Noname manuscript No.
(will be inserted by the editor)

An Efficient Genetic Algorithm for Setup Time Minimization in PCB Assembly

**Abel García-Nájera · Carlos A. Brizuela ·
Israel M. Martínez-Pérez**

Manuscript submitted to The International Journal of Advanced Manufacturing Technology

Abstract A central aspect of Surface Mount Technology (SMT) systems is the assembly of printed circuit boards (PCBs) which requires the resolution of many optimization problems. One of these problems arises when assembling many types of PCBs on a single machine. In this case the main goal becomes the minimization of the setup times. That is, the time required to modify the feeder rack in order to have all components needed by the next type of PCB to be assembled. Achieving such a minimization goal will provide the system with improved productivity and flexibility capabilities. In order to minimize the setup time we propose a genetic algorithm that uses a group-based representation with a series of specialized genetic operators. A set of 90 instances is proposed as a test bed for the single machine many-types of PCB problem. The proposed algorithm accomplishes the best known result for a benchmark instance of the problem and outperforms, in terms of assembly time, a well known heuristic on the set of proposed instances.

Keywords PCB assembly · genetic algorithms · grouping · sequencing.

1 Introduction

Flexible manufacturing systems (FMS) aim at solving production problems of mid volume (200-20,000 parts per year) [8] and mid-variety parts. The flexibility of the FMS is characterized by how well it responds to changes in the product design and the production schedules, and these are a key-factor in electronics assembly, especially printed circuit board (PCB) assembly. Efficient operations in PCB assembly

Abel García-Nájera
Departamento de Matemáticas Aplicadas y Sistemas, Universidad Autónoma Metropolitana Unidad Cuajimalpa, Av. Vasco de Quiroga 4871, Col. Santa Fe Cuajimalpa, C.P. 05300, México, D.F., México
E-mail: agarcian@correo.cua.uam.mx

Carlos A. Brizuela and Israel Martínez-Pérez
Computer Sciences Department, CICESE, Carretera Ensenada-Tijuana No. 3918, Zona Playitas, C.P. 22860, Ensenada, B.C. México
E-mail: {cbrizuel, isaraelmp}@cicese.mx

will result in a reduction of the production costs and an increase of the competitiveness. Flexibility is therefore becoming essential for PCB assembly. Literature identifies different types of manufacturing flexibilities [9], [11]. Among these are: i) **Machine flexibility**. The number of operations that can be performed by a machine without setup time. ii) **Routing flexibility**. The ability to manufacture a product by alternating routes through the system. iii) **Process flexibility**. The set of product types that can be produced by the system without major setups. iv) **Product flexibility**. The ease of introducing products into an existing product mix. v) **Volume flexibility**. The ability to vary production volume economically.

It is therefore not surprising that, during the past 20 years, PCB assembly literature has been driven by the desire of reducing set-up, production, and component feed times [25] to improve production line productivity and flexibility.

The manufacturing of a single PCB type on a single machine requires the resolution of very important decision problems. Some of them are the *slot assignment problem* (SAP) and the *pick and place sequence problem* (P^2SP) [13]. If we consider the manufacturing of multiple PCB types on a single machine, which we refer to as the M1P problem, at least two additional problems are present: the *PCBs grouping problem* (PGP) and the *PCB groups sequencing problem* (PGSP). The last two problems, PGP and PGSP are NP-hard [36]. We shall concentrate in this paper on both problems.

Group Technology (GT) is a manufacturing concept which takes advantage of parts similarities by grouping them together based on design or manufacturing properties, thus alleviating the production problems of a proliferation of products with decreasing life expectancy and improved quality [4]. In PCB assembly, grouping the boards in order to minimize the number of changes in the feeder rack, is directly related to the minimization of the total assembly time. When a specific PCB type is assembled, the feeder rack should contain all components needed to perform the task. Similarly, when a group of PCB types is going to be assembled, all needed components by all PCB types should be present in the feeder rack. Once the assembly of a PCB group is finished, the feeder rack needs to be updated in order to handle the assembly task for the next group of PCBs [37].

Due to its complexity, most of the solutions proposed for the M1P problem are essentially based on heuristics. Ammons *et al.* [1] presented an integer lineal programming model, in which they assume that all PCB groups are already formed and intend to allocate the component types to the different assembly machines. The model is solved using a branch and bound algorithm. Crama *et al.* [6] proposed a solution based on a hierarchical decomposition of the problem, that is, the problem is divided in five independent subproblems, and for each of them, well-known heuristics and local search methods are employed. Leon and Peters [27] studied six feeder rack setup strategies, focusing in particular on the decisions associated to feeder changes and the PCB assembly. Smed *et al.* [37] compared five heuristics for grouping and sequencing problems over randomly generated instances, concentrating on two different paradigms for the grouping problem. Balakrishnan and Vanderbeck [3] provided a mathematical model for the problem, and proposed two *column generation* methods, one for solving the problem and the other to obtain lower bounds. Salonen *et al.* [34] proposed several heuristics for PCB grouping and PCB groups sequencing, based on the family setup and on the decompose and sequence strategies, where they specify a cost function which takes into account the number of groups and the required changes in the feeder rack. A purely theoretical work was presented by Crama *et al.* [7], where they revise and formulate a mathematical model for all subproblems which are part of the problem. A more recent work by Narayanaswami and Iyengar [30], suggested a heuristic for PCB grouping, based on the similarity of each pair of PCBs, and another one for the PCB groups sequencing, based on the similarity of the generated groups. Jeong [23] proposed an entropy-based group setup strategy which combines component similarity and geometric similarity simultaneously. The entropy method is used to determine the weight of each similarity by capturing the importance of each similarity in different production environments. Ashayeri and Selen [2] developed a production planning and scheduling formulation to determine the component machine allocations, as well as a PCB sequence. Two strategies were proposed: one focusing on minimal number of changeovers and the other on minimal process time. Most of the works previously described, completely separate the M1P problem from the others involved in the assembly of a single PCB type (i.e. P^2SP and SAP), mainly because of the increase in problem complexity [27].

One of the main approaches for solving PCB assembly problems is genetic algorithms (GAs) [28,32,19,13], which are inspired by Darwin's theory of evolution. That means, GAs use an evolutionary process to improve solution's quality. The good results obtained by this kind of algorithms have inspired many researchers to apply it for scheduling problems, including those in FMS environments, where it has proven to be an adequate strategy [?]. In this paper, we propose a genetic algorithm for the manufacturing of multiple PCB types on the pick-and-place machine, which is a commonly used machine in industry. The rationale for applying this metaheuristics to the M1P problem is because it allows the representation of groups and their sequencing in such a way that the genetic operators like crossover and mutation explore the group and sequencing spaces in an effective and efficient manner.

When developing a new method it is important to have a set of benchmarks where a comparison with other approaches can be performed. However, for the M1P model there is a lack of such instances. A few instances [30] make the exception. This fact motivates us to generate and make it publicly available 90 different instances for comparison purposes.

The remainder of the paper is organized as follows. Section 2 reviews relevant work for single machine scheduling with sequence-dependent setup times. Section 3 states the problem we are dealing with and describes some previous approaches based on evolutionary algorithms for the PCB assembly problems on a single machine. In Section 4 we explain the genetic algorithm proposed to solve the problem. Section 5 describes the experimental setup and results. We conclude this paper in Section 6 with a brief discussion about possible implications as well as future directions of our work.

2 A closely related problem: Single machine scheduling with sequence-dependent setup times

In view of the more general context of the single machine model, setup time minimization has been widely considered in literature for different approaches. Rubin and Ragatz [?] proposed a Branch and Bound algorithm which was able to solve small instances up to 45 jobs. Bigras et al. [?] designed a hybrid approach of

Branch and Bound with linear programming which obtains optimal results for larger instances than those previously employed by Rubin and Ragatz. Due to the complexity of the problem, several heuristic approaches have been also used to solve the problem. Tan and Narasimhan [?] developed a simulated annealing algorithm for minimizing tardiness on a single processor with sequence-dependent setup times. A memetic algorithm for the same problem was later proposed by Franca et al. [?]. In this algorithm, a hierarchically structured population and several neighborhood reduction schemas were used to obtain better computational results than previous approaches. Gupta and Smith [?] presented a competitive greedy randomized adaptive search procedure that uses a new cost function in the construction phase along with variable neighborhood search in the improvement phase. Liao and Juan [?] solved a related scheduling criterion, the weighted tardiness, on a single machine environment with sequence-dependent setup times. The proposed ACO algorithm has several features, including a novel parameter for the initial pheromone trail and other for adjusting the timing of local search. Gagne and coworkers [?] modeled an ant colony optimization algorithm which uses look-ahead information in the transition rule of the algorithm for the minimization of the total tardiness. In a later work [?], these authors introduced a hybrid algorithm with Tabu and neighborhood search, which represents the best approach found in literature for this problem. Evolutionary approaches have been also proposed for sequence-dependent setup times on single machine environments. The first genetic algorithm was presented by Rubin et al. [?], which consisted of a SGA algorithm for minimization of the total tardiness. Armentano and Mazzini [?] proposed for the same problem a genetic algorithm, whose control parameters were adjusted by using a statistical method. For their part, Sioud et al. [?] introduced a genetic algorithm that integrates the RPMX crossover operator. This operator takes into account the relative and absolute position of a job in the sequence, outperforming previous evolutionary approaches found in literature, but still less efficient than the Tabu/NS of Gagne et al. [?]. The same authors developed a constraint-based genetic algorithm, which uses an ILOG API C++ in the crossover operator of the genetic algorithm [?]. Continuing their work, a novel crossover operator that combines concepts of constraint programming, multi-objective evolutionary algorithms, and ant colony optimization was proposed [?]. Numerical experiments demonstrated the efficiency of the algorithm by generating competitive solutions to those obtained by other state-of-the-art approaches.

The M1P model becomes a single machine scheduling problem if we restrict the number of elements in each group to one. Therefore, a description of a more general model, the M1P, is decribed next.

3 Problem Statement

The PCB assembly task consists in placing a number of electronic components of pre-specified types at pre-specified locations on a PCB [7]. Automatic assembly machines can be mainly classified into two categories: concurrent and sequential. The concurrent assembly machine, such as the chip shooter (CS) machine and multi-head (MH) machine, performs the pick-up and placement operations simultaneously. The sequential placement machine, such as the pick-and-place (PAP) machine considered in this study, performs the the pick-up and placement oper-

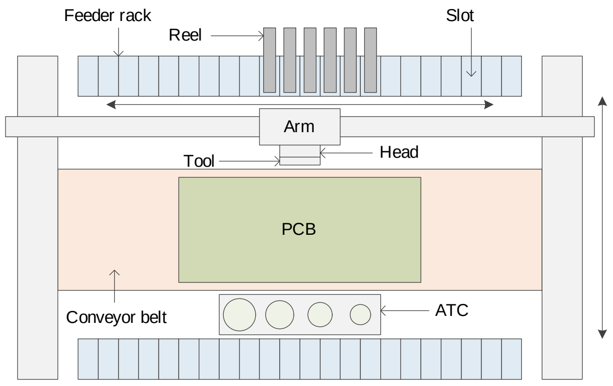

Fig. 1 Pick-and-place assembly machine.

ations one-by-one. The PAP machine, schematically shown in Figure 1, works as follows.

The PCB to be manufactured is transferred through the machine on a conveyor belt. There may be one or two *feeder* racks aside the conveyor belt, which have a certain number of available *slots*. *Reels* contain *components* which are going to be placed on the PCB, and each of them contains only one *component type*: resistors, capacitors, transistors and integrated circuits, among others. Each slot contains only one reel, while reels may occupy one or more contiguous slots. The machine has a *robotic arm*, which has one or more *heads*. Heads pick up components from the reels on the feeder racks and place them on the PCB, by means of using an appropriate *tool*. Each component type can be picked up with a subset of tools, that is, one head with a specific tool can only pick up components from a limited set of component types. Tools are changed in the *automatic tool changer* (ATC) when the next component cannot be picked up with the current tool [24].

When manufacturing a single PCB type, it is required that the time to assemble a PCB [13] be minimized, since this is the only variable that minimizes the total assembly time. Once the assembly of a PCB batch is done, some changes in the feeder rack are required to manufacture the next batch of PCBs. This has to be done for every set of PCBs to be manufactured.

When manufacturing multiple PCB types, the total assembly time can be minimized if we propose a method to reduce:

1. The number of times that the feeder rack needs to be reconfigured. We can appropriately group the PCB types in order to reduce this number. This is called the PCBs grouping problem (PGP).
2. The total number of reel changes in the feeder rack. We can properly sequence the PCB groups to decrease this number. This is known as the PCB groups sequencing problem (PGSP).

There are some strategies to follow in order to minimize the previous objectives [1]:

1. Single setup strategy. It tries to configure a machine to produce a family of PCB types, using a single setup. It assumes enough capacity of the feeder rack.

2. Multi-setup strategy. Since the feeder rack capacity is limited, there are occasions in which the single setup strategy cannot be considered, but it is required to do additional setups within a family.

From these strategies, the most efficient ones for small batches are *decompose and sequence* [1,27,37,35,30] and *partition and repeat* [3,7], both belonging to the multi-setup strategies. The decompose and sequence strategy includes both the PGP and PGSP problems.

Let us first formulate the PGP and PGSP problems. Given N PCB types $(1,...,N)$, M component types $(1,...,M)$, and the corresponding M_i component types the PCB type i contains $(M_i \leq M, \forall\, i \in \{1,...,N\})$, PGP consists in finding a set of groups $\mathbf{G}$ of PCB types such that the number of resulting groups $|\mathbf{G}|$ is minimized, without exceeding the number of components R that can be allocated into the machine (number of available slots). We assume, without loss of generality, that each component type uses only one slot and that $M \geq N$.

In addition, we can define the following variables. Let a_{ki} be a data input equals to 1 if the component type k is used in PCB type i, and 0 otherwise. Let x_{ij} be a decision variable equals to 1 if PCB type i is assigned to group j, and 0 otherwise. Let y_j be a decision variable equals to 1 if the group j is formed, and 0 otherwise.

Finally, let

$$z_{kj} = \left\lceil \frac{1}{M} \sum_{i=1}^{N} x_{ij} a_{ki} \right\rceil, \tag{1}$$

where $z_{kj} = 1$ if the component type k is used by at least one PCB in group j, and 0 otherwise.

With this notation, the PGP problem can be modeled as:

$$\underset{x_{ij}, y_j, z_{kj} \in \{0,1\}}{\text{minimize}} \; |\mathbf{G}| = \sum_{j=1}^{N} y_j \tag{2}$$

subject to

$$\sum_{k=1}^{M} z_{kj} y_j \leq R, \qquad \forall\, j \in \{1,...,N\} \tag{3}$$

$$\sum_{j=1}^{N} x_{ij} y_j = 1, \qquad \forall\, i \in \{1,...,N\} \tag{4}$$

$$x_{ij} \in \{0,1\}, \qquad \forall\, i,j \in \{1,...,N\} \tag{5}$$

$$y_j \in \{0,1\}, \qquad \forall\, j \in \{1,...,N\} \tag{6}$$

$$z_{kj} \in \{0,1\}, \qquad \forall\, k \in \{1,...,M\}, j \in \{1,...,N\} \tag{7}$$

Constraint (3) requires that the number of component types used in each PCB group does not exceed the number of available slots (R). Constraint (4) requires that one PCB type must be assigned to only one group.

From this model we can observe that the PGP can be seen as a *bin packing problem*, which is known to be NP-hard [14]. In this case, the *bins* are the PCB groups, the *objects* are the PCB types and the *maximum capacity of bins* is the number of available slots.

Now, given the set $\mathbf{G}$ of non-empty PCB groups, PGSP consists in sequencing the groups in $\mathbf{G}$, in such a way that the total number of feeder changes C in the feeder rack is minimized.

Let w_{jl} be a decision variable equals to 1 if PCB group l is manufactured immediately after PCB group j, and 0 otherwise.

The PGSP problem can be modeled as follows:

$$\underset{w_{jl}\in\,\{0,1\}}{\text{minimize}}\, C = \sum_{j\in\mathbf{G}}\sum_{l\in\mathbf{G}}\sum_{k=1}^{M}(z_{kj}-z_{kl})^2 w_{jl} \tag{8}$$

subject to

$$\sum_{l\in\mathbf{G}} w_{jl} = 1, \qquad \forall\, j \in \mathbf{G} \tag{9}$$

$$\sum_{j\in\mathbf{G}} w_{jl} = 1, \qquad \forall\, l \in \mathbf{G} \tag{10}$$

$$\sum_{j\in\mathbf{G}}\sum_{j\in\mathbf{G}} w_{jl} \leq |\mathbf{Q}| - 1, \qquad \forall\, \mathbf{Q} \subset \mathbf{G} \tag{11}$$

$$w_{jl} \in \{0,1\}, \qquad \forall\, j,l \in \{1,...,N\} \tag{12}$$

The term $c_{il} = \sum_{k=1}^{M}(z_{kj} - z_{kl})^2 w_{jl}$ in (8), indicates the number of feeder changes in the feeder rack when finishing the manufacturing of PCB group j and starting with PCB group l. For example, if PCB group j uses component type k and PCB group l does not, or viceversa, the term between parenthesis becomes 1. On the other hand, if both PCB groups either use or do not use component type k, this term is 0.

Constraint (9) states that only one PCB group must be manufactured right after PCB group j, while (10) stipulates that only one PCB group must be manufactured exactly before PCB group l. Constraint (11) requires that, for each subset $\mathbf{Q} \subset \mathbf{G}$ of PCB groups, the manufacturing sequence does not form a cycle.

It is not hard to show that, for the model presented above, the PGSP is equivalent to the *Traveling Salesman Problem*, which is known to be NP-hard [33].

3.1 Related work: evolutionary approaches

To date, a number of evolutionary approaches have been employed to analyze the PCB assembly problem on a single machine. Most of these approaches focused on the application for only one type of assembly machine. Here, we briefly review some of the most relevant heuristics proposed in the field.

Leu *et al.* [28] proposed the first genetic algorithm for solving the PCB assembly problem in three different types of assembly machines: the PAP machine, the moving board PAP machine, and the CS machine. In the case of the PAP and CS machines, the authors divided the chromosome into two parts, the first represented the component assembly sequence, and the second one represented the feeder assignment. During the optimization process, both parts are simultaneously optimized, that is, the genetic operators are applied to each part, and the resultant chromosome is used to evaluate the cost function. The initial population is randomly generated. The genetic operators include an order-based crossover, an inversion operator, a rotation operator, and a mutation operator. Genetic operators are in this order applied to the population for a fixed number of generations. The evaluation function depends on the specific characteristics of each assembly machine, which basically evaluates the total travel distance in the assembly machine. In the case of the moving board PAP machine, the whole chromosome was used to represent the assembly sequence, and the genetic operators remained the same.

An application for the PCB assembly problem in the PAP machine was presented by Maimon and Braha [29]. This algorithm mainly focused on minimizing the total number of feeder changes while manufacturing different PCB types, since the authors estimated that the setup activities consume approximately 35-40% of the machines's usable time in the PCB manufacture industry. The chromosome represented a solution as a permutation of the indices of the PCB types. The initial population is randomly generated. For each individual, the algorithm calculates the total number of component switches based on the Keep Tool Needed Soonest (KTNS) policy. Individuals are randomly chosen for reproduction via roulette wheel selection. Two parents are selected at a time and genetic operators, order-based crossover and mutation, are applied to generate two offspring. Crossover operation is validated in such a way that two valid offspring (permutations) are guaranteed. The genetic algorithm gave much better results when compared to a benchmark heuristic based on a Traveling Salesman Problem formulation.

Ong and Khoo [32] also investigated the application of genetic algorithms for the PAP machine. They concentrated on optimizing the sequence of component placements onto a PCB as well as the arrangement of component types in the feeder rack on a single machine. The objective was to minimize the traveled distance of the placement head during the assembly of all components onto the PCB. A second criterion allowed the duplication of component types on the reels. To this end, the chromosome is divided into two parts. One part represented the component assembly sequence and the second represented the reel assignment. The initial population is randomly generated. During (order-based) crossover, two offspring are generated and each of these offspring is subject to mutation and inversion. Half of the population is selected based on its fitness function, which basically calculates the total distance of the placement head from the starting point to the feeder, picks a component and places it on the assigned location on the board and goes back to the starting point. This process is iterated until a fixed number of generations. The best parameters settings obtained in this work were found to be favorable compared to those obtained by Leu *et al.* [28].

Hardas *et al.* [17] designed an experiment to determine the best representation and crossover type, crossover rate, and mutation rate to use for solving a component sequencing problem. They considered a PCB consisting of 10 components which are placed on a single-headed placement machine. Three different representations (path, ordinal, and adjacency) and six appropriate crossover operators (partially mapped, ordered, cycle, classical, alternating edges, and heuristic) were evaluated at three different mutation rates and at 11 crossover rates. Two algorithm response variables, the total distance traveled by the placement head and the algorithm solution efficiency (measured as number of generations and algorithm solution time), were used to evaluate the different GA applications. The combination of representation and crossover operator, along with mutation rate, were found to be the most significant parameters in the algorithm design. In particular, path representation with order crossover was found to produce the best solution as measured by the total distance traveled as well as the solution generation efficiency. Increasing the mutation rate led to slightly improved solutions in terms of head travel, but also resulted in increased solution time.

Ho and Ji [21] proposed a hybrid genetic algorithm, which incorporated the nearest neighbor heuristic, the iterated swap procedure, and the 2-opt local search heuristic in order to improve solution quality. Each chromosome is formed of two parts: the first representing the component sequencing and the second the feeder arrangement. The initial population is generated so that the first part of each chromosome is constructed by the nearest neighbor heuristic, while the second part is generated randomly. During this initialization step, each chromosome is improved as follows: the iterated swap procedure (ISP) is performed on the first part, while the 2-opt local search heuristic is applied to the second part. The roulette wheel selection method is performed to select a pair of chromosomes to undergo a modified order crossover. Heuristic mutation and inversion mutation are applied. After an offspring is produced, the first part of the chromosome is improved by the ISP, while the second part is improved by the 2-opt local search heuristic.

Neammanee and Reodecha [31] developed a memetic algorithm, which integrates a genetic algorithm, the Minimum Slack Time (MST) scheduling rule, the KTNS policy, and a local search procedure, for minimizing the PCB scheduling's total weighted tardiness. MST rule is used for generating the initial population. Then, two-point crossover and shift mutation are performed. In order to improve solution quality, a local search procedure is executed.

Some approaches have been proposed for the CS machine. For example, Dikos *et al.* [10] employed a genetic algorithm to optimize the assignment of the feeder carriage in this machine. The chromosome represented all distinct components in their assigned locations inside the feeder carriage. The initial population is randomly generated. A roulette wheel and tournament parent selection are used as selection mechanisms. The fitness of each individual is simply calculated by counting the number of additional feeder slots that the feeder carriage is moving, which are not free, i.e., incur a cycle time penalty. The (order-based) crossover and mutation operators are applied to the population with a probability p_c and p_m, respectively. According to their experimental results, the roulette wheel selection method generated better average solutions than tournament selection, since a less favorable next generation provides more variety. This model was later improved in [38], where several operators and selection schemes were investigated in order to find a good combination within the domain of the feeder allocation problem.

In addition, Ho and Ji [18,20] proposed a hybrid genetic algorithm to optimize the sequence of component placements as well as the arrangement of component types to feeders for a CS machine. The objective of the problem was to minimize the total assembly time. The developed GA hybridizes different search heuristics including the nearest-neighbor heuristic (NNH), the 2-opt heuristic, and an iterated swap procedure (ISP). To this end, a chromosome is divided into two parts, the first one represented the assembly sequence, and the second one indicated the sequence of feeders to be visited by the placement head. The fitness function is given by the total assembly time, which is the summation of all dominating times of components, i.e., the longest one among the traveling time of the moving table between two components, the traveling time of the feeder carrier between two feeders, and the indexing time of the turret. The roulette wheel is used as selection mechanism. The genetic operators employed in this algorithm include a modified version of the classical order crossover operator as well as the two mutations: heuristic and inversion. The algorithm works as follows. The NNH is used to generate an initial solution for the first part of the chromosomes in the initial population. The idea of the NNH is to start with the first component ramdomly, then to select the next component as close as possible to the previous one from those unselected components to form the placement sequence until all components are selected. The second part is generated randomly. The 2-opt local search heuristic is then performed on the second part of the chromosome. This heuristic calculates, for one parent, all possible two swaps in order to generate offspring, and the best offspring replaces the parent if the offspring has a shorter assembly time than the parent. Afterwords, ISP heuristic is performed for the first part of each initial solution generated by NNH. This heuristic simply exchanges the positions and neighbors of two genes to create 5 different offspring. If the best offspring is better than the parent, the parent is replaced. The whole population is then evaluated, followed by the selection procedure, and the rest of the genetic operators. The NNH is now applied to the second part of the chromosomes, while the 2-opt heuristic is applied to the first one. The best chromosome is chosen at each iteration. The performance of the HGA was superior to that of the simple GA proposed by Leu *et al.* in terms of the total assembly time. This approach is the base for the extended version of Ho and Ji [19].

Chyu and Chang [5] proposed a method that first groups the component types that can be processed at the same machine speed. Then, the minimum spanning tree technique is employed to perform feeder duplications, reducing the distance effect between components of each type. Finally, a genetic algorithm with 2-opt local search, using a feeder arrangement list as solution representation, is applied to determine the component placement sequence.

There also exist applications for the multi-head (MH) machine. Lee *et al.* [26] applied a genetic algorithm to solve the problem of minimizing the PCB assembly time for this machine. The PCB assembly time on the MH machine is dependent on two decision problems. First, a reel assignment problem determines which reel is to be assigned to which slot on the feeder rack. Next, a sequencing problem determines the sequence of pick-and-place movements of the arm, assuming that each reel's position on the racks is fixed by the solution of the reel assignment problem. The chromosome consisted of four parts, each of them representing the reel assignment, the reel-groups, the reel-group assignment, and component-cluster sequence. The fitness function is simply defined by the resulting pick time and tool change time. The initial population is randomly generated. Two parents are selected at a time and a partial crossover operator is used to generate two offspring. The exchange, inversion, and rotate mutation operators are used. Crossover and mutation probabilities did not remain fixed during optimization, instead, a parameter control strategy was used depending on the fitness value of the solutions in each generation. To evaluate the proposed algorithm, the authors compared its performance with that of a heuristic algorithm commonly used in industry, obtaining better results.

Jeevan *et al.* [22] also presented an evolutionary model for the sequencing problem in the MH machine. Unlike the previous approach, the chromosome is only used to encode the sequence of component placements, while a constraint fitness function takes into account component placements with and without tool changes. The initial population is randomly generated. The individuals are then evaluated by the fitness function, and those individuals with the best scores are selected via tournament selection mechanism. The genetic operators, an order-based crossover and mutation operator, are then applied to the population for a specific number of generations. Experimental results showed significant distance reduction when shifting from single to dual head, and from triple to quadruple head. Unfortunately, the performance algorithm was not compared with that of Lee *et al.* [26].

Gyorfi and Wu [16] generalized the genetic algorithm proposed by Leu *et al.* [28] for solving the P^2SP and SAP problems in PAP machines. They generalized the problem formulation to include multiple-placement tool configurations of PAP machines and showed that the generalized model reduces to the model of Leu *et al.* [28] for the single-placement tool case.

From all the reviewed approaches, only one [30] analyzed the M1P problem we study in this paper, the rest only focused on the manufacturing of a single PCB type on a single machine.

4 A group-based genetic algorithm for the M1P problem (GBGA-M1P algorithm)

The combinatorial complexity behind the M1P problem can be modeled as a grouping problem. Falkenauer [12], describes a representation based on groups, which is well suited for grouping in general and, for our particular case, grouping of PCBs. To the best of our knowledge, this is one of the first works to solve this specific problem by means of an evolutionary algorithm. This algorithm is based on the well-known group-based representation [12], proportional selection (roulette wheel selection), and specialized genetic operators.

4.1 Group Based Representation

The combinatorial complexity behind the M1P problem can be modeled as a grouping problem. Falkenauer [12] describes a representation based on groups, which is well suited for grouping in general and, for our particular case, grouping of PCBs.

In this representation, each chromosome consists of two parts (Figure 2). In the first part, each locus represents the object number and its allele the identifier (label) for the group it belongs to, as it is suggested by Falkenauer [12]. The second part of the chromosome represents the formed groups. It is important to emphasize that, in grouping problems, the only important thing is the members of a group not the group identifier itself.

We propose to use the second part of the chromosome as the assembly sequence, i.e. the PCB groups assembly order is given from left to right, as illustrated in Figure 2. In this part, the locus indicates the assembly sequence and its allele corresponds to the group label.

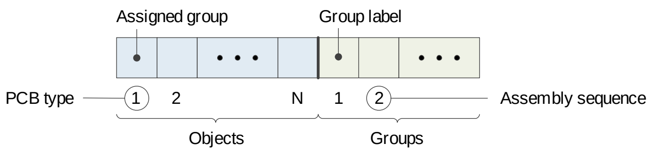

Fig. 2 Individual representation for the M1P problem consists of a two-part chromosome: the first part corresponds to the PCB types assignment to groups and the second part corresponds to the PCB groups assembly sequence.

4.2 Fitness Function

The proposed fitness function considers two main factors: the number of generated groups $|\mathbf{G}|$ and the number of feeder changes C in the feeder rack during the whole manufacturing process. The objective is to minimize these criteria, hence the fitness function f is given by:

$$f = \frac{\alpha}{|\mathbf{G}|} + \frac{1}{C}, \tag{13}$$

where the coefficient α is introduced to equalize the orders of magnitude of $1/C$ and $1/|\mathbf{G}|$.

In fact, this is a multi-objective problem because the two criteria, C and $|\mathbf{G}|$, we are trying to minimize are in conflict with each other.

4.3 Crossover

We use the crossover operator proposed by Falkenauer [12], which is only applied to the second part of the chromosome. An example of this opeator is shown in Figure 3. First, random groups are selected, for instance, B and D for Parent 1, and E for Parent 2 (Figure 3(a)). Then, the PCB types assigned to the groups selected in Parent 2 (i.e., PCB types 2 and 6 are assigned to group E) are copied into the first part of Offspring 1, and the group itself is copied into the left most position in the second part (Figure 3(b)). The groups and their assigned PCB types on Parent 1 are copied into Offspring 1, if there is no interference with the PCB types already copied from Parent 2. In our example, only groups A and C are copied into Offspring 1, since PCB types 2 and 6 are already assigned to group E (Figure 3(c)).

Finally, if there exist unassigned elements, assign them to the existing groups using the heuristic known as *first fit decreasing* (FFD) [15], in case they fit in any of them. For example, in view of the Figure 3(d), PCB type 1 was unassigned, and after applying FFD, it was assigned to group C. In case the unassigned PCB types do not fit in any of the existing groups, new groups are created.

Offspring 2 is generated in a similar manner, but with the parents' role inverted.

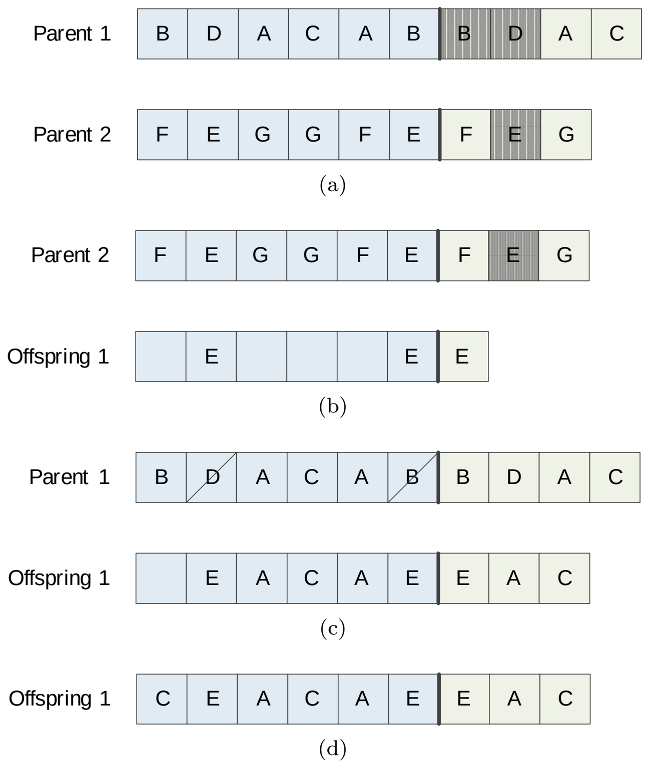

Fig. 3 Crossover operator: (a) Random selection of crossover points; (b) Copy of the PCB types assigned to the PCB groups selected in Parent 2 into Offpring 1; (c) Copy of the PCB groups from Parent 1 into Offpring 1; and (d) FFD applied to unassigned PCB types.

4.4 Mutation

Like for crossover operator, the mutation operator is also applied to the second part of the representation. In general, there exist three main strategies to mutate a solution: to create a new group, to eliminate one group, or to interchange elements between groups [12]. The proposed mutation operator works as follows.

First, randomly select a group from the second part of the chromosome. For instance, in view of the Figure 4(a), group D is selected. Next, copy all elements of all groups with the exception of those elements assigned to the group selected in the previous step. In Figure 4(b) we can see that the PCB types assigned to groups A, B and C are copied from the original individual to its mutation. Finally, apply FFD [15] to assign the elements belonging to the group selected in the first step. In our example, PCB type 2 was originally assigned to the selected group D, and after applying FFD it was reassigned to group C (Figure 4(c)). However, if the unassigned elements do not fit in the existing groups, create a new one, as in Figure 4(d), where PCB type 2 was assigned to a new group E.

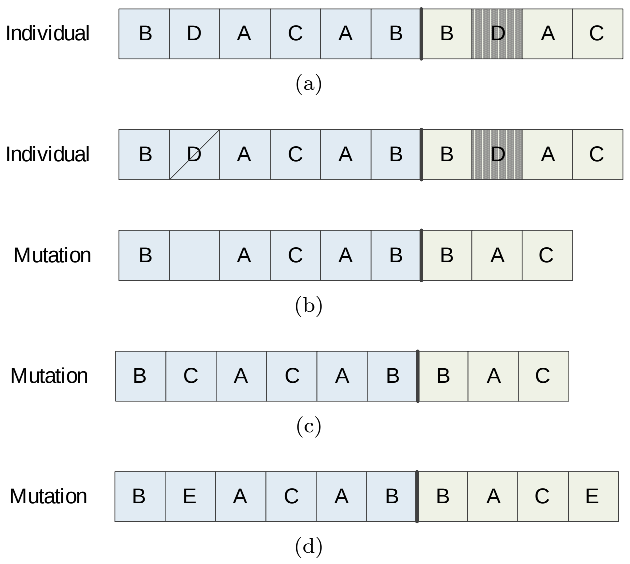

Fig. 4 Mutation operator: (a) Random selection of a PCB group; (b) Copy of the PCB groups from the original individual into the mutation, except those in the previously selected PCB group; (c) Case where the PCB types of the selected PCB group (D) can be assigned to other PCB groups; and (d) Case where a new PCB group needs to be created.

5 Computational results for the GBGA-M1P algorithm

In order to assess the solution quality obtained by our algorithm, we apply it to the only one instance publicly available for this problem [30]. Narayanaswami and Iyengar [30] described an instance consisting of 12 PCB types, each of them containing from 10 to 17 different component types from a universe of 30 and a feeder rack with 20 slots. They also proposed a grouping strategy that combines the feeder rack contents into the similarity measure (Jaccard similarity index) for efficient grouping. The groups of PCBs are then sequenced using a procedure that resembles greedy tree traversal. We will refer to their proposed approach as the NI heuristic.

The validation of the proposed approach is made by evaluating the number of groups and the number of feeder rack changes the algorithm generates. This is because the setup time is closely related to these two criteria as it is shown in the Computational results section.

We applied our GBGA-M1P for solving the benchmark instance mentioned above. The algorithm was run 30 times. We considered a population size of 25 individuals, 300 generations, crossover probability of 1.00, and mutation probability of 0.05. Results are shown in Table 1. It is clear that the results produced by the GA outperforms the ones obtained by NI heuristic in both, the number of PCB groups formed and the number of feeder changes.

Narayanaswami and Iyengar [30] also used the KTNS policy, which tries to keep in the feeder rack some component types that will be used in future groups, if there are available slots for them, in order to reduce the number of changes.

Table 1 Results for the instance of Narayanaswami and Iyengar [30].

PCB	NI heuristic		GBGA-M1P	
types	Groups	Changes	Groups	Changes
12	6	62	5	45

After applying this policy in the NI heuristic, the number of changes are reduced to 56, and the number of groups remain the same. In this study, this policy is not applied to the GA, but if it would be used, the number of feeder changes would not be increased, due to the fact that this policy intends to reduce this number.

5.1 Instance Generation Method

A method to generate PCB types is described in [13]. We used this method to generate 5 sets of 38 different PCB types, that is, 190 different PCB types. With these PCB types we formed 90 instances[1] for the problem studied here. These instances vary from 20 to 158 PCB types, containing from 112 to 787 components, which are from 16 to 89 types.

5.2 Results

The NI heuristic as well as the GBGA-M1P were applied to the generated instances. This time, the population size was set to 100 and the GBGA-M1P run for 500 generations. Crossover and mutation rates remained the same as those used in the previous instance. These parameters were selected after a non exhaustive trial and error process. Results for the number of created PCB groups and the required number of feeder changes are shown in Table 2, for instances with 20 to 78 PCB types, and in Table 3, for instances with 100 to 158 PCB types.

Both tables 2 and 3 have three main columns. The first main column identifies the instance and the number of PCB types the instance contains. The second main column presents the results for the number of PCB groups formed with the NI heuristic and with the GBGA-M1P. For the latter method, the best solution quality obtained out of 30 runs (Best), the average quality (Avg.), the standard deviation (SD), and the percentage of improvement (%Sav.) on the NI heuristic are shown. The improvements are computed considering the GBGA-M1P average results. The third main column presents the results for the number of required feeder changes and has the same structure as the second main column.

We can observe that, in all cases, the number of formed PCB groups was decreased between 8% and 23% with the GBGA-M1P. Considering the number of required feeder changes, we see that, the GBGA-M1P found solutions to 17 instances which decrease the number feeder changes up to approximately 5%. For the remaining instances, solutions found by the GBGA-M1P increase the number of changes up to nearly 3%.

[1] http://www.cicese.mx/ cbrizuel/PCB/pcb2.html

Table 2 Results obtained by the NI heuristic and the GBGA-M1P.

Instance		PCB groups					Feeder changes				
Id.	PCBs	NI	Best	Avg.	SD	%Sav.	NI	Best	Avg.	SD	%Sav.
1	20	13	11	11.00	0.00	15.38	1611	1536	1549.33	0.36	3.83
2	20	12	10	10.00	0.00	16.67	1422	1383	1401.07	0.54	1.47
3	20	14	12	12.00	0.00	14.29	1698	1644	1660.27	0.49	2.22
4	22	17	15	15.07	1.66	11.35	2115	2039	2048.50	0.33	3.14
5	22	14	12	12.00	0.00	14.29	1817	1778	1785.83	0.24	1.72
6	22	15	12	12.00	0.00	20.00	1849	1736	1754.10	0.53	5.13
7	24	18	15	15.00	0.00	16.67	2246	2158	2172.70	0.28	3.26
8	24	17	14	14.40	3.40	15.29	2113	2044	2069.77	0.73	2.05
9	24	12	11	11.00	0.00	8.33	1682	1608	1619.93	0.35	3.69
10	26	20	17	17.00	0.00	15.00	2472	2402	2419.97	0.41	2.10
11	26	22	19	19.00	0.00	13.64	2730	2665	2682.73	0.29	1.73
12	26	18	15	15.00	0.00	16.67	2228	2142	2161.37	0.46	2.99
13	28	22	18	18.00	0.00	18.18	2620	2547	2569.60	0.38	1.92
14	28	16	14	14.00	0.00	12.50	2134	2083	2101.30	0.38	1.53
15	28	23	18	18.00	0.00	21.74	2719	2579	2608.03	0.43	4.08
16	50	34	27	27.00	0.00	20.59	4262	4148	4190.87	0.37	1.67
17	50	38	31	31.00	0.00	18.42	4771	4664	4702.00	0.38	1.45
18	50	35	30	30.00	0.00	14.29	4547	4519	4547.30	0.38	-0.01
19	52	37	29	29.50	1.69	20.27	4692	4568	4600.67	0.36	1.95
20	52	37	32	32.23	1.31	12.89	4945	4901	4946.93	0.38	-0.04
21	52	36	30	30.23	1.40	16.03	4785	4694	4735.47	0.37	1.04
22	54	41	37	37.00	0.00	9.76	5239	5259	5306.73	0.39	-1.29
23	54	36	29	29.00	0.00	19.44	4635	4537	4580.23	0.38	1.18
24	54	38	33	33.00	0.00	13.16	5023	5018	5056.37	0.32	-0.66
25	56	41	33	33.23	1.27	18.95	5169	5044	5096.20	0.48	1.41
26	56	45	38	38.07	0.66	15.40	5705	5701	5745.37	0.40	-0.71
27	56	38	30	30.00	0.00	21.05	4775	4648	4681.63	0.40	1.96
28	58	46	38	38.00	0.00	17.39	5692	5648	5683.87	0.39	0.14
29	58	40	33	33.03	0.54	17.43	5138	5110	5141.57	0.40	-0.07
30	58	39	34	34.57	1.43	11.36	5197	5195	5253.13	0.49	-1.08
31	70	54	43	43.00	0.00	20.37	6710	6647	6675.13	0.24	0.52
32	70	48	39	39.60	1.24	17.50	6302	6266	6308.63	0.36	-0.11
33	70	47	38	38.93	0.64	17.17	6150	6106	6157.20	0.39	-0.12
34	72	53	43	43.00	0.00	18.87	6690	6667	6702.83	0.31	-0.19
35	72	51	41	41.00	0.00	19.61	6554	6511	6552.37	0.28	0.02
36	72	51	40	40.57	1.22	20.45	6501	6361	6419.40	0.48	1.26
37	74	54	46	46.00	0.00	14.81	7066	7062	7133.63	0.39	-0.96
38	74	50	44	44.03	0.41	11.94	6703	6762	6811.20	0.30	-1.61
39	74	51	41	41.53	1.20	18.57	6552	6509	6567.70	0.44	-0.24
40	76	54	44	44.23	0.96	18.09	6961	6905	6949.70	0.39	0.16
41	76	52	42	42.10	0.71	19.04	6716	6653	6700.57	0.32	0.23
42	76	52	43	43.00	0.00	17.31	6774	6772	6808.63	0.32	-0.51
43	78	55	46	46.00	0.00	16.36	7220	7214	7260.07	0.39	-0.55
44	78	54	44	44.20	0.90	18.15	7052	7020	7071.57	0.37	-0.28
45	78	53	44	44.23	0.96	16.55	6971	6950	7020.27	0.35	-0.71

Continues in Table 3...

On average, when using the GBGA-M1P, there is a saving of 17.13% in the number of PCB groups and there is a slight increase of 0.29% in the number feeder changes compared with the results obtained by the NI heuristic.

The summary of these results is shown in figures 5 and 6, for the number of PCB groups and the number of feeder changes, respectively. The horizontal axis in these figures corresponds to the instance identifiers. For each instance, there is a

Table 3 Results obtained by the NI heuristic and the GBGA-M1P.

Instance		PCB groups					Feeder changes				
Id.	PCBs	NI	Best	Avg.	SD	%Sav.	NI	Best	Avg.	SD	%Sav.
...continued from Table 2											
46	100	73	58	58.43	1.05	19.96	9269	9256	9300.77	0.32	-0.34
47	100	73	61	61.20	0.65	16.16	9538	9579	9632.63	0.27	-0.99
48	100	66	54	54.20	0.74	17.88	8656	8668	8736.73	0.38	-0.93
49	102	73	60	59.97	0.68	17.85	9439	9466	9541.53	0.36	-1.09
50	102	70	59	59.20	0.68	15.43	9318	9369	9426.50	0.28	-1.16
51	102	71	57	57.23	0.74	19.39	9069	9097	9147.87	0.30	-0.87
52	104	74	62	62.60	0.78	15.41	9749	9829	9886.83	0.35	-1.41
53	104	72	62	62.03	0.29	13.85	9738	9887	9932.27	0.27	-1.99
54	104	70	60	60.00	0.00	14.29	9345	9484	9529.97	0.25	-1.98
55	106	70	58	58.17	0.64	16.90	9249	9285	9364.23	0.33	-1.25
56	106	77	63	63.40	0.77	17.66	9977	9978	10057.23	0.31	-0.80
57	106	77	62	61.40	0.80	20.26	9729	9720	9800.37	0.35	-0.73
58	108	73	58	58.97	0.53	19.22	9562	9519	9583.70	0.34	-0.23
59	108	77	63	63.07	0.40	18.09	10007	10008	10074.77	0.30	-0.68
60	108	74	61	61.93	0.40	16.31	9792	9885	9948.93	0.30	-1.60
61	120	81	70	70.87	0.48	12.51	10931	11177	11230.23	0.23	-2.74
62	120	81	65	65.33	0.82	19.35	10529	10615	10668.37	0.31	-1.32
63	120	87	72	72.20	0.55	17.01	11358	11436	11513.93	0.31	-1.37
64	122	85	68	68.03	0.26	19.96	10923	10951	11015.90	0.29	-0.85
65	122	89	70	70.27	0.63	21.04	11178	11213	11279.07	0.32	-0.90
66	122	88	72	72.20	0.55	17.95	11365	11502	11559.20	0.31	-1.71
67	124	86	68	68.80	0.58	20.00	11036	11061	11181.10	0.33	-1.31
68	124	85	70	70.70	0.74	16.82	11177	11272	11363.93	0.40	-1.67
69	124	86	70	70.93	0.35	17.52	11187	11338	11389.03	0.26	-1.81
70	126	88	73	73.77	0.57	16.17	11547	11720	11786.47	0.25	-2.07
71	126	83	71	71.10	0.42	14.34	11239	11406	11478.90	0.31	-2.13
72	126	91	72	72.13	0.47	20.74	11514	11532	11621.17	0.33	-0.93
73	128	93	72	72.40	0.68	22.15	11451	11505	11565.73	0.26	-1.00
74	128	92	75	75.13	0.45	18.34	11900	12061	12113.13	0.30	-1.79
75	128	89	73	73.47	0.68	17.45	11466	11658	11715.77	0.26	-2.18
76	150	102	85	84.93	0.42	16.74	13428	13721	13783.33	0.24	-2.65
77	150	102	85	84.70	0.54	16.96	13446	13627	13700.40	0.25	-1.89
78	150	110	87	87.60	0.56	20.36	13862	14036	14122.23	0.27	-1.88
79	152	108	86	85.87	0.58	20.49	13821	13902	14002.27	0.28	-1.31
80	152	104	83	83.60	0.59	19.62	13447	13597	13678.83	0.28	-1.72
81	152	111	92	92.20	0.43	16.94	14389	14672	14729.43	0.22	-2.37
82	154	108	88	88.40	0.55	18.15	13952	14172	14245.27	0.29	-2.10
83	154	108	90	90.13	0.38	16.55	14181	14418	14502.93	0.29	-2.27
84	154	108	89	89.60	0.62	17.04	14118	14379	14460.13	0.30	-2.42
85	156	106	86	86.87	0.49	18.05	13902	14144	14193.03	0.28	-2.09
86	156	112	92	91.97	0.34	17.88	14489	14711	14802.47	0.23	-2.16
87	156	110	89	89.63	0.61	18.52	14297	14459	14550.00	0.24	-1.77
88	158	107	89	89.47	0.63	16.38	14198	14450	14529.67	0.33	-2.34
89	158	111	90	90.93	0.39	18.08	14489	14670	14769.57	0.30	-1.94
90	158	109	88	88.90	0.45	18.44	14227	14448	14504.20	0.28	-1.95
Average						17.13					-0.29

bar which represents the savings percentage of the PCB groups and the number of feeder changes, respectively, obtained by the GBGA-M1P over the NI heuristic. It is clear that, in the case of the number of PCB groups, the GBGA-M1P obtained favorable results when they are compared with those obtained by the NI heuristic.

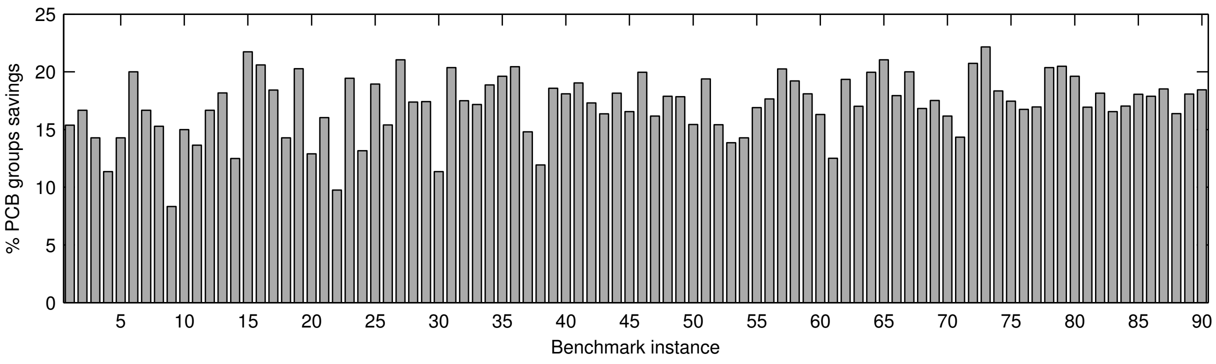

Fig. 5 Savings percentage of the number of formed PCB groups using the GBGA-M1P over the NI heuristic.

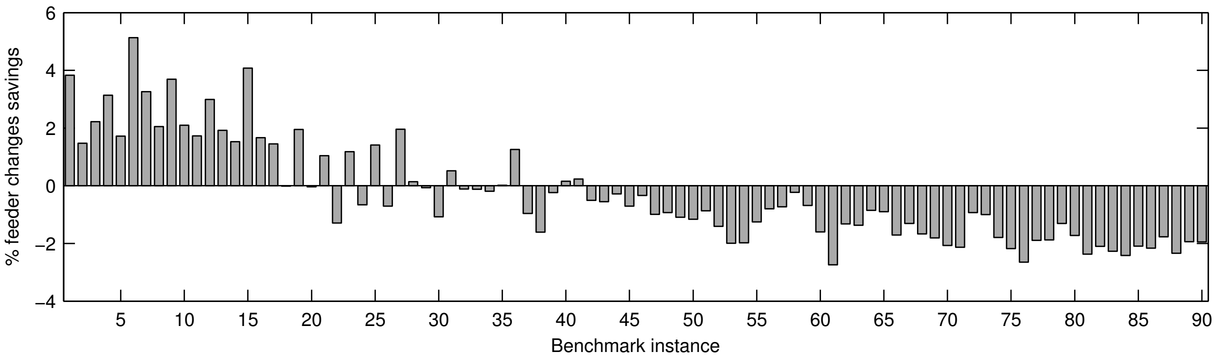

Fig. 6 Savings percentage of the number of required feeder changes using the GBGA-M1P over the NI heuristic.

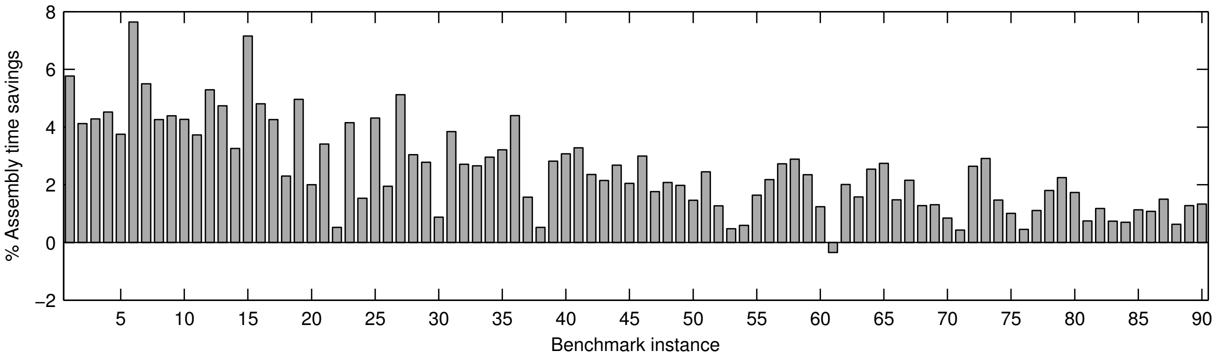

Fig. 7 Savings percentage of the assembly time using the GBGA-M1P over the NI heuristic.

On the other hand, for the number of feeder changes, the GBGA-M1P obtained positive results for the majority of the first half instances.

Given that the GBGA-M1P did not obtain solutions that reduce the number of feeder changes for all instances, we are going to further analyze these results.

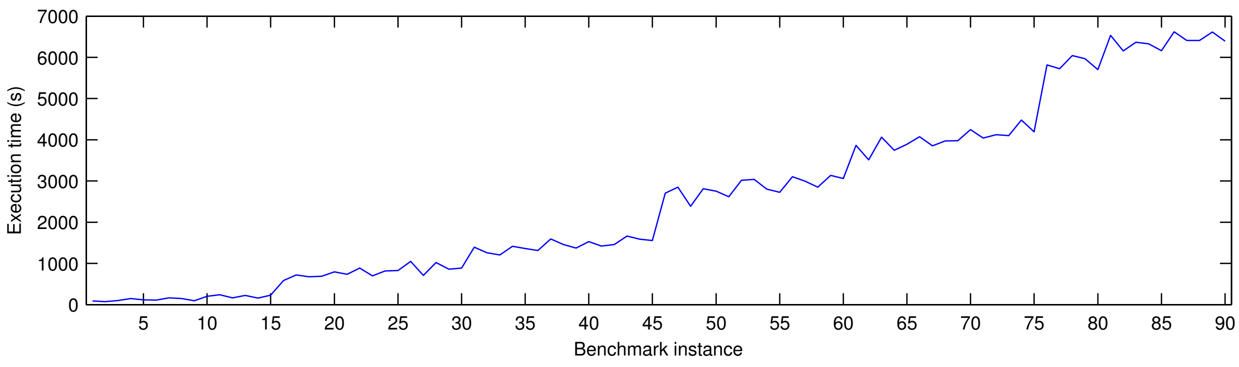

Fig. 8 Average execution time of the GBGA-M1P.

According to Salonen *et al.* [36], a single component feeder of a placement machine can be changed typically in 1-2 minutes, however it may take, for instance, 15-25 minutes to prepare the machine for the component setup operations, because the starting of one or several component changes requires extra manual work by the personnel. Therefore, they propose a cost function of the setup operations for a set of PCB assembly jobs on a single machine as a weighted sum of the number of PCB groups ($|\mathbf{G}|$) and the number of feeder changes (C), that is

$$T(\mathbf{G}, C) = t_g|\mathbf{G}| + t_c C \tag{14}$$

where t_g and t_c are time factors for the number of PCB groups and for the number of feeder changes, respectively. For the experiments we set $t_g = 2.0$ and $t_c = 25$, following what is observed in the work of Salonen *et al.* [36].

Table 4 shows the total assembly time of the set of PCBs for both the NI heuristic and the GBGA-M1P, which has the following structure. The first two columns identify the instance and the number of PCB types the instance contains, respectively. The second column presents the assembly time given by the solutions from the NI heuristic. Fourth and fifth columns show the best and the average assembly time given by the solutions found with the GBGA-M1P. Finally, the last columns presents the per cent improvement of the GBGA-M1P over the NI heuristic. The improvements are computed considering the GA average assembly times.

We can observe that the GBGA-M1P was able to find solutions which definitely reduce the assembly time, since, for all but instance 61, results are positive. These savings represent up to nearly 8% in time and, on average, there is an assembly time reduction of approximately 2.5%. These results are graphically represented in Figure 7.

Thus, even though the proposed approach was not capable of finding solutions which reduce the number of feeder changes for all instances, those solutions actually lead to a faster assembly operation of the sets of PCBs.

It is also important to know what the execution time of the proposed GA is. To this end, Figure 8 shows the average execution time of the GBGA-M1P for each problem instance.

As it might be expected, the execution time of the GBGA-M1P increases as the size of the benchmark instance increases. For the smallest instances, the GBGA-

Table 4 Assembly time, in hours, obtained by the NI and the GBGA-M1P.

Instance			Proposed GA		
Id.	PCBs	NI	Best	Avg.	%Sav.
1	20	32.27	30.18	30.41	5.77
2	20	28.70	27.22	27.52	4.12
3	20	34.13	32.40	32.67	4.28
4	22	42.33	40.23	40.42	4.52
5	22	36.12	34.63	34.76	3.75
6	22	37.07	33.93	34.24	7.64
7	24	44.93	42.22	42.46	5.50
8	24	42.30	39.90	40.50	4.26
9	24	33.03	31.38	31.58	4.39
10	26	49.53	47.12	47.42	4.27
11	26	54.67	52.33	52.63	3.73
12	26	44.63	41.95	42.27	5.29
13	28	52.83	49.95	50.33	4.74
14	28	42.23	40.55	40.86	3.26
15	28	54.90	50.48	50.97	7.16
16	50	85.20	80.38	81.10	4.81
17	50	95.35	90.65	91.28	4.26
18	50	90.37	87.82	88.29	2.30
19	52	93.62	88.22	88.97	4.96
20	52	97.83	95.02	95.88	2.00
21	52	94.75	90.73	91.52	3.41
22	54	104.40	103.07	103.86	0.52
23	54	92.25	87.70	88.42	4.15
24	54	99.55	97.38	98.02	1.53
25	56	103.23	97.82	98.78	4.31
26	56	113.83	110.85	111.62	1.95
27	56	95.42	89.97	90.53	5.12
28	58	114.03	109.97	110.56	3.04
29	58	102.30	98.92	99.46	2.78
30	58	102.87	100.75	101.96	0.88
31	70	134.33	128.70	129.17	3.84
32	70	125.03	120.68	121.64	2.71
33	70	122.08	117.60	118.84	2.66
34	72	133.58	129.03	129.63	2.96
35	72	130.48	125.60	126.29	3.21
36	72	129.60	122.68	123.89	4.40
37	74	140.27	136.87	138.06	1.57
38	74	132.55	131.03	131.87	0.52
39	74	130.45	125.57	126.77	2.82
40	76	138.52	133.42	134.26	3.07
41	76	133.60	128.38	129.22	3.28
42	76	134.57	130.78	131.39	2.36
43	78	143.25	139.40	140.17	2.15
44	78	140.03	135.33	136.28	2.68
45	78	138.27	134.17	135.43	2.05
46	100	184.90	178.43	179.36	3.00
47	100	189.38	185.07	186.04	1.76
48	100	171.77	166.97	168.20	2.08
49	102	187.73	182.77	184.01	1.98
50	102	184.47	180.73	181.78	1.46
51	102	180.73	175.37	176.31	2.45
52	104	193.32	189.65	190.86	1.27
53	104	192.30	190.62	191.38	0.48
54	104	184.92	183.07	183.83	0.59
55	106	183.32	178.92	180.31	1.64
56	106	198.37	192.55	194.04	2.18
57	106	194.23	187.83	188.92	2.73
58	108	189.78	182.82	184.30	2.89
59	108	198.87	193.05	194.19	2.35
60	108	194.03	190.17	191.62	1.24
61	120	215.93	215.45	216.70	-0.35
62	120	209.23	204.00	205.03	2.01
63	120	225.55	220.60	221.98	1.58
64	122	217.47	210.85	211.94	2.54
65	122	223.38	216.05	217.26	2.74
66	122	226.08	221.70	222.74	1.48
67	124	219.77	212.68	215.02	2.16
68	124	221.70	217.03	218.86	1.28
69	124	222.28	218.13	219.37	1.31
70	126	229.12	225.75	227.18	0.85
71	126	221.90	219.68	220.94	0.43
72	126	229.82	222.20	223.74	2.64
73	128	229.60	221.75	222.93	2.91
74	128	236.67	232.27	233.19	1.47
75	128	228.18	224.72	225.88	1.01
76	150	266.30	264.10	265.11	0.45
77	150	266.60	262.53	263.63	1.11
78	150	276.87	270.18	271.87	1.80
79	152	275.35	267.53	269.15	2.25
80	152	267.45	261.20	262.81	1.73
81	152	286.07	282.87	283.91	0.75
82	154	277.53	272.87	274.25	1.18
83	154	281.35	277.80	279.27	0.74
84	154	280.30	276.73	278.34	0.70
85	156	275.87	271.57	272.75	1.13
86	156	288.15	283.52	285.03	1.08
87	156	284.12	278.07	279.85	1.50
88	158	281.22	277.92	279.44	0.63
89	158	287.73	282.00	284.05	1.28
90	158	282.53	277.47	278.78	1.33
Average					2.54

M1P executed in minutes, while for the largest instances, it run in less than two hours.

One last question that emerges is that if the execution time of the GBGA-M1P is significant for the overall process duration, that is, computing solutions plus assembly process. Table 5 shows this information, which structure is the following. The first column identifies the problem instance. The second column is the assembly time related to the solutions found by the NI heuristic. The third and fourth columns are the assembly time related to the solutions found by the GBGA-M1P and its corresponding execution time. The fifth column corresponds to the overall process duration, that is the addition of the assembly time plus the execution time, values in the third and fourth columns, respectively. Finally, the last column is the reduction in time for the overall process obtained with the GBGA-M1P over the NI heuristic.

We can see that the execution time of the GBGA-M1P only affects instances 71 and 76, for which there is an increase in the overall process duration. Instance 61 also presents an increase in the overall process duration, however, it is affected because of the assembly time (see Table 4). For the remaining 87 instances, the solutions found by the GBGA-M1P obtained a reduction in the overall process duration of up to nearly 8% compared to the assembly time found by the NI heuristic. On average, there is a saving in the overall process duration of approximately 2.2%.

It is worth mentioning at this point that, improving the NI heuristic does not guarantee that our proposed method is the best over all existing methods, the GBGA-M1P achieves a relative superiority over the NI algorithm, regarding the total assembly time and the number of groups generated.

6 Conclusions and Future Work

We have introduced a genetic algorithm for the manufacturing of multiple PCB types on the pick-and-place machine. The objective function was to minimize the PCB assembly time, which implies the simultaneous minimization of the number of formed PCB groups and the number of feeder changes. This ensures the minimization of the setup times and allows the system to be machine and product flexible. The genetic algorithm was run for solving a publicly available instance as well as a set of 90 instances proposed here. The obtained solutions were compared with those achieved by the NI heuristic previously reported in the literature.

Results show that, when our approach is used for solving the benchmark instance, the number of PCB groups is reduced from 6 to 5 and the number of feeder changes is reduced from 62 to 45, i.e. the proposed GA achieves the best known result to date for this particular instance. When the genetic algorithm is run for solving the set of 90 instances, the number of formed PCB groups is reduced by 17.13% while the number of required feeder changes slightly increases by 0.29%, on average. If we consider the assembly time then the improvement of the GA over the NI heuristics is of 2.54%. Therefore, it can be deduced that, the proposed algorithm may help to accomplish the goal of SMT systems, particularly PCB assembly, since the time to change a product type is minimized. We make the instances used in this work publicly available for future comparisons.

Future work is aimed at studying another feeder rack setup strategies and other manufacturing machine types, as well as the comparison with other heuristics. Furthermore, we look forward to tackle the multi-objective problem, considering the two objectives, PCB groups and feeder changes in two independent objective functions.

Table 5 Overall process duration, computing time plus assembly time, in hours, obtained by the NI and the GBGA-M1P.

Inst.	NI	Proposed GA			
Id.	proc.	Asm.	Comp.	Proc.	%Sav.
1	32.27	30.41	0.02	30.43	5.69
2	28.70	27.52	0.02	27.54	4.05
3	34.13	32.67	0.03	32.70	4.20
4	42.33	40.42	0.04	40.46	4.42
5	36.12	34.76	0.03	34.80	3.66
6	37.07	34.24	0.03	34.27	7.56
7	44.93	42.46	0.05	42.51	5.40
8	42.30	40.50	0.04	40.54	4.17
9	33.03	31.58	0.03	31.61	4.31
10	49.53	47.42	0.06	47.47	4.16
11	54.67	52.63	0.07	52.70	3.61
12	44.63	42.27	0.05	42.32	5.19
13	52.83	50.33	0.06	50.39	4.63
14	42.23	40.86	0.04	40.90	3.16
15	54.90	50.97	0.06	51.03	7.05
16	85.20	81.10	0.16	81.26	4.62
17	95.35	91.28	0.20	91.48	4.06
18	90.37	88.29	0.19	88.48	2.09
19	93.62	88.97	0.19	89.16	4.76
20	97.83	95.88	0.22	96.10	1.77
21	94.75	91.52	0.20	91.72	3.19
22	104.40	103.86	0.25	104.11	0.28
23	92.25	88.42	0.19	88.62	3.94
24	99.55	98.02	0.23	98.25	1.31
25	103.23	98.78	0.23	99.01	4.09
26	113.83	111.62	0.29	111.91	1.69
27	95.42	90.53	0.20	90.72	4.92
28	114.03	110.56	0.28	110.85	2.79
29	102.30	99.46	0.24	99.69	2.55
30	102.87	101.96	0.25	102.20	0.65
31	134.33	129.17	0.39	129.56	3.56
32	125.03	121.64	0.35	121.99	2.43
33	122.08	118.84	0.33	119.18	2.38
34	133.58	129.63	0.39	130.02	2.66
35	130.48	126.29	0.38	126.67	2.92
36	129.60	123.89	0.36	124.26	4.12
37	140.27	138.06	0.44	138.50	1.26
38	132.55	131.87	0.41	132.27	0.21
39	130.45	126.77	0.38	127.15	2.53
40	138.52	134.26	0.42	134.68	2.77
41	133.60	129.22	0.40	129.61	2.98
42	134.57	131.39	0.41	131.80	2.06
43	143.25	140.17	0.46	140.63	1.83
44	140.03	136.28	0.44	136.72	2.37
45	138.27	135.43	0.43	135.87	1.74
46	184.90	179.36	0.75	180.11	2.59
47	189.38	186.04	0.79	186.84	1.35
48	171.77	168.20	0.66	168.86	1.69
49	187.73	184.01	0.78	184.79	1.57
50	184.47	181.78	0.76	182.54	1.04
51	180.73	176.31	0.73	177.04	2.04
52	193.32	190.86	0.84	191.70	0.84
53	192.30	191.38	0.84	192.23	0.04
54	184.92	183.83	0.78	184.61	0.16
55	183.32	180.31	0.76	181.07	1.23
56	198.37	194.04	0.86	194.90	1.75
57	194.23	188.92	0.83	189.75	2.31
58	189.78	184.30	0.79	185.09	2.47
59	198.87	194.19	0.87	195.06	1.91
60	194.03	191.62	0.85	192.47	0.81
61	215.93	216.70	1.07	217.77	-0.85
62	209.23	205.03	0.98	206.00	1.54
63	225.55	221.98	1.13	223.11	1.08
64	217.47	211.94	1.04	212.98	2.06
65	223.38	217.26	1.08	218.34	2.26
66	226.08	222.74	1.13	223.87	0.98
67	219.77	215.02	1.07	216.09	1.67
68	221.70	218.86	1.10	219.96	0.78
69	222.28	219.37	1.10	220.48	0.81
70	229.12	227.18	1.18	228.36	0.33
71	221.90	220.94	1.12	222.06	-0.07
72	229.82	223.74	1.15	224.89	2.15
73	229.60	222.93	1.14	224.07	2.41
74	236.67	233.19	1.24	234.43	0.94
75	228.18	225.88	1.16	227.04	0.50
76	266.30	265.11	1.62	266.72	-0.16
77	266.60	263.63	1.59	265.22	0.52
78	276.87	271.87	1.68	273.55	1.20
79	275.35	269.15	1.66	270.81	1.65
80	267.45	262.81	1.58	264.40	1.14
81	286.07	283.91	1.82	285.72	0.12
82	277.53	274.25	1.71	275.96	0.57
83	281.35	279.27	1.77	281.04	0.11
84	280.30	278.34	1.76	280.09	0.07
85	275.87	272.75	1.71	274.46	0.51
86	288.15	285.03	1.84	286.87	0.45
87	284.12	279.85	1.78	281.63	0.88
88	281.22	279.44	1.78	281.22	0.00
89	287.73	284.05	1.84	285.88	0.64
90	282.53	278.78	1.78	280.55	0.70
Average					2.18

Acknowledgment

This work was supported by Mexican CONACYT under grant C01-45811.

References

1. Ammons, J.C., Carlyle, M., Crammer, L.L., DePuy, G., Ellis, K., McGinnis, L.F., Tovey, C.A., Xu, H.: Component allocation to balance workload in printed circuit card assembly systems. IIE Transcations **26**, 265–275 (1997)
2. Ashayeri, J., Selen, W.: A planning and scheduling model for onsertion in printed circuit board assembly. European Journal of Operational Research **183**(2), 909–925 (2007)
3. Balakrishnan, A., Vanderbeck, F.: A tactical planning model for mixed-model electronics assembly operations. Operations Research **47**(3), 395–409 (1999)
4. Bellman, R.: Intelligent heuristic for fms scheduling using grouping. Journal of Intelligent Manufacturing **2**, 387–395 (1991)
5. Chyu, C.C., Chang, W.S.: A genetic-based algorithm for the operational sequence of a high speed chip placement machine. The International Journal of Advanced Manufacturing Technology **36**(9), 918–926 (2008)
6. Crama, Y., Flippo, O.E., van de Klundert, J., Spieksma, F.C.R.: The assembly of printed circuit boards: A case with multiple machines and multiple board types. European Journal of Operational Research **98**(3), 457–472 (1997)
7. Crama, Y., van de Klundert, J., Spieksma, F.C.R.: Production planning problems in printed circuit boards assembly. Discrete Applied Mathematics **123**(1-3), 339–361 (2002)
8. Crama, Y., Spieksma, A.O.F.: Production Planning in Automated Manufacturing. Springer-Verlag (1994)
9. Das, S.: The measurement of flexibility in manufacturing systems. International Journal of Flexible Manufacturing Systems **8**, 67–93 (1996)
10. Dikos, A., Nelson, P.C., Tirpak, T.M., Wang, W.: Optimization of high-mix printed circuit card assembly using genetic algorithms. Annals of Operations Research **75**(1), 303–324 (1997)
11. ElMaraghy, H.A.: Flexible and reconfigurable manufacturing systems paradigms. The International Journal of Flexible Manufacturing Systems **17**, 261–276 (2006)
12. Falkenauer, E.: A hybrid grouping genetic algorithm for bin packing. Journal of Heuristics **2**, 5–30 (1996)
13. Garcia-Najera, A., Brizuela, C.A.: PCB assembly: An efficient genetic algorithm for slot assignment and component pick and place sequence problems. In: Proceedings of the 2005 IEEE Congress on Evolutionary Computation, vol. 2, pp. 1485–1491 (2005)
14. Garey, M.R., Johnson, D.S.: Computers and intractability: A guide to the theory of NP-Completeness. W. H. Freeman and Company (1999)
15. Gen, M., Cheng, R.: Genetic algorithms and engineering optimization. John Wiley & Sons, Inc. (2000)
16. Gyorfi, J.S., haur Wu, C.: An efficient algorithm for placement sequence and feeder assignment problems with multiple placement-nozzles and independent link evaluation. IEEE Transactions on Systems, Man, and Cybernetics, Part A **38**(2), 437–442 (2008)
17. Hardas, C.S., Doolen, T.L., Jensen, D.H.: Development of a genetic algorithm for component placement sequence optimization in printed circuit board assembly. Computers & Industrial Enginbeering **55**(1), 165–182 (2008)
18. Ho, W., Ji, P.: Component scheduling for chip shooter machines: a hybrid genetic algorithm approach. Computers & Operations Research **30**(14), 2175–2189 (2003)
19. Ho, W., Ji, P.: A genetic algorithm to optimise the component placement process in PCB assembly. The International Journal of Advanced Manufacturing Technology **26**(11), 1397–1401 (2005)
20. Ho, W., Ji, P.: A genetic algorithm approach to optimising component placement and retrieval sequence for chip shooter machines. The International Journal of Advanced Manufacturing Technology **28**, 556–560 (2006)
21. Ho, W., Ji, P.: An integrated scheduling problem of pcb components on sequential pick-and-place machines: Mathematical models and heuristic solutions. Expert Systems with Applications **36**(3), 7002–7010 (2009)
22. Jeevan, K., Parthiban, A., Seetharamu, K.N., Azid, I.A., Quadir, G.A.: Optimization of PCB component placement using genetic algorithms. Journal of Electronics Manufacturing **11**(1), 69–79 (2002)
23. Jeong, I.J.: An entropy based group setup strategy for pcb assembly. In: 2006 International Conference on Computational Science and Its Applications, pp. 698–707. Springer (2006)
24. van Laarhoven, P.J.M., Zijm, W.H.M.: Production preparation and numerical control in PCB assembly. International Journal of Flexible Manufacturing Systems **5**(3), 187–207 (1993)

25. Lambert, S., Abdulnor, G., Drolet, J., Cyr, B.: Flexbility analysis of a surface mount technology electronic assembly plant: An integrated model using simulation. The International Journal of Flexible Manufacturing Systems **17**, 151–167 (2006)
26. Lee, W., Lee, S., Lee, B., Lee, Y.: A genetic optimization approach to operation of a multi-head surface mounting machine. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences **E83-A**(9), 1748–1756 (2000)
27. Leon, V.J., Peters, B.A.: A comparison of setup strategies for printed circuit board assembly. Computers & Industrial Engineering **34**(1), 219–234 (1998)
28. Leu, M.C., Wong, H., Ji, Z.: Planning of component placement/insertion sequence and feeder setup in PCB assembly using genetic algorithm. Transactions of the ASME **115**, 424–432 (1993)
29. Maimon, O., Braha, D.: A genetic algorithm approach to scheduling pcbs on a single machine. International Journal of Production Research **36**(3), 761–784 (1998)
30. Narayanaswami, R., Iyengar, V.: Setup reduction in printed circuit board assembly by efficient sequencing. International Journal of Advanced Manufacturing Technology **26**(3), 276–284 (2004)
31. Neammanee, P., Reodecha, M.: A memetic algorithm-based heuristic for a scheduling problem in printed circuit board assembly. Computers & Industrial Engineering **56**(1), 294–305 (2009)
32. Ong, N., Khoo, L.P.: Genetic algorithm approach in PCB assembly. Integrated Manufacturing Systems **10**(5), 256–265 (1999)
33. Papadimitriou, C.H., Steiglitz, K.: Combinatorial Optimization: Algorithms and Complexity. Dover Publications, Inc. (1998)
34. Salonen, K., Johnsson, M., Smed, J., Johtela, T., Nevalainen, O.: A comparison of group and minimum setup strategies in PCB assembly. In: Proceedings of Group Technology/Cellular Manufacturing World Symposium, vol. 1, pp. 95–100 (2000)
35. Salonen, K., Smed, J., Johnsson, M., Nevalainen, O.: Job grouping with minimum setup in PCB assembly. In: Proceedings of Group Technology/Cellular Manufacturing World Symposium, vol. 1, pp. 221–225 (2003)
36. Salonen, K., Smed, J., Johnsson, M., Nevalainen, O.: Grouping and sequencing pcb assembly jobs with minimum feeder setups. Robotics and Computer-Integrated Manufacturing **22**(4), 297–305 (2006)
37. Smed, J., Johnsson, M., Puranen, M., Leipälä, T., Nevalainen, O.: Job grouping in surface mounted component printing. Tech. Rep. 196, Turku Centre for Computer Science (1998)
38. Wang, W., Nelson, P.C., Tirpak, T.M.: Optimization of high-speed multi-station SMT placement machines using evolutionary algorithms. IEEE Transactions on Electronics Packaging Manufacturing **22**(2), 137–146 (1999)