Coupling a genetic algorithm approach and a discrete event simulator to design mixed-model un-paced assembly lines with parallel workstations and stochastic task times

Lorenzo Tiacci*

Università degli Studi di Perugia - Dipartimento di Ingegneria, Via Duranti, 93 – 06125 – Perugia, Italy

Abstract

In the paper, an innovative approach to deal with the Mixed Model Assembly Line Balancing Problem (MALBP) with stochastic task times and parallel workstations is presented. At the current stage of research, advances in solving realistic and complex assembly line balancing problem, as the one analyzed, are often limited by the poor capability to effectively evaluate the line throughput. Although algorithms are potentially able to consider many features of realistic problems and to effectively explore the solution space, a lack of precision in their objective function evaluation (which usually includes a performance parameter, as the throughput) limits in fact their capability to find good solutions. Traditionally, algorithms use indirect measures of throughput (such as workload smoothness), that are easy to calculate, but whose correlation with the throughput is often poor, especially when the complexity of the problem increases. Algorithms are thus substantially driven towards wrong objectives. The aim of this paper is to show how a decisive step forward can be done in this filed by coupling the most recent advances of simulation techniques with a genetic algorithm approach. A parametric simulator, developed under the event/object oriented paradigm, has been embedded in a genetic algorithm for the evaluation of the objective function, which contains the simulated throughput. The results of an ample simulation study, in which the proposed approach has been compared with other two traditional approaches from the literature, demonstrate that significant improvements are obtainable.

Keywords: mixed-model assembly line; balancing; un-paced lines; asynchronous lines; stochastic task times; paralleling; discrete event simulation.

1 Introduction

The design of an assembly line is a complex problem that, as many other industrial problems, has to take into account two fundamental aspects: performances and costs. Performances of an assembly line are mainly related to its throughput, i.e. to the number of products that can be completed in the unit time. Costs are related to the amount of resources (labor and equipment) needed to complete all the tasks. The assembly line balancing problem is fundamentally a trade-off problem between these two factors. There are many ways to formulate Assembly Line Balancing Problems (ALBPs), as it will be described in the next paragraph. However, both these aspects have to be taken into account in some way, or in the objective function or in the problem constraints.

One of the main issues while solving ALBPs is that is difficult to evaluate the throughput of a complex assembly line. While costs of a determined line configuration can be easily calculated from the amount of

* Corresponding Author: Tel.: +39-075-5853741; fax +39-075-5853736.
E-mail address: lorenzo.tiacci@unipg.it (L.Tiacci).

resources employed, it is often not easy to calculate its throughput with an acceptable degree of precision. This difficulty is correlated to the features considered in the ALBP. It is low for simple lines, with a limited number of tasks, deterministic completion times, and where only one product can be produced. However, when more realistic features are considered it becomes much higher. So for example if a large number of tasks have to be considered, with stochastic times of completion, and/or when multiple products have to be assembled in a mixed-model way, with a specified sequence, or with the possibility to utilize parallel workstations in each workcentre, then it becomes very challenging to predict the throughput of a determined line configuration. In this context, it becomes consequently very difficult to compare different design alternatives on the basis of their performances.

As Bukchin (1998) outlined in his study, the only way to accurately evaluate the throughput of complex assembly lines would be to perform a simulation study. However, this is unfortunately very time consuming, because it requires to build a simulation model each time a different design alternative has to be evaluated. This does not fits with the most part of solving methods presented in literature, which provide the evaluation of a large number of possible solutions to find the final one.

For this reason until now, researchers have utilized indirect 'measures' of the throughput, which can be easily calculated from a determined line configuration, without the need to perform a simulation run. These measures are based upon the assumption that the way workloads are allocated to workstations (in terms of variability inside each workstation and among different workstations) can have a direct influence on the line throughput. Unfortunately, the correlation between the effective throughput and these measures are often poor. Furthermore, the more the complex is the line, the more is expected this correlation to be low.

So at this stage of research, advances in solving more realistic and complex assembly line balancing problem are limited by the poor capability to effective evaluate the line throughput. Although algorithms are potentially able to consider many features of realistic problems and to effectively explore the solution space, a lack of precision (or a sort of bias) in their objective function evaluation limits in fact their capability to find good solutions. In summary, algorithms are driven towards wrong objectives.

The aim of this paper is to show how a decisive step forward can be done in this filed by coupling the most recent advances of simulation techniques with a genetic algorithm approach. In particular, the adoption of event and object oriented simulation approaches has recently allowed to build parametric simulation models that can be embedded in genetic algorithms procedures for the effective evaluation of their objective function (Tiacci, 2012).

In the paper one of the most complex problem in assembly lines is considered: mixed-model lines, with stochastic task times of completion and parallel workstations allowed. There are few algorithms presented in literature capable to deal with all these features at the same time. In this work we introduce a new genetic algorithm approach in which a parametric simulator is embedded, and compare it with other two methods, namely a Simulation Annealing (SA) approach and a Genetic Algorithm (GA) approach, that on the contrary utilize indirect measures of throughput in their objective functions. The results of an ample simulation study, reported in Section 6, demonstrate the radical improvements obtainable by the proposed approach.

The paper is organized as follows. In the next section, a literary review on assembly line balancing problems and the related operational objectives is carried out. In Section 3 the particular problem taken into consideration is described in detail. The GA algorithm approach, coupled to the parametric simulator for the objective function evaluation, is described in Section 4. Section 5 is dedicated to the design of experiment for the evaluation of the proposed approach, which is also compared with other two approaches from the literature. In Section 6 results are reported and discussed.

2 Literary Review

In this paper we focus our attention to a very complex assembly line balancing problem (described in detail in Section 3), which provides the following features: mixed-model un-paced line, with parallel workstations, stochastic task times of completion; the objective takes into consideration equipment and labor cost. The

literary review is organized in two sections. Section 2.1 is related to assembly line balancing problems that fit with our case. Section 2.2 is related to how the objectives of this type of problem have been approached in literature.

2.1 Assembly line balancing problems

The Assembly Line Balancing Problem (ALBP) consists in assigning tasks to workstations, while optimizing one or more objectives without violating any restriction imposed on the line (e.g. precedence constraints among tasks). The basic version is the so-called Simple Assembly Line Balancing Problem (SALBP) and provides a single-model, paced line with fixed cycle time and deterministic task times. For an overview of exact methods and heuristics developed to solve the SALBP see Scholl and Becker (2006). One of the most restricting assumptions of SALBP is related to the production of a single model. In fact, pressure for manufacturing flexibility, due to the growing demand for customized products, has led to a gradual replacement of simple assembly lines with mixed-model assembly lines, in which a set of similar models of a product can be assembled simultaneously.

The related problem that arises is the so called MALBP (Mixed-model Assembly Line Balancing Problem), that is much more complicated because of the variability of assembly times of different models assigned to a specific workstation. In mixed-model production, set-up times between models could be reduced sufficiently enough to be ignored and usually all models are variations of the same base product and only differ in specific customizable product attributes (Boysen et al., 2008). Studies published in the last years utilize different approaches to solve it, such as: Simulated Annealing (McMullen and Frazier, 1998; Simaria and Vilarinho, 2001; Vilarinho and Simaria, 2002), Ant techniques (McMullen and Tarasewich, 2003; McMullen and Tarasewich, 2006; Yagmahan, 2011), Genetic Algorithms (Akpinar and Bayhan, 2011; Haq et al., 2006; Simaria and Vilarinho, 2004; Tiacci et al., 2006; Zhang and Gen, 2011), and other heuristics (Askin and Zhou, 1997; Bukchin et al., 2002; Jin and Wu, 2003; Karabati and Sayin, 2003; McMullen and Frazier, 1997; Merengo et al., 1999).

The MALBP can be seen as a particular case of the more general GALBP (Generalized Assembly Line Balancing Problem), which embrace many features related to more realistic problems, such as cost functions, equipment selection, paralleling, stochastic task times. For a comprehensive classification of the possible features of the GALBP see Becker and Scholl (2006) and Boysen et al. (2007).

Stochastic task times in particular are a difficult issue to deal with. Tasan and Tunali (2008), in their survey on genetic algorithms applied to ALBP, outlined that only one article (Suresh and Sahu, 1994), out of the 29 analysed, dealt with stochastic processing times. Non deterministic task times in fact complicate the prediction of line performances, because blocking and starvation phenomena are accentuated. Furthermore, blocking and starvation are also induced by the variability of task times of different models assigned to a specific workstation. Thus, the combination among mixed-model lines and stochastic task times is particularly challenging.

Another feature characterizing many real assembly lines configuration is the possibility to implement some type of parallelism, for example utilizing parallel workstations performing the same task set. The aim of using parallel stations is often to perform tasks with processing time larger than the desired cycle time. However, also if any given task time does not exceed cycle time, the possibility to replicate workstations may be desirable, because it enlarges the space of feasible solutions of the balancing problem, including many feasible and potentially better balanced configurations (Tiacci et al., 2006; Vilarinho and Simaria, 2002).

There are not many algorithm in literature that takes into consideration the same amount of modeling options as our case at the same time, such as, in particular, stochastic task times, multiple products produced in a mixed-model way, parallel workstations (without limits regarding the maximum number of replicas per work centre), and a fitness function that considers cycle time performance as well as total labour and equipment costs. In fact, the one considered is a realistic industrial setting, which differs significantly from the simplified theoretical problem. Such particularities are very difficult to include in mathematical models and

generally increase significantly the complexity of the problem to be solved. This can explain why they are generally not considered together in the literature (Lapierre et al., 2006). Of the above mentioned studies, the only ones that address all these issues at the same time are the works from McMullen and Frazier (1998), McMullen and Tarasewich (2003), McMullen and Tarasewich (2006) and Tiacci et al. (2006).

2.2 Operational objectives and their measures

As already mentioned in the introduction, the two basic objectives that have to be taken into consideration in the assembly line balancing problems are performances and costs. In the existing assembly line literature, the performance and the cost objective are usually treated by fixing one of the two as a constraint, and trying to optimize the other one. Thus in type-1 problem one tries to minimize the number of stations for a given cycle time, while in the so called type-2 problem one tries to minimize the cycle time for a given number of workstations. The most general problem version is addressed as the type-E problem, which tries to maximize the line efficiency by simultaneously minimizing the cycle time and the number of workstations. This can be achieved for example through a unique objective function by applying objective-specific weighting factors.

These types of problem versions, which have been originally addressed to the SALBP, can be extended to the MALBP, where mixed models production is considered. However, in this case, the sequence of models that enter the line, together with the variability of task times of different models assigned to a specific workstation, can cause work overload in many stations. To avoid these difficulties, different objectives for assembly line balancing for workload smoothing have been introduced in the literature. For an overview and a comparison among objectives to smoothen workload see Emde et al. (2010).

Methods that are developed to seek station assignments that lead to more balanced workloads across stations and across products are motivated to limit the effect, on the realized cycle time, of the sequencing of different models on the assembly line. However one should remind that workload smoothing it is not an objective in itself, but a (supposed) mean to achieve a high and stable throughput (Tiacci, 2012). Workload smoothing objectives are substantially utilized, instead of throughput, because the throughput of a mixed model line is difficult to estimate, while measures related to workload smoothing, for given tasks assignments, can be calculated straightforwardly. However using workload smoothing as objective, as outlined by Karabati and Sayin (2003), remain to be an approximate approach, for the very reason that the effects of the sequencing decision on the line throughput are not incorporated explicitly.

Most of the studies in literature that deal with mixed model assembly lines use some of these ‘indirect measures’ of throughput. However, their correlation with the throughput should be verified. The only two studies that deal with this issue in literature are those by Bukchin (1998) and by Venkatesh and Dabade (2008). From their works, it is possible to conclude that is difficult to find a measure of general validity for all the possible issues of the problem. Furthermore, a loss of correlation of the same measures with the simulated throughput is expected as the number of features that try to capture the complexity of real cases increases (e.g. mixed models, sequencing policies, parallel workstations, stochastic task times, etc.) (Tiacci, 2012).

How anticipated in the introduction, in this paper we present an innovative approach that, thanks to the last advances of the research on simulation architectures and techniques, allow to overcome the limit to use, for the objective function evaluation, indirect measures of throughput instead of the simulated one. This is performed by coupling a genetic algorithm and an event/object oriented parametric simulator.

3 The problem

The problem is to assign a certain number of tasks to a number of WC, while respecting precedence constraints, and trying to maximize the objective function.

3.1 Notation

i the task index ($i = 0, \ldots, N\text{-}1$)

j the model index ($j = 0, \ldots, M\text{-}1$)
k the work centre index ($k = 0, \ldots, P\text{-}1$)
ct the imposed cycle time;
ct_{eff} the realized average cycle time;
N the number of tasks of the problem;
N_k the number of tasks assigned to work centre k;
M the number of models;
P the total number of work centres in the line;
t_{ij} the average time required to execute the task i on model type j;
σ_{ij} the standard deviation of the time required to execute the task i on model type j;
cv the coefficient of variation of t_{ij};
W_k the number of workers (or workstations) in work centre k;
E_k the number of pieces of equipment in work centre k ($E_k = W_k \cdot N_k$);
CW annual cost of a worker;
CE annual cost of a piece of equipment;

3.2 Tasks

A set of N tasks (numbered with $i = 0,\ldots, N\text{-}1$) has to be performed in the line in order to complete each product. Because we are dealing with mixed model lines, the number of models (types of product) to be assembled can be higher than one, and it is indicated by M (numbered with $j = 0, \ldots, M\text{-}1$). Input data are thus represented by an N x M matrix t_{ij} whose elements represent the average completion time of task i on model type j. If the completion of a model does not require the execution of a certain task, this would result in a 0 in the corresponding matrix element.
In order to take into account another important feature of real assembly lines, stochastic task times have to be considered. As many authors in literature, task times are considered normally distributed, which is considered to be realistic in most cases of human work. The standard deviation σ_{ij} of the completion time of task i for model j is taken equal to its mean value (t_{ij}) multiplied by the coefficient of variation cv ($\sigma_{ij} = cv \cdot t_{ij}$).

3.3 Line features

In the line, each operator has a WorkStation (WS) where he performs one or more tasks. A cost CW is associated to each operator. Each Work Centre (WC) consists of either one WS, for the case of non-paralleling, or multiple parallel WSs (see Fig. 1).

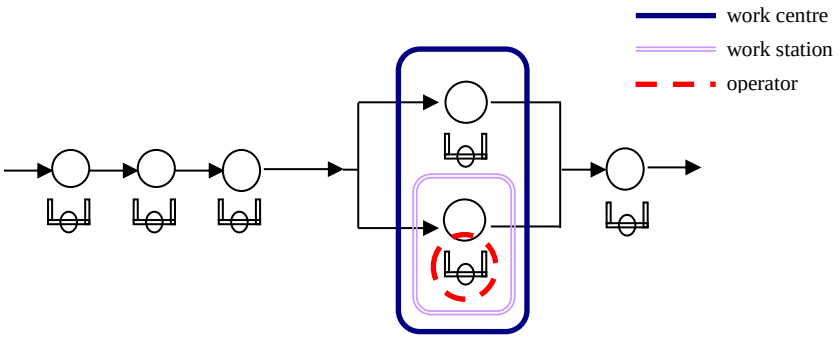


Figure 1. An assembly line with parallel workstations.

'Paralleling' means that when a WC may consist of two or more workstations. In this case, the tasks assigned to the WC are not shared among the WSs, but each WS performs all of them. Thus, an increment of

production capacity of the WC is obtained through the addition of one (or more) WS and the related operator who performs the same set of tasks. Each task requires a different piece of equipment to be performed, to which is associated a cost *CE*. So if one station is parallelized there is an extra cost due to equipment duplication.
The line is asynchronous, that is as well as blockage and starvation are possible. There are no buffers between WCs. One WC with multiple WSs is considered busy if every WS inside is busy. If a WS finishes its work on a workpiece while the subsequent WC is still busy, the workpiece cannot move on, and remains in the current WS keeping it busy ('blocking after processing' policy); the WS will be released (i.e. will be able to process another workpiece) only when the workpiece leaves the WS. The first WC is never starved (there is always raw material for the first WC) and the last station is never blocked (there is always storage space for the finished product).
Models enter the line with a fixed sequence, able to comply with the demand proportion for each model. Set-up times between models are considered negligible.

3.4 Constraints

The way tasks are assigned to WC is constrained by the precedence relations among tasks (which impose a partial ordering, reflecting which task has to be completed before others).

3.5 Objective (Fitness function)

The objective to minimize is the Normalized Design Cost (*NDC*), introduced by Tiacci et al. (2006). The *NDC* takes into consideration the two fundamentals aspects between the trade-off exists: the economic aspect and the performance aspect.
The economic aspect is evaluated through the Design Cost, that is the total annual cost for labour and equipment costs of the line configuration:

$$DC = \sum_{k=0}^{P-1} \left(CE \cdot E_k + CW \cdot W_k \right) \tag{1}$$

The performance aspect is taken into consideration introducing the realized cycle time (ct_{eff}), i.e. the actual average cycle time of the line, that also correspond to the throughput inverse. The realized cycle time can be different from the imposed one (*ct*), but if ct_{eff} is higher than the *ct* a penalty in it the evaluation of the line configuration has to be considered. This is attained by increasing the Design Cost value if $ct_{eff} \geq ct$:

$$NDC = \begin{cases} DC & if \quad ct_{eff} < ct \\ DC \cdot \left[1 + z \cdot \left(\dfrac{ct_{eff} - ct}{ct} \right)^2 \right] & if \quad ct_{eff} \geq ct \end{cases} \tag{2}$$

where *z* is a penalty factor that can be assigned according to the required observance of *ct*. Note that for z equal to infinity the constraint related to the imposed cycle time becomes strict.

4 Coupling GA and event-oriented simulation

In this section, a genetic algorithm approach to solve the defined problem is described. The GA is coupled with an object event/object oriented simulator that is used to calculate the individuals' fitness (Section 4.5).

4.1 Representation and decoding

Each individual is represented through a couple of chromosomes. They both are vectors having a number of elements equal to *N*. The first chromosome contains an ordered sequence of all tasks. Its order corresponds to the same order they are assigned to WCs. The second chromosome establishes the WC the task is given to. In this chromosome the WCs are numbered in ascending order starting from the first one in the line, to the last one. So when a task is assigned to a different WS with respect to the preceding task, the corresponding value in the second chromosome will be incremented by one (see Fig. 2).

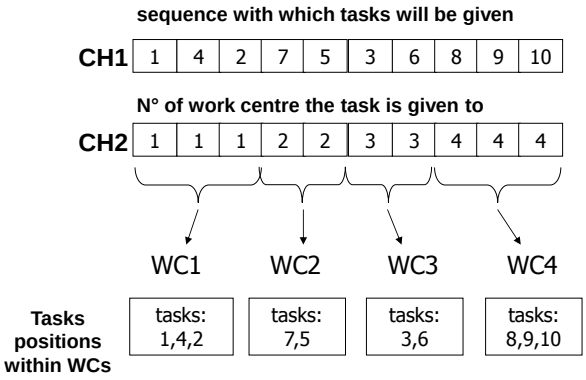


Figure 2. Chromosomal representation.

Once all tasks have been assigned to WCs, it remains to calculate the number of workers (i.e. WSs) in each WC. This is obtained by imposing a certain probability of completion, *pc*, of all tasks assigned to each WC. By summing the average task times and variances of tasks assigned to a WC, and using the inverse normal distribution function, it is possible to calculate the time needed to complete all tasks with a probability *pc*. Indicating with T_{limit} this time, the number of workers W_i assigned to the WC will be equal to the minimum integer value so that $ct \cdot W_i \geq T_{limit}$.

4.2 Initial population

The initial population is created randomly, respecting the precedence constraints. The first chromosome is generated following the adopted by Kim et al. (1996):

Step 1. Form an initial available set of tasks having no predecessors, and create an empty string (i.e. an empty chromosome).
Step 2. Terminate, if the available set is empty. Otherwise, go to Step 3.
Step 3. Select a task from the available set at random, and append it to the string.
Step 4. Update the available set by removing the selected task and by adding every immediate successor of the task if all the immediate predecessors of the successor are already in the string. Go to Step 2.

Note that in Step 4 the available set is updated with tasks satisfying precedence constraints so that it always ensures the generation of a feasible sequence.
To generate the second chromosome, a value equal to 1 is initially assigned to the first position of the vector. Then the subsequent position can assume the same value as the previous one, or a value increased by 1. This option is made at random, with a probability of 50%.

4.3 Crossover

The Crossover operator is applied during the reproduction phase. Two individuals, namely p1 and p2 (parent1 and parent2), generate two children s1 and s2. The crossover operator involves both chromosomes.

A single point crossover operator is applied for the first chromosome (Fig. 3a). A cut point is chosen randomly in one parent (p1), and the elements after the cut point are copied into the same position of the offspring s1. These elements are removed from the other parent p2, and its remaining elements are copied into the initial positions of s1 in the same order as they appear in p2. In this way, a feasible sequence that satisfies precedence restrictions is generated, avoiding also duplication or omissions of tasks.
For the second chromosome (CH2) the following method is adopted (Fig. 3b). The chosen cut point is the same as the one of the first chromosome. The values in the preceding positions are copied in the same order from p1 to s1. Let *d* represent the difference between the last copied value of p1 and the corresponding value of p2. The values of the positions to the right of the cut point of the first chromosome of p2 are copied in the same positions in s1 after being increased by *d*. Note that in this way the first value after the cut-point in s1 can assume or a value equal to preceding gene (position), or a value increased by 1.
The second offspring s2 is generated inverting p1 and p2 roles.

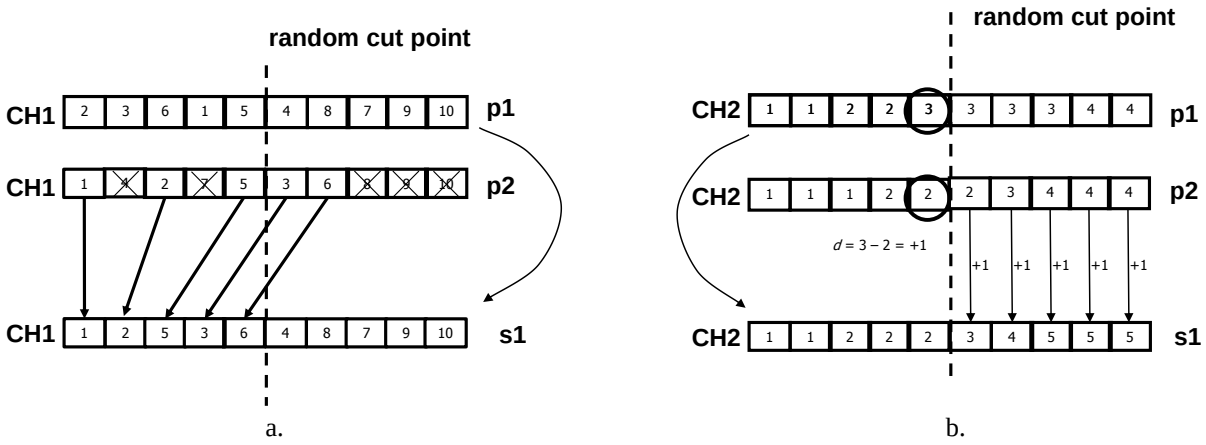
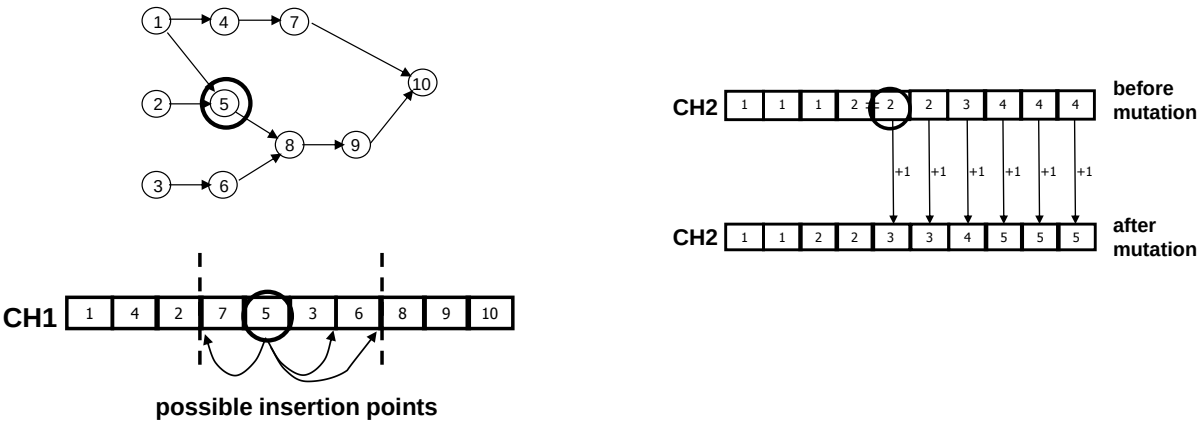


Figure 3. Crossover operator.

4.4 Mutation

The mutation operator is also applied to both chromosomes. For the first chromosome the feasible insertion method is adopted (Kim et al., 1996). A task is randomly selected (eg. task 5 in Fig. 4a) and its position is swapped with another feasible position in the chromosome. Because in a feasible sequence the selected task must follow all of its immediate predecessors and precede all of its immediate successors, the potential positions are consecutive, forming a substring in the parent. Among the potential insertions positions, one is randomly chosen.
Regarding the second chromosome, the mutation takes place in the following way: a position in the chromosome is randomly chosen. Let *p* represent the value contained in this position, and *l* the value of the preceding position. Naturally will be $l \leq p$. If $l = p$, then all the values from *p* (included) onwards are increased by one (Fig. 4b). This means that one WC has been divided into two. If $l < p$ then all the values from *p* onwards are decreased by one. This means that two WCs have been united.



a. b.

Figure 4. Mutation operator.

4.5 Fitness evaluation: the Assembly Line Simulator (ALS)

The fitness of an individual is the Normalize Design Cost (eq. (2)). When an individual is decoded, all the info related to the number of WCs in the line, the number of workers in each WC, and the tasks (and thus the pieces of equipment) assigned to each WC are known. We will use the term 'line configuration' to refer to this set of information. So the *DC* of the line configuration associated to the individual can be calculated straightforwardly through eq. (1).

The evaluation of the effective cycle time ct_{eff} of the line configuration (needed to calculate the *NDC*) is performed by exploiting the integration capabilities of an Assembly Line Simulator (ALS) (Tiacci, 2012). ALS is a parametric event-oriented simulator, developed in Java, capable to immediately calculate the average cycle time of a complex line by simply receiving as inputs the task times durations, the line configuration (number of WCs, of parallel workstations in each WC, of tasks assigned to each WC), and the sequence of models entering the line. Thanks to its object-oriented architecture, different lines configurations can be created by the mean of very simple array representations, so that the time required for building the model is zeroed. ALS has been developed following an event oriented paradigm: the simulation run is performed only by events scheduling, rescheduling, cancelling etc.; by avoiding the use of 'processes', execution times are kept very low with respect to the analogous process-oriented version (Tiacci and Saetta, 2007), arriving to outperform Arena.

These characteristics allow it to be effectively coupled to those algorithms and procedures where numerous variants of line configurations have to be simulated, and the evaluation of a fitness function (which includes some line performances indicator, such as the cycle time in our case) has to be performed several times. ALS allows to overcome the limit of using traditional measures (not simulation-based) of the performance parameter (the line throughput or, equivalently the effective cycle time) that are poorly correlated to its real value (as f.e. the approaches that will be described in Sections 5.1.1 and 5.1.2).

To embed ALS in the GA, the genetic algorithm procedure has been developed using the same programming environment of the simulator (Java). ALS has then been imported as a library (for details refer to ALS documentation). ALS is contained in a package named `lineSimulator`, which provides the public class `Simulation`, whose constructor's arguments represent the line configuration that has to be simulated. When a Simulation object is created, the simulation model of the assembly line is created, the simulation is performed and outputs (f.e. average cycle time, average flow time, average WIP) are stored in the object attributes. To evaluate the average cycle time ct_{eff} of each line configuration related to an individual, we set the simulation run performed by ALS to end when 3000 loads have been completed.

The methodology to generate different sequences of models entering the line has been chosen to reflect the different nature of the balancing and the sequencing problems. The balancing problem has a long-term nature, concerning the design phase of the assembly line. The sequencing problem has a short-term nature, concerning the daily production. In other words, the exact production sequence is not usually already known

during the design stages of the assembly line (balancing problem), but only a few days before production starts.For this reason the different design alternatives during the searching procedure are evaluated just assuming a determined demand proportion among models, without specifying any exact sequence. This is carried out by exploiting the possibility provided by ALS to generate random sequences of models while respecting a determined demand proportion among models.
Once obtained c_{teff} it is possible calculate the individual fitness function through eq. (2).

4.6 GA iterations and resulting general structure

The GA structure is depicted in Fig. 5. After creating the initial population of N individuals, during the reproduction all the individuals are randomly coupled. Each couple of parents generates a couple of children (as described in Section 4.4) that are added to the initial population. Each individual has now a certain probability *mr* (mutation rate) of undergoing a mutation (Section 4.5). Each mutated individual is added to the population, but the original one is also maintained in the population.
In order to maintain high population diversity, an elimination strategy has been implemented during the selection phase. Let define the 'distance' between two individuals as the absolute difference between their fitness values (Section 4.3). This phase provides a spacing procedure among individuals so that the distance between any two individuals in the resulting population is always higher than a threshold value Δf. At this purpose, the population is sorted. Firstly, the individual with the best fitness is considered as reference, and all the individuals whose distance from that one is lower than Δf are eliminated from the population. The closer individual that satisfies the threshold is then taken as reference, and the procedure continue iteratively until all the population have been spaced.
After terminating the spacing procedure, only the first N individuals of the resulting population survive and pass the next generation. After a number G of iterations the algorithm stops and the individual with the best fitness is considered the final solution.

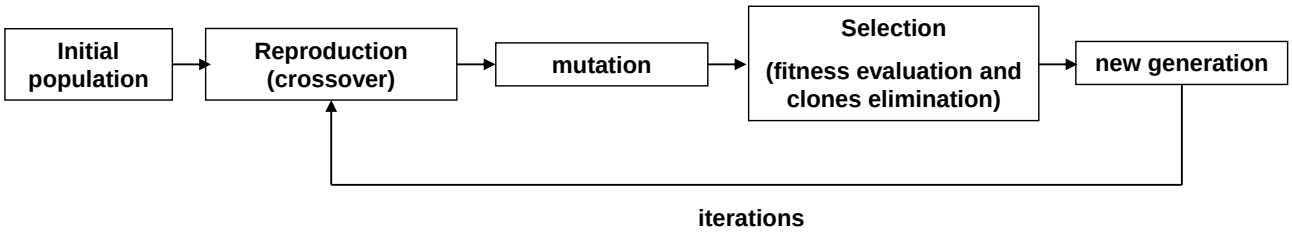


Figure 5. GA iterations and general structure.

In our implementation we adopted a strategy to dynamically set the threshold value Δf at each iteration. For the first iteration, Δf is set equal to a percentage d_1 of the fitness of the best individual. Than it decreases linearly in such a way that equals zero after a certain percentage d_2 of the total number of iterations G, and remains zero for the residual iterations:

$$\Delta f\,(iter)=[bestfit \cdot d_1 \cdot (1 - iter/(d_2 \cdot G))]^{+} \tag{3}$$

where *bestfit* is the fitness value of the best individual in the current population, and *iter* is the current iteration number. In this way, high population diversity is initially guaranteed, in order not to be too early trapped in local optima. On the contrary, a local search behavior is induced in the last iterations by not discarding clones and very similar individuals, and refining the best solution by searching among the closest neighbors.

5 Design of experiment

5.1 Experiment 1

The proposed approach has been tested on 5 problems (Bartholdi, 1993; Gunther et al., 1983; Rosenberg and Ziegler, 1992; Scholl, 1993; Wee and Magazine, 1981) of different size, varying the number of tasks between a minimum of 25 and a maximum of 297. With the aim of exploring a larger number of cases, each of the 5 problems has been solved for 4 different cycle times (see Table 3). Furthermore, each of the 20 problems obtained has been solved for two different mixes (M = 2 products and M = 4 products), obtaining a total number of 40 different explored cases.

The problems precedence diagrams can be downloaded from the dataset of Scholl (2012), in which the number of tasks and precedence constraints between tasks are defined for a single model. The task times t_{ij} for both the 2 and the 4 model instances that have been created for our study are available at (Tiacci, 2014). A coefficient of variation cv = 0.3 has been utilized to obtain task times standard deviations. As it was mentioned in Section 4.5, during the searching procedure only the demand proportion among models has been assumed to be known. In 2 models problems, the demand has been equally distributed with 50% for m#1 and 50% for m#2. In 4 models problems, the following mix has been assumed: 50% m#1, 20% m#2, 20% m#3, 10% m#4. Sequences of models have been randomly generated while respecting these demand proportions. The annual cost of a worker (CW) and of a piece of equipment (CE) have been considered respectively equal to 30000 and 3000 €/y. The values of the genetic algorithm parameters utilized in the study are N = 40; mr = 0.55; G = 100, pc = 0.5, d_1 = 0.03%, d_2 = 70%.

The algorithm has been compared, on the same problems instances, with other two approaches proposed in literature, namely a Simulated Annealing approach (McMullen and Frazier, 1998) and a Genetic Algorithm approach (Tiacci et al., 2006).

The final solutions of each approach have been evaluated on the basis of their Normalized Design Cost, considering a penalty coefficient z = 20 (see eq. (2)). To obtain the *NDC* value the final solutions has been simulated 10 times using ALS, and the average value of the cycle time has been used, together with the number of workers and pieces of equipment, to calculate the *NDC* trough eq.(2). The simulations carried out through ALS, for the final evaluation of solutions, consider deterministic sequences of models entering the line: the sequence 1212121212 for 2 models problems; 1213141213 for 4 models problems (McMullen and Frazier, 1998). This is to reflect the fact that it is practically impossible to have the exact information about the models sequence at the time of the assembly line design decision, but that an exact sequence is determined when production starts. These sequences have been chosen so that the products are not being introduced into the production system at a steady rate, as would be desirable in many JIT systems. Using these specific sequences for the evaluation of the final solutions, instead of the random sequences used during the searching procedure, allows drawing more meaningful conclusions about the robustness of the solutions found by the algorithm.

Because of the stochastic nature of the algorithms, the solution found for an instance changes if the procedure is repeated using a different random seed. For this reason, each of the 40 instances has been solved 10 times, and results here reported are related to the best performance chosen among these five solutions.

The difference that we want to outline between the approach presented here in and the two studies from the literature is related to the objective function evaluation. In these studies, *indirect* measures of the cycle time have been utilized to compare different solutions during the search procedures. This forced to create objective functions that do not contain explicitly the cycle time value, and thus cannot be equal to the real objective of the problem as defined by eq. (2). It is noteworthy that, on the contrary, by embedding ALS in the genetic algorithm proposed here in, it is possible to calculate during the procedure any desired objective function, as *NDC*, that explicitly contains the effective cycle time value.

In the next two subsections the objective function utilized in the two studies are reported.

5.1.1 Simulated annealing approach

We have implemented all the six different versions of the Simulated Annealing algorithm proposed by McMullen and Frazier (1998), which differ from one another in the objective function that are minimized (see Table 1).

type		name	description
single objective	minimize	DC	design cost
		S	smoothness index
		L	probability of lateness across all work centers
multi objective	minimize	DC&L	= DC + 1.67 * L
		DC&3L	= DC + 3 * 1.67 . L
		3D&L	= 3 * DC + 1.67 * L

Table 1. Objective functions of the Simulated Annealing algorithm

The first three are single-objective functions, whereas the second three are multi-objective and weigh differently (1:1, 1:3, 3:1) the cost parameter (DC) and the performance parameters (L and S). For more details about S (which is an index intended to distribute work into the workstations as evenly as possible), L (representing the probability of lateness across all WCs) and the normalizing weight equal to 1.67, the latter which appears in the three multi-objective functions, refer to the McMullen's publication. The number of iterations for each temperature has been set equal to the number of tasks of the problem. All the other values of the setting parameters of Simulated Annealing that are not quoted by us (such as for instance annealing temperature, cooling rate, etc.) are the same assumed by McMullen and Frazier (1998).

5.1.2 Genetic Algorithm with traditional fitness function

In the study presented by Tiacci et al. (2006) the declared objective function of the problem was the *NDC*. However, due to the impossibility to correctly estimate the cycle time of a solution when calculating its fitness, a different function was used to evaluate each individual during the procedure:

$$fit = k \cdot LE_p + MV_p \tag{4}$$

This fitness function, that has to be maximized, takes into account two indexes, which are concerned with the two different aspects of the solution: the line efficiency (*LE*) and the model variability (*MV*). *LE*, introduced by Driscoll and Thilakawardana (2001), represents, with a range from 0 to 100%, the attainment of a good utilization of the line, being a direct measure of the economic convenience of a solution. *MV* has been introduced by Bukchin (1998), and it is a measure of the variability of assembly times of a certain model assigned to different WCs. Bukchin (1998) showed that the model variability is one of the indexes with the highest (inverse) correlation to simulated throughput in a mixed model assembly line. In their original formulation, both *LE* and *MV* did not take into account paralleling. LE_p and MV_p in eq. (4) are new formulations, proposed by Tiacci et al. (2006), that take into account paralleling .

Because MV_p is an a-dimensional parameter that may not necessarily have a range between 0 and 100%, as LE_p does have, they also introduced the coefficient *k*, which allows to weigh the two indexes in the most coherent way. After some tuning operations on the algorithm, they assigned to *k* the value of 4.

5.2 Experiment 2

The structure of Experiment 2 is the same of experiment 1, but refers to a more realistic scenario, in which many of the simplifying assumptions of the first experiment have been relaxed, such as a limited number of models, the same pieces of equipment costs and the same coefficient of variation for all tasks. To do this a

new instance has been generated, taking the same precedence diagram of the most complex instance in literature, the one from Scholl with 297 tasks.
Firstly a coefficient of variation cv_i for each task i has been randomly generated from a uniform distribution with range [0.1, 0.5]. Analogously, different pieces of equipment costs CE_i have been randomly generated from a uniform distribution with range [100, 10.000].
Then, average completion times for 200 different models have been randomly generated, taking the task time t_i of the Scholl's instance as reference. To do this, different degrees of customization and optional equipment have been considered. Basically, tasks fall into three different categories (Golz et al., 2012):

- common tasks have to be performed into every model. In this case the average task time is the same for all models, and equal to t_i;
- OR-tasks are optionally performed depending on the customer specifications. This kind of task are modelled assuming that they are performed only for one model (randomly chosen), with an average completion time t_i. For all the other models the completion time is set equal to 0;
- XOR-tasks represent variants that are performed in every model. These tasks have been generated by randomly assigning to each model an average completion time uniformly distributed in the range $[0.7t_i, 1.3t_i]$.

In this experiment, the degree of variability is defined by the fraction of the three different task categories as low, medium and high. In the low task variability instances the distribution of tasks is defined as 70% for common and 15% each for OR and XOR-tasks, while these fractions are 50 and 25% in the medium and 30 and 35% in the high part variability instances.

Low	Medium	High
70% common tasks	50% common tasks	30% common tasks
15% OR tasks	25% OR tasks	35% OR tasks
15% XOR tasks	25% XOR tasks	35% XOR tasks

Table 2. Degree of tasks' variability

The demand proportion of each model and the exact model sequences for the final evaluation has been generated for each instance in the following way. An exact sequence is firstly generated by appending each model to the sequence a number of times drawn from a uniform distribution with range [1, 4]. Then, the obtained sequence is randomly shuffled. The demand proportion for each model is calculated considering the same proportion observed in the deterministic sequence. All the other settings are the same of experiment 1. The instances data can be downloaded from . (Tiacci, 2014)

6 Results

Table 3 shows the results obtained with the GA coupled with ALS for the 40 instances of the experiment 1. The CPU time needed to solve each instance, with respect to algorithm that use non-simulation-based measures of throughput, is higher. This is due substantially to the extra-time required to simulate the line configuration each time that the fitness function of an individual is evaluated. However, the total CPU times required to solve an instance ranges from about 100 to 1600 sec., depending on the instance complexity. This amount of time is acceptable considering the long-term nature of the problem and the expensiveness of the decisions that have to be taken (design of an assembly line).
It is possible to note as the effective cycle time obtained is quite near to the imposed one, and always slightly higher (3.89% on the average). This means that the algorithm takes advantage of the low penalty applied to the *DC* for very little *ct* overshooting, to find very efficient solutions.

Table 4 shows the solution found by the algorithm for the 35 tasks, 2 models problem by Gunther with an imposed cycle time equal to 30. The line configuration provides 12 WCs, in three of which paralleling is utilized. All the solutions related to each of the 40 problems are available at (Tiacci, 2014).

Tables 5, 6 and 7 show the comparison between the approach presented here-in (that is indicated as GA+ALS), the Simulated Annealing (SA) approach in all its 6 variants related to the objective functions described in section 5.1.1, and the Genetic Algorithm with traditional fitness function described in section 5.1.2 (indicated as GA+Trad.).

The superiority of the GA+ALS approach is evident from Table 5, in which the *NDC* is reported. The last column of the table (%diff) shows the percentage difference between the *NDC* obtained by the GA+ALS approach and the best result (highlighted in bold) obtained by the other two approaches (SA and GA+Trad.) for the corresponding problem. The GA+ALS approach obtains always the lower values, with improvements that reach almost the 20% in some instances.

Table 6 is related to the effective cycle times (ct_{eff}) of the solutions. Defining the Absolute Percentage Deviation (*APD*) between the effective cycle time and the imposed cycle time, $APD = |(ct_{eff} - ct)/ct|$, results show that the GA+ALS approach obtains, on the average, the lower values than all the other approaches. Furthermore, the ct_{eff} values obtained by GA+ALS are close to *ct* for almost all the 40 considered issues, as demonstrated by the maximum value of the *APD* (3.44%). On the contrary, the other approaches provide in at least some cases higher deviations values, as the higher maximum value of their *APD* demonstrates. It is noteworthy that in some cases SA provides effective cycle times that are far below the planned *ct*, but the corresponding solutions are very costly (Table 7).

Regarding the cost parameter *DC* (Table 7), the better results are almost always obtained from GA+trad and SA+3DC&L. However, the effective cycle time achieved in the corresponding instances by these two algorithms largely exceeds the planned one, and this penalizes the evaluation of the solution in terms of *NDC*.

The superiority of the GA+ALS approach with respect to the GA+Trad is remarkable. As described in section 5.1.2, the GA+Trad has been developed declaring the *NDC* as objective to minimize. Thus, the fitness function tuning (i.e. the determination of the weighting factor *k* in eq. (4) through a trial and error procedure) has been performed with this scope. Despite this, GA+ALS outperforms the traditional approach in all the instances. This fact outlines one of the main drawbacks of the traditional approaches: using indirect measures of cycle time forces the adoption of fitness function that does not coincide with the real objective; and this in turns requires preliminary trial and error tuning procedures to 'align' the fitness function as much as possible to the real objective. However, of course, this alignment will never bring to better result than using a fitness function that corresponds to the real objective.

Tables 8-10 show the results related to experiment 2. The values here refer, as in the preceding experiment, to the best solutions obtained by each algorithm. Figures 6-7 shows the results related to the full factorial design of experiment 2, considering all the repetitions made for each combination of factors. The factors considered are: the algorithm utilized ("*alg*"), the task times variability ("*Variability*"), and the imposed cycle time ("*ct*"). We reported the main effect plots of each factors on *NDC*, *APD* and *DC* (Figure 6), and the interaction effect plot between the *alg* and *ct* (Figure 7). All the other interaction effects among factors resulted non-significant.

Analyzing the results some considerations can be drawn. Even in this case, the lower values of the *NDC* are obtained in all the instances by GA+ALS, which also outperforms all the other algorithms in terms of average *APD*. Only SA+L obtains similar performances in terms of *APD*, but through much more costly line configurations.

The tasks variability does not seem to have an evident influence on the quality of the solution. This is valid both in terms of the main performance indicator, the *NDC*, and in terms of its components, i.e. the *DC* and the respect of the imposed cycle time (measured through the *APD*).

On the contrary, the factor that seems to have the higher impact on performances is the imposed cycle time. Lower values of the imposed *ct* lead to worst performances in terms of *NDC*, *APD* and *DC*. A possible

explanation for this behavior is that, when cycle time is lower, the combinations of tasks that can be allocated into a WorkCentre in a feasible way decrease. The consequent reduction of the solution space may be overcome using parallel workstations. Paralleling on the other hand introduces extra-cost due to equipment duplications, and increases the difficulty to evaluate the effective cycle time of the line through non-simulation based metrics.

If this is true, due to the simulation-based nature of the objective function evaluation, GA+ALS is expected to perform better than the other algorithms in terms performance decrement. This is exactly what can be observed from the interactions plot between *ct* and the *alg* for *NDC* (Figure 7). It is clear from the graphic that the performance decrement, observed when *ct* decreases, is much less evident when GA+ALS is utilized, with respect to the all the other algorithms.

Problem	**size**	***M***	***ct***	ct_{eff}	*DC* (x1000)	*NDC* (x1000)	CPU time* (sec)
Roszieg	25	2	18	18.0	345	345	102
			14	14.0	441	441	115
			11	11.2	540	545	113
			9	9.2	654	658	126
Gunther	35	2	54	55.6	447	455	140
			41	41.8	609	614	141
			36	37.1	654	666	159
			30	31.1	759	781	166
Wee-mag	75	2	34	34.8	2109	2134	456
			28	28.8	2478	2517	398
			24	24.6	3015	3053	381
			21	21.2	3360	3366	395
Barthold	148	2	470	482.8	894	907	537
			403	406.3	996	997	544
			378	376.0	1050	1050	558
			351	358.8	1092	1103	568
Scholl	297	2	1515	1549.3	2871	2900	1513
			1394	1423.6	3084	3112	1542
			1301	1323.2	3198	3217	1539
			1216	1236.2	3429	3448	1605
Roszieg	25	4	18	18.1	345	345	104
			14	14.1	441	442	116
			11	11.3	543	551	112
			9	9.2	657	667	126
Gunther	35	4	54	55.6	453	461	140
			41	41.4	609	610	160
			36	36.8	669	675	160
			30	30.9	771	784	159
Wee-mag	75	4	34	34.5	2118	2128	492
			28	28.6	2532	2554	400
			24	24.9	2973	3048	409
			21	21.2	3345	3353	385
Barthold	148	4	470	469.5	924	924	542
			403	416.3	984	1005	541
			378	383.1	1068	1072	530
			351	358.9	1092	1103	540
Scholl	297	4	1515	1560.0	2901	2952	1501
			1394	1429.6	3066	3106	1520
			1301	1328.2	3231	3259	1536
			1216	1254.5	3423	3492	1567

*The experiment has been performed on a computer with an Intel Core Processor i5-2400 (3.1 GHz), with Java 7 Update 25, running under Windows 7 Professional Edition.

Table 3. Experiment 1: results for the 40 instances of the GA approach coupled with ALS (GA+ALS)

WC#	WS assigned (W_i)	tasks assigned
0	1	16, 0
1	2	9, 11
2	1	1, 4, 5, 6
3	1	7, 13, 8
4	1	17, 14, 2
5	1	12
6	1	3
7	2	18, 19, 10
8	1	15
9	1	20, 21, 24, 25
10	1	22, 29, 30
11	1	31, 23, 26, 33
12	3	32, 27, 34, 28
ct_{eff} = 37.09; *DC* = 654000; *NDC* = 665928		

Table 4. GA+ALS solution for Gunther 35, 2 models, *ct* = 36.

Problem	**size**	***M***	***ct***	GA+ ALS	Simulated Annealing						GA+ Trad.	%diff
					DC	S	L	DC&L	DC&3L	3DC&L		
Roszieg	25	2	18	345	397	486	404	379	377	**347**	364	-0.7%
			14	441	**478**	558	512	516	518	536	583	-8.3%
			11	545	**571**	717	665	617	615	581	667	-4.7%
			9	658	**696**	834	757	734	731	813	857	-5.8%
Gunther	35	2	54	455	499	648	541	492	491	**487**	519	-7.0%
			41	614	**675**	756	740	750	706	690	842	-10.0%
			36	666	848	849	771	775	774	**768**	979	-15.3%
			30	781	**886**	993	906	1058	1054	1056	1086	-13.4%
Wee-mag	75	2	34	2134	2686	2460	**2403**	2791	2783	2918	2863	-12.6%
			28	2517	3768	**2793**	2896	3395	3386	3268	4088	-11.0%
			24	3053	4126	**3375**	3379	4068	3976	4611	5362	-10.6%
			21	3366	3711	3756	3646	3549	**3548**	4005	3970	-5.4%
Barthold	148	2	470	907	1194	1497	1305	1134	1194	1224	**1030**	-13.5%
			403	997	1284	1809	1404	1284	1240	1344	**1168**	-17.1%
			378	1050	1252	1539	1458	1320	**1146**	1233	1302	-9.1%
			351	1103	1203	1854	1522	1271	1394	1335	**1196**	-8.5%
Scholl	297	2	1515	2900	3321	4437	5291	3268	3329	**3214**	3367	-10.8%
			1394	3112	3461	5010	5498	3480	**3442**	3681	3609	-10.6%
			1301	3217	3911	5088	6007	3754	3990	3909	**3552**	-10.4%
			1216	3448	**4068**	5766	6831	4152	5075	4545	4137	-18.0%
Roszieg	25	4	18	345	388	486	405	383	384	381	**370**	-7.2%
			14	442	**488**	612	513	559	560	543	586	-10.5%
			11	551	**572**	714	666	624	628	590	686	-3.8%
			9	667	**698**	834	758	755	755	774	924	-4.7%
Gunther	35	4	54	461	**512**	657	541	522	520	537	565	-11.1%
			41	610	763	798	741	762	760	**657**	899	-7.7%
			36	675	776	849	**771**	812	809	919	974	-14.2%
			30	784	**890**	993	907	1110	1117	1075	1143	-13.5%
Wee-mag	75	4	34	2128	2707	2418	**2400**	2537	2535	2822	3155	-12.8%
			28	2554	3806	**2830**	2901	3424	3425	3363	4752	-10.8%
			24	3048	4191	**3385**	3389	4023	4017	4181	5440	-11.1%
			21	3353	3718	3756	3643	3463	**3459**	4056	4045	-3.1%
Barthold	148	4	470	924	1068	1359	1173	**965**	985	970	973	-4.4%
			403	1005	1064	1386	1382	**1040**	1087	1208	1107	-3.5%
			378	1072	1194	1584	1434	1137	1260	1187	**1124**	-4.9%
			351	1103	**1103**	1683	1509	1155	1199	1195	1194	0.0%
Scholl	297	4	1515	2952	**2959**	4245	5291	3117	3055	3272	3224	-0.2%
			1394	3106	**3447**	4860	5547	3513	3614	3456	3697	-11.0%
			1301	3259	**3819**	5250	6005	4410	4473	4013	3870	-17.2%
			1216	3492	4127	5772	6836	**3868**	4161	4455	4333	-10.8%

Table 5. Experiment 1: Normalized Design Cost, *NDC*.

Problem	**size**	***M***	***ct***	GA+ ALS	Simulated Annealing						GA+ Trad.
					DC	S	L	DC&L	DC&3L	3DC&L	
Roszieg	25	2	18	18.0	20.0	14.0	18.3	19.5	19.5	18.3	19.6
			14	14.0	14.9	12.9	14.2	15.4	15.5	15.6	16.1
			11	11.2	11.4	9.7	11.2	12.1	12.1	11.7	12.4
			9	9.2	9.5	7.9	9.5	9.7	9.7	10.1	10.3
Gunther	35	2	54	55.6	58.4	47.5	56.8	57.8	57.8	58.2	59.9
			41	41.8	44.8	35.9	43.8	45.7	44.5	44.6	47.7
			36	37.1	40.7	8.0	35.4	39.1	39.0	39.6	42.4
			30	31.1	32.1	26.6	30.7	34.5	34.4	34.6	35.0
Wee-mag	75	2	34	34.8	38.4	30.6	35.3	38.8	38.8	39.3	39.4
			28	28.8	32.8	27.4	29.2	31.9	31.9	32.0	33.9
			24	24.6	27.9	25.3	24.7	27.9	27.5	28.6	29.8
			21	21.2	22.6	20.9	21.7	22.2	22.2	23.1	23.2
Barthold	148	2	470	482.8	471.2	310.9	452.2	442.0	467.8	455.9	516.0
			403	406.3	402.0	320.2	391.9	402.0	413.4	396.6	449.3
			378	376.0	386.6	284.5	376.7	364.2	388.6	351.1	424.5
			351	358.8	354.8	274.4	353.1	361.6	369.5	369.4	379.6
Scholl	297	2	1515	1549.3	1690.9	1427.4	1563.6	1691.9	1683.8	1704.1	1695.6
			1394	1423.6	1535.3	1328.0	1449.1	1559.5	1535.2	1600.4	1560.1
			1301	1323.2	1474.3	1224.3	1314.7	1440.6	1471.1	1491.1	1429.4
			1216	1236.2	1362.5	1141.7	1311.3	1368.2	1436.9	1426.6	1375.8
Roszieg	25	4	18	18.1	19.7	14.0	18.3	19.6	19.6	19.6	19.7
			14	14.1	15.1	11.9	14.2	15.7	15.8	15.6	16.1
			11	11.3	11.4	9.3	11.2	12.1	12.1	11.8	12.4
			9	9.2	9.5	8.0	9.5	9.8	9.8	9.9	10.5
Gunther	35	4	54	55.6	58.9	47.4	56.8	59.4	59.4	60.1	60.6
			41	41.4	45.8	34.7	43.8	45.9	45.9	44.1	48.5
			36	36.8	39.2	32.9	35.4	40.2	40.2	41.6	42.0
			30	30.9	32.2	26.6	30.7	34.8	34.8	34.7	35.4
Wee-mag	75	4	34	34.5	38.4	31.6	35.3	37.9	37.9	38.8	40.1
			28	28.6	32.9	28.2	29.3	32.0	32.0	32.1	35.0
			24	24.9	28.0	25.3	24.7	27.7	27.6	28.1	29.9
			21	21.2	22.6	20.9	21.7	21.9	21.9	23.3	23.3
Barthold	148	4	470	469.5	521.1	363.4	471.9	505.9	509.4	512.4	507.4
			403	416.3	424.9	311.0	419.4	414.7	430.8	453.0	443.1
			378	383.1	413.0	288.9	374.4	410.2	423.4	415.3	408.0
			351	358.9	361.6	325.6	327.3	371.3	375.6	370.5	378.7
Scholl	297	4	1515	1560.0	1615.2	1405.0	1563.6	1647.5	1620.6	1717.3	1670.8
			1394	1429.6	1531.9	1329.9	1456.8	1546.8	1566.2	1567.3	1572.9
			1301	1328.2	1456.5	1228.0	1313.7	1500.8	1518.3	1501.1	1466.3
			1216	1254.5	1365.0	1174.9	1311.6	1348.5	1367.0	1424.2	1393.2
Avg. *APD* from *ct*				2.02%	8.92%	13.28%	3.37%	9.84%	10.12%	11.20%	13.87%
Max. *APD* from *ct*				3.79%	17.38%	77.81%	7.86%	16.05%	18.17%	19.14%	24.91%

Table 6. Experiment 1: effective cycle time, ct_{eff}

Problem	**size**	***M***	***ct***	GA+ ALS	Simulated Annealing						GA+ Trad.
					DC	S	L	DC&L	DC&3L	3DC&L	
Roszieg	25	2	18	345	315	486	402	330	330	345	315
			14	441	438	558	510	426	426	426	405
			11	540	558	717	660	519	519	534	510
			9	654	654	834	717	654	654	630	600
Gunther	35	2	54	447	441	648	513	447	447	435	420
			41	609	576	756	678	594	615	597	546
			36	654	630	849	771	678	678	639	603
			30	759	804	993	897	732	732	720	693
Wee-mag	75	2	34	2109	2019	2460	2334	1986	1986	1968	1905
			28	2478	2376	2793	2787	2442	2442	2331	2166
			24	3015	2682	3192	3327	2685	2772	2661	2469
			21	3360	3339	3756	3567	3330	3330	3324	3231
Barthold	148	2	470	894	1194	1497	1305	1134	1194	1224	864
			403	996	1284	1809	1404	1284	1224	1344	924
			378	1050	1239	1539	1458	1320	1128	1233	999
			351	1092	1200	1854	1521	1248	1320	1266	1056
Scholl	297	2	1515	2871	2616	4437	5184	2568	2667	2451	2622
			1394	3084	2871	5010	5331	2715	2856	2559	2811
			1301	3198	2886	5088	5994	3051	2973	2739	2973
			1216	3429	3153	5766	6084	3162	3057	2841	3075
Roszieg	25	4	18	345	330	486	402	330	330	330	315
			14	441	438	612	510	426	426	435	405
			11	543	558	714	660	522	522	534	510
			9	657	654	834	717	657	657	651	603
Gunther	35	4	54	453	441	657	513	435	435	429	435
			41	609	597	798	678	594	594	588	537
			36	669	672	849	771	636	636	618	624
			30	771	804	993	897	735	735	720	690
Wee-mag	75	4	34	2118	2019	2418	2334	2010	2010	2010	1917
			28	2532	2373	2826	2787	2442	2442	2349	2121
			24	2973	2682	3192	3327	2748	2748	2634	2481
			21	3345	3339	3756	3567	3330	3330	3294	3240
Barthold	148	4	470	924	864	1359	1173	864	864	834	864
			403	984	1005	1386	1338	1023	993	924	924
			378	1068	1020	1584	1434	993	978	993	999
			351	1092	1083	1683	1509	1083	1092	1125	1062
Scholl	297	4	1515	2901	2721	4245	5184	2703	2784	2412	2661
			1394	3066	2883	4860	5331	2832	2769	2640	2781
			1301	3231	2970	5250	5994	2997	2871	2724	2925
			1216	3423	3174	5772	6084	3126	3180	2808	3042

Table 7. Experiment 1: Design Cost, *DC*.

Tasks Variability	**size**	***M***	***ct***	GA+ ALS	Simulated Annealing						GA+ Trad.	%diff
					DC	S	L	DC&L	DC&3L	3DC&L		
low	297	200	1515	3227	3911	5570	6452	3839	3489	3684	**3336**	-3.28%
			1394	3380	3962	6243	6851	**3849**	4157	3957	4008	-12.17%
			1301	3559	4504	7375	7348	4405	**3898**	4607	4072	-8.72%
			1216	3721	4993	6345	7582	4849	4508	5039	**4378**	-15.01%
medium	297	200	1515	3195	**3584**	6193	6399	3674	3588	3693	3609	-10.85%
			1394	3351	**4020**	5902	7271	4122	4111	4188	4066	-16.64%
			1301	3514	4350	6648	7636	4331	4040	4657	**3993**	-11.98%
			1216	3727	4299	8050	7998	4382	4432	4301	**4198**	-11.23%
high	297	200	1515	3235	3660	5479	6606	3787	3687	3829	**3411**	-5.15%
			1394	3429	4255	6079	7186	**4071**	4204	4146	4075	-15.78%
			1301	3567	4432	6868	7416	4530	4226	4856	**3858**	-7.54%
			1216	3787	4666	6816	7940	4477	4379	5133	**4364**	-13.22%

Table 8. Experiment 2: Normalized Design Cost, *NDC*.

Tasks Variability	**size**	***M***	***ct***	GA+ ALS	Simulated Annealing						GA+ Trad.
					DC	S	L	DC&L	DC&3L	3DC&L	
low	297	200	1515	1556	1713	1401	1537	1701	1671	1702	1634
			1394	1429	1564	1283	1410	1551	1582	1579	1567
			1301	1335	1493	1180	1336	1474	1436	1508	1439
			1216	1226	1400	1126	1250	1397	1381	1424	1369
medium	297	200	1515	1537	1678	1447	1577	1697	1683	1708	1681
			1394	1426	1559	1287	1435	1569	1583	1596	1565
			1301	1337	1480	1240	1325	1474	1454	1517	1443
			1216	1246	1358	1170	1266	1364	1364	1380	1352
high	297	200	1515	1544	1684	1399	1550	1695	1678	1712	1635
			1394	1420	1582	1276	1380	1569	1579	1588	1563
			1301	1319	1467	1248	1282	1493	1464	1521	1413
			1216	1276	1384	1167	1262	1360	1363	1427	1354
Avg. *APD* from *ct*				2.30%	12.87%	6.42%	2.46%	12.73%	12.08%	14.76%	10.72%
Max. *APD* from *ct*				4.94%	15.12%	9.28%	4.08%	14.86%	13.61%	17.37%	12.61%

Table 9. Experiment 2: effective cycle time, ct_{eff}

Tasks Variability	**size**	***M***	***ct***	GA+ ALS	Simulated Annealing						GA+ Trad.
					DC	S	L	DC&L	DC&3L	3DC&L	
low	297	200	1515	3181	2911	5570	6424	2948	2881	2821	2971
			1394	3338	3054	6243	6833	3066	3047	2926	3061
			1301	3512	3136	7375	7245	3257	3209	3063	3326
			1216	3716	3426	6345	7466	3363	3289	3177	3321
medium	297	200	1515	3181	2913	6193	6195	2851	2881	2791	2911
			1394	3316	3136	5902	7147	3137	3004	2949	3121
			1301	3462	3152	6648	7586	3200	3166	2997	3229
			1216	3683	3379	8050	7741	3377	3420	3148	3362
high	297	200	1515	3211	2929	5479	6538	2951	2994	2863	3031
			1394	3406	3121	6079	7186	3099	3107	2990	3151
			1301	3554	3340	6868	7416	3154	3213	3089	3361
			1216	3611	3378	6816	7723	3500	3388	3201	3473

Table 10. Experiment 2: Design Cost, *DC*.

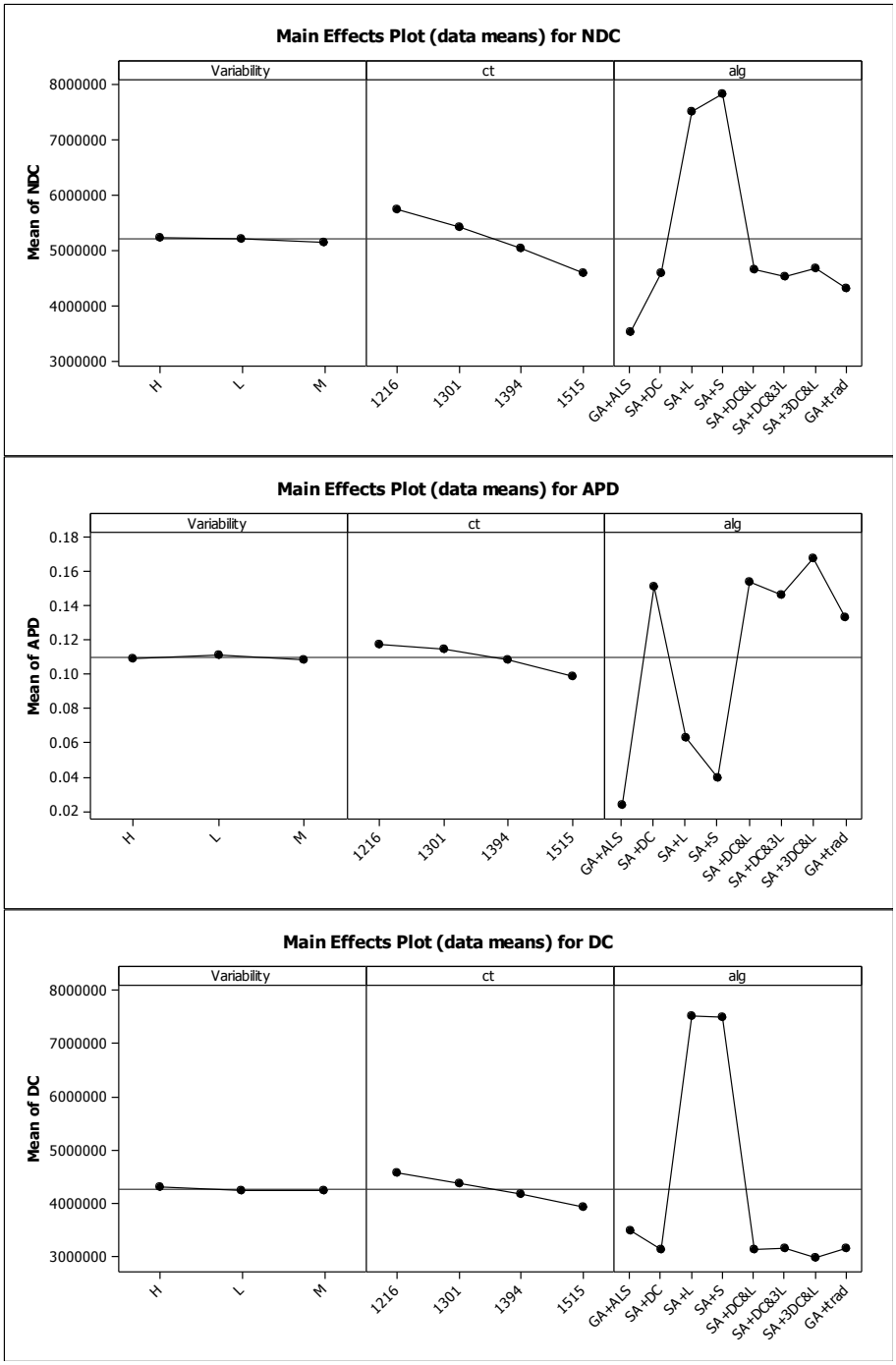


Figure 6. Main effects plots for the factorial experiment 2.

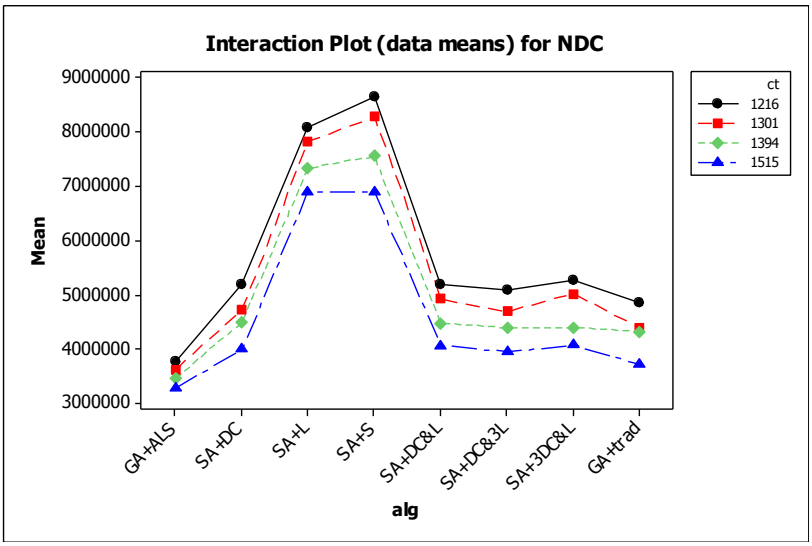


Figure 7. Interaction plot (*alg*ct*) for NDC for the factorial experiment 2.

7 Conclusions

In the paper, an innovative approach to solve the Mixed Model Assembly Line Balancing Problem (MALBP) with stochastic task times and parallel workstations is presented.

A genetic algorithm that considers all the features of this complex problem has been presented. To evaluate the fitness function of each individual in each generation, a parametric simulator, named ALS, has been embedded in the genetic algorithm procedure. ALS has been built undo the event/object oriented paradigm. Its fastness and flexibility allow its utilization in those algorithms and procedures where the evaluation of a fitness function (which includes some performance parameters, as the throughput or the cycle time) has to be performed a large number of times. It allows overcoming the limit of using others measures of throughput (or cycle time), presented in literature, that are poorly correlated to its real value when the complexity of the line increases.

The proposed approach has been compared with other two approaches presented in literature (a Simulated Annealing and a Genetic Algorithm approaches), that use *indirect* measures of the cycle time, not simulation based, to compare different solutions during the search procedures. This forced to create objective functions that do not contain explicitly the cycle time as variable, and thus cannot be identical to the real objective of the problem as defined by eq. (2).

The benefits of the proposed approach, in terms of achievement of the final objective, are relevant, as demonstrated by the results of a comparative study performed on an ample set of problems. Another important advantage is that there is no more need to perform trial and error procedures to tune the weighting coefficients of traditional objective functions that contains indirect measures of the cycle time, and that for this reason have to be 'aligned' with the real objective.

The promising results achieved in this work suggest continuing to exploit the latest advances in discrete event simulation techniques to solve more complex problems.

For example, due to the computational complexities involved, the assembly line balancing problem and the sequencing problems are usually addressed in literature independently of each other, although they are closely interrelated. A possible improvement in the GA approach presented here in could be to find a representation that allows considering the sequence of models entering the line as a variable itself. By using ALS for the evaluation of the fitness function one could explicitly consider the effect of different sequencing policies on the line throughput (and thus on the objective function), taking at the same time the best sequencing and balancing decisions.

Similarly, exploiting the capabilities of ALS to simulate line configurations with buffers among WCs, it could be possible to approach simultaneously the balancing and buffer allocation problems, that are usually treated separately in literature but are in fact closely interrelated.

8 References

Akpinar, S., Bayhan, G.M., 2011. A hybrid genetic algorithm for mixed model assembly line balancing problem with parallel workstations and zoning constraints. Eng Appl Artif Intel 24, 449-457.

Askin, R.G., Zhou, M., 1997. A parallel station heuristic for the mixed-model production line balancing problem. International Journal of Production Research 35, 3095-3105.

Bartholdi, J.J., 1993. Balancing 2-Sided Assembly Lines - a Case-Study. International Journal of Production Research 31, 2447-2461.

Becker, C., Scholl, A., 2006. A survey on problems and methods in generalized assembly line balancing. European Journal of Operational Research 168, 694-715.

Boysen, N., Fliedner, M., Scholl, A., 2007. A classification of assembly line balancing problems. European Journal of Operational Research 183, 674-693.

Boysen, N., Fliedner, M., Scholl, A., 2008. Assembly line balancing: Which model to use when? International Journal of Production Economics 111, 509-528.

Bukchin, J., 1998. A comparative study of performance measures for throughput of a mixed model assembly line in a JIT environment. International Journal of Production Research 36, 2669-2685.

Bukchin, J., Dar-El, E., Rubinovitz, J., 2002. Mixed model assembly line design in a make-to-order environment. Computers & Industrial Engineering 41, 405-421.

Driscoll, J., Thilakawardana, D., 2001. The definition of assembly line balancing difficulty and evaluation of balance solution quality. Robotics and Computer-Integrated Manufacturing 17, 81-86.

Emde, S., Boysen, N., Scholl, A., 2010. Balancing mixed-model assembly lines: a computational evaluation of objectives to smoothen workload. International Journal of Production Research 48, 3173-3191.

Golz, J., Gujjula, R., Gunther, H.O., Rinderer, S., Ziegler, M., 2012. Part feeding at high-variant mixed-model assembly lines. Flex Serv Manuf J 24, 119-141.

Gunther, R.E., Johnson, G.D., Peterson, R.S., 1983. Currently practiced formulations for the assembly line balance problem. Journal of Operations Management 3, 209-221.

Haq, A.N., Rengarajan, K., Jayaprakash, J., 2006. A hybrid genetic algorithm approach to mixed-model assembly line balancing. Int J Adv Manuf Tech 28, 337-341.

Jin, M.Z., Wu, S.D., 2003. A new heuristic method for mixed model assembly line balancing problem. Computers & Industrial Engineering 44, 159-169.

Karabati, S., Sayin, S., 2003. Assembly line balancing in a mixed-model sequencing environment with synchronous transfers. European Journal of Operational Research 149, 417-429.

Kim, Y., Kim, J., Kim, Y., 1996. Genetic algorithms for assembly line balancing with various objectives. Computers & Industrial Engineering 30, 397-409.

Lapierre, S., Ruiz, A., Soriano, P., 2006. Balancing assembly lines with tabu search. European Journal of Operational Research 168, 826-837.

McMullen, P., Frazier, G., 1998. Using simulated annealing to solve a multiobjective assembly line balancing problem with parallel workstations. International Journal of Production Research 36, 2717-2741.

McMullen, P., Tarasewich, P., 2003. Using ant techniques to solve the assembly line balancing problem. Iie Transactions 35, 605-617.

McMullen, P.R., Frazier, G.V., 1997. A heuristic for solving mixed-model line balancing problems with stochastic task durations and parallel stations. International Journal of Production Economics 51, 177-190.

McMullen, P.R., Tarasewich, P., 2006. Multi-objective assembly line balancing via a modified ant colony optimization technique. International Journal of Production Research 44, 27-42.

Merengo, C., Nava, F., Pozzetti, A., 1999. Balancing and sequencing manual mixed-model assembly lines. International Journal of Production Research 37, 2835-2860.

Rosenberg, O., Ziegler, H., 1992. A comparison of heuristic algorithms for cost-oriented assembly line balancing. Zeitschrift für Operations Research 36, 477-495.

Scholl, A., 1993. Data of assembly line balancing problems. Schriften zur Quantitativen Betriebswirtschaftslehre 16, 1-28.

Scholl, A., 2012. Data sets for SALBP. http://www.assembly-line-balancing.de.

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

Simaria, A.S., Vilarinho, P.M., 2001. The simple assembly line balancing problem with parallel workstations - A simulated annealing approach. International Journal of Industrial Engineering-Theory Applications and Practice 8, 230-240.

Simaria, A.S., Vilarinho, P.M., 2004. A genetic algorithm based approach to the mixed-model assembly line balancing problem of type II. Computers & Industrial Engineering 47, 391-407.

Suresh, G., Sahu, S., 1994. Stochastic Assembly-Line Balancing Using Simulated Annealing. International Journal of Production Research 32, 1801-1810.

Tasan, S.O., Tunali, S., 2008. A review of the current applications of genetic algorithms in assembly line balancing. J Intell Manuf 19, 49-69.

Tiacci, L., 2012. Event and object oriented simulation to fast evaluate operational objectives of mixed model assembly lines problems. Simulation Modelling Practice and Theory 24, 35-48.

Tiacci, L., 2014. Coupling a genetic algorithm approach and a discrete event simulator to design mixed-model un-paced assembly lines with parallel workstations and stochastic task times - SUPPLEMENTARY RESOURCES. www.researchgate.net/profile/Lorenzo_Tiacci.

Tiacci, L., Saetta, S., 2007. Process-oriented simulation for mixed-model assembly lines, Proceedings of the 2007 Summer Computer Simulation Conference. Society for Computer Simulation International, San Diego, USA, pp. 1250-1257.

Tiacci, L., Saetta, S., Martini, A., 2006. Balancing mixed-model assembly lines with parallel workstations through a genetic algorithm approach. International Journal of Industrial Engineering-Theory Applications and Practice 13, 402-411.

Venkatesh, J.V.L., Dabade, B.M., 2008. Evaluation of performance measures for representing operational objectives of a mixed model assembly line balancing problem. International Journal of Production Research 46, 6367-6388.

Vilarinho, P., Simaria, A., 2002. A two-stage heuristic method for balancing mixed-model assembly lines with parallel workstations. International Journal of Production Research 40, 1405-1420.

Wee, T.S., Magazine, A., 1981. An efficient branch and bound algorithm for an assembly line balancing problem - part II: maximize the production rate. Working Paper No. 151, University of Waterloo, Waterloo.

Yagmahan, B., 2011. Mixed-model assembly line balancing using a multi-objective ant colony optimization approach. Expert Syst Appl 38, 12453-12461.

Zhang, W.Q., Gen, M., 2011. An efficient multiobjective genetic algorithm for mixed-model assembly line balancing problem considering demand ratio-based cycle time. J Intell Manuf 22, 367-378.