Management and Production Engineering Review
Volume 3 • Number 1 • March 2012 • pp. 4–17
DOI: 10.2478/v10270-012-0001-5

MULTI-OBJECTIVE APPROACH FOR PRODUCTION LINE EQUIPMENT SELECTION

H. Chehade[1], A. Dolgui[2], F. Dugardin[1], L. Makdessian[3], F. Yalaoui[1]

[1] *Université de Technologie de Troyes, Institut Charles Delaunay, France*
[2] *Ecole des Mines de Saint-Etienne, Centre for Industrial Engineering and Computer Science, France*
[3] *University of Tichrine, Faculty of Mechanical and Electrical Engineering, Syria*

Corresponding author:
Hicham Chehade
Institut Charles Delaunay, UMR CNRS 6279 STMR
Laboratoire d'Optimisation des Systèmes Industriels (LOSI)
Université de Technologie de Troyes
12, rue Marie Curie, BP2006 – 10010 Troyes cedex, France
phone: 0033 3 25 71 84 55
e-mail: Hicham.chehade@utt.fr

Received: 2 November 2011
Accepted: 10 January 2012

Abstract
A novel problem dealing with design of reconfigurable automated machining lines is considered. Such lines are composed of workstations disposed sequentially. Each workstation needs the most suitable equipment. Each available piece of equipment is characterized by its cost, can perform a set of operations and requires skills of a given level for its maintenance. A multi-objective approach is proposed to assign tasks, choose and allocate pieces of equipment to workstations taking into account all the problem parameters and constraints. The techniques developed are based on a genetic algorithm of type NSGA-II. The NSGA-II suggested is also combined with a local search. These two genetic algorithms (with and without local search) are tested for several line examples and for two versions of the considered problem: bi-objective and four-objective cases. The results of numerical tests are reported. What is the most interesting is that the assessment of these algorithms is accomplished by using three measuring criteria: the direct measures of gaps, the measures proposed by Zitzler and Thiele in 1999 and the distances suggested by Riise in 2002.

Keywords
production line design, line balancing, equipment selection, multi-objective optimization.

Introduction

The design and analysis of production systems has been broadly discussed in the literature [1–6]. Several standard scientific problems have been formulated such as the optimal process planning, facility layout, line balancing, buffer allocation, equipment selection, etc.

This paper deals with line balancing and equipment selection while designing machining lines. Note that we consider a relatively general case where:

– Each piece of equipment can be used to accomplish not only one but a set of different technological tasks.

– At the line design stage, for each task there is a set of various types of equipment each of which can be used to execute the task, one piece of equipment must be selected from this set.

In a previous works, we have already studied models and algorithms for combinatorial optimization of machining lines with a single criterion and several criteria. In this paper, we focus on the line balancing and the equipment selection problems for a special type of such lines: reconfigurable automated machining lines where for each available piece of equipment we know the maximal set of tasks that can be executed with the equipment, but in each design decision we use only a subset of tasks. We propose some adaptations and improvements for previously suggested multi-objective algorithms and study their effectiveness via a series of numerical tests. Furthermore, to compare the performances of these algo-

rithms different measuring techniques will be used to provide a broader perspective.

When a new machining line is designed (or an existing line is reconfigured for a new product) a corresponding line balancing and equipment selection problem has to be solved. Because of precedence constraints between tasks and the need to choose equipment for each workstation, we have to take into account these constraints as well as others related to equipment compatibility. This leads to a very complex combinatorial optimisation problem. Such a line is designed (or reconfigured) for manufacturing a given product. At the beginning of any design or redesign, all tasks required for manufacturing the product should be known. Then, the problem is to define the workstations, i.e. to assign tasks and pieces of equipment to workstations such that a criterion is (or several criteria are) optimized.

The rest of the paper is organized as follows. Section 2 presents an analysis of the state of the art in this domain. In Sec. 3, the problem statement is given and a Pareto optimization model is developed for the considered machining lines. The optimization algorithms are briefly explained in Sec. 4. Tests and comparisons of algorithms are presented in Sec. 5. Some discussions on possible extensions of the proposed approach are given in Sec. 6. Finally, conclusions are reported in Sec. 7.

State of the art and motivation

In literature, a similar simplified problem is known as assembly line balancing (ALB). The Simple Assembly Line Balancing problem (SALBP) deals with grouping tasks (non-divisible work elements) into workstations taking into account precedence relations between tasks and a constraint on line cycle time or number of workstations. The tasks are executed at each workstation sequentially. The cycle time of the assembly line is determined by the workstation with the maximum workload. Two principal types of SALBP are studied: SALBP-1 attempts to minimize the number of workstations for a required cycle time T_0, while SALBP-2 attempts to minimize the cycle time for a given number w of workstations. Comprehensive surveys on these problems and they generalisations were published, for example, in [7–13]. However, actual industrial problems are usually more complex. Often, the assignment of tasks to workstations requires equipment selection for each station to do the required job efficiently. This should consider also equipment compatibility constraints, ability of equipment to execute the considered tasks, etc. In this case, we have a more complex combinatorial problem than SALBP which is often called the *line balancing and equipment selection problem.*

Recently, several new generalisations of ALB, named transfer line balancing problems (TLBP), were proposed for mass production machining lines. Lines with sequential activation of equipment at each station were studied in [14, 15]. Lines with parallel activation of pieces of equipment at each station were considered in [16]. Lines with mixed equipment activation at each station were addressed in [17], etc. Some specificities of such machining lines are: *i*) the tasks of the same equipment (multi-spindle head) are executed in parallel, i.e. simultaneously, so the equipment working time is equal to the maximum of its task times; *ii*) if equipment is selected for a line design, all the tasks of this equipment will be executed here (we cannot execute only a sub-set of equipment tasks).

Thus, the literature is awash with different ALB models for standard manual assembly lines with sequential execution of tasks. We have also some experience to apply ALB approaches for mass production machining lines with parallel execution of tasks at workstations where both line balancing and equipment selection problems are examined. Nevertheless, it can be concluded that, often, when considering problems of assembly and/or machining line balancing and equipment selection, only scalar optimization techniques are developed. They optimize only one of the following criteria: equipment cost (investments), occupied area, workstation loads, etc., see for example [18–20]. Again, in real life industrial situations, the problems are usually far more complex, because there are several conflicting criteria, and all of them should be considered simultaneously.

Therefore, the *motivation of this work* is to suggest an approach of multi-objective line balancing and equipment selection for reconfigurable automated machining lines. Lines where all pieces of equipment of each workstation are activated simultaneously will be considered. However, in contrast with our previous work dedicated to design of mass production machining lines, this paper deals with the case where only a sub-set of tasks of each piece of equipment can be used and not necessarily all the tasks. This gives more flexibility for design decisions, simplifies future modifications and is a property of reconfigurable machining lines.

In this study, more than one objective function will be treated. In literature, multi-objective problems are often reduced to a corresponding single criterion optimization problem via a weighted sum of initial criteria [21]. A major drawback of this approach lies in the difficulty to obtain the required

weights for the considered criteria. Moreover, this technique gives only one solution. Usually, decision makers prefer a set of acceptable solutions to apply instead of a single option. Therefore, the goal of this paper is to develop a Pareto[1] optimization approach. This approach provides a set of solutions giving decision makers some leeway in their decisions.

A Pareto multi-objective optimization problem can be defined as follows [22]:

Find vectors of decision variables $X = [x_1,\ x_2, \ldots, x_z]^T$ that satisfy

Q inequality constraints:

$$g_i(X) \geq 0, \qquad i = 1, \ldots, Q, \tag{1}$$

P equality constraints:

$$h_i(X) = 0, \qquad i = 1, \ldots, P \tag{2}$$

and optimize (minimize or maximize) a vector of objective functions:

$$F(X) = [f_1(X), f_2(X), \ldots, f_r(X)]^T. \tag{3}$$

In other words, the result is a particular set of vectors $\{ X^* = [x_1^*,\ x_2^*, \ldots, x_z^*] \}$ which provides a compromise for all the objectives (1) among the set of all points in the space of decision variables that satisfy the constraints (1) and (2). Thus, a Pareto-optimal set $\{ X^* \}$ is composed of only feasible solutions, and called the set of non-dominated solutions [23]. None of these solutions (vectors of decision variables) can be considered as better than an other one from this set [24].

In this paper, we have developed multi-objective genetic algorithms based on the property of Pareto optimality. Fonseca and Fleming [25] presented three techniques for this type of multi-objective evolutionary algorithms. In the first, the fitness is determined via an aggregation of all objectives for the solution (linear sum of the criteria). The second works on different populations at the same time, each population is associated with one criterion. The third is based on Pareto ranking of a single population (Niche Ranking Techniques). Sarker [26] present another genetic algorithm based on the Pareto optimality. A review on genetic techniques used for solving multi-objective problems can be found in the paper of Coello [27]. Clearly, the increasingly used and arguably the most powerful are the following algorithms: NPGA of Horn [28], NSGA of Srinivas and Deb [24], and SPEA of Zitzler and Thiele [29].

There exist multi-objective approaches for assembly line balancing. A multi-objective line balancing problem was studied by Ponnambalam [30], with as objectives the number of workstations, total dead time, and load smoothing between workstations. The authors suggested a genetic algorithm using an approach based on a weighted sum of criteria. A similar approach was presented in the paper of Younes [31] for a flexible manufacturing system. Their criteria were: 1) cost of transferring a part, from one station to another, and 2) load smoothing between stations. A Multi-Objective Group Genetic Algorithm (MOGA) was suggested for the design of a hybrid assembly line in Rekiek [32]. This algorithm was enriched by introducing a Branch and Cut (B&C) and the Prometee (Preference Ranking Organisation METHod for Enrichment Evaluation) techniques.

In our previous publications [33] and [34], we used the well known Multi-start method with one, two and four criteria for which the best parameters settings were determined. We also built a method based on the NSGA-II algorithm to treat, with certain effectiveness, the case where the equipment selection is the sole problem (line balancing was already resolved). In this paper, we develop further this NSGA-II for the more general case of both line balancing and equipment selection. Two versions of the algorithm are provided here: with local search (NSGA_{LS}) and without local search (NSGA_{WLS}). These two methods will be tested on a set of randomly generated problems and compared using three types of different measures: the direct measurement of gaps between objective functions, the measures proposed by Ziztler and Thiele [29] and the distances suggested by Riise [35].

Problem statement

Background

A similar single criterion optimization problem for equipment selection and line balancing was examined in the paper of Makdessian [33] to minimize the investment cost. Three approaches, a Branch and Bound algorithm, a heuristic based on a truncated enumeration, and a genetic algorithm, were presented. In [34], this model was reformulated for a multi-objective equipment selection with Pareto optimization approach, but without line balancing. Two multi-objective algorithms were proposed: a Multi-start algorithm derived from Sysoev and Dolgui [22] and a genetic algorithm of type NSGA-II (the second version of Non Dominated Sorted Genetic Algorithm), inspired from the work of Lacomme [36]. The model and the algorithms proposed in this pa-

[1] A French-Italian economist Pareto (1848–1923) first developed the principle of multi-objective optimization for use in economics. His theory became collectively known as Pareto's optimality concept.

per are based on some preliminary ideas and results of [34]. However, they are more general and more realistic, because here we consider both line balancing and equipment selection problems with additional constraints.

Problem formulation

Each piece of equipment can be used for processing a set of tasks. For each task, there is a set of available alternative pieces of equipment. Any equipment has advantages and drawbacks. A machining line is designed for a given product, but with the possibility to reconfigure for another product in the future. This is feasible because of the modular principle of line design and the possibility to use only partially the equipment already installed.

Let N be the set of all tasks required to manufacture one item of the considered product, M be the set of available pieces of equipment. Let $n = |N|$ be the number of tasks required to manufacture one product item and $m = |M|$ the number of available types of equipment which can be used for this line, respectively. Let Eq_j be the equipment of type j, $j \in M$ and $SetEq_j$ be the set of all tasks that can be executed with Eq_j, $j = 1, \ldots, m$. If Eq_j is assigned to workstation k, then a sub-set $N_{kj} \subseteq$ SetEqj of tasks will be executed with this equipment at workstation k, $k = 1, \ldots, w$. Each available piece of equipment is characterized by its cost, can perform a set of operations and requires skills of a given level for its maintenance. Of course, usually a task executed with different equipment requires different times. Several pieces of equipment of different types can be installed on a workstation.

We will also use the following notations:

Ec_j	cost of a piece of equipment of type j,
$Prod_L$	throughput of line (number of items produced per year),
$Area_j$	area occupied by equipment of type j,
SL_k	Technical complexity of station k, this defines the skill level required for a worker employed for maintenance.

The following assumptions are introduced:

- Precedence relations between tasks $(i = 1, ..., n)$ from N are given;
- The set M of all available pieces of equipment is also given;
- Cost, set of operations and complexity level for each equipment type are known;
- Compatibility constraints between pieces of equipment are known;
- Inclusion constraints for the tasks obliging execution of certain tasks at the same workstation are also known;
- Task processing times are known, deterministic and depend on the type of equipment employed;
- A task can be performed at any workstation, if the station is provided suitable equipment;
- Longest task does not exceed the predetermined cycle time T_0;
- Tasks are grouped in sets, all the tasks of set Nk_j will be executed with equipment j at workstation k;
- Setup, tool changing, material handling, loading and unloading times are negligible or included in the processing times of the tasks;
- All tasks assigned to pieces of equipment are executed simultaneously, thus the equipment time is equal to the maximum of task times for tasks executed with this equipment;
- Objective line cycle time is equal to $T_0 = AvTime/ProdL$, where AvTime is the available working time per year.

Considering the fact that some basic ideas and techniques of the suggested multi-objective genetic algorithms have been already published in [34], this paper presents only a short description of the improved algorithms and is focused on the justification and the application of this approach, the new ideas and the main extensions of the methods, but most of all on the experimental tests of the algorithms, their analysis and some discussions about the further developments for this research.

Optimization criteria

Let w be the number of workstations in a design decision (in a feasible line) and X be the vector of decision variables which represent an assignment of tasks and pieces of equipment to workstations.

The vector criterion of our problem is:

$$Optimize \;\; F(X) = \{f_1(X),\; f_2(X),\; f_3(X),\; f_4(X)\}. \tag{4}$$

The first criterion is related to the minimization of the investment cost:

$$Minimize \;\; f_1(X) = \sum_{k=1}^{w} \sum_{j \in M_k} Ec_j(X), \tag{5}$$

where M_k is the set of equipment assigned to workstation k. Only one piece of equipment of each type can be assigned to a workstation, nevertheless several pieces of equipment of the same type can be employed in the line but at different workstations.

The second criterion is the maximization of the line throughput rate, i.e. the number of items produced per year. The line throughput rate is equal to the throughput rate of the bottleneck workstation:

$$\begin{aligned} &Maximize \;\; f_2(X) = ProdL(X) \\ &= Min \; k = 1, 2, \ldots, w(Prod \; k(X)). \end{aligned} \tag{6}$$

The third criterion furnishes a line with a minimum occupied space:

$$Minimize \;\; f_3(X) = \sum_{k=1}^{w} \sum_{j \in M_k} Area_j(X). \tag{7}$$

And finally, the last criterion, considers the technical complexity of workstations, and thus the required skill level for the workers employed for the line maintenance:

$$\begin{aligned} &Minimize \;\; f_4(X) = SL(X) = Max \; k \\ &= 1, 2, \ldots, w(SLk(X)). \end{aligned} \tag{8}$$

This function is rather particular and considers the technical complexity of the line taking into account the equipment assigned to workstations. Required skill level for a workstation is equal to maximum of required skill levels for pieces of equipment of this workstation. Here, we adopted arbitrarily a scale from 0 to 9. A piece of equipment with $SL = 9$ requires the highest possible skill level for workers employed for its maintenance (note also that this correspond also to the highest worker salary). This information is important and is used to identify the requirements to carry out the maintenance activities.

Constraints

The following additional notations are used for modeling the problem constraints:

w_0 maximal number of workstations in the line (i.e. $w \leq w_0$);

e_k number of pieces of equipment assigned to workstation k;

e_0 maximum authorized number of pieces of equipment per station (i.e. $e_k \leq e_0$);

Eq_j piece of equipment of type j, $j \in M$;

M_k subset of equipment from M assigned to workstation k;

SetEq_j maximal set of tasks which can be executed with equipment of type j, $j \in M$;

N_{kl} set of tasks executed with l-th equipment of workstation k;

N_k set of tasks executed at workstation k, $N_k = \bigcup_{q=1}^{e_k} N_{kq}$.

The constraints introduced in the previous section can be represented as follows:

i) A partial order relation over the set N can be represented by an acyclic graph $G^p = (N, D^p)$. An arc $(i_1, i_2) \in N \times N$ belongs to set D^p if and only if task i_2 must be executed after task i_1. From this graph, for each set $M_k = \{j_1, \ldots, j_{ek}\}$, $j_l \in M$ and vector $N_k = (N_{k1}, \ldots, N_{ke_k})$, $N_{kl} \subseteq$ SetEq_{jl}, the set of tasks $Pred(N_k)$ which must be executed before workstation k can be deduced;

ii) Exclusion conditions for the pieces of equipment of the same workstation can be represented by a graph $\overline{G}^{ES} = (M, \overline{D}^{ES})$ in which a pair $(j_1, j_2) \in M \times M$ belongs to the set $\overline{D}^{ES}$ if and only if the piece of equipment Eq_{j_1} and the piece of equipment Eq_{j_2} cannot be allocated to the same workstation.

iii) Inclusion conditions for the tasks of the same workstation can be represented by the graph $G^I = (N, D^I)$ such that a pair $(i_1, i_2) \in N \times N$ belongs to the set D^I if and only if tasks i_1 and i_2 must be allocated to the same workstation.

Thus, a feasible solution $S = (E, O)$, where $E = (M_1, \ldots, M_w)$ and $O =< \{N_{11}, \ldots, N_{1e_1}\}, \ldots, \{N_{w1}, \ldots, N_{we_w}\} >, M_k = \{j_1, \ldots, j_{ek}\}, N_{kl} \in$ SetEq_{jl}, $j_l \in M$, satisfies the following constraints:

$$\bigcup_{k=1}^{w} \bigcup_{l=1}^{e_k} N_{kl} = N \tag{9}$$

$$\begin{aligned} &N_{k'l'} \cap N_{k"l"} = \emptyset, \\ &k'l' \neq k"l"; k', k" = 1, \ldots, w; l' = 1, \ldots, e_{k'}; \\ &l" = 1, \ldots, e_{k"} \end{aligned} \tag{10}$$

$$Pred(N_k) \subseteq \bigcup_{r=1}^{k-1} \bigcup_{q=1}^{e_r} N_{rq}, \quad k = 1, \ldots, w \tag{11}$$

$$\bigcup_{l=1}^{e_k} N_{kl} \cap c \in \{\emptyset, c\}, c \in D^I, k = 1, \ldots, w \tag{12}$$

$$\begin{aligned} &(M_k(l'), M_k(l")) \notin \bar{D}^{ES}, \\ &k = 1, \ldots, w; l' = 1, \ldots, e_k - 1; l" = l' + 1 \end{aligned} \tag{13}$$

$$e_k \leq e_0, k = 1, \ldots, w \tag{14}$$

$$w \leq w_0. \tag{15}$$

Constraints (9)–(10) state that all tasks from N must be assigned and only once. Relation (11) defines the precedence constraints between tasks. Expression (12) determines the inclusion constraints for the tasks, i.e. the necessity to execute the corresponding tasks at the same workstation. Relation (13) deals with the compatibility of equipment at the same workstation (exclusion constraints, i.e. some pieces of equipment cannot be assigned to the same workstation); Expressions (14)–(15) provide limits on the number of workstations and pieces of equipment for each workstation.

The optimization algorithm for the problem (5)–(15) is described in the next section.

Method developed

As stated in the previous section, this problem is a special case of the multi-objective line balancing and equipment selection (resource planning) problem. In [33] and [34], some preliminary results of this study were reported. The choice of NSGA-II for a similar problem was made based on a literature review and some tests. In this work, some further improvements to this approach are suggested for a more general problem.

First, we need to define the solution coding and a population of solutions.

For our problem, we have used the following solution coding: each solution of w workstations is composed of w genes. Each gene k (i.e. workstation k) is represented by a data structure composed of an area of integer values where each value j signifies Eq_j and corresponding tasks $N_{kj} \subseteq$ SetEqj. An example of this solution is presented in Fig. 1. In this figure, there are 4 workstations. The first one is equipped with one piece of Eq_2 which executes tasks a, b and c. The second workstation has 2 pieces of equipment: one of type Eq_2 and second of type Eq_3, the first executes tasks d and e, and the second, tasks f and g. The third workstation is equipped with Eq_4 which executes tasks k and l. The last station has one piece of Eq_3 which executes s and z. Note that in this example Eq_2 can execute at least (a, b, c, d, e) and Eq_3 can execute (f, g, s, z), but only a part of these tasks are executed at workstations 1, 2 and 4, respectively.

2 (*a*, *b*, *c*)	**2 (*d*, *e*); 3 (*f*, *g*)**	**4 (*k*, *l*)**	**3 (*s*, *z*)**

Fig. 1. Example of solution encoding.

Let P_t be the population of such solutions at the beginning of the iteration t.

As for all genetic algorithms, our NSGA-II consists of two phases: 1) Initialization of population, and 2) Genetic procedures (selection, crossover, mutation, etc.) to evolve the population.

As a first step, the initial population P1 is created with S_p randomly generated solutions. For that, we use an algorithm similar to COMSOAL for SALBP, see for example [14], i.e. we create a list of candidate equipment and select one piece of equipment from this list randomly.

The main procedures of the genetic operations of NSGA-II algorithms are as follows. Before applying the selection operator, the solutions are ranked. This process identifies several Pareto fronts: front(k), $k = 1, 2, \ldots$. If $S \in$front(k), this means that the solution Pt(S) is classified in k-th front. The ranking is made so that non-dominated solutions occupy the first Pareto front. Therefore, the rank number 1 is assigned to the best solutions. For the rest of the solutions, the non-dominated ones are sought and then they are classified in the second Pareto front, and so on. This ranking procedure is stopped when all solutions are classified in fronts.

In this procedure, one starts to fill the first Pareto front. The solutions are compared to each other to check for dominance relations. Let X be the considered solution. If X dominates Y, then Y is added to the set composed of solutions that are worse than X. Otherwise, if Y dominates X, then the number of solutions better than X increases by 1, i.e. Nmbrbetter(X)=Nmbrbetter(X)+1. This procedure is repeated until X is compared with all other solutions from the current population. If after comparisons Nmbrbetter(X)=0, then X is added to the first front, and so on. Finally, we obtain the first complete front. The procedure for the other Pareto fronts is the same. After applying this ranking procedure, we will have all the solutions assigned to Pareto fronts and to each solution a rank is allotted. Let *Rank* be a vector where element *Rank*(S) represents the number of the Pareto front in which solution Pt(S) is assigned.

Another particularity of NSGA-II algorithms consists in calculating distances between solutions of the same Pareto front. This procedure is called "Crowding Distance Assignment" or Margin calculation.

The margin or crowding distance is as follows: Once the crossover and mutation operations are processed, a population with a size (2S) equal to twice the size of the initial population (S) is obtained. Therefore, we cannot accommodate all the fronts in the new parent population of size S. The non-accommodated fronts are simply deleted. When the last allowed front is being considered, we may have more solutions in the last front than the remaining slots in the new population. In order to avoid arbitrarily choosing some individuals, we apply a niching strategy by choosing the individuals that can assure diversity among those considered. For this reason, we calculate the crowding distance for each individual measuring the Euclidean distance between its two closest neighbours in each front. Figure 2 shows a two-objective example for crowding distance assignment.

The tournament prefers the most isolated individuals. Boundary points must always be selected that is why their crowding distance is set to ∞ in the algorithm below. Let $F_o[i]$ be the fitness function of the solution i according the objective o ($o = 1, 2$). The function $F[i]_d$ adopted to calculate the crowding

distance is shown in Algorithm 1. Thereafter, those strings having largest crowding distance values are chosen to be inserted in the new parent population. Once the non-dominated sorting is over, the new parent population, M_{t+1}, is created by choosing solutions of different non-dominated fronts.

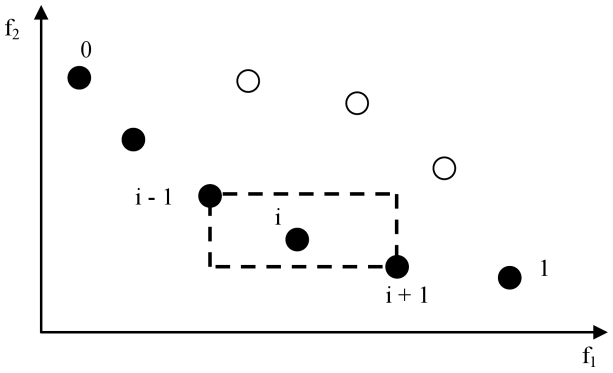

Fig. 2. Crowding distance assignment.

Algorithm 1: Crowding distance assignment
Let χ *the number of solutions in the considered front*
for $i \leftarrow 1$ *to* χ *do*
  $F[i]_d = 0$
end for
for $o \leftarrow 1$ *to* 3 *do*
  Sort the solutions according to each objective o
  Set $F[1]_d$ *and* $F[\chi]_d$ *to* ∞
  for $i \leftarrow 2$ *to* (χ-1) *do*
    $F[i]_d = F[i]_d + (Fo[i+1] - Fo[i-1])$
  end for
end for

If two individuals of the population belong to the same rank, we consider as "better", or more interesting for us or with greater fitness, the one with the larger crowding distance.

Thus, the fitness function of NSGA-II is defined as follows: let X and Y be two solutions from the current population. We consider that X is better than Y if $Rank(X) < Rank(Y)$, or $Rank(X) = Rank(Y)$ and $Margin(X) > Margin(Y)$.

The principal steps of our NSGA-II are shown below (Algorithm 2).

Algorithm 2: Overall structure of our NSGA-II ($NSGA_{WLS}$)
Create an initial population P_t *with Sp solutions, where Sp is the size of population*
Evaluate all the solutions from P_t *using the problem criteria*
Sort P_t *by non domination*
Compute the crowding distance for each solution from P_t
Repeat
Create an offspring population C_t *of* S_p *solutions using genetic operators: selection, crossing, mutation and reparation of unfeasible solutions (two parents from* P_t *generate two children)*
Evaluate each new solution (with all the criteria)
Fusion C_t *and* P_t
Sort the resulting population O_t *of 2*Sp solutions and ranking them in front*(k), k =1,2, ...
Compute the crowding distance for each solution from O_t
$P_{t+1} \leftarrow \varnothing$
$k \leftarrow 1$
While $|P_{t+1}| + |front(k)| \leq Sp$ ***do***
Add front(k) *to* P_{t+1}
$k \leftarrow k+1$
end while
$Missing \leftarrow Sp - |P_{t+1}|$
if $Missing \neq 0$ ***then***
Sort the solutions of front(k) *by decreasing order of their crowding distances*
for$j \leftarrow 1$ ***to*** $Missing$ ***do***
Add the j-th solution of front(k) *to* P_{t+1}
end for
end if
until *Stopping criterion*

This algorithm is based on the algorithm suggested in [37]. Here, $|front(k)|$ means the number of solutions in k-th Pareto front. As a first step, the initial population is generated and solutions are sorted in Pareto fronts. For each solution $P_1(S)$ a margin $Margin(S)$ is calculated. Then, at each iteration of the genetic process, new solutions are generated via a selection of parents and application of genetic operators such as crossover, mutation, etc.

Parents are selected using the *binary tournament*: two solutions are randomly selected from the population, the fittest is kept as the first parent, and the second is rejected. As mentioned above, the "fitness" function uses solution ranks and crowding distances. If two solutions belong to different ranks, the solution of smallest rank is the fittest. When these two solutions belong to the same front, the tournament procedure prefers the most isolated one, i.e. with largest *crowding distance*, to better cover the optimal Pareto set. Two new candidates are randomly selected again to find a second parent with the same procedure.

After selecting two parents, an OX crossover is applied.

To carry out genetic jumps in the algorithm, the operator of mutation is used. We have applied a traditional mutation (random change of a gene, i.e. a piece of equipment is replaced with another randomly). This mutation operator is applied with a probability of 0.01.

After the crossover and mutation operations, we check the feasibility of offspring. For each new so-

lution, we start with a gene and check if all the constraints are respected. Two types of constraints are checked: precedence relations among tasks and compatibility for pieces of equipment assigned to the same stations. If there is a violated constraint, the gene considered is randomly modified: new piece(s) of equipment are randomly selected from the set of equipment whose elements respect these constraints. If there is no feasible solution, then we come back to the previous gene, otherwise go to the next gene, and so on.

Clones are detected and eliminated during the insertion of new individuals into population at the step of fusion of C_t and P_t.

In literature, for a better trade-off between diversification and intensification, the traditional mutation operator is sometimes replaced with a local search, see for example [36–39], such algorithms are called *memetic algorithms*. Note that, in 1999, Deb was the first to mention the possibility to hybridize NSGA-II with a local search. Nevertheless, a local search was rarely used with genetic algorithms of type NSGA-II. Thus, we have developed two genetic algorithms: *i*) $NSGA_{LS}$, with traditional mutation and a local search, and *ii*) $NSGA_{WLS}$, without local search, i.e. only with random mutations.

Note that several questions appear when incorporating the local search in a NSGA, for example: at which step (or iteration) should the local search be applied?

Our $NSGA_{LS}$ is as shown in Algorithm 3.

Algorithm 3: NSGA-II with local search ($NSGA_{LS}$)
Create an initial population P_t of size Sp
Evaluate the solutions from P_t
Sort P_t by non-domination
Compute the crowding distance for each solution from P_t
Repeat
Create an offspring population C_t of Sp solutions using genetic operators: selection, crossing, mutation and reparation of unfeasible solutions (two parents from P_t generate two children for C_t)
Evaluate each new solution
Fusion C_t and P_t
*Sort the resulting population O_t of 2*Sp solutions by non-domination*
Compute the crowding distance for each solution from O_t
$P_{t+1} \leftarrow \varnothing$
$k \leftarrow 1$
While $|P_{t+1}| + |front(k)| \leq Sp$ ***do***
Add all solutions from front(k) to P_{t+1}
$k \leftarrow k+1$
end while
$Missing \leftarrow Sp - |P_{t+1}|$
if $Missing \neq 0$ ***then***
Sort the solutions of front(k) by decreasing order of the crowding distance
for $j \leftarrow 1$ ***to*** *Missing* ***do***
Add the j-th solution of front(k) to P_{t+1}
end for
end if
After each ten iterations,
for *all solutions of the current population*
for *each station of the solution selected*
search for new available and compatible piece of equipment which improves the solution
if *such a piece of equipment exists,* ***then***
replace the current piece of equipment with this new equipment
end if
end for
replace the selected solution with the improved one
end for
until *Stopping criterion*

In the algorithm, the local search is implemented as follows: for each workstation, all non-assigned pieces of equipment to this workstation are tested for possible replacements of the current equipment. There are two conditions when to replace the equipment of a workstation:

– The new equipment meets compatibility constraints.

– The new equipment improves the current solution.

The local search procedure is situated at the end of Algorithm 2 just before checking the stopping criterion, i.e. when a new population is obtained after crossover, mutation, etc. Local search is not applied at each iteration of the genetic algorithm, but only after each ten iterations.

The algorithms $NSGA_{LS}$ (Algorithm 3) and $NSGA_{WLS}$ (Algorithm 2) have been tested. The results are reported in the next section.

Numerical experiments

All developments were done in C++ (Borland C++) and tested on a Pentium 4 computer under Windows 2000. Two versions of NSGA-II were tested: $NSGA_{LS}$, i.e. with an additional procedure of local search applied to all the individuals of the population each ten iterations, and $NSGA_{WLS}$, without this additional local search. The following subsections describe the computational experiments as well as the measuring criteria used to compare the algorithms studied. Two types of results are presented: the first for only two criteria (throughput and cost) and the

second for all the four objective functions mentioned above.

Tested instances

The computational evaluation is based on five problem families: {($|N|$=7, $|M|$=3); ($|N|$=10, $|M|$=5); ($|N|$=20, $|M|$=8); ($|N|$=30, $|M|$=15); ($|N|$=50, $|M|$=20)}, where $n = |N|$ is the number of tasks, and $m = |M|$ is the number of available equipment types. As aforesaid, only one piece of equipment of each type can be assigned to a workstation, nevertheless several pieces of equipment of the same type can be employed in the line but at different workstations. Thus, in the instances tested, the number of available units of equipment of each type had no limit.

For each family of problems, five different instances were generated (25 problems in total).

The size of the population (number of individuals) S_p depends on the size of the instance. The maximal size of the population was fixed at 200. The initial population of size $30 \times |M| \times |N|$ was generated by using a COMSOAL-like heuristic, as for example in [6, 14].

These values have been chosen after a preliminary study, see [34]. Task times as well as the precedence graphs were generated randomly. Compatibility among pieces of equipment belonging to the same workstation is considered to be 100%, i.e. this type of constraint was not used in the tests. The area occupied by equipment has been generated randomly between 10 and 50 units. The cost of each piece of equipment has been generated taking into account its capacity to execute the tasks: that which is able to execute more tasks is more expensive. Worker skill levels required for maintenance of each piece of equipment have been modelled with numbers from 0 to 9.

An additional preliminary study on selecting the stopping criterion has been done. First, the algorithms have been tested with only the stop condition of 1000 iterations. For the problems with 50 tasks and 20 types of equipment, the Pareto-optimal solutions have been obtained in 7068 seconds. Nevertheless, no improvement was observed compared with the tests of the same series with run time of 120 seconds. Therefore, the stop condition has been fixed at 120 seconds for all algorithms and instances.

Two types of tests have been accomplished: the first for only two criteria and the second for all the four objective functions. The same parameters of algorithms have been used for both the bi-objective and four-objective studies.

Measuring criteria

Two multi-objective algorithms: $NSGA_{LS}$ and $NSGA_{WLS}$ were evaluated. The comparison was made between their Pareto fronts of non-dominated solutions, i.e. the Pareto fronts obtained with these algorithms. In order to compare the quality of the Pareto fronts of non-dominated solutions, three measures were used:

Gaps direct measurement of gaps (see sub-section 5.4.1);

μ, μ^*, μ_d distances proposed by Riise [35];

$C1$, $C2$ measure of Zitzler [29].

Let n_1 be the number of solutions in the Pareto front of non-dominated solutions obtained with the first method, we will denote this front $F1$. The Pareto front obtained with the second method will be denoted $F2$ and the notation n_2 will be used for the number of non dominated solutions in $F2$. Note that $F1$ and $F2$ are the fronts of non-dominated solutions, i.e. the first Pareto fronts obtained with corresponding algorithms. In other words, $F1$=*front*(1) for the first method and $F2$=*front*(1) for the second method, when the algorithms are stopped.

The distance of Riise μ is computed as shown in (16), where d_X is a distance between a solution X belonging to the Pareto front $F1$, and its orthogonal projection on the Pareto front $F2$. The value of μ is negative if $F1$ is under $F2$ which means that $F1$ is better than $F2$ for the considered objectives, and positive, otherwise.

$$\mu = \sum_{X \in F1} d_X. \tag{16}$$

As the value of μ depends on the number of solutions n_1 in front $F1$, the normalization is usually applied as follows:

$$\mu^* = \frac{\mu}{n_1}. \tag{17}$$

In order to quantify the distance between the two Pareto fronts $F1$ and $F2$, μ^* is replaced with the μ_d measure. This provides a value that splits off the front $F1$ from the front $F2$ and it is calculated as shown in (18), for the bi-objective case:

$$\mu_d = \frac{\frac{1}{n_1} \sum_{X \in F1} d_X}{\sqrt{(f_{1\max} - f_{1\min})^2 + (f_{2\max} - f_{2\min})^2}}, \tag{18}$$

where $f_{1\max} = \max\{f_1(X) \mid X \in F1 \cup F2\}$, $f_{1\min} = \min\{f_1(X) \mid X \in F1 \cup F2\}$, $f_{2\max} = \max\{f_2(X) \mid X \in F1 \cup F2\}$ and $f_{2\min} = \min\{f_2(X) \mid X \in F1 \cup F2\}$; $f_1(X)$ and $f_2(X)$ are the two objective functions of the problem. For the four-objective case,

the formula is similar but with $\sqrt{\sum_{i=1}^{4}(f_{i\max}-f_{i\min})^2}$ in the denominator, i.e. considering all four functions $f_1(X)$, $f_2(X)$, $f_3(X)$ and $f_4(X)$.

The measure of Zitzler $C1$ represents the percentage of solutions in $F1$ dominated by at least one solution in $F2$. Considering that the measure is not symmetric, it is advised to compute $C2$ as well, which represents the percentage of solutions in $F2$ dominated by at least one solution from $F1$. In conclusion, $F1$ is better than $F2$ if $C1$ is smaller than $C2$.

Bi-objective case

First, we have tested our algorithms for the bi-objective case since maximizing throughput rate and minimizing cost are generally the most important criteria and often used for these kinds of design problems. The following indicators will be employed to compare the results of our two algorithms: that of *Ziztler* and *Thiele* [29] and of *Riise* [35], both were explained in the previous sub-section. Note that these two types of measures have been successfully applied in previous work to compare two Pareto fronts obtained by two different multi-objective algorithms, see [40, 41].

Test results are reported in Table 1 where n_1 and n_2 are the numbers of non dominated solutions in fronts $F1$ and $F2$, respectively. For the test families, $n = |N|$ is the number of tasks and $m = |M|$ is the number of available equipment types. $C1$ and $C2$ clearly show that $NSGA_{LS}$ outperforms $NSGA_{WLS}$ in all instances.

Table 1
Measuring indicators for the bi-objective case.

n; m	n_1	n_2	$C1$	$C2$	μd
7; 3	12	10	0.25	0.10	−0.1
10; 5	10	9	0.40	0.11	−0.13
30; 15	16	7	0.88	0.00	−0.29
20; 8	11	10	0.73	0.10	−0.24
50; 20	13	12	0.85	0.00	−0.32

Note: in all the configurations $C1 \gg C2$ since $C1$ ∼70% and $C2$ ∼0–10% which proves that $NSGA_{LS}$ dominates $NSGA_{WLS}$. Moreover $\mu_d < 0$, which means that the first (optimal) Pareto front of the $NSGA_{LS}$ algorithm is under that of the $NSGA_{WLS}$ algorithm. The last statement means that the Pareto front of the $NSGA_{LS}$ algorithm is more suitable for the two considered objectives: it maximizes throughput rate and minimizes the cost of the line. Therefore, we can conclude that by applying local search in our algorithm we improve without doubt the quality of the results.

Four objective case

Direct comparison

To select the best algorithm, both versions have been tested and the following gaps have been computed as shown in Eq. (19) where F_NSGA_{WLS} and F_NSGA_{LS} are the best values of the corresponding objective functions obtained with $NSGA_{WLS}$ and $NSGA_{LS}$, respectively. The value of Gap is calculated for each adopted criterion.

$$Gap = \frac{(F\_NSGA_{WLS}) - (F\_NSGA_{LS})}{Max\ \{F\_NSGA_{WLS}, F\_NSGA_{LS}\}}. \quad (19)$$

The average values of the different criteria and the gaps are presented in Tables 2, 3 and 4. Columns Av_c, Av_t, Av_a and Av_s in Tables 2 and 3 report the average values of cost, throughput, occupied area and skill level, respectively. Similarly, columns Gap_c, Gap_t, Gap_a and Gap_s in Table 4 refer to cost, throughput, occupied area and skill level necessary for maintenance.

Table 2
Results of $NSGA_{LS}$.

n; m	Av_c	Av_t	Av_a	Av_s
7; 3	21670	8483	114	7.5
	15853	7926	140	0
	21037	9100	116	8.83
	12584	6118	56	0
	23569	1001	124	7.2
10; 5	21343	8783	154	8
	20299	7832	193	2
	26208	9131	207	2.33
	15730	6118	70	3
	23817	9853	166	2.8
30; 15	131669	6528	764	6.85
	126457	6364	559	5
	125994	6426	838	8
	128749	6388	584	6.75
	117491	6923	584	8.5
20; 8	87785	7624	511	6.08
	59615	6449	294	3.25
	55374	7542	322	5.35
	41168	6651	281	2.66
	49439	7558	264	7
50; 20	167588	6070	900	6.52
	235072	7027	1458	8.12
	180831	6002	1479	4.5
	195979	6705	1418	3.16
	207883	6855	1375	6.68

Table 3
Results of $NSGA_{WLS}$.

n; m	Av_c	Av_t	Av_a	Av_s
7; 3	22137	8258	150	7.33
	15853	7926	140	0
	21190	9100	116	8.83
	12584	6118	56	0
	23569	1001	124	7.2
10; 5	21343	8783	154	8
	20387	7832	208	2.5
	26208	9131	207	2.33
	21041	6388	138	8.25
	23817	9853	166	2.8
30; 15	107440	6629	876	9
	90080	6535	885	8.6
	96935	7691	1099	7.5
	105563	6704	970	7.33
	95770	7082	936	9
20; 8	70442	8045	581	7
	74224	6466	528	7.75
	68252	7543	365	5.5
	57815	6561	360	5.22
	66943	7573	499	7
50; 20	213470	6116	1681	8
	237586	6658	1507	9
	200295	6111	1847	8
	222123	6700	1936	6.92
	236347	7278	1804	8

Table 4
Gaps for results obtained with algorithms tested.

n; m	Gap_c	Gap_t	Gap_a	Gap_s
7; 3	−0.021	0.026	−0.240	0.023
	0	0	0	0
	−0.007	0	0	0
	0	0	0	0
	0	0	0	0
10; 5	0	0	0	0
	−0.004	0	−0.072	−0.200
	0	0	0	0
	−0.250	−0.042	−0.490	−0.636
	0	0	0	0
30; 15	−0.184	0.015	−0.128	−0.238
	−0.287	0.029	−0.367	−0.418
	−0.230	0.164	−0.230	0.063
	−0.180	0.047	−0.397	−0.079
	−0.185	0.023	−0.375	−0.055
20; 8	−1.974	0.052	−0.119	−0.131
	−0.197	−0.003	−0.440	−0.580
	−0.188	0	−0.116	−0.027
	−0.930	0.014	−0.218	−0.490
	−0.261	−0.002	−0.266	0
50; 20	−0.215	−0.008	−0.464	−0.184
	−0.106	0.053	−0.033	−0.097
	−0.097	−0.018	−0.199	−0.437
	−0.118	0.001	−0.267	−0.542
	−0.121	−0.058	−0.237	−0.164

Table 5
Comparison of gaps.

(a)	
$Gap_{c(LS)} < Gap_{c(WLS)}$	$Gap_{t(LS)} > Gap_{t(WLS)}$
76% (19/25)	**40%** (10/25)
$Gap_{c(LS)}=Gap_{c(WLS)}$	$Gap_{t(LS)}=Gap_{t(WLS)}$
24% (6/25)	**32%** (8/25)
(b)	
$Gap_{a(LS)} < Gap_{a(WLS)}$	$Gap_{s(LS)} < Gap_{s(WLS)}$
72% (18/25)	**60%** (15/25)
$Gap_{a(LS)}=Gap_{a(WLS)}$	$Gap_{s(LS)}=Gap_{s(WLS)}$
28% (7/25)	**28%** (7/25)

It is evident from the results in Tables 4 and 5 that the $NSGA_{LS}$ algorithm gives better results (smaller gaps).

Multi-objective comparison

The algorithms $NSGA_{LS}$ and $NSGA_{WLS}$ have been also compared by using the measuring criteria proposed in Zitzler and Thiele [29]. The parameters of the algorithms are the same as in the previous sub-section. Each non dominated front obtained by $NSGA_{LS}$ is compared with $NSGA_{WLS}$ thanks to $C1$ and $C2$ measures. Remember that $C1$ shows the ratio of solutions in the Pareto front obtained with $NSGA_{WLS}$ dominated by, at least, one solution obtained with $NSGA_{LS}$. Conversely, $C2$ represents the number of non dominated solutions obtained with $NSGA_{LS}$ that are dominated by, at least, one solution of Pareto front obtained with $NSGA_{WLS}$. The values of $C1$ and $C2$ are reported in Table 6.

Table 6
Measuring indicators for the four-objective case.

n; m	n_1	n_2	$C1$	$C2$
7; 3	14	12	0.071	0
10; 5	15	9	0.133	0
30; 15	12	11	0.083	0
20; 8	8	13	0.375	0
50; 20	16	9	0.125	0

$C1$ and $C2$ show that $NSGA_{LS}$ is never dominated by $NSGA_{WLS}$ which allows us to conclude, as in the bi-objective case, on a relative efficiency of local search in this type of algorithm. Nevertheless, $NSGA_{WLS}$ is dominated in only few cases. This can be explained since this problem presents four objectives: it is difficult to discriminate between two solutions while the number of objectives is increasing. Therefore, most of the solutions are not dominated by others, since the search space is increasing when the number of objectives is also growing.

Discussion on possible extensions of the model

The model (5)–(15) has been developed for the design of automated machining lines with multi-spindle heads activated in parallel. For this case, all the tasks of the same workstation are executed in parallel. Thus, we ignored the cycle time constraint, because it is sufficient to examine all available pieces of equipment before optimisation and eliminate those with processing times longer than the cycle time T_0. Nevertheless, the algorithms developed can be also used for the cases when the cycle time constraint cannot be ignored (machining lines with sequential activation of spindle heads, manual assembly lines, etc.). In these cases, it is necessary to add to the model the following expressions:

$$\sum_{j \in M_k} TimeEq_j \leq T_0, \qquad k = 1, \ldots, w, \tag{20}$$

where $TimeEq_j$ is the time necessary to execute all tasks assigned to equipment j at workstation k, this time is estimated as $Max\{t_{ij}\}$ for multi-spindle machining equipment, and as $\sum t_{ij}$ for manual assembly lines, here t_{ij} is the time of task i executed with equipment j.

Another non restrictive assumption of the model concerns the objective function $f_1(X)$ which represents the investment cost. Here again, this is due to the lines considered where the investment is the major cost component. For manual assembly lines, for example, the worker salaries should be also included. This can be accomplished by introducing salaries $Ws_k(X)$ of workers assigned to workstation k, $k = 1, \ldots, w$. The criterion $f_1(X)$ is then modified as follows:

$$Minimize\ f_1(X) = \sum_{k=1}^{w} \left(Ec_k(X) + Ws_k(X)\right) \tag{21}$$

where the equipment cost $Ec_k(X)$ and worker wages $Ws_k(X)$ are put on the same scale using known methods [5]. Note that $Ec_k(X)$ and $Ws_k(X)$ depend on pieces of equipment assigned to workstation k (costs of pieces of equipment and wages for workers necessary for this equipment).

Criterion $f_4(X)$ considers only the required skills for the maintenance team. For manual assembly lines, it is more important to take into account the required operator skills for workstations. The corresponding criterion can be formalised as follows:

$$\begin{aligned} &Minimize\ f_4(X) = \Delta SL(X) \\ &= \mathrm{Max}_{k=1,2,\ldots,w}(SL_k(X)) \\ &- \mathrm{Min}_{k=1,2,\ldots,w}(SL_k(X)). \end{aligned} \tag{22}$$

A piece of equipment with $SL = 9$ requires the highest possible skill level (note also that the corresponding workstation has the highest salary cost). $\Delta SL = 0$ represents the case where all workers are perfectly interchangeable. The case when $\Delta SL = 9$ is most constraining with a maximum skill difference.

In some cases, as for example in the semiconductor industry, there are also compatibility constraints between equipment of neighbouring workstations [22]. For this case, exclusion conditions for the pieces of equipment of neighbouring workstations can be represented by a graph $\overline{G}^{EN} = (M, \overline{D}^{EN})$ in which a pair $(j_1, j_2) \in M \times M$ belongs to the set $\overline{D}^{EN}$ if and only if piece of equipment Eq_{j_1} and piece of equipment Eq_{j_2} cannot be allocated to two neighbouring workstations. The following constraint is then added in the model:

$$\begin{gathered} \left(M_k\left(l'\right), M_{k+1}\left(l''\right)\right) \notin \overline{D}^{EN}, \\ k = 1, \ldots, w-1;\ \ l' = 1, \ldots, e_k; \\ l'' = 1, \ldots, e_{(k+1)}. \end{gathered} \tag{23}$$

Other extensions are possible and can be easily added to the model (5)–(15).

Conclusion

A new problem of multi-objective line balancing and equipment selection (resource planning) was introduced in this paper. This problem deals with automated machining lines. Each workstation can contain one or more pieces of equipment. The problem is to select pieces for each workstation from a given set of all available equipment. The goal is to configure a machining line for production of a given product in large series while optimizing some criteria. The difference with the models known in literature consists in the fact that we use a sub-set of tasks of each available piece of equipment and not necessary all the tasks that this equipment can execute. This is a new requirement due to the concept of reconfigurable manufacturing systems. The possibility to employ any sub-set of a given set of tasks which can be executed with selected equipment, facilitates a future reconfiguration of the line. At the same time, this increases the combinatorial complexity of the problem as compared with the case where the set of tasks for each piece of equipment is fixed.

After presenting the problem statement, an algorithm of type NSGA-II was developed. Then, it was enriched with a procedure of local search adapted for the multi-objective optimization. We compared the versions of the algorithm with ($NSGA_{LS}$) and without ($NSGA_{WLS}$) local search using three mea-

suring criteria: gaps for each criterion; distances of Riise and measure of Zitlzler. They are employed to compare the optimal Pareto front obtained with $NSGA_{LS}$ and the optimal Pareto front of $NSGA_{WLS}$, for each test example. The results of numerical tests on several families of randomly generated examples, have demonstrated that based on these measures, the NSGA-II with the additional local search ($NSGA_{LS}$) greatly outperforms the same NSGA-II without this local search procedure ($NSGA_{WLS}$).

Regarding the perspectives, the testing of other possible crossovers or generating the initial population by using other heuristics might be a subject of future research. Furthermore, it might be interesting to add other industrial constraints to be closer to real industrial situations.

Furthermore, another promising way to pursue might be the application of parallel calculation techniques for these NSGA-II algorithms.

The results of this paper encourage the use of similar techniques for other problems and different types of production systems.

A more comprehensive comparison with other multi-objective algorithms as NPGA and SPEA could be interesting. Still another path to examine is to integrate the random factors in the model, such as machine breakdowns, maintenance times, etc. Finally, it may be possible to take into account some user preferences interactively making the approach more attractive to management.

The authors thank Chris Yukna for his help with English.

References

[1] Askin R.G., Standridge C.R., *Modeling and Analysis of Manufacturing Systems*, John Wiley & Sons, 1993.

[2] Hitomi K., *Manufacturing System Engineering*, Taylor & Francis, 1996.

[3] Scholl A., *Balancing and Sequencing of Assembly Lines*, Physica-Verlag, 1999.

[4] Dashchenko A.I. (Ed.), *Manufacturing Technologies for Machines of the Future 21st Century Technologies*, Springer, 2003.

[5] Dolgui A., Proth J.-M., *Les Systèmes de Production Modernes*, 2 volumes, Hermès Science, London, 2006.

[6] Dolgui A., Proth J.-M., *Supply Chain Engineering: Useful Methods and Techniques*, Springer, 2010.

[7] Baybars I., *A survey of exact algorithms for the simple assembly line balancing*, Management Science, 32, 909–932, 1986.

[8] Talbot F.B., Patterson J.H., Gehrlein W.V., *A comparative evaluation of heuristic line balancing techniques*, Management Science, 32(4), 430–454, 1986.

[9] Ghosh S., Gadnon R.J., *A comprehensive literature review and analysis of the design, balancing and scheduling of assembly systems*, International Journal of Production Research, 27, 637–670, 1989.

[10] Erel E., Sarin S., *A survey of the assembly line balancing procedures*, Production Planning and Control, 9(5), 414–434, 1998.

[11] Rekiek B., Dolgui A., Delchambre A., Bratcu A., *State of art of assembly lines design optimization*, Annual Reviews in Control, 26(2), 163–174, 2002.

[12] Scholl A., Becker C., *State-of-the-art exact and heuristic solution procedures for simple assembly line balancing*, European Journal of Operational Research, 168, 666–693, 2006.

[13] Guschinskaya O., Dolgui A., *Equilibrage de lignes de production: état de l'art*, Journal Européen des Systèmes Automatisés, 44, 1081–1119, 2010.

[14] Dolgui A., Finel B., Guschinsky N., Levin G., Vernadat F., *An heuristic approach for transfer lines balancing*, Journal of Intelligent Manufacturing, 16, 159–171, 2005.

[15] Dolgui A., Finel B., Guschinsky N., Levin G., Vernadat F., *MIP approach to balancing transfer lines with blocks of parallel operations*, IIE Transactions, 38(10), 869–882, 2006.

[16] Belmokhtar S., Dolgui A., Guschinsky N., Levin G., *An integer programming model for logical layout design of modular machining lines*, Computers and Industrial Engineering, 51(3), 502–518, 2006.

[17] Dolgui A., Ihnatsenka I., *Branch and bound algorithm for a transfer line design problem: Stations with sequentially activated multi-spindle heads*, European Journal of Operational Research, 197, 1119–1132, 2009.

[18] Graves S.C., Holmes R.C., *Equipment selection and task assignment for multiproduct assembly system design*, International Journal of Flexible Manufacturing Systems, 1, 31–50, 1988.

[19] Szadkowski J., *Critical path concept for multitool cutting processes optimization*, Manufacturing, Modeling, Management and Control: A Proceedings Volume of the IFAC Symposium, Vienna, Austria, 393–398, 1997.

[20] Bukchin J., Tzur M., *Design of flexible assembly line to minimize equipment cost*, IIE Transactions, 32, 585–598, 2000.

[21] Ishibushi H., Yoshida T., Murata T., *Balance between genetic search and local search in memetic algorithms for multiobjective permutation flowshop scheduling*, IEEE Transactions on Evolutionary Computation, 7(2), 204–223, 2003.

[22] Sysoev V., Dolgui A., *A Pareto optimization approach for manufacturing system design*, Proceedings of the International Conference on Industrial Engineering and Production Management (IEPM'99), book 1, 75–83, 1999.

[23] Collette Y., Siarry P., *Optimisation Multiobjectif*, Edition Eyrolles, 2000.

[24] Srinivas N., Deb K., *Multiobjective function optimization using non-dominated sorting genetic algorithms*, Evolutionary Computation Journal, 2(3), 221–248, 1994.

[25] Fonseca C.M., Fleming P.J., *An overview of evolutionary algorithms in multi-objective optimisation*, Evolutionary Computation, 3(1), 1–16, 1995.

[26] Sarker R., Liang Ko-H., Newton C., *A new multiobjective evolutionary algorithm*, European Journal of Operational Research, 140, 12–23, 2002.

[27] Coello C., *A comparative survey of evolutionary-based multiobjective optimization techniques*, Knowledge and Information Systems, 1, 269–308, 1999.

[28] Horn J., Nafpliotis N., *Multiobjective optimization using the Niched Pareto Genetic Algorithm*, IlliGAL Report No.93005, University of Illinois at Urbana-Champaign, Urbana, Illinois, USA, 1993.

[29] Zitzler E., Thiele L., *Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach*, IEEE Transactions on Evolutionary Computation, 3, 257–271, 1999.

[30] Ponnambalam S.G., Aravindan P., Mogileeswar Naidu G., *A multi-objective genetic algorithm for solving assembly line balancing problem*, International Journal of Advanced Manufacturing Technology, 16, 341–352, 2000.

[31] Younes A., Ghenniwa H., Areibi S., *An adaptive genetic algorithm for multi-objective flexible manufacturing systems*, Proceedings of the Genetic and Evolutionary Computation Conference, 1241–1248, 2002.

[32] Rekiek B., Pellichero F., De Lit P., Falkenauer E., Delchambre A., *A resource planner for hybrid assembly lines*, Proceedings of the 15th International Conference CAR & FOF'99, vol. 1, MW6-18–MW6-23, 1999.

[33] Makdessian L., Yalaoui F., Dolgui A., *Optimisation de lignes de production, Partie I: une approche monocritère*, Journal of Decision Systems, 17, 313–336, 2008.

[34] Makdessian L., Yalaoui F., Dolgui A., *Optimisation de lignes de production, Partie II: une approche multicritère*, Journal of Decision Systems, 17, 337–368, 2008.

[35] Riise A., *Comparing genetic algorithms and tabu search for multiobjective optimization*, Proceedings of the IFORS conference, Edinburgh, July 8-12, 2002.

[36] Lacomme P., Prins C., Sevaux M., *A genetic algorithm for a bi-objective capacitated arc routing problem*, Computers & Operations Research, 33, 3473–3493, 2006.

[37] Deb K., Agrawal S., Pratap A., Meyarivan T., *A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II*, Proceedings of Parallel Problem Solving from Nature VI, 849–858, 2000.

[38] Deb K., *Multi-objective genetic algorithms: Problem difficulties and construction of test problems*, Evolutionary Computation Journal, 7(3), 205–230, 1999.

[39] Deb K., Pratap A., Agarwal S., Meyarivan T., *A fast and elitist multi-objective genetic algorithm: NSGA-II*, IEEE Transactions on Evolutionary Computation, 6(2), 182–197, 2002.

[40] Chehade H., Amodeo L., Yalaoui F., *A new hybrid multiobjective algorithm for assembly lines design*, Proceedings of the World Congress in Computer Science, Computer Engineering and Applied Computing, Las Vegas, USA, July 13–16, 2009.

[41] Dugardin F., Amodeo L., Yalaoui F., *Méthodes multi-objectif pour l'ordonnancement de lignes réentrantes*, Journal of Decision Systems, 18(2), 233–257, 2009.