SPLIT_RTLB.v1 : Readme file

October 2, 2019

The JAVA project SPLIT_RTLB.v1 represents the code of the resolution method described in the article entitled *A new split-based hybrid metaheuristic for the Reconfigurable Transfer Line Balancing Problem.*

The objective of the document is to give to the reader the necessary guidelines in order to:

- Properly install the necessary software and run the code.
- Fix the parameters of the resolution method based on user's choice.
- Describe the input/output format.
- Run the resolution method on a RTLB instance.
- Retrieve the results obtained in the paper.

The code is open-source and can be used/modified if needed. We recommend to cite the article *A new split-based hybrid metaheuristic for the Reconfigurable Transfer Line Balancing Problem.* Please feel free to contact us at: *youssef.lahrichi.contact@gmail.com* for any comment/suggestion/question about the code: this will help the authors enhance the solver.

Installation

The code is given as a JAVA project built thanks to ECLIPSE. We recommend the reader to download eclipse at the following link:

https://www.eclipse.org/downloads/

Then download the SPLIT_RTLB.zip file containing the project and unzip the file into an new file named SPLIT_RTLB.

Open eclipse and select a workspace of your choice. Then click on file -> import -> General -> Existing Projects into Workspace -> Select root directory and browse to SPLIT_RTLB -> Select All -> Finish.

Expand the project. The main function is contained in the src/RLTB.java class.

Project overview

The main classes contained in the project are:

- **RTLB:** This class represents an instance of the RTLB problem.
- **Solution:** This class represents a solution to the problem.
- **Sequence:** This class represents a giant sequence.
- **Split:** This class provides the algorithmic components of the split algorithm.

The other classes can be found in: /doc/package-summary.html (generated with Javadoc).

Input file

we take back the example from the article to illustrate it corresponding input file.

Example

The small instance is described by the following data (same notations as in the paper):

- The part requires the execution of 7 operations numbered from 1 to 7 ($n = 7$).
- At most **5** workstations can be used. ($s_{max} = 5$).

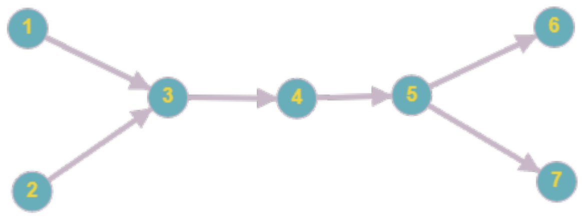


Figure 1: Precedence graph.

- Precedence constraints are given by:

$$P = \{(1,3),(2,3),(3,4),(4,5),(5,6),(5,7)\}$$

  represented by figure 1.

- $N_{max} = 3$, Maximum number of operations that could be assigned to a workstation.

- $M_{max} = 3$, Maximum number of machines that could be hosted by a workstation.

- Processing times are represented in the following table:

i	1	2	3	4	5	6	7
d_i	1.5	1	3.5	1.5	2.5	3	1

- Setup times are represented in the follwing table:

$t_{i,j}$	$j = 1$	2	3	4	5	6	7
$i = 1$	0	0.5	1	1	1	1	1
2	1	0	0.5	1	1	1	1
3	0.5	1	0	1	1	1	1
4	1	1	1	0	0.5	1	1
5	1	1	1	0.5	0	1	1
6	1	1	1	1	1	0	0.5
7	1	1	1	1	1	0.5	0

- $\mathbf{C} = 2.5$, cycle time.

- Inclusion and exclusion constraints are given by:

$$I = \{(1,2)\}, E = \{\{5,6\}\}$$

- Accessibility constraints are given by the following table:

Pos \ *i*	1	2	3	4	5	6	7
1	x	x	x	x		x	x
2	x	x	x	x		x	x
3	x	x	x		x	x	x
4	x	x	x		x	x	x

Then the input file corresponding to the previous example is as follows:

Input file: example.txt

```
Number of operations: /7/;

Maximal number of stations: /5/;

Maximal number of operations per wokstation: /3/;

Maximal number of machines per station: /3/;

Cycle time: /2.5/;

Processing times:
1 = 1.5
2 = 1
3 = 3.5
4 = 1.5
5 = 2.5
6 = 3
7 = 1
/;

Accessbiblity: Number of part-fixing positions /100/
1 . 1
1 . 2
1 . 3
1 . 4
2 . 1
2 . 2
```

```
2 . 3
2 . 4
3 . 1
3 . 2
3 . 3
3 . 4
4 . 1
4 . 2
5 . 3
5 . 4
6 . 1
6 . 2
6 . 3
6 . 4
7 . 1
7 . 2
7 . 3
7 . 4
/;

Precedence constraints:
1 . 3
2 . 3
3 . 4
4 . 5
5 . 6
5 . 7
/;

Inclusion sets:
1 . 1
1 . 2
/;

Exclusion sets:
1 . 5
1 . 6
/;
```

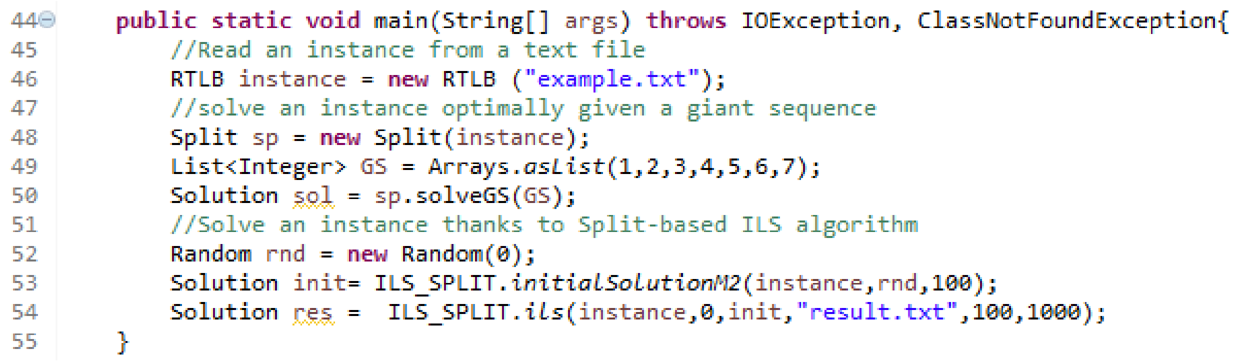
```
Setup times:
1.1=0
1.2=0.5
1.3=1
1.4=1
1.5=1
1.6=1
1.7=1
2.1=1
2.2=0
2.3=0.5
2.4=1
2.5=1
2.6=1
2.7=1
3.1=0.5
3.2=1
3.3=0
3.4=1
3.5=1
3.6=1
3.7=1
4.1=1
4.2=1
4.3=1
4.4=0
4.5=0.5
4.6=1
4.7=1
5.1=1
5.2=1
5.3=1
5.4=0.5
5.5=0
5.6=1
5.7=1
6.1=1
```

```
6.2=1
6.3=1
6.4=1
6.5=1
6.6=0
6.7=0.5
7.1=1
7.2=1
7.3=1
7.4=1
7.5=1
7.6=0.5
7.7=0
/;
```

Warning

Please respect the file format (blank lines and spaces) to avoid compiling errors. We can provide the instances from *Borisovsky, Pavel A., Xavier Delorme, and Alexandre Dolgui. 2013. "Genetic algorithm for balancing reconfigurable machining lines."Computers Industrial Engineering.* in the imput format provided the agreement of the autors.

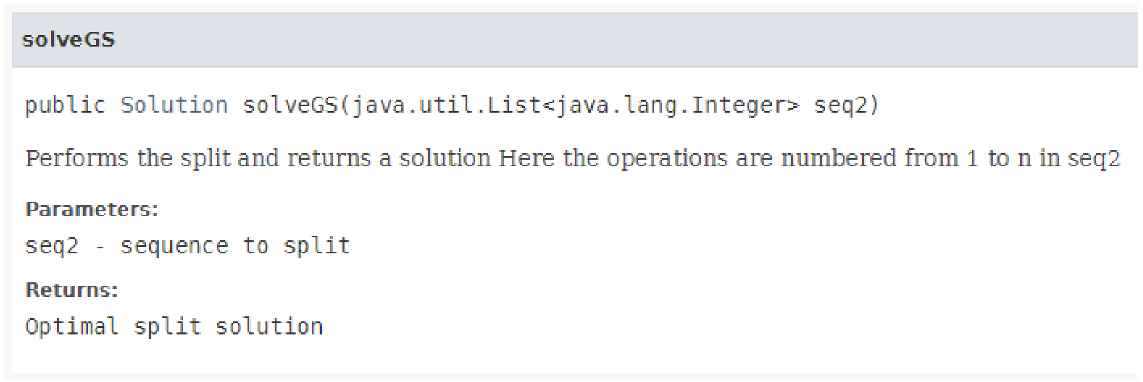
Code to read/solve an instance



```java
    public static void main(String[] args) throws IOException, ClassNotFoundException{
        //Read an instance from a text file
        RTLB instance = new RTLB ("example.txt");
        //solve an instance optimally given a giant sequence
        Split sp = new Split(instance);
        List<Integer> GS = Arrays.asList(1,2,3,4,5,6,7);
        Solution sol = sp.solveGS(GS);
        //Solve an instance thanks to Split-based ILS algorithm
        Random rnd = new Random(0);
        Solution init= ILS_SPLIT.initialSolutionM2(instance,rnd,100);
        Solution res =  ILS_SPLIT.ils(instance,0,init,"result.txt",100,1000);
    }
```

At line 46, an instance is read from the file "example.txt". It is the instance of 7 operations described previously.

At lines 48-50, the instance is solved optimally respecting the giant sequence "1,2,3,4,5,6,7". The optimal solution is stored in the local variable sol (its cost can be retrieved thanks to sol.cost()). The signature of the function solveGS is as follows:



solveGS

```
public Solution solveGS(java.util.List<java.lang.Integer> seq2)
```

Performs the split and returns a solution Here the operations are numbered from 1 to n in seq2

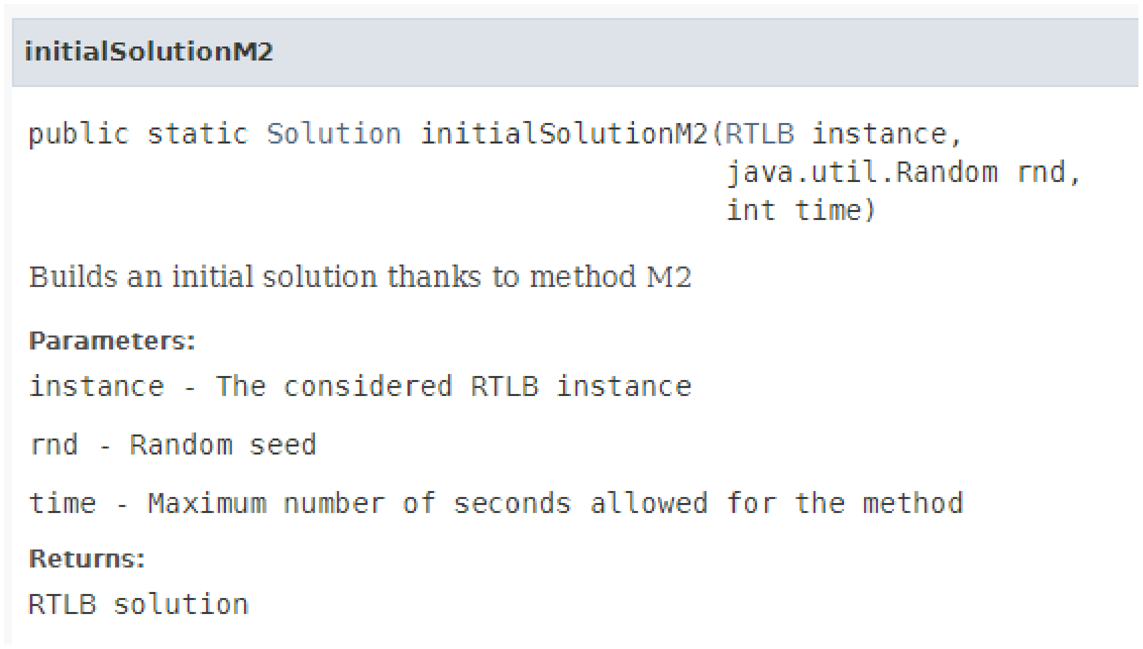
Parameters:

`seq2 - sequence to split`

Returns:

`Optimal split solution`

At lines 52-53, an initial solution is built thanks to method M2. The signature of the method initialSolutionM2 is as follows:



initialSolutionM2

```
public static Solution initialSolutionM2(RTLB instance,
                                         java.util.Random rnd,
                                         int time)
```

Builds an initial solution thanks to method M2

Parameters:

`instance - The considered RTLB instance`

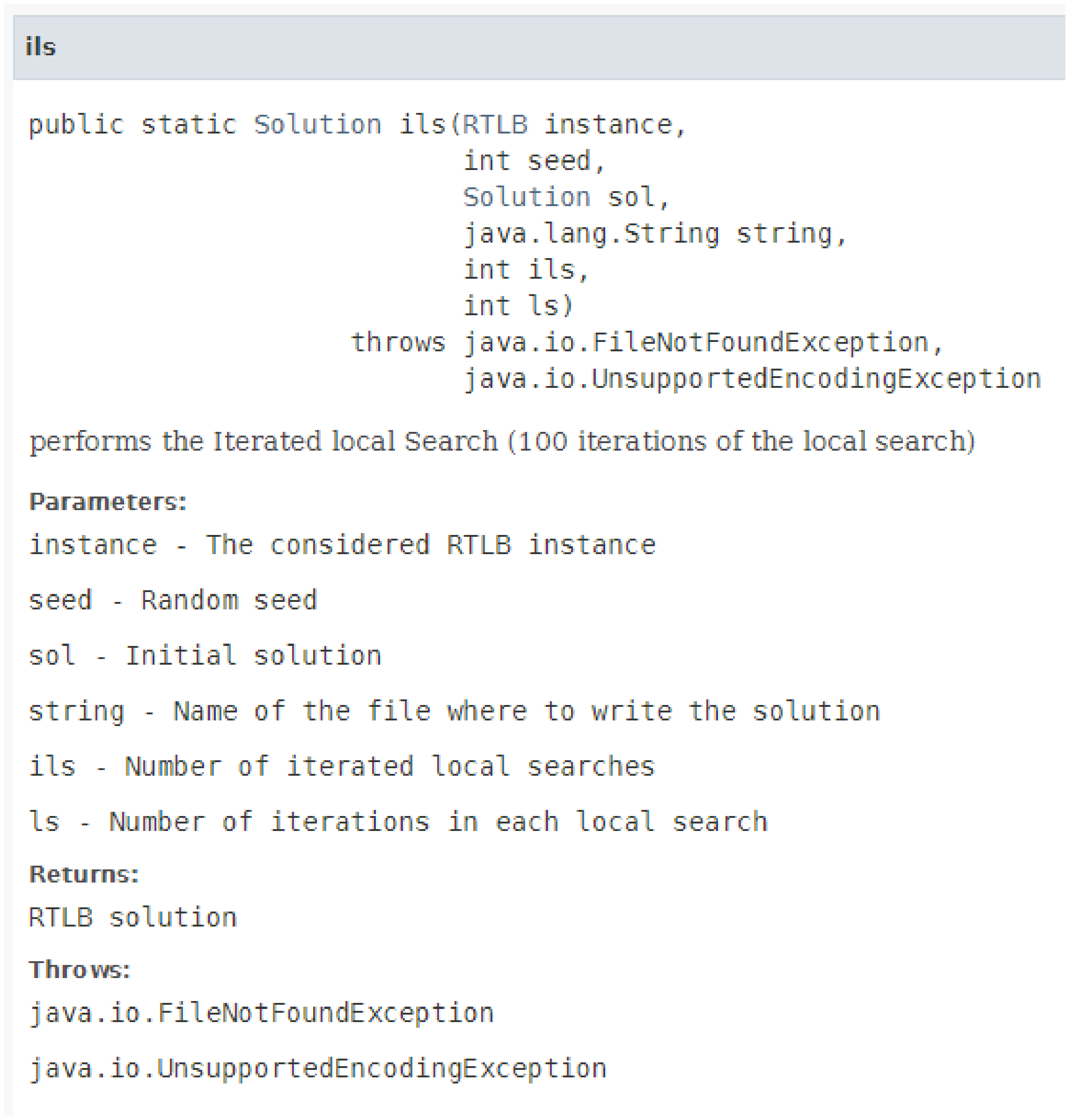
`rnd - Random seed`

`time - Maximum number of seconds allowed for the method`

Returns:

`RTLB solution`

A line 54, an ILS is performed. The signature of the function ils is as follows:



ils

```
public static Solution ils(RTLB instance,
                           int seed,
                           Solution sol,
                           java.lang.String string,
                           int ils,
                           int ls)
                    throws java.io.FileNotFoundException,
                           java.io.UnsupportedEncodingException
```

performs the Iterated local Search (100 iterations of the local search)

Parameters:

`instance` - The considered RTLB instance

`seed` - Random seed

`sol` - Initial solution

`string` - Name of the file where to write the solution

`ils` - Number of iterated local searches

`ls` - Number of iterations in each local search

Returns:

RTLB solution

Throws:

`java.io.FileNotFoundException`

`java.io.UnsupportedEncodingException`

The output file describes the solution and its cost. Its format is as follows:

Output file: result.txt

```
Initial solution feasible
Cost (Number of machines) of initial solution 7
Number of stations of initial solution 4
50 iterated local searches of 100 iterations each.
Cost (Number of machines) 7
Number of stations 4
Time 0.071704353

Workstation number 1 contains the following operations [1, 2, 3]
Workstation number 1 uses the following number of machines 3

Workstation number 2 contains the following operations [4]
Workstation number 2 uses the following number of machines 1

Workstation number 3 contains the following operations [5]
Workstation number 3 uses the following number of machines 1

Workstation number 4 contains the following operations [7, 6]
Workstation number 4 uses the following number of machines 2
```

How to retrieve the article results ?

To retrieve the results of the paper, change the two last arguments of ils according to the desired parameters and change seed from 0 to 9 (10 independent runs).