RAIRO Operations Research
RAIRO Oper. Res. **41** (2007) 193–211
DOI: 10.1051/ro:2007018

METAHEURISTICS BASED ON BIN PACKING FOR THE LINE BALANCING PROBLEM

Michel Gourgand[1], Nathalie Grangeon[2] and Sylvie Norre[2]

Abstract. The line balancing problem consits in assigning tasks to stations in order to respect precedence constraints and cycle time constraints. In this paper, the cycle time is fixed and the objective is to minimize the number of stations. We propose to use metaheuristics based on simulated annealing by exploiting the link between the line balancing problem and the bin packing problem. The principle of the method lies in the combination between a metaheuristic and a bin packing heuristic. Two representations of a solution and two neighboring systems are proposed and the methods are compared with results from the literature. They are better or similar to tabu search based algorithm.

Keywords. Flow-shop, stochastic, Markovian analysis, simulation, metaheuristic.

Mathematics Subject Classification. 90Bxx.

In the literature, most of the line balancing problems come from automotive industry and deal with vehicle assembly lines. In 1986, Baybars [4] proposes a classification of these problems and defines the basic problem called SALBP (Simple Assembly Line Balancing Problem). This problem consists in assigning tasks to stations in order to respect precedence constraints and cycle time constraint. According to the considered objective, two kinds of problem are defined:

SALBP1: the cycle time is fixed. The objective is to minimize the number of stations.

Received October 6, 2006. Accepted November 15, 2006.

[1] Université Blaise Pascal, LIMOS CNRS UMR 6158, ISIMA, BP 10125, 63173 Aubière Cedex, France; {gourgand,grangeon}@isima.fr

[2] Université Blaise Pascal, LIMOS CNRS UMR 6158, Antenne IUT de Montluçon, Avenue Aristide Briand, 03100 Montluçon, France; norre@moniut.univ-bpclermont.fr

SALBP2: the number of stations is fixed. The objective is to minimize the cycle time.

In this paper, we consider the SALBP1 problem. Few works concern the proposition of metaheuristics based on simulated annealing for the SALBP1. These works [6,16,18,19,26] consider classical neighboring systems (permutation, insertion). Obtained results are worse than other methods such as tabu search [9,27]. Our objective is to use metaheuristics based on simulated annealing by exploiting the link between the line balancing problem and the Bin Packing problem.

In the first part, we present our hypotheses and notations. In the second part, we give a state of the art for the SALBP1 and we detail the link with the Bin Packing problem. The third part presents the principle of the proposed methods and the neighboring systems. These metaheuristics are compared together and with methods from the literature in the fourth part.

1. Statement of the problem

We consider the SALBP1. The hypotheses are the following:

- the line parameters are known (configuration, length, ...);
- the tasks are linked by precedence constraints;
- a task is performed by a unique station;
- a station can perform any task;
- at a given time, a station can perform at most one task;
- the time to perform a task does not depend on the assigned station;
- all the tasks must be assigned.

The notations are the following:

V**:** set of tasks linked by a precedence graph. $|V| = n$. The two fictitious tasks: B and E are the first and last task of the graph.

c**:** cycle time,

t_j**:** time to perform task j, $j = 1, n$. $t_B = t_E = 0$,

m**:** number of used stations,

S_k**:** station load, set of tasks assigned to station k ,

$t(S_k)$**:** time of station k (sum of the duration of the tasks assigned to station k).

The problem consists in assigning tasks to stations in order to minimize the number of used stations. The assignment must verify the following constraints:

C1: precedence constraints between the tasks: if task j_1 precedes task j_2, then, either j_1 is assigned to a station before the station assigned to j_2, or j_1 and j_2 are assigned to the same station.

C2: cycle time constraint: the time of station k must be lower or equal to the cycle time:

$$t(S_k) = \sum_{j \in S_k} t_j \leq c, \forall k = 1, m.$$

TABLE 1. Duration of the tasks.

Task	1	2	3	4	5	6	7	8	9	10	11
Duration	4	38	45	12	10	8	12	10	2	10	34

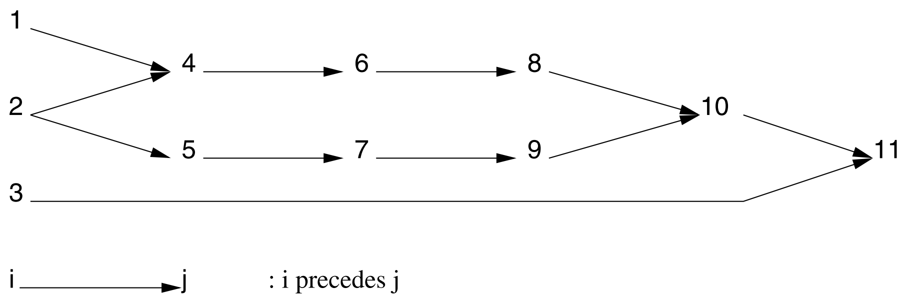

FIGURE 1. Precedence constraints between the tasks ("Mansoor" instance).

TABLE 2. An optimal solution.

Station	**1**	**2**	**3**
Tasks	2, 5, 7, 9	1, 3, 4	6, 8, 10, 11
$t(S_k)$	62	61	62

Example 1. Table 1 and Figure 1 show the "Mansoor" instance of SALBP1 with 11 tasks.

If $c = 62$, an optimal solution is composed of 3 stations. Table 2 presents an optimal assignment of the tasks to the stations.

2. State of the art and link with the Bin Packing

This state of the art is composed of two parts. In the first part, we consider the exact and approached methods proposed for the SALBP1. In the second part, we present the link between the line balancing problem and the Bin Packing problem.

2.1. State of the art

In the literature, a lot of papers deal with SALBP1. A recent state of the art can be found in [23]. Table 3 shows the number of references cited in [23] according to the proposed method: exact methods or approached methods.

Exact methods are the first studied methods. A lot of mathematical models such as [4] and [28] have been proposed. A detailed survey on such models is given in the chapter 2 of the book [24]. But [23] indicates that the resolution of such models by using classical methods is not a realistic choice to solve real instances. Other methods are branch and bound and dynamic programming.

TABLE 3. Papers cited in [23] according to the proposed solution method.

Exact methods	
Dynamic programming Branch and bound techniques	≈ 30 references
Approached methods	
Truncated branch and bound techniques Construction heuristics	≈ 30 references
Genetic algorithm	≈ 20 references
Tabu search algorithm	5 references
Simulated annealing based methods	3 references
Ant colonies	2 references

Concerning approached methods, two kinds of methods can be distinguished: heuristics and metaheuristics. A large variety of heuristic approaches has recently been proposed. States of the art for this kind of methods are given in [2, 8, 25], ... Most of the heuristics are greedy algorithms based on priority rules. The priority rules are computed according to the time to perform the tasks and the precedence relations. Two kinds of heuristics are distinguished: station-oriented heuristics and task-oriented heuristics. They differ by the manner in which the tasks are selected out of the set of available tasks.

- Station-oriented heuristics [23]: the heuristic starts with the first station ($k = 1$). The following stations are considered successively. In each iteration, a task with highest priority which is assignable to the current station k is selected and assigned. When station k is loaded maximally, it is closed, and the next station $k + 1$ is opened.
- Task-oriented heuristics [23]: Among available tasks, one with highest priority is chosen and assigned to the earliest station to which it is assignable.

Task oriented heuristics can be divided into "Immediate Update First" and "General First Fit" [29]. They depend on whether the set of available tasks is updated immediately after assigning a task or after assigning all currently available tasks. Experimental results, in [27], show that station-oriented heuristics obtain better results than task-oriented heuristics, but no theoretical dominance exists. Heuristics COMSOAL (COmputerized Method for Sequencing Tasks on Assembly Line) [3] and RPW (Ranked positional Weight) [15] are well known examples.

Concerning the metaheuristics, genetic algorithms are privileged, [1, 11, 13, 21, 22]. A study of these papers [5] lets us to conclude that it is difficult to implement the proposed methods because of the lack of details concerning the generation of the initial population, the admissibility of the generated children, ... [23] indicate that the results obtained by these methods are not significant because these methods are seldom compared with the methods of the literature, and in general not tested on known data sets and if they are, in fact the simplest instances are retained. Some papers are interested in the Tabu method [9, 17, 27] or

TABLE 4. Link between the line balancing and the Bin Packing.

Terms from the line balancing problem	Terms from the Bin Packing problem
Station	Bin
Task	Object
Cycle time	Size of bin
Duration of task	Size of object

in the simulated annealing based methods [16,18,19,26]. Two kinds of neighboring systems are proposed:

- insertion: a task i (assigned to station k_1) is randomly chosen. This task is assigned to a station k_2 ($k_2 \neq k_1$) randomly chosen among the set of stations which allows to respect the precedence constraints and the cycle time constraint;
- permutation: two tasks i_1 (assigned to station k_1) and i_2 (assigned to station k_2), not subjected to precedence constraints, are permuted such as the cycle time constraint is verified.

Scholl and Becker [23] cite two papers about the implementation of ant colonies [7], [20].

2.2. Link with the Bin Packing problem

Authors, like [13, 29] consider the parallel between the SALBP1 and the one dimensional Bin Packing problem. They show that the SALBP1 can be reduced to a Bin Packing by omitting the precedence constraints. The objects correspond to the tasks, the bins to the stations. The distribution of objects in bins is similar to the assignment of tasks to stations. In the Bin Packing problem, the sum of the size of the objects in the same bin can not be greater than the size of the bin. In the SALBP1, the sum of the duration of the tasks assigned to the same station must be lower or equal to the cycle time. The link between the terms of both problems is given by Table 4.

In the literature, a lot of methods allow to propose quickly a solution for the Bin Packing problem. The Next Fit heuristic considers the objects according to a sequence and assigns them to a current bin. If an object can not fit for the current bin, the bin is closed and a new bin is opened. The new bin becomes the current bin. In [10], we can see that this heuristic is linear in time and a performance guaranty is given by:

$$NF(L) \leq 2.OPT(L) - 1, \ \forall L$$

where

- L is an object sequence of any size to be placed in bins of the same size;
- $NF(L)$ is the number of bins obtained by the Next Fit heuristic;
- $OPT(L)$ is the optimal number of bins.

As a closed bin is never reopened, it is obvious that the efficiency of the heuristic lies in the input object sequence: the heuristic, applied to two different sequences,

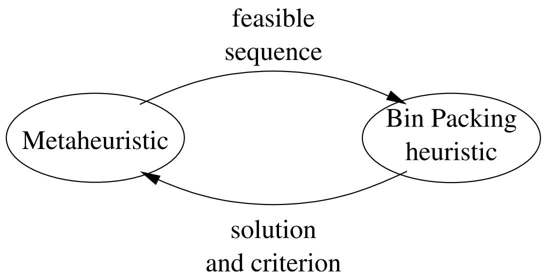

FIGURE 2. Combination between a metaheuristic and a Bin Packing heuristic.

may construct two different solutions in term of number of used bins or composition of bins. The proposed method for the line balancing problem lies in this remark.

An improvement of the Next Fit heuristic is the Best Fit heuristic. In this heuristic, the bins are never closed. All the bins are considered for insertion of an object. If no bin can be assigned to an object, then a new bin is created, else the object is assigned to the most filled bin that can accept it. The performance guaranty of the heuristic is:

$$BF(L) \leq \frac{17}{10}.OPT(L) + 2, \ \forall L$$

where $BF(L)$ is the number of bins obtained by the Best Fit heuristic.

In the literature, the heuristics Next Fit and Best Fit, dedicated to the Bin Packing problem have been adapted to the line balancing problem. The adaptation of these heuristics [13,24,29] consists in considering only the sequence of admissible tasks (all the predecessors of the tasks have been assigned to a station) instead of considering all the sequence of tasks. This list of admissible tasks changes during the heuristic.

To our knowledge, no metaheuristic which exploits this link with the Bin Packing problem exists for the line balancing problem. Few papers concern the proposition of metaheuristics for the SALBP1 because of the difficulty to build feasible neighbor solutions. The existing neighboring systems do not work well. In this paper, we propose to go further into the parallel between the SALBP1 and the Bin Packing problem.

3. Proposed method

The principle of the proposed method lies in the combination between a metaheuristic and a Bin Packing heuristic (Fig. 2). At each iteration, the metaheuristic provides to the Bin Packing heuristic a sequence which respects the precedence constraints. From this sequence, the Bin Packing heuristic builds a solution (assignment of the tasks to the stations) and computes corresponding performance criterion.

We need to propose:

- a representation of a sequence;
- a Bin Packing heuristic to build feasible solutions;
- a neighboring system that modifies the sequence at each iteration;
- one or more performance criteria to compare two solutions.

3.1. Representation of a sequence

A sequence is represented by a sequence of tasks:

$$\sigma = \{B, \sigma_1, \sigma_2 ..., \sigma_n, E\}$$

where σ_i is the task assigned to the ith position in the sequence. This sequence must respect the precedence constraints. An obvious initial sequence consists in sorting the tasks according to the decreasing number of successors.

Remark 1. For the instance described by Table 1 and Figure 1, the computation of the number of successors allows to build the sequence s_σ:

$$\sigma = \{B, 1, 2, 4, 5, 6, 7, 8, 9, 10, 3, 11, E\}$$

3.2. Heuristic

We propose to represent a solution by a list s_σ, composed of tasks and separators (|). A separator symbolizes a change of station. This sequence can be built either by the Next Fit heuristic (Algorithm 1), either by the Best Fit heuristic (Algorithm 2).

For example, $\{B, |, \sigma_1, \sigma_2, |, \sigma_3, \sigma_4, |, ..., \sigma_n, |, E\}$ means that tasks σ_1 and σ_2 are assigned to the first station, tasks σ_3 and σ_4 are assigned to the second station, ...

In the Next Fit heuristic, the tasks are considered according to the sequence σ. The task σ_i is assigned to the current station if the cycle time constraint [**C2**] is verified. Else, the current station is closed and a new station is created. This heuristic builds a feasible solution which respects the precedence constraints [**C1**] if the input sequence respects them and the cycle time constraint [**C2**].

Remark 2. For the problem described by Table 1 and Figure 1, the Algorithm 1 builds for σ, the solution:

$$s_\sigma = \{B, |, 1, 2, 4, |, 5, 6, 7, 8, 9, 10, |, 3, |, 11, |, E\}$$

This solution is composed of 4 stations.

In the Best Fit heuristic, the tasks are considered according to the sequence σ. All the stations are considered for insertion of a task. The task σ_i is assigned to the most filled station that allows the task σ_i to respect the precedence constraints: the first considered station for insertion is the station with the highest number that contains a predecessor of the task. If no station can be assigned to the task, then

Algorithm 1 Next Fit heuristic for the line balancing

Input: σ, a sequence of tasks
Output: s_σ, m, $(t(S_k), k = 1, m)$
$k := 1$
$t(S_k) := 0$
$s_\sigma := \{B, |\}$
for $i := 1$ to n **do**
  if $c - t(S_k) < t_{\sigma_i}$ **then**
    $k := k + 1$
    $s_\sigma := s_\sigma \cup \{|\}$
    $t(S_k) := 0$
  end if
  $s_\sigma := s_\sigma \cup \{\sigma_i\}$
  $t(S_k) := t(S_k) + t_{\sigma_i}$
end for
$s_\sigma := s_\sigma \cup \{|, E\}$
$m := k$

a new station is created. This heuristic builds a feasible solution which respects the precedence constraints [**C1**] if the input sequence respects them and the cycle time constraint [**C2**].

Remark 3. For the instance described by Table 1 and Figure 1, the Algorithm 2 builds for the initial sequence σ, the solution:

$$s_\sigma = \{B, |, 1, 2, 4, 6, |, 5, 7, 8, 9, 10, |, 3, |, 11, |, E\}$$

This solution is composed of 4 stations.

3.3. Metaheuristic

We consider the combination with metaheuristics based on simulated annealing: stochastic descent, Kangaroo algorithm [14] and Improved Solution Kangaroo Algorithm [12]. In this part, we present the combination between these metaheuristics and a Bin Packing heuristic.

3.3.1. *Stochastic descent*

The basic algorithm is the stochastic descent, which accepts the neighbor solution if its criterion is better or equal to the criterion of the current solution. This algorithm allows generally to find a local minimum. The principle algorithm is described by Algorithm 3.

The stop criterion is, for instance: maximum number of iterations reached, or optimal solution obtained.

Algorithm 2 Best Fit heuristic for the line balancing

Input: σ, a sequence of tasks
Output: s_σ, m, $(t(S_k), j = 1, K)$
$k := 1$
$t(S_k) := 0$
$t(S_0) := -1$ // fictitious station
$s_\sigma := \{B, |\}$
for $i := 1$ to n **do**
  $k_{best} := 0$
  $j_1 :=$ highest number of the station that contains a predecessor of σ_i in σ
  for $j := j_1$ to k **do**
    if $c - t(S_j) < t_{\sigma_i}$ and $t(S_j) > t(S_{k_{best}})$ **then**
      $k_{best} := j$
    end if
  end for
  if $k_{best} = 0$ **then**
    $k := k + 1$
    $t(S_k) := 0$
    $k_{best} := k$
    $s_\sigma := s_\sigma \cup \{|\}$
  end if
  Insert $\{, \sigma_i\}$ in s_σ such that σ_i is assigned to the station k_{best}
  $t(S_{k_{best}}) := t(S_{k_{best}}) + t_{\sigma_i}$
end for
$s_\sigma := s_\sigma \cup \{|, E\}$
$m := k$

3.3.2. *Kangaroo algorithm*

A simple way to leave a local minimum is to restart a stochastic descent from a new randomly chosen starting point. This scheme introduces the successive descents algorithm. The Kangaroo algorithm (Algorithm 4) follows this scheme but the new starting point is obtained by perturbing the local minimum. This algorithm allows, after a stochastic descent to accept any solution (by using another neighboring system) and to start again with a new stochastic descent.

Two neighboring systems are used, a first one $\mathcal{V}$ for the stochastic descent, and a second one $\mathcal{W}$, generally larger than $\mathcal{V}$, for the perturbation (called kangaroo jump). This algorithm has been proved to converge in probability [14] if neighboring system $\mathcal{W}$ satisfies the accessibility property. The proof of the convergence lies in the fact that the kangaroo algorithm builds a Markov chain where any state can lead to an absorbing state and the absorbing states constitute the global optimal set.

A is the maximum number of iterations without improvement. *best* corresponds to the best found solution. k is the number of iterations since the last improvement.

Algorithm 3 Combination between a stochastic descent and a Bin Packing heuristic

Input: stop criterion, a sequence σ of tasks
Output: a solution s_σ
Compute s_σ, by applying the Bin Packing heuristic with σ as input sequence.
while unsatisfied stop criterion **do**
Choose uniformly and randomly σ' in the neighboring system $\mathcal{V}$ of σ
Compute $s_{\sigma'}$, by applying the Bin Packing heuristic with σ' as input sequence
if $s_{\sigma'}$ is better than s_σ **then**
$\sigma := \sigma'$
$s_\sigma := s_{\sigma'}$
end if
end while
s_σ is the solution of the combination

Algorithm 4 Combination between the Kangaroo algorithm and a Bin Packing heuristic

Input: stop criterion, a sequence σ of tasks, $A > 0$
Output: a solution s_{best}
Compute s_σ, by applying the Bin Packing heuristic with σ as input sequence.
$k := 0$, $\sigma_{best} = \sigma$, $s_{best} := s_\sigma$
while necessary **do**
if $(k < A)$ **then**
Choose uniformly and randomly σ' in the neighboring system $\mathcal{V}$ of σ
Compute $s_{\sigma'}$, by applying the Bin Packing heuristic with σ' as input sequence
if $s_{\sigma'}$ is better or equal to s_σ **then**
if $s_{\sigma'}$ is better than s_σ **then**
k:= 0
if $s_{\sigma'}$ is better than s_{best} **then**
$\sigma_{best} := \sigma'$, $s_{best} := s_{\sigma'}$
end if
end if
$\sigma := \sigma'$, $s_\sigma := s_{\sigma'}$
end if
$k := k + 1$
else
Choose uniformly and randomly σ' in the neighboring system $\mathcal{W}$ of σ
Compute $s_{\sigma'}$, by applying the Bin Packing heuristic with σ' as input sequence
$k := 0$
if $s_{\sigma'}$ is better than s_{best} **then**
$\sigma_{best} := \sigma'$, $s_{best} := s_{\sigma'}$
end if
$\sigma := \sigma'$, $s_\sigma := s_{\sigma'}$
end if
end while
s_{best} is the solution of the combination

3.3.3. *Improved Solution Kangaroo Algorithm*

This method, also called ISKA is an improvement of the Kangaroo Algorithm. The improvement consists in starting the new stochastic descent from the best found solution. The line 20 in Algorithm 4 becomes:

20: Choose uniformly and randomly σ' in the neighboring system $\mathcal{W}$ of σ_{best}

3.3.4. *Proposed neighboring systems*

In her thesis, Boutevin [6] interested in the proposition of a metaheuristic based on simulated annealing. Our objective is to improve the obtained results by the proposition of a better neighboring system than the existing neighboring systems [6]. One of the problems was the difficulty to compute a feasible neighbor solution because of the constraints.

3.3.4.1 Classical neighboring system

A classical neighboring system consists in choosing randomly a task and inserting it at a new position in the sequence. The new position is randomly chosen among the positions which respect the precedence constraints [**C1**]. The Algorithm 5 is the classical proposed neighboring system: the moved task is chosen among the set of tasks that can be moved (step 4 and 8), according to the precedence constraints.

Algorithm 5 First proposed neighboring system

1: **Input**: σ, a sequence of tasks
2: **Output**: σ', a sequence of task, neighbor of σ
3: $\sigma' := \sigma$
4: **repeat**
5:     Choose randomly a task σ_i
6:     Compute i_1, the position of the nearest predecessor of σ_i in σ
7:     Compute i_2, the position of the nearest successor of σ_i in σ
8: **until** $i_1 + 1 \neq i_2 - 1$
9: Choose randomly a new position i', $i' \in [i_1 + 1, i_2]$, $i' \neq i$
10: Insert σ_i into position i' in σ'

Remark 4. In the sequence $\sigma = \{B, 1, 2, 4, 5, 6, 7, 8, 9, 10, 3, 11, E\}$ built for the example (Tab. 1 and Fig. 1),

- task 3 can be inserted in a position between positions 1 to 9 (B is in position 0).
  For example, the insertion in σ, of task 3 in position 4 leads to the neighbor solution:
  $$\sigma' = \{B, 1, 2, 4, 3, 5, 6, 7, 8, 9, 10, 11, E\}$$
  and the Algorithm 1 provides the solution:
  $$s_{\sigma'} = \{B, |, 1, 2, 4, |, 3, 5, |, 6, 7, 8, 9, 10, |, 11, |, E\}$$

This solution is different from the solution in the previous remark, but the number of stations is identical.

- task 7 can be inserted in position 5 or 7.
  The insertion in σ, of task 7 in position 5 leads to the neighbor solution:

$$\sigma'' = \{B, 1, 2, 4, 5, 7, 6, 8, 9, 10, 3, 11, E\}$$

and the Algorithm 1 provides the solution:

$$s_{\sigma''} = \{B, |, 1, 2, 4, |, 5, 7, 6, 8, 9, 10, |, 3, |, 11, |, E\}$$

This solution is identical to the solution in the previous remark. The task 7 is assigned to the same station.

3.3.4.2 Improved neighboring system

The classical neighboring system may construct sequences that lead to the same solutions after applying the Bin Packing heuristic. An improved neighboring system, which allows to reduce the number of identical solutions, consists in moving the task to another station. This can be easily done in the case of the combination with the Next Fit heuristic: the new position of a task depends on the position of the predecessors and the successors, and on the position of the separators which correspond to the station assigned to the task. The new position is chosen such that the task will be assigned to another station. The Algorithm 6 describes this neighboring system.

Remark 5. In the solution $s_\sigma = \{B, |, 1, 2, 4, |, 5, 6, 7, 8, 9, 10, |, 3, |, 11, |, E\}$,

- task 3 can be inserted between positions 1 and 9 in σ.
  For example, the insertion in σ of task 3 in position 4 leads to the neighbor solution:

$$\sigma' = \{B, 1, 2, 4, 3, 5, 6, 7, 8, 9, 10, 11, E\}$$

and the Algorithm 1 provides the solution:

$$s_{\sigma'} = \{B, |, 1, 2, 4, |, 3, 5, |, 6, 7, 8, 9, 10, |, 11, |, E\}$$

- task 7 can not be moved.
- task 10 can be inserted in position 11 in σ. The insertion in σ of task 10 in position 11 leads to the neighbor solution:

$$\sigma'' = \{B, 1, 2, 4, 5, 6, 7, 8, 9, 3, 10, 11, E\}$$

and the Algorithm 1 provides the solution:

$$s_{\sigma''} = \{B, |, 1, 2, 4, |, 5, 6, 7, 8, 9, |, 3, 10, |, 11, |, E\}$$

The solutions $s_{\sigma'}$ and $s_{\sigma''}$ are different from the initial solution, but the number of stations is identical.

Algorithm 6 Second neighboring system

Input: σ, a sequence of tasks, s_σ the corresponding solution
Output: σ', a sequence of tasks, neighbor of σ
repeat
  Choose randomly a task σ_i
  Compute i_1, the position of the nearest predecessor of σ_i in σ
  Compute i_2, the position of the nearest successor of σ_i in σ
  Compute $i_3 < i$, the position of the nearest separator in s_σ
  Compute $i_4 > i$, the position of the nearest separator in s_σ
until $i_1 + 1 \leq i_3$ or $i_4 + 2 \leq i_2$
if $i_1 + 1 \leq i_3$ and $i_4 + 2 \leq i_2$ **then**
  Choose a new position i', $i' \in [i_1 + 1; i_3 + 1] \cup [i_4 + 2; i_2]$, $/s_{\sigma_{i'}} \neq |$
else
  if $i_1 + 1 \leq i_3$ **then**
    Choose a new position i', $i' \in [i_1 + 1; i_3 + 1]$, $/s_{\sigma_{i'}} \neq |$
  else
    Choose a new position i', $i' \in [i_4 + 2; i_2]$, $/s_{\sigma_{i'}} \neq |$
  end if
end if
Insert σ_i into position i' in s_σ
$\sigma' := s_\sigma - \{|\}$

3.4. Performance criteria

The objective is the minimization of the number of used stations (m). However, a lot of solutions have the same number of used stations. So, we proposed to use finest criteria. By finest criteria, we mean criteria which allow to detect promising solutions, *i.e.* solutions in which some stations are a little loaded and some stations nearly loaded rather than solutions in which stations are all half loaded [13]. It will be easier to empty the little loaded stations by loading the nearly loaded stations.

We propose to use the following criteria:

- ratio of the load of the most loaded station on the load of the least loaded station:

$$f_1 = \max_{k=1,m} \{t(S_k)\} / \min_{k=1,m} \{t(S_k)\}$$

  The objective is to maximize f_1.
- criterion proposed by [13] for the Bin Packing problem:

$$f_2 = \frac{1}{m} \sum_{k=1}^{m} \left(\frac{t(S_k)}{c} \right)^2.$$

  The objective is to maximize f_2.

These criteria allow to compare two sequences σ and σ': sequence σ' is better than sequence σ if:

Version 1 $m(\sigma') \leq m(\sigma)$
Version 2 $m(\sigma') < m(\sigma)$ or $(m(\sigma') = m(\sigma)$ and $f_1(\sigma') >= f_1(\sigma))$
Version 3 $f_2(\sigma') \geq f_2(\sigma)$

Criteria are computed by using either Next Fit heuristic (Algorithm 1), either Best Fit heuristic (Algorithm 2) according to the combination described by Figure 2. $m(\sigma)$ (resp. $m(\sigma')$) is the number of stations for sequence σ (resp. σ').

4. Computational experiments

We have tested the following scenarios: (H, M, N, C) where:

- Heuristic (H): Algorithm 1, Algorithm 2;
- Metaheuristic (M): stochastic descent (algo 3), Kangaroo algorithm (algo 4), ISKA;
- Neighboring system (N): classical (algo 5), improved (algo 6);
- Criterion (C): version 1, version 2, version 3.

The data sets are the 269 data sets from the assembly line balancing library: `http://www.assembly-line-balancing.de/`.

In the following, the term "method" refers to a scenario (H, M, N, C). As metaheuristics are stochastic algorithms, we have run $Nb_{rep} = 10$ replications of each method with the following parameters: 1000000 iterations and 20000 iterations before a jump for KA and ISKA.

Chiang [9] describes the application of tabu search on $Nb_{instances} = 64$ data sets chosen among the 269 data sets. Table 5 presents a comparison between our results for the $Nb_{instances} = 64$ data sets and results obtained by [9]. Four different versions of the tabu search are developed: best improvement with task aggregation, best improvement without task aggregation, first improvement with task aggregation, and first improvement without task aggregation.

For each instance $j, j = 1, Nb_{instances}$ and each method, we define:

$m_{i,j}$: the number of stations obtained by replication i, $i = 1, Nb_{rep}$ for instance j;
$m_{opt,j}$: the optimal number of stations for instance j;
$m_{min,j}$: the smallest number of station obtained for instance j

$$m_{min,j} = \min_{i=1}^{Nb_{rep}} m_{i,j};$$

rel_j: the standard deviation between $m_{min,j}$ and $m_{opt,j}$ for instance j

$$rel_j = 100 * (m_{min,j} - m_{opt,j})/m_{opt,j}.$$

For each method and each version of the performance criterion, we give:

#opt: the number of instance where the optimal solution is obtained at least one time by the 10 replications (*i.e.* $m_{min,j} = m_{opt,j}$).

#opt10: the number of instance where the optimal solution is obtained by the 10 replications (*i.e.* $m_{i,j} = m_{opt,j}, i = 1, Nb_{rep}$);

avg.rel: the average standard deviation from the optimal solution.

$$avg.rel = 1/Nb_{instances} \sum_{j=1}^{Nb_{instances}} rel_j;$$

max.rel: the maximum standard deviation from the optimal solution.

$$max.rel = \max_{j=1,Nb_{instances}} rel_j.$$

We obtain better results than [9]. The best combination is the combination with Best Fit heuristic and classical neighboring system: the optimal solution is obtained at each replication.

Table 6 presents results for the proposed combination for all the $Nb_{instances} = 269$.

In the majority of the cases, ISKA and the Kangaroo algorithm obtain the best results, except for versions 1 and 2 of the performance criteria with the classical neighboring system. However, ISKA and the kangaroo algorithm are the most robust methods: on the 10 replications, they more often obtain an optimal solution.

For versions 1 and 2 of the performance criteria, the results obtained with the improved neighboring system are worse than classical neighboring system, not only in term of number of optimal solutions obtained, but also in term of standard deviation. Only version 3 of the performance criteria obtains slightly better results with the improved neighboring system. The improved neighboring system does not bring anything concerning the quality of the obtained solution. A study of the behavior of the performance criterion during the metaheuristic shows that this neighboring system only accelerates the convergence of the method during the first 10000 iterations.

Version 1 of the performance criterion is the most naive version and obtains worse results. Indeed, a great number of sequences have the same criterion, if only this criterion is considered. Versions 2 and 3 of the performance criterion allow to order the sequences by privileging the sequences which lead to a greater disparity in the occupation of the stations.

Whatever the used neighboring system, version 3 of the performance criterion allows to obtain the greatest number of optimal solutions. However, it is the version 2 which allows to have a more robust behavior: among the 10 replications, the optimal solution is more often obtained (203 optimal solutions are obtained by the 10 replications of Kangaroo algorithm).

Table 5. Comparison with the results obtained by [9] (64 data sets among the 269 ones).

Scenario with Next Fit heuristic and classical neighboring system									
Criterion	Version 1			Version 2			Version 3		
Metaheuristic	SD	K	ISKA	SD	K	ISKA	SD	K	ISKA
#opt	59	59	59	60	60	60	60	60	60
Percentage of optimality	92,19	92,19	92,19	93,75	93,75	93,75	93,75	93,75	93,75
#opt10	56	56	56	59	59	59	51	57	57
Percentage of optimality	87,5	87,5	87,5	92,19	92,19	92,19	79,69	89,06	89,06
Scenario with Next Fit heuristic and improved neighboring system									
Criterion	Version 1			Version 2			Version 3		
Metaheuristic	SD	K	ISKA	SD	K	ISKA	SD	K	ISKA
#opt	63	63	63	60	60	60	58	60	60
Percentage of optimality	98,44	98,44	98,44	93,75	93,75	93,75	90,63	93,75	93,75
#opt10	60	60	61	59	59	59	44	55	55
Percentage of optimality	93,75	93,75	95,31	92,19	92,19	92,19	68,75	85,94	85,94
Scenario with Best Fit heuristic and classical neighboring system									
Criterion	Version 1			Version 2			Version 3		
Metaheuristic	SD	K	ISKA	SD	K	ISKA	SD	K	ISKA
#opt	**64**	**64**	**64**	**64**	**64**	**64**	**64**	**64**	**64**
Percentage of optimality	100	100	100	100	100	100	100	100	100
#opt10	**64**	**64**	**64**	**64**	**64**	**64**	**64**	**64**	**64**
Percentage of optimality	100	100	100	100	100	100	100	100	100

Tabu search from [9]				
Metaheuristic	Best improvement with task aggregation	Best improvement without task aggregation	First improvement with task aggregation	First improvement without task aggregation
#opt	51	62	51	62
Percentage of optimality	79.7	96.9	79.7	96.9

Table 6 presents also some results obtained by [25] for the same instances. In this paper, the proposed methods are two tabu methods, called PrioTabu and EurTabu. The difference between the two methods is the heuristic to build the initial solution. For each method, we give #opt, avg.rel. and max.rel. previously defined.

We obtain better results than [25] in term of number of optimal solutions.

If we consider all the scenario (H, M, N, C), we obtain 241 optimal solution for the 269 data sets. The 28 instances for which we do not obtain the optimal solution are described by Table 7.

Table 6. Comparison with the results obtained by [25] (269 data sets).

Scenario with Next Fit heuristic and classical neighboring system									
Criterion	Version 1			Version 2			Version 3		
Metaheuristic	SD	K	ISKA	SD	K	ISKA	SD	K	ISKA
#opt	195	192	193	212	211	207	218	223	224
#opt10	158	160	158	182	189	190	132	171	170
avg.rel	1.37	1.34	1.45	1.00	0.97	1.06	1.12	0.97	0.95
max.rel	33.33	33.33	33.33	33.33	33.33	33.33	33.33	33.33	33.33
Scenario with Next Fit heuristic and improved neighboring system									
Criterion	Version 1			Version 2			Version 3		
Metaheuristic	SD	K	ISKA	SD	K	ISKA	SD	K	ISKA
#opt	181	182	182	204	208	211	215	225	225
#opt10	164	167	165	181	187	189	114	164	167
avg.rel	1.69	1.62	1.45	1.10	1.03	0.94	1.31	0.90	0.94
max.rel	14.29	14.29	14.29	33.33	33.33	33.33	33.33	33.33	33.33
Scenario with Best Fit heuristic and classical neighboring system									
Criterion	Version 1			Version 2			Version 3		
Metaheuristic	SD	K	ISKA	SD	K	ISKA	SD	K	ISKA
#opt	198	196	198	219	216	213	217	221	224
#opt10	186	187	186	200	203	202	176	195	197
avg.rel	1,13	1,14	1,13	0,68	0,72	0,76	0,77	0,67	0,62
max.rel	14,29	14,29	14,29	14,29	14,29	14,29	14,29	14,29	14,29

Tabu search and truncated branch and bound from [25]				
Method	Prio Tabu		Eur Tabu	
Parameters	L=50	L=500	L=50	L=500
#opt	191	200	212	214
avg.rel	1.14	0.86	0.67	0.63
max.rel	14.29	7.69	7.69	7.69

Table 7. Instances for which we do not obtain the optimal solution.

Name of the instance	Cycle length
ARC111	11570
barthol2	85 121 146
lutz3	110
scholl	1394 1452 1483 1515 1584 1659 1742 1787 1834 1883 1935 1991 2049 2111 2177 2247 2322 2402 2488 2580 2680 2787
tonge70	151

5. Conclusion

In this paper, we have studied the SALBP1, a classical theoretical line balancing problem. To solve this problem, we have proposed a combination between metaheuristic and heuristic by exploiting the link between the line balancing problem

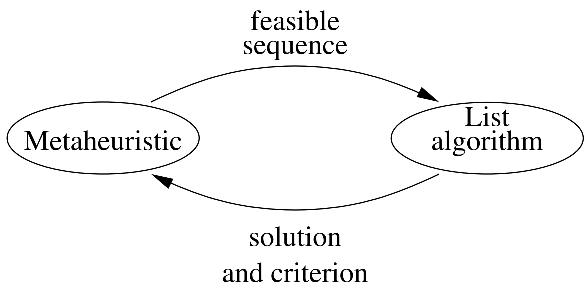

Figure 3. Combination between a metaheuristic and a list algorithm.

and the Bin Packing problem. To implement this combination, we have proposed two neighboring systems, an adaptation of the Next Fit and the Best Fit heuristics and performance criteria. The proposed methods have been tested on instances from the literature. Our results are better than the results from [9] and similar to the results from [25].

The principle of the proposed method can be generalized to other problems with constraints like precedence constraints. Figure 3 presents this generalization.

Our further works concern an industrial line balancing problem. We plan to take into account more constraints such as incompatibilities between operations, compulsory assignment, ...

References

[1] E.J. Anderson and M.C. Ferris, Genetic algorithms for combinatorial optimization: The assembly line balancing problem. *ORSA J. Comput.* **6** (1994) 161–174.

[2] M. Amen, Heuristic method for cost-oriented assembly line balancing: A survey. *Inter. J. Prod. Econ.* **68** (2000) 1–14.

[3] A.L. Arcus, Comsoal a computer method of sequencing operations for assembly lines. *Inter. J. Prod. Res.* **4** (1966) 259–277.

[4] I. Baybars, A survey of exact algorithms for the simple assembly line balancing problem. *Manage. Sci.* **32** (1986) 909–932.

[5] C. Boutevin, L. Deroussi, M. Gourgand and S. Norre, *Supply chain Optimisation*, chapter Hybrid methods for line balancing problems, edited by A. Dolgui, J. Soldek, O. Zaikin, Kluwer Academic Publishers (2004).

[6] C. Boutevin, *Problème d'Ordonnancement et d'Affectation avec Contraintes de Ressources de type RCPSP et Line Balancing.* Ph.D. thesis, Université Blaise Pascal, Clermont-Ferrand (2003).

[7] J. Bautista and J. Pereira, Ant algorithms for assembly line balancing. In Berlin Springer, editor, *Ant algorithms, Third International Workshop, ANTS 2002 Proceedings*, *Bruxelles (Belgique)*, edited by M. Dorigo, G. Di Caro, M. Sampels *Lect. Notes Comput. Sci.* **2463** (2002) 65–75.

[8] T.K. Bhattacharjee and S. Sahu, *A critique of some current assembly line balancing techni.* Technical report, Indian Institute of Technology, Kharagpur, India (1987).

[9] W.C. Chiang, The application of a tabu search metaheuristic to the assembly line balancing problem. *Ann. Oper. Res.* **77** (1998) 209–227.

[10] E.G. Coffman Jr., M.R. Garey and D.S. Johnson, Approximation algorithms for bin packing: A survey, in *Approximation Algorithms for NP-Hard Problems*, edited by Dorit S. Hochbaum (1997) 46–93.

[11] A.L. Corcoran and R.L. Wainwright, Using libga to develop genetic algorithms for solving combinatorial optimization problems. *Appl. Handbook Genetic Algorithms* **1** (1995) 144–172.

[12] L. Deroussi, *Heuristiques, métaheuristiques et systèmes de voisinage.* Ph.D. thesis, Université Blaise Pascal, Clermont-Ferrand II (2002).

[13] E. Falkenauer and A. Delchambre, A genetic algorithm for bin packing and line balancing, in *Proceedings of IEEE International Conference on Robotics and Automation (ICRA92), Los Alamitos, Californie* (1992) 1186–1192.

[14] G. Fleury, *Méthodes stochastiques et déterministes pour les problèmes NP-difficiles.* Ph.D. thesis, Université Blaise Pascal, Clermont-Ferrand II (1993).

[15] W.B. Helgeson and D.P. Birnie, Assembly line balancing using the rank positional weight technique. *J. Ind. Eng.* **12** (1961) 394–398.

[16] A. Heinrici, A comparison between simulated annealing and tabu search with an example for the production planning, in *Operations Research Proceedings, Amsterdam, 1993*, edited by Dyckhoff *et al.* Springer, Berlin (1994) 498–503.

[17] S.D. Lapierre, A. Ruiz and P. Soriano, Balancing assembly lines with tabu search. *Eur. J. Oper. Res.* (2004) in press.

[18] P.R. McMullen and G.V. Frazier, Using simulated annealing to solve a multiobjective assembly line balancing problem with parallel workstations. *Inter. J. Prod. Res.* **36** (1998) 2717–2741.

[19] V. Minzu and J.M. Henrioud, Stochastic algorithm for task assignment in single or mixed-model assembly lines. *APII-JESA* **32** (1998) 831–851.

[20] P.R. McMullen and P. Tarasewich, Using ant techniques to solve the assembly line balancing problem. *IEE Transactions* **35** (2003) 605–617.

[21] B. Rekiek, *Assembly Line Design: Multiple Objective Grouping Genetic Algorithm and the Balancing of Mixed-model Hybrid Assembly Line.* Ph.D. thesis, Université Libre de Bruxelles (2000).

[22] J. Rubinovitz and G. Levitin, Genetic algorithm for assembly line balancing. *Inter. J. Prod. Econ.* **41** (1995) 444–454.

[23] A. Scholl and C. Becker, State-of-the-art exact and heuristic solution procedures for simple assembly line balancing. *European Journal of Operational Research, special issue on Balancing of Automated Assembly and Transfer Lines*, edited by A. Dolgui **168** (2006) 666–693.

[24] A. Scholl, *Balancing and Sequencing of Assembly Lines.* Physica-Verlag Heidelberg, New-York (1999).

[25] A. Scholl and R. Klein, Balancing assembly lines effectively – a computational comparison. *Eur. J. Oper. Res.* **114** (1999) 50–58.

[26] G. Suresh and S. Sahu, Stochastic assembly line balancing using simulated annealing. *J. Production Res.* **32** (1994) 1801–1810.

[27] A. Scholl and S. Voss, Simple assembly line balancing – heuristic approaches. *J. Heuristics* **2** (1996) 217–244.

[28] F.B. Talbot and J.H. Patterson, An integer programming algorithms with network cuts for solving the single model assembly line balancing problem. *Manage. Sci.* **32** (1984) 85–99.

[29] T.S. Wee and M.J. Magazine, Assembly line as generalized bin packing. *Oper. Res. Lett.* **1** (1982) 56–58.