Iterated Local Search

Jean-Louis Bouquard

Beijing Institute of Technology

Optimization Methods

Content

1. Previously on Optmization Methods
2. The initial solution
3. Experimentation

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.

```
1:  currentvalue ← f(s)
2:  bestvalue ← currentvalue
3:  bestneighbor ← s
4:  for t in neighborhood(s) do
5:      if (f(t) < bestvalue) then
6:          bestvalue ← f(t)
7:          bestneighbor ← t
8:      end if
9:  end for
10: Improved ← (bestvalue < currentvalue)
11: 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 q_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 q_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.