Available online at www.sciencedirect.com
SCIENCE DIRECT®
Computers & Industrial Engineering 46 (2004) 141–157
computers & industrial engineering
www.elsevier.com/locate/dsw

An improved genetic heuristic to support the design of flexible manufacturing systems

Lídia Lampreia Lourenço[a,*], Margarida Vaz Pato[b]

[a]*Department of Mathematics, Faculty of Sciences and Technology, New University of Lisbon, and Operations Research Centre, Faculty of Sciences, University of Lisbon, Quinta da Torre, 2829-516 Monte de Caparica, Portugal*
[b]*Department of Mathematics, Faculty of Economics and Business Administration, Technical University of Lisbon, and Operations Research Centre, Faculty of Sciences, University of Lisbon, Rua do Quelhas, 6, 1200-781 Lisboa, Portugal*

Abstract

In industry, when flexible manufacturing systems are designed within a group technology approach, numerous decision-taking processes emerge requiring control of the multiple characteristics of the system. In this context, several grouping problems are identified within the scope of combinatorial optimisation. Such is the case of the part families with precedence constraints problem, which is defined in order to set up families where the total dissimilarity among the parts placed in the same family is minimal and precedence constraints, as well as capacity constraints arise when grouping parts. The present paper describes the use of an improved genetic heuristic to tackle this problem. It comprises a standard genetic heuristic with appropriate operators, improved through specific local search. In order to study the performance of the improved genetic approach, a special purpose constructive heuristic plus an earlier version of the genetic heuristic were implemented. CPLEX software was used from a binary linear formulation for this problem. Computational results are given from the experiment performed using test instances partly taken from the literature while others were semi-randomly generated. The improved genetic heuristic produced optimal solutions for most of the shortest dimension test instances and acted positively in relation to the constructive heuristic results, over almost all the instances. As for the CPLEX it found optimal solutions only for the small instances, besides which for the higher dimensioned instances CPLEX failed to obtain any integer solutions at all, in 10 h running time. Therefore, these experiments demonstrate that the improved genetic is a good tool to tackle high dimensioned test instances, when one does not expect an exact method to find an optimal solution in reasonable computing time.

Keywords: Flexible manufacturing systems; Part families problem; Genetic heuristics

* Corresponding author. Fax: +351-212-948-391.
E-mail addresses: lll@fct.unl.pt (L.L. Lourenço), mpato@iseg.utl.pt (M.V. Pato).

0360-8352/$ - see front matter © 2004 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cie.2004.01.001

1. Introduction

In large organisations, the grouping of different elements on the basis of global objectives may be a way whereby specific goals can be more easily attained, by acting on the elements of each group alone. A case of this grouping arises in industry when designing flexible manufacturing systems on the basis of group technology. This may therefore be regarded as an alternative to the technology inherent in conventional production.

These flexible manufacturing systems have the particular feature of sharing a production philosophy founded on previously given orders. Each order consists of a small batch of products, while a high number of different orders are made during the same period of time. This calls for the introduction of different products, the incorporation of new materials and the adoption of different technological processes, all of which occur within a short period. To achieve enhanced competitiveness, a group-technology based flexible manufacturing system is required. This should be designed to enable efficient processing of small or medium-sized groups of different products, thus assuring a simplification of the productive process, through a reduction in the flow of tools, materials and products handled, besides a saving in production costs as a result of this system.

The two following industries share such particularities: the micro-electrical and the metal-cutting industry. For instance, in the Hewlett-Pacard micro-electrical company the circuit boards produced require different components. These are inserted in the actual boards by using automatic machines whose storage capacity is limited to no more than a given number of components (Hilllier & Brandeau, 1998; Ng, 2000). Another situation concerns the metal cutting plants. Here a set of computer-controlled tooling machines are needed to perform specific metal-part manufacturing operations, using auxiliary components: tools, tool magazines, pallets, fixtures, grippers and equipment to handle parts (Stecke, 1986). Each tool magazine has a limited storage capacity, although, provided the tools are loaded into the magazine, the machine automatically changes one tool for another to perform a different operation.

If we analyse these examples, we find that in both customized productions there are many types of printed circuit boards or metal parts, but the number that must be produced of each type is small. Another important similarity peculiar to both industries is that common components are inserted in different printed circuit boards, and identical tools are employed to manufacture different types of parts. Although the number of common components/tools is not that high, it is significant to cause this feature to be borne in mind when planning the productive process. Moreover, precedence constraints in both industrial processes should not be overlooked. Careful structuring of these production systems is a must. This involves firstly contemplating family grouping of circuit boards that incorporate identical components, or metal parts that share identical tools and later the assignment of each family to a machine or cell of machines. This structuring process helps one to secure considerable gains in terms of time and energy and, subsequently, to significant final manufacturing profits.

In such contexts the problem of part families with precedence constraints (PFP), described in Section 2, arises. Section 3 is devoted to the improved genetic heuristic. Here the encoding of each solution through a chromosome is given, as is characterisation of the genetic operators that modify the chromosome population, besides the improved local search procedure to be applied to the most fitted chromosomes of the population. Section 4 refers to two mathematical programming models for the PFP, both with binary variables but one is quadratic and the other linear. Section 5 concerns the computational experiment undertaken to compare the results of the genetic heuristics (the basic and the improved versions) with those obtained by a special purpose constructive heuristic and by the CPLEX applied to

the binary linear model. Lastly, Section 6 presents the conclusions and considerations regarding future research.

2. The part families with precedence constraints problem

In presenting the problem of part families with precedence constraints, let us consider an industrial plant, based on a flexible manufacturing system of parts. One could use the nomenclature from the micro-electrical context. In this case, instead of parts the system would produce circuit boards.

One must know the N parts (or batches of parts) ordered and, for each of them, in the case of part i, the tools used to produce part i, its processing time p_i regarded as independent of the machine, whose parts take precedence over it in terms of production. In this organisation of the productive process, one agrees to form $K(K < N)$ part families at the most. The aim is to ensure that each family of parts shares similar manufacturing features.

Note that, once the families are set up on the strength of the PFP, they will be assigned to machines that could well be organised into cells, which leads to another problem that is not addressed in this work. For the issue of machine cell formation, see the papers by Cheng, Goh, and Lee (1996), Chen and Srivastava (1994) and Ganesh and Srinivasan (1994). As regards the problem of grouping parts into families, considerably neglected in the literature, refer to the paper by Viswanathan (1995) which links this problem with the formation of machine cells. In a recent paper, Çatay, Vakharia, and Erengüç (2002) study the problem of grouping components into families and propose a specific grouping algorithm for a p-median formulation, whose objective is to maximise the sum of similarities between each component and the group median. Unfortunately, although the authors mention capacity and precedence constraints, they do not consider them in their computational experience, which accounts for the impossibility of comparing the results with those of the current approach.

Before beginning to process the family-grouped parts, where the machine cells are single, each machine should have previously loaded in its magazine the tools required to perform the various production operations of the parts allotted to it. Remember that the processing of a family should be performed with no interruptions in swapping tools in the machine magazine. This therefore introduces a limitation to the dimension of the families. The production time of each family also introduces a constraint arising from the machines' technological requirements. As a result, two types of capacity constraints emerge. One refers to the maximum number of parts assigned to family k, say M_k, while the other requires that the total production time of the parts for family k does not exceed the bound parameter T_k. Viswanathan (1995) deals precisely with constraints of this nature, imposed by the number of parts to be assigned to a machine or cell of machines. He points out that this parameter is usually given by the production design analyst.

As for the precedence relations between parts, these arise from the fact that many flexible systems are intended for products requiring activities occurring at different levels of its production structure. One such example appears in a manufacturing system with partial assembly and final assembly sections. In such circumstances, the parts produced in the manufacturing section are later used to build part assemblies that are applied to the final product. In this way, some parts may only be incorporated in these unfinished products if they, besides others, have already been manufactured. An $N \times N$ matrix is

therefore assumed to be known and its element of line i and of column j, g_{ij}, defines the precedence between parts i and j, in other words

$$g_{ij} = \begin{cases} 1, & \text{if part } i \text{ directly precedes part } j, \\ 0, & \text{otherwise} \end{cases}$$

Precedence relations were identified by Finke and Kusiak (1985) in the formation of groups of parts. These authors state that, in flexible manufacturing systems, precedence constraints are far more important than others, such as the due date constraints.

So that organisation of the productive process respects the precedence relations between the parts, a specific ordering among the families is stipulated: the grouping of families should be such that the succeeding parts of another for the precedence relations are either placed in the same family or are placed in families of a higher index of the preceding part. Processing of these families will subsequently be undertaken in a rising index order, that is, the parts of a family will only be produced when all the lower index family parts have been produced.

A further problem, which is not addressed in this paper, concerns determining of the processing sequence of the parts within each family. The purpose is to avoid violation of precedence relations within the same family.

The optimisation criteria that determines the grouping of the parts into families concerns minimisation of the sum of dissimilarities among parts within the same family. However, like other group technology approaches, where the dissimilarity possesses a binary nature, the present model is based on a real non-negative dissimilarity which expresses the tools requirements. In fact, as mentioned above, specific tools for each part are used in the production process and the intention is to change tools as little as possible during the production of parts of a family. It is therefore considered that the dissimilarity between two parts i and j, expressed as d_{ij}, is the result of the ratio between the number of different tools and the total number of tools involved in producing these parts:

$$d_{ij} = \frac{1}{A_{ij}} \sum_{f=1}^{F} \delta(u_{if}, u_{jf})$$

where F is the total number of tools in the system; A_{ij}, the total number of tools required to produce part i and part j and

$$\delta(u_{if}, u_{jf}) = \begin{cases} 1, & \text{if } u_{if} \neq u_{jf}, \\ 0, & \text{otherwise} \end{cases}$$

with

$$u_{if} = \begin{cases} 1, & \text{if tool } f \text{ is used in processing part } i, \\ 0, & \text{otherwise} \end{cases}$$

Note that, in this case, the dissimilarity is given by the Hamming distance, also used in Finke and Kusiak (1985) to obtain the costs involved in changing tools in the magazine. However, in their study, the context of the problem differed from that of the present one.

In short, the problem of part families with precedence constraints, denoted in brief terms by PFP, requires that one determines the grouping of N parts into, at the most, K families. The forming of families

would be subject to capacity constraints concerning the number of parts and processing time, besides precedence constraints. The criterion for grouping is given by minimisation of the total dissimilarity among parts within the same family.

This PFP characterisation was inspired by a problem due to Kusiak (1985) for which this author presented a p-median type formulation. In this work Kusiak makes use of the properties of the parts' design to define the dissimilarity, but he does not consider capacity or precedence relations.

As was previously mentioned, capacity constraints appear, for instance, in Viswanathan (1995). Here, the problem studied is the building of cells of machines and, at the same time, the assignment of parts to machines. A quadratic formulation with binary variables is presented where the variables stand for the assignment of machines to cells and for parts processed by machines. However, our integer linear formulation for the PFP (Section 4), could be adapted and extended to the problem of Viswanathan and also to the generalized group technology model that considers cell formation with constraints. Note that, should our formulation be extended to the cell formation problem, it can handle the problem involving many types of machines or multiple machines of the same type.

The PFP was also motivated by the problem proposed by Gunther, Johnson, and Peterson (1983), whose objective is to attribute operations that use common tools to the same workstation. However, the PFP was neither formulated nor handled by these authors as it differs from the assembly-line balancing problem, which is their main focus of research.

Finally, one should point out that the issue is easily defined and formulated, though hard to solve. This is the case with so many other combinatorial optimisation problems. In fact, in Garey and Johnson (1979) the NP-hard class was attributed to a grouping problem. This is a particular case of the PFP, resulting from the assumptions that only one part directly precedes all the others and that the capacities are vast enough to allow all parts to be included in each family. Hence, the PFP is NP-hard too.

The use of heuristics, namely of the genetic type, has enabled one to find good-quality solutions for many of these difficult problems, at low computing time, see Reeves (1993). Remember that production takes place as soon as a set of orders is received and there is little time to plan the new manufacturing system design. This is why the answer must always be promptly found.

In Section 3 the authors describe the improved genetic algorithm developed for the PFP, by genetically encoding the solutions, defining respective fitness, besides the selection, crossover and mutation operators. In addition, the local search procedure to improve solutions is briefly explained.

3. The improved genetic heuristic

A genetic heuristic is a kind of random search, applicable to a vast set of optimisation problems of a difficult nature. Beneath its origin one finds concepts based on biology and on the theory of evolution (Goldberg, 1989).

For the problem addressed, the authors propose a genetic algorithm with local search that draws on its specific characteristics. This algorithm, to be seen in summarised form in Fig. 1, is the product of an improvement to a previous version developed by Lourenço and Pato (2001). This version differs in terms of the crossover and mutation operators and the attempt to make the solutions feasible once the operators are applied, rather than immediately eliminating them in the event of non-feasibility. It also differs in view of the application of local search from the best solution in all generations, as opposed to the previous version, where local search only acted on the entire population of the last generation.

Step 0	generate 50 chromosomes for the initial population evaluate the fitness of each chromosome *chromosome** := chromosome with the highest fitness value *generation* := 1
Step 1	if *generation* = 500 then stop select 50 chromosomes for the transitory population,selection operator randomly choose 24 chromosomes and recombine them,crossover/repair operator *chromosome** := chromosome with the highest fitness value randomly choose 20 chromosomes and mutate them - mutation/repair operator *chromosome** := chromosome with the highest fitness value apply the improvement algorithm to *chromosome** update population for the *generation*+1
Step 2	*generation* := *generation*+1 go to step 1

Fig. 1. Improved genetic algorithm.

This genetic algorithm operates with a fixed dimension population of 50 chromosomes, which point to 50 feasible PFP solutions. The initial population is obtained from constructive plus improvement algorithms and from randomly conducted processes. Beginning with this population, the operators act over 500 generations, as explained below.

The three operators, applied in each generation of the genetic algorithm, are that of selection, crossover/repair and mutation/repair. Therefore the last two include a feasibility repair procedure. In each generation the crossover/repair operator works on less than half of the population's chromosomes, and the mutation/repair occurs in 2/5 of the chromosomes at the most. The chromosomes resulting from crossover and mutation are then analysed as to their feasibility. Should any of the PFP constraints be violated, an attempt is made to repair feasibility. When, following the crossover/repair operation child-chromosomes are already feasible, the parents are replaced by the children only if the offspring are best fitted, whereas, in the mutation/repair situation, the mutants are always inserted in the population. At the end of each generation, the chromosome with the highest fitness value, the *chromosome**, undergoes improvement. In keeping with an elitist policy (Rudolph, 1996), besides being preserved as the best chromosome created to date, it is also put back into the population.

After presenting the encoding process for the solutions of the PFP and the fitness function, a description of these procedures is given.

3.1. Encoding, fitness function and selection operator

The first step in constructing a genetic algorithm for a given problem is to find a suitable representation scheme for each solution, in other words, to define the respective chromosome.

Inclusion or otherwise of a part in a family is an inducement for a possible binary encoding of the solutions, but requires $N \times K$ genes. As in a previous study (Pato & Lourenço, 1997) the authors opted in favour of another encoding, in which each solution is represented through a non-binary vector with only

N components—genes—where the position of each gene in the vector represents the index of the part, and the value of the gene represents the index of the family to which the part belongs in the respective solution. By using appropriate genetic operators, solution feasibility is, in the majority of cases, efficiently kept, as the operators do not damage feasibility. Even if they do, an attempt to build a feasible solution from it is always performed by using a straightforward procedure.

One must determine the fitness of each chromosome to support decisions such as keeping it in the population or eliminating it. Fitness of a given chromosome cr_p is, in this application, given directly by the following value

$$apt(cr_p) = \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} d_{ij} - \sum_{k=1}^{K} \sum_{\substack{i,j \in k \\ i<j}} d_{ij}$$

where d_{ij} indicates the dissimilarity between parts i and j, and k is the family index. Thus, the fitness of each chromosome is obtained by subtracting the dissimilarity between the pairs of parts placed in the same family from the dissimilarity between all pairs of parts to be grouped.

Each generation begins with the application of the selection operator 50 times to choose the transitory population on which the other two are to act. The purpose behind the existence of selection, in all the generations, is to provide the most fitted chromosomes with a greater probability of staying in the population while gradually eliminating the lower fitness ones. The technique employed in the selection process is the roulette.

3.2. Crossover/repair operator

This operator randomly chooses two chromosomes from the transitory population 12 times and, for each pair, determines a crossover point between positions 2 and $n - 1$. The values of all the genes located after the crossover point are changed between these two chromosomes, whereas the values of the genes before this point remain constant. Consequently, each of the two new chromosomes (child 1 and child 2) stays with the first parts, in the same families as one of the progenitors, and the remaining parts placed in the same families as that of the other progenitor.

Example 1. Suppose we have 10 parts and 5 families and the precedence relations between the parts are: $1 \rightarrow 2$; $1 \rightarrow 5$; $2 \rightarrow 7$; $3 \rightarrow 4$; $4 \rightarrow 5$; $5 \rightarrow 6$; $6 \rightarrow 8$; $7 \rightarrow 8$; $8 \rightarrow 9$; $9 \rightarrow 10$. The crossover/repair operates for this example as shown in Fig. 2 below. It is assumed that, here, both child-chromosomes are already feasible, without the need for repair.

parent-chromosome 1	1	2	1	1	**2**	2	3	3	4	5
parent-chromosome 2	2	2	1	1	2	3	3	4	4	5

child-chromosome 1	1	2	1	1	2	**3**	**3**	**4**	**4**	**5**
child-chromosome 2	2	2	1	1	2	**2**	**3**	**3**	**4**	**5**

Fig. 2. Crossover/repair operator illustration.

This operator proves to be fairly effective in preserving feasibility. However, violations of precedence constraints or of family capacity can be created, in which case the repair procedure acts in the following way.

One begins by analysing child-chromosome 1. If the precedence constraints are not fulfilled, one compares the index of the family of each part failing to satisfy the precedence constraints with the indexes of the families of their successors, and the violating part is transferred to the family with the lowest index of its successors. In this way these violations cease to exist. After this correction the precedence constraints are satisfied, although violation in the families' capacity may occur. In this case, for each of the families where capacity violation exists, the parts contained in the current family will move to the family whose index immediately follows, in a decreasing index order, until violation ceases to occur. Following this repair, child-chromosome 1 is once more studied as to its feasibility for precedence constraints. If it is already feasible, then its fitness is calculated and it substitutes the least fitted progenitor in the transitory population, provided it is better than the latter.

Precisely the same procedure is used for child-chromosome 2. Once its feasibility is reached and its fitness calculated, if it is better than that of at least one of the two chromosomes of the current pair, then it substitutes the least fitted one. Mention should be made that, before child-chromosome 2 begins to be analysed, one of the progenitors may already have been replaced by child-chromosome 1. This procedure eliminates both descendants, should they fail to meet the feasibility requirements or fail to be better than the progenitor with the least fitness value.

One should add that crossover/repair is applied to the transitory population's chromosomes 12 times. In other words, through this operator, up to 24 modifications may be carried out on the chromosomes of a given generation.

3.3. Mutation/repair operator

The chromosomes to be subjected to mutation are randomly selected In each one, four genes are randomly chosen. For each of the first two selected genes, that is, for the respective parts, one verifies, among all their successors which one is the successor's family with the lowest index. If the part chosen is in a family which is two or more indexes lower in relation to the index found, then this part reverts to the family whose index is immediately below the one found. Otherwise, the part chosen moves to the family of the index found. As for the two other selected genes (parts), a similar reasoning is employed, but this applies to the predecessor parts of those chosen. Thus, for each of the parts chosen one determines the highest index from among those of the families to which their predecessors belong. If the part we are addressing is in a family whose index is two units higher, or more, in relation to the determined index, then the part chosen is placed in the family whose index immediately follows it, otherwise it is transferred to the family of this index. Should parts be selected with neither successors nor predecessors, then the mutation is not applied. The maximum number of mutation/repair operator applications on each generation is 20.

Example 2. Using the same data of Example 1, the mutation/repair operator is clarified in Fig. 3. Let the random positions be: 3, 5, 2 and 8.

chromosome	1	**2**	**1**	3	**3**	4	4	**5**	5	5
mutant	1	**1**	**2**	3	**4**	4	4	**4**	5	5

Fig. 3. Mutation/repair operator illustration.

Note that the mutation process does not spoil the precedence constraints, although it does violate the families' capacities with a certain degree of frequency. In that case, one tries to repair the capacity constraints violation and, when this succeeds, the mutant is analysed once more with regard to the precedence constraints. Then, should the mutant be feasible, it replaces the original chromosome in the transitory population.

As it is not always possible to repair the violations, it has been empirically verified that, at the end of the algorithm, the rate of feasible mutants is around 75% of the number of mutations undertaken throughout all the generations, whereas, the number of feasible chromosomes more or less coincides with the total number of chromosomes produced by the crossover/repair operator.

3.4. Elitism and the improvement heuristic

At the end of each generation an improvement heuristic is applied to the chromosome with the highest fitness value, in other words, the *chromosome**. This heuristic analyses transfers of parts from families to others, without violating the PFP constraints. If it were given a specific neighbourhood, this might be considered a local search.

The algorithm briefly described in Fig. 4 begins by a positioning in the highest index family—current family or family k—while retaining the information of the parts to be found there, besides the respective

Step 1	let the current solution be the one corresponding to *chromosome** k:=K (index of the last family)
Step 2	P:=[$i \in$ family 1,...,k-1: successors of part i are already attributed to higher index families]
Step 3	if $i \in P$, which may be placed in family k, does not exist, then go to step 5 select $i^* \in P$ which can be placed in k and produce the biggest decrease in total dissimilarity
Step 4	attribute i^* to family k update information regarding the predecessors of i^* update the parameters of the families involved in the transfer include in P all the parts which have no successors for attribution go to step 3
Step 5	if k=1 or P = Ø then stop otherwise, k:= k-1 go to step 2

Fig. 4. Improvement algorithm.

processing time (Step 1). Once this family has been studied, the current one will become the one whose index is immediately below it and successively, in the same way, as far as family 1.

Analysis of the current family—family k—consists in choosing for transfer to this family, from among all the parts that are not to be found in it, nor in the higher index families and have no successors to attribute (all successors of i are already assigned to this current family or to the higher index families), the part responsible for the greatest reduction in the total dissimilarity and that does not violate the family capacity constraints, Step 3. Let it be part i^*.

Then it updates the information regarding the predecessors of the re-assigned part, the capacities of family k and of the family to whom i^* belonged. Also, for every immediate predecessor of i^*, the number of successors to be attributed falls by one unit, the number of parts of family k is increased by one unit and its processing time is added to the time concerning part i. Lastly, for the family to which the transferred part belonged, the following up-dating is performed: the number of parts is reduced by one and total time is reduced by the processing time of i^*. The value of the dissimilarity corresponding to the current solution is also updated. This completes Step 4.

One now proceeds to the selection of another part that does not belong to the current family, and Steps 3 and 4 are repeated until all parts of P violate the current family's capacities or no parts, in families with an index below k and without successors, remain to be attributed. On encountering this situation the current family becomes the family whose index is immediately below, that is, $k - 1$. If $k = 1$ or P is empty, then the algorithm ends.

After applying this improvement method to the chromosome with the highest fitness value (*chromosome**), the latter is updated and it substitutes the least fitted one in the transitory population (the next generation's population).

One of the outputs of the improved genetic heuristic is precisely the feasible solution for PFP corresponding to the last updated *chromosome** which, through its total dissimilarity provides an upper bound on the optimal value of the PFP. In Section 4 the authors present mathematical formulations developed in order to test the quality of the upper bounds relative to the heuristic solutions for the PFP.

4. Binary formulations

A natural formulation for the PFP is developed by means of a quadratic function of binary variables pointing to the placing of parts into families. In fact, if the objective is to minimise this function and if the constraints for the building of families are formalised on the basis of the above-mentioned binary variables, the following quadratic formulation may be presented:

$$\min \sum_{k=1}^{K} \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} d_{ij} x_{ik} x_{jk} \tag{4.1}$$

$$\text{s.to} \sum_{k=1}^{K} x_{ik} = 1 \qquad i = 1, \ldots, N \tag{4.2}$$

$$\sum_{k=1}^{K} k x_{ik} \leq \sum_{k=1}^{K} k x_{jk} \qquad \text{for each pair of parts } (i,j) : \; g_{ij} = 1 \tag{4.3}$$

$$\sum_{i=1}^{N} x_{ik} \leq M_k \qquad k = 1, \ldots, K \tag{4.4}$$

$$\sum_{i=1}^{N} p_i x_{ik} \leq T_k \qquad k = 1, \ldots, K \tag{4.5}$$

$$x_{ik} = 0, 1 \qquad i = 1, \ldots, N;\ k = 1, \ldots, K \tag{4.6}$$

The objective function (4.1) represents total dissimilarity, in other words, the sum of the dissimilarity between all the pairs of parts assigned to the same family. Eq. (4.2) impose that each part belong to one family and one family alone. Each of the precedence constraints (4.3), defined for a pair of parts (i, j) such that i is an immediate predecessor of j, ensures that if i precedes j then j cannot be placed in a family with an index lower than that of the family in which part i is placed. The cardinality capacity constraints (4.4) do not allow families to be set up with a number of parts above the given bound. Time capacity constraints (4.5) prevent the setting-up of families whose total part processing time exceeds the given time bound. Lastly, the constraints (4.6) associate a given part i to a family k ($x_{ik} = 1$) or otherwise ($x_{ik} = 0$).

The PFP approach through this natural formulation becomes unviable as a practical mean of determining the optimal solution as its objective function is quadratic and the variables are binary. Below the PFP is formalised with linear functions alone.

A linear reformulation for the model (4.1)–(4.6) is obtained by substituting the product of variables $x_{ik}x_{jk}$ by the new variable y_{ij} and by adding equations linking the primitive, natural variables, x_{ik} and x_{jk}, with the new ones, y_{ij}. The binary variable y_{ij} indicates whether parts i and j are in the same family $y_{ij} = 1$, or if in different families ($y_{ij} = 0$), for all $i, j = 1, \ldots, N$, $i < j$, which results in the following mixed linear programming problem (Lourenço & Pato, 2001):

$$\min \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} d_{ij} y_{ij} \tag{4.7}$$

$$\text{s. to } y_{ij} \geq x_{ik} + x_{jk} - 1; \qquad i = 1, \ldots, N-1,\ j = i+1, \ldots, N,\ k = 1, \ldots, K \tag{4.8}$$

constraints (4.2)–(4.6)

$$y_{ij} \geq 0; \qquad i = 1, \ldots, N-1,\ j = i+1, \ldots, N \tag{4.9}$$

Note that, in the optimal solution, the constraints (4.8) force each variable y_{ij} to be equal to 1 if parts i and j are in the same family. Should these parts be in different families, the variable y_{ij} contributes with the value 0 to the objective function. This formulation in binary linear programming has already been used by Aronson and Klein (1989) for a clustering problem applied to software design.

The formulation (4.2)–(4.9) may now be applied to solve the PFP, using standard software, at least for the instances whose dimension is not very high. The fact that PFP is NP-hard announces computational difficulties when tackling high dimensioned instances.

However, the lower bounds obtained from the linear relaxation of this formulation are very poor and in most instances nil. This has to do with the fact that linear relaxation enables each part to remain

divided into several fractions of the unit, and these could be attributed to different families. In this way, the second member of a constraint (4.8) is negative which, together with the objective function of the problem, causes the variable y_{ij} be equal to 0 in the optimal solution of the linear relaxation. The rare exception to a nil bound arises when there exists only two families and, in this case, for two parts i and j, the variable y_{ij} could be greater than 0 rather than 0, because those two parts remain split and at least one is above 0.5. This case forces the return of a positive value to y_{ij} through constraint (4.8). Hence, the total dissimilarity will not be equal to 0, but is still very low.

Section 5 reports on a computational experiment performed with the improved genetic heuristic algorithm, as well as with a constructive and an earlier genetic heuristics, both developed for the PFP, and a standard algorithm applied to the formulation (4.2)–(4.9).

5. Computational experiment

The computational experiment was carried out with instances partly taken from the literature and from OR-Library and Scholl (1995). Below, in Table 1 is displayed the data for these test instances. Most of the parameters were obtained from OR-Library. As for maximum number of parts per family, they were (pseudo)randomly generated. In that way, the constraints are frequently effective. The dissimilarity matrix was also randomly generated with elements in the interval [0,1] as, in the current model the dissimilarity represents a proportion between the number of different tools used to produce two parts and the total number of tools.

Use was made of CPLEX version 8.1 software and the other algorithms were programmed in Pascal. All the algorithms were executed in a 1500 MHz Pentium IV PC with 256 RAM.

Table 2 displays information for each instance of the PFP: on parameters N and K in columns (2) and (3), on the number of variables and constraints of formulation (4.2)–(4.9), respectively, in columns (4) and (5), and on some computational results, including the execution times in the remaining columns. Hence, column (6) indicates the minimum dissimilarity and computing time relative to CPLEX applied to the formulation (4.2)–(4.9). Columns (7) and (8) show upper bounds and the computing time obtained, respectively, from a constructive heuristic and from an earlier version of a genetic heuristic (Pato & Lourenço, 1997). The same results from the improved genetic heuristic are found in column (9), whereas column (10) displays the generation where the best solution was found within the improved genetic heuristic.

Note that, for the last instance, only ten different chromosomes could be found to form the population, through the algorithm that generates the initial population. For this reason, in this case, one decided to reduce, in the improved genetic algorithm, the number of mutations to 5 and the number of crossovers to 10.

As expected, comparison of the improved genetic heuristic, column (9) of Table 2, with the constructive heuristic, column (7) of the same table, demonstrated that the genetic approach focussed throughout this paper behaved far more favourably.

An analysis of this table also shows that the improved genetic heuristic (column (9)) performed better than the earlier genetic version (column (8)) in most instances, though it did not consume much more computing time. This was particularly so in the most difficult cases, where the CPLEX binary linear code failed to operate after 10 h' running time. As for the first five, lower

Table 1
Data regarding the test instances

Parameters	Instances				
	Mertens	Bowman	Jaeschke	Jackson	Mansoor
N	7	8	9	11	11
K	2	5	3	4	3
No. of direct precedence relationships	6	8	11	13	11
Part's processing time	1–6	3–17	1–6	1–7	2–45
Maximum no. of parts per family	5	5–7	3–8	6–9	6–8
Maximum time per family	14–20	40–47	15–17	15–21	125–149
Dissimilarity between two parts	0.1–1.0	0.1–1.0	0.1–1.0	0.1–1.0	0.1–1.0
Parameters	Instances				
	Mitchell	Roszieg	Heskia	Buxey	Sawyer
N	21	25	28	29	30
K	5	6	5	7	7
No. of direct precedence relationships	27	32	39	36	32
Part's processing time	1–9	1–13	1–108	19–21	1–25
Maximum no. of parts per family	4–10	5–7	6–10	5–10	6–18
Maximum time per family	29–44	27–42	39–590	59–83	58–84
Dissimilarity between two parts	0.1–1.0	0.1–1.0	0.1–1.0	0.1–1.0	0.1–1.0
Parameters	Instances				
	Lutz1	Gunther	Kilbridge	Hahn	Warnecke
N	32	35	45	53	58
K	7	8	8	8	10
No. of direct precedence relationships	38	45	62	82	70
Part's processing time	100–1400	1–30	3–29	40–1775	7–52
Maximum no. of parts per family	6–9	7–11	4–12	9–12	10–12
Maximum time per family	53–84	3500–4700	90–138	115–158	3604–5153
Dissimilarity between two parts	0.11.0	0.1–1.0	0.1–1.0	0.1–1.0	0.1–1.0
Parameters	Instances				
	Tonge	Wee-Mag	Arc83	Lutz2	Lutz3
N	70	75	83	89	89
K	23	30	21	15	24
No. of direct precedence relationships	86	87	113	118	118
Part's processing time	1–152	5–27	233–2881	1–10	1–74
Maximum no. of parts per family	5–10	5–10	6–10	4–12	5–10
Maximum time per family	324–457	55–102	8375–10,713	21–49	155–213
Dissimilarity between two parts	0.1–1.0	0.1–1.0	0.1–1.0	0.1–1.0	0.1–1.0
Parameters	Instances				
	Mukherje	Arc111	Barthold1	Barthold2	Scholl
N	94	111	148	148	297
K	19	27	14	40	50
No. of direct precedence relationships	181	176	175	175	300

(*continued on next page*)

Table 1 (*continued*)

Parameters	Instances				
	Mukherje	Arc111	Barthold1	Barthold2	Scholl
Part's processing time	8–123	10–5200	7–811	5–383	9–1386
Maximum no. of parts per family	8–11	7–11	11–19	10–19	11–19
Maximum time per family	354–549	7702–10,172	789–1156	175–268	2800–4086
Dissimilarity between two parts	0.1–1.0	0.1–1.0	0.1–1.0	0.1–1.0	0.1–1.0

dimensioned instances, both genetic versions came up with the optimal solutions. The improved version obtained at least one more optimum.

It should be remembered that the generation of the improved genetic, where the best feasible solution was found, is very high for most instances (see column (10)). This indicates that the genetic operators are, in fact, acting along the generations of the algorithm. Note that the maximum number of generations is 500.

As for application of the CPLEX to formulation (4.2)–(4.9) for PFP, it also managed to solve those small instances. However, for the high-dimensioned cases the optimal value remains unknown. Moreover, linear relaxation lower bounds, not given in Table 2, were, as expected, very poor. In fact, the lower bound from the linear relaxation is 0 for all instances, with the exception of the Mertens instance, as it has only two families. Hence, each part is split into two fractions, each equal to 0.5, or one of them greater than 0.5. This second situation leads to the fact that, in the constraint (4.8), the value of y_{ij} (for a pair i and j) is greater than 0.5. As mentioned in Section 4, in this case the value of the objective function of the linear relaxation is greater than 0, though very low.

One should add that comparison with computing results obtained by other authors cannot be performed because the PFP has not yet been studied in the way it is tackled here. At least the authors of the current paper are aware of such studies. Even the problem studied in Scholl (1995) is different from the PFP because it has another objective function, which explains the impossibility of comparing the results of this paper with his.

In conclusion, the improved genetic heuristic proved to be a very successful method for tackling the smallest instances of this difficult combinatorial optimisation problem. As for the higher dimensioned cases, although the quality of respective solutions is not known, the improved genetic heuristic is, at least, an easy tool to consistently generate a population of feasible solutions, at a low computing time, even when the standard software behaved very poorly.

6. Closing comments

This paper presents the part families with precedence constraints problem (PFP), within a flexible manufacturing system, for which an improved genetic heuristic is proposed. This hybrid heuristic, based on a genetic heuristic together with a simple local search procedure, provided very good feasible solutions, at least for the cases where the optimum is known. For the other instances, it always generated feasible solutions and also better quality solutions than the best known ones, at low computing expenses.

Table 2
Results of the computational experiment

(1) Instances	(2) N	(3) K	(4) No. of variables total/binary	(5) No. of constraints	(6) CPLEX optimal value* upper bound** (time)	(7) Constructive heuristic upper bound (time)	(8) Genetic earlier version upper bound (time)	(9) Improved genetic upper bound (time)	(10) Improved genetic best generation
Mertens	7	2	35/14	59	3.9* (0.01 s)	3.9 (0 s)	3.9 (160 s)	3.9 (200)	1
Bowman	8	5	68/45	166	0.9* (0.09 s)	0.9 (0 s)	0.9 (160 s)	0.9 (74 s)	17
Jaeschke	9	3	63/27	134	5.5* (0.03 s)	5.5 (0 s)	5.5 (240 s)	5.5 (70 s)	1
Jackson	11	4	99/44	252	3.3* (0.30 s)	5.0 (0 s)	3.3 (91 s)	3.3 (48 s)	96
Mansoor	11	3	88/33	193	4.4* (0.11 s)	6.8 (0 s)	4.4 (420 s)	4.4 (640 s)	74
Mitchell	21	5	315/105	1108	15.4* (380.48 s)	20.3 (0 s)	16.5 (340 s)	16.5 (280 s)	9
Roszieg	25	6	450/150	1869	17.4* (1927.70 s)	22.8 (0 s)	18.3 (40 s)	17.4 (310 s)	124
Heskia	28	5	518/140	1967	23.3** (36,000 s)	29.1 (0 s)	24.2 (41 s)	23.5 (180 s)	222
Buxey	29	7	609/203	2921	17.8** (36,000 s)	26.0 (0.s)	19.4 (45 s)	18.8 (200 s)	6
Sawyer	30	7	645/210	3121	18.4** (36,000 s)	24.7 (0 s)	19.0 (43 s)	17.0 (200 s)	483
Lutz1	32	7	720/224	3556	24.1** (36,000 s)	30.9 (0 s)	26.3 (51 s)	23.7 (340 s)	81
Gunther	35	8	875/280	4856	25.0** (36,000 s)	36.1 (0.03 s)	23.3 (292 s)	23.4 (57 s)	345
Kilbridge	45	8	1350/360	8043	46.1** (36,000 s)	55.0 (0.02 s)	44.1 (72 s)	37.7 (320 s)	272
Hahn	53	8	1802/424	11,175	81.9** (36,000)	94.6 (0.05 s)	79.2 (130 s)	72.3 (330 s)	300
Warnecke	58	10	1933/580	16,678	72.1** (36,000 s)	79.3 (0.01 s)	61.2 (93 s)	61.2 (350 s)	394
Tonge	70	23	4025/1610	55,747	42.6** (36,000 s)	41.0 (0.02 s)	26.9 (100 s)	20.6 (460 s)	434
Wee-Mag	75	30	5025/2250	83,472	26** (36,000 s)	24.1 (0.04 s)	14.8 (110 s)	11.9 (490 s)	464
Arc83	83	21	5146/1743	71,701	– (36,000 s)	75.4 (0.10 s)	54.4 (130 s)	51.3 (530 s)	443
Lutz2	89	15	5251/1335	58,977	– (36,000 s)	114.9 (0.20 s)	108.1 (140 s)	104.5 (600 s)	273
Lutz3	89	24	6052/2136	94,239	– (36,000 s)	57.1 (0.30 s)	64.5 (130 s)	53.6 (410 s)	321
Mukherje	94	19	6157/1786	83,362	– (36,000 s)	82.6 (0.20 s)	66.0 (160 s)	65.1 (460 s)	421
Arc111	111	27	9102/2997	165,176	– (36,000 s)	105.6 (0.50 s)	84.8 (300 s)	53.5 (170 s)	493
Barthold1	148	14	12,950/2072	152,643	– (36,000 s)	344.6 (1 s)	315.0 (220 s9	306.5 (400 s)	309
Barthold2	148	40	16,798/5920	435,523	– (36,000 s)	97.4 (1 s)	84.9 (230 s)	78.5 (340 s)	493
Scholl	297	50	58,806/14,850	2198,497	– (36,000 s)	378.4 (1 s)	319.5 (320 s)	304.7 (240 s)	467

Note that, even for some of the smaller instances tested, the software CPLEX failed to obtain the optimal solution.

The improved genetic heuristic may be useful to support flexible production planning with the characteristics found in Section 2. In fact, if one considers practical situations in which the total number of parts may reach 5000 (Suresh & Kay, 1998), then the part families with precedence constraints problem cannot at the moment, in real time, be optimally solved. Hence, this heuristic may be used to approach a solution to the PFP. It also can act as a starting point to develop another group technology methodology within flexible manufacturing systems: for instance, solving the cell formation problem with the simultaneous assignment of the machines to cells and the parts to machines.

Acknowledgements

The authors are grateful to the two anonymous referees for their valuable comments. This research was partially supported by POCTI/ISFL/152 project.

References

Aronson, J. E., & Klein, G. (1989). A clustering algorithm for computer assisted process organization. *Decision Sciences*, *20*, 730–745.

Çatay, B., Vakharia, A., & Erengüç, S. (2002). A methodology for component grouping and printed circuit board scheduling in an openshop manufacturing environment, working paper http://bear.cba.ufl.edu/vakharia/vita/openshop.doc.

Cheng, C. H., Goh, C.-H., & Lee, A. (1996). Solving the generalized machine assignment problem in group technology. *Journal of the Operational Research Society*, *47*, 794–802.

Chen, W.-H., & Srivastava, B. (1994). Simulated annealing procedures for forming machine cells in group technology. *European Journal of Operational Research*, *75*, 100–111.

Finke, G., & Kusiak, A. (1985). Network approach to modelling of flexible manufacturing modules and cells. *RAIRO, Automatique-Productique informatique industrielle (APII)*, *19*(4), 359–370.

Ganesh, M. V., & Srinivasan, G. (1994). A heuristic algorithm for the cell formation problem. *Computers and Industrial Engineering*, *26*(1), 93–201.

Garey, M. R., & Johnson, D. S. (1979). Computers and intractability: A guide to the theory of NP-completeness. San Francisco: W.H. Freeman.

Goldberg, D. E. (1989). Genetic algorithms in search, optimization, and machine learning. Reading, MA: Addison-Wesley.

Gunther, R. E., Johnson, G. D., & Peterson, R. S. (1983). Currently practiced formulations for the assembly line balance problem. *Journal of Operations Management*, *3*(4), 209–221.

Hilllier, M. S., & Brandeau, M. L. (1998). Optimal component assignment and board grouping in printed circuit board manufacturing. *Operations Research*, *46*(5), 675–689.

Kusiak, A. (1985). The part families problem in flexible manufacturing systems. *Annals of Operations Research*, *3*, 279–300.

Lourenço, L. L., & Pato, M. V. (2001). *A binary linear programming formulation for the part families with precedence constraints problem. Proceedings of the IIE annual conference*, Atlanta: Publication of the Institute of Industrial Engineering, (CD-ROM production of EP).

Ng, M. (2000). Clustering methods for printed circuit board insertion problems. *Journal of the Operational Research Society*, *51*(10), 1205–1211.

OR-Library. http://www.ms.ic.ac.uk//info.html.

Pato, M. V., & Lourenço, L. L. (1997). A standard genetic algorithm for clustering with precedence constraints. *Investigação Operacional*, *17*, 71–86.

Reeves, C. R. (1993). Modern heuristic techniques for combinatorial problems. Oxford: Blackwell.

Rudolph, G. (1996). *Convergence of evolutionary algorithms in general search spaces*. *Proceedings of the third IEEE conference on evolutionary computing*, IEEE Press: Piscataway, NJ, pp. 50–54.

Scholl, A. (1995). Balancing and sequencing of assembly lines. Heidelberg: Physica-Verlag.

Stecke, K. E. (1986). A hierarchical approach to solving machine grouping and loading problem of flexible manufacturing systems. *European Journal of Operational Research*, *24*, 369–378.

Suresh, N. C., & Kay, J. M. (1998). Group technology and cellular manufacturing. Boston: Kluwer.

Viswanathan, S. (1995). Configuring cellular manufacturing systems: a quadratic integer programming formulation and a simple interchange heuristic. *International Journal of Production Research*, *33*(2), 361–376.

CPLEX Version 8.1 (2002). Using CPLEX callable Library. Incling Village, NV: ILOG.