_2 \leq n, & C_{2,j_2} + p_{2,j_2} \leq C_{2,j_1} + HV \times (1 - Z_{j_1,j_2}) & (4) \end{array}$$

A first model
Constraints

Constraints:

$$\forall (j_1, j_2), 1 \leq j_1 < j_2 \leq n,\; \begin{aligned} C_{1,j_1} + p_{1,j_1} &\leq C_{1,j_2} \textit{ if } Z_{j_1,j_2} = 0 \\ C_{1,j_2} + p_{1,j_2} &\leq C_{1,j_1} \textit{ if } Z_{j_1,j_2} = 1 \end{aligned}$$

$$\begin{aligned} \forall (j_1, j_2), \quad & C_{1,j_1} + p_{1,j_1} \leq C_{1,j_2} + HV \times Z_{j_1,j_2} & (1) \\ 1 \leq j_1 < j_2 \leq n,\; & C_{1,j_2} + p_{1,j_2} \leq C_{1,j_1} + HV \times (1 - Z_{j_1,j_2}) & (2) \end{aligned}$$

$$\begin{aligned} \forall (j_1, j_2), \quad & C_{2,j_1} + p_{2,j_1} \leq C_{2,j_2} + HV \times Z_{j_1,j_2} & (3) \\ 1 \leq j_1 < j_2 \leq n,\; & C_{2,j_2} + p_{2,j_2} \leq C_{2,j_1} + HV \times (1 - Z_{j_1,j_2}) & (4) \end{aligned}$$

A first model
Constraints

Constraints:

$$\forall (j_1, j_2), 1 \leq j_1 < j_2 \leq n, \; C_{1,j_1} + p_{1,j_1} \leq C_{1,j_2} \textit{ if } Z_{j_1,j_2} = 0$$
$$C_{1,j_2} + p_{1,j_2} \leq C_{1,j_1} \textit{ if } Z_{j_1,j_2} = 1$$

$$\begin{aligned} \forall (j_1, j_2), \quad & C_{1,j_1} + p_{1,j_1} \leq C_{1,j_2} + HV \times Z_{j_1,j_2} && (1) \\ 1 \leq j_1 < j_2 \leq n, \; & C_{1,j_2} + p_{1,j_2} \leq C_{1,j_1} + HV \times (1 - Z_{j_1,j_2}) && (2) \end{aligned}$$

$$\begin{aligned} \forall (j_1, j_2), \quad & C_{2,j_1} + p_{2,j_1} \leq C_{2,j_2} + HV \times Z_{j_1,j_2} && (3) \\ 1 \leq j_1 < j_2 \leq n, \; & C_{2,j_2} + p_{2,j_2} \leq C_{2,j_1} + HV \times (1 - Z_{j_1,j_2}) && (4) \end{aligned}$$

A first model
Constraints and objective function

$$\forall j, 1 \leq j \leq n, \quad C_{1,j} + p_{1,j} \leq C_{2,j} \tag{5}$$

$$T_j \geq C_{2,j} - d_j \tag{6}$$

Objective function

$$Minimize \sum_{j=1}^{n} T_j$$

A first model
Constraints and objective function

$$\forall j, 1 \leq j \leq n, \quad C_{1,j} + p_{1,j} \leq C_{2,j} \tag{5}$$

$$T_j \geq C_{2,j} - d_j \tag{6}$$

Objective function

$$Minimize \sum_{j=1}^{n} \overline{T}_j$$