A Double Grouping Problem in Generalized Cell Formation Solved Simultaneously by an Adapted Grouping Genetic Algorithm (SIGGA)

Emmanuelle VIN*[1], Pascal FRANCQ[2] and Alain DELCHAMBRE[1]
[1]*Beams Department, Université libre de Bruxelles (ULB), Brussels, B-1050, Belgium.*
** Corresponding author: Tel.: +(32) 2650-2814; Fax: +(32) 2650-4724; E-mail: Emmanuelle.vin@ulb.ac.be*
[2]*CoDE Department, Université libre de Bruxelles (ULB), Brussels, B-1050, Belgium.*

Abstract
This paper addresses the cell formation problem with alternative process plans and machine capacity constraints. The problem of Generalized Cell Formation consists in defining the preferential process and the preferential routing for each part optimizing the grouping of machines into independent manufacturing cells. The problem is reduced in two grouping problems: grouping operations (including the selection of process and routing) into machines (yielding flows between the machines) and grouping machines into cells. To solve these two interdependent grouping problems, we proposed an adapted grouping genetic algorithm (SIGGA). The chromosome is composed by two parts encoding both problems. The encoding is based on group encoding. The genetic operators are focalized on groups instead of treated objects. These operators are applied on both part of the chromosome to simultaneously evolve the population on both grouping problems. The used heuristics and operator are fully detailed. The algorithm has been tested on four literature cases presented by Sofianopoulou.

***Keywords*:** Cell Formation, Genetic Algorithm.

1. Introduction

Cellular production systems are an important application of group technology, which consists in decomposing system into sub-systems and in grouping similar things together. Cellular manufacturing systems are based on the creation and the management of several production cells. These cells are composed by complementary machines placed as close as possible and dedicated to a product family. A principal problem for the implementation of these cells is precisely the cell formation problem.

During the last years, the cell formation problem has been addressed in numerous works. Several methods have been presented and can be classified by different ways. We can use the classification proposed by Joines [12] to group the resolution methods into different categories as:

- Part Family identification methods
  - Classification and coding
  - Relational data bases
- Part-Machine grouping methods
  - Array based methods
  - Hierarchical clustering (similarity coefficient)
  - Non-hierarchical clustering
  - Graph theory
  - Mathematical programming
  - Artificial intelligence
  - Search heuristic

Joines presents a complete review of production oriented manufacturing cell formation techniques. In the artificial intelligence category, methods are grouped into Neural Network [19], Simulated annealing method [28], Tabu Search [18] or Genetic Algorithm.

Initially, the cell formation problem was really simple and the methods used to solve the creation of product families were much uncomplicated [12]. With time, the problem has evolved with the data complexity. New parameters have been taken into account to solve the cell formation problem. New complex methods have been proposed to solve these problems. These methods can be classified with regard to the complexity of the problem, and with regard to the production parameters taken into account such as sequence operation [10], cost [4], alternative process plans [10], part volumes [18], machine capacity [22], flexibility [5], labor-related factor (training, balanced work loads) [23], multiple objectives…

To take into account all these parameters, proposed methods make complex. Algorithms combine different methods as a multicriteria decision maker with a cell formation method [26], or two different methods to add the efficacy of both resolutions. In particular, to take into account process alternative, we consider not one but two grouping problems. The first one consists in allocating a machine for each operation respecting capability and capacity constraints (RP, Resource Planning problem). The second grouping problem tries to make independent cells in assigning a cell for each machine (CF, Cell Formation problem).

With both problems, a question arises: how to solve these two groupings and in which order? The resolution can be sequential, semi-simultaneous or completely simultaneous. The sequential resolution finds a solution for the second problem based on the result found for the first problem or conversely. When two problems are interdependent, a good solution for the first problem does not imply a good solution for the second problem. To avoid the inconvenient of the sequential resolution, the semi-simultaneous resolution is based on several iterations of the sequential resolution. But in this case, the second solution hardly depends on the first solution. Finally, the simultaneous resolution permits to optimize both problems simultaneously with each iteration.

Section 2 presents a state of the art based on the complexity of the parameters taken into account and on the simultaneous resolution. In section 3, the problem is

described with all used parameters and all used constraints. The proposed method explained in section 4 is based on the grouping genetic algorithm GGA and permits to solve simultaneously both grouping problems. The different characteristics of the implementation are explained in section 5. A case study is presented in section 6, before concluding in section 7.

2. Previous Works

2.1 Genetic algorithm

Genetic algorithms (GAs) are in order to explore and exploit a large space search to obtain a good solution [11]. Many authors use them to solve a cell formation problem. Venugopal and Narendran [25], Joines et al. [13] or Onwubolu and Mutingi [21] minimize the total cell load variation with a GA. Zhao and Wu [30] and Uddin and Shanker [24] used GA to solve a cell formation problem with alternative routes. Goncalves and al. [9], Filho and Tiberti [8] and Mahdavi and al. [19] presented a method based on a genetic algorithm to solve cell formation problem without alternative process plans.

2.2 Alternative Process

Suresh [23] proposes a classification and a review based on these parameters used for a more complex cell formation problem. Nevertheless, its review does not make any distinction between different alternative process plans. We proposed a distinction between different definition of routing and process plans in a previous paper ([26]). In this classification, we described the routing as the specific machine sequence to achieve a part. In contrast to routing, process is defined by a sequence of operations and then a sequence of machine types. When we talk about alternative process in this paper, we use several processes to define a part. Between the routing and the alternative process, we found cases with a few duplicate machines ([15] [4]), or with several routing defining a part ([15] [10] [18] [3] [1] [28]), or one process by part ([2] [6] [23] [29] [27]).

In working with alternative process, we need to choose for each part which process is the best and for each operation, the best machine to manufacture it. The cell formation is reduced in the choice of process, the choice of routing in this process, and grouping machine into cells. Authors using these complete alternative process plans are not frequent ([22] [24]). The resolution of these three problems can be sequential, semi-simultaneous and real simultaneous as explained in the introduction..

Many authors use the first strategy. Gupta [10] proposed a two-step algorithm to solve this problem. One routing is definitely determined for each part, respecting machine capacity constraints. Next, cell formation is achieved. The drawback of this method is its sequential approach. Routing selection is performed once and the flexibility given by alternative routings is not used to minimize inter-cellular moves.

To evolve the solutions taking into account flexibility in the choice of routing, authors use the second strategy. Nagi et al. [20] proposed an (semi-simultaneous) iterative method solving the two distinct sub-problems: cell formation, tackled with a heuristic and routing selection, addressed with the Simplex method. The use of the simplex limits the size of the considered problem. Caux et al. [3] proposed an approach based on simulated annealing and a branch-and-bound algorithm in order to perform routing selection and inter-cellular moves minimization simultaneously. Sofianopoulou [22] proposed an adapted simulated annealing-based heuristic. A mathematical model assigns randomly part type to a particular process plan and finds for this configuration, a machine-to-cell assignment minimizing the intercellular movement. This method is tested for duplicate machines and/or alternative process plans (defined as a sequence of machines). Vin and al. [27] proposed a semi-simultaneous method based on a genetic algorithm to solve the selection of preferential routing. Given an operation assignment, another integrated genetic algorithm finds the "best" associated grouping of machines into cell. Lei and Wu [17] defined an algorithm in which an initial solution is first generated by a new similarity coefficient-based method and is later improved iteratively by a tabu search algorithm.

The second strategy is often used. However, the methods proposed by these authors are rarely adapted for the complete alternative process plans where the search space is larger and the resolution more complex.

It is interesting to note that the complete problem with the process choice, the routing choice and the cell formation is always solved by maximum two strategies. The process choice is always done during the resolution of routing choice.

In this paper, we solve a cell formation problem with complete alternative process plans. We propose a new adapted algorithm (SIGGA, Simultaneous resolution by a Grouping Genetic Algorithm) based on a grouping genetic algorithm (GGA) to solve simultaneously both problems:

- The routing and process selection problem with the allocation of operations on a specific machine, yielding flows between the machines (resource planning problem with several constraints and criteria);
- The grouping of machines into independent cells (cell formation problem).

3. Description of the Problem

3.1 Notation

Indices

t Machines types index (TM_t=Machine type t). t=1,2,..,nt

m Machines index (M_m=Machine m). m=1,2,..,nm

i Products index (P_i=Product i). i=1,2,..,np

j Process index (Pr_{ij}=Process j of product i). j=1,2,.., npr_i

k Operations index (O_{ijk}=Operation k of process j of product i). k=1,2,.., no_{ij}

c Cells index (C_c=Cell c) c=1,2,.., n_c

Parameters

d_m Availability of machine m.

Q_i Quantity of product i.

T_{ijk} Average operating time of operation O_{ijk}.

T_{ijkm} Operating time if O_{ijk} on machine m.

n_c Maximum number of cells.

nm_c Maximum number of machines in cell c.

Necessary data and hypotheses are presented hereunder. A machine type has different capabilities in terms of operation types. Each machine m, unique, is characterized by an availability parameter dm, which is equal to its capacity value times its availability rate. This latter value takes the

possible failures into account. Each machine belongs to at least one type and can belong to several types if it is a multi-functional machine.

Each product is defined by a set process (Process = a sequence of npr_i operations $\{o_{ij1}, o_{ij2},\ldots, o_{ij\ npri}\}$). Each operation is not performed on one given machine, but is defined as an operation type that can be accomplished on one machine type (lathe, grinding machine, etc.). So each operation can be performed on all machines belonging to its type. The duration of each operation can be fixed for the considered machine type (average operating time, T_{ijk}), or particularized to a specific machine (operating time, T_{ijkm}).

3.2. Formulation

Decision variables

x_{ij} = 1 if process j of product i is used (= 0 otherwise).

y_{ijkm} = 1 if operation O_{ijk} is achieved on machine m (= 0 otherwise).

y_{ijkm^*} = 1 if operation O_{ijk} can be achieved on machine m (= 0 otherwise).

z_{mc} = 1 if machine m is in cell c (= 0 otherwise).

When the algorithm assigns an operation O_{123} to a specific machine M_5, variable x_{12} is put at 1 to specify that process j of product i is used in the solution. This variable implies that all other variables $x_{1j\neq2}$ of the same product (P_1) are put at 0. In this case, all operations belonging to $P_{1j\neq2}$ cannot be used in the grouping solution. To complete this notation, decision variable y_{1235} is also equal to 1.

Decision variable zmc is used to compute moves between cells as a function of the assignation of machines in each cell.

RP Constraints

$$\sum_{j=1}^{npr_j} x_{ij} = 1 \quad \forall i \tag{1}$$

$$\sum_{i=1}^{np}\sum_{j=1}^{npr_j}\sum_{k=1}^{no_{ij}} Q_i.T_{ijkm}.y_{ijkm} \le d_m \quad \forall m \tag{2}$$

$$if\,(y_{ijkm} = 1) => Q_i.T_{ijkm}. > 0 \quad \forall i,j,k,m \tag{3}$$

Constraint (1) represents the process selection. As explained above, only one process can be chosen by product. Second constraint defines the machine charge. This charge cannot exceed the machine availability. Constraint (3) determines that an operation assigned to a specific machine must have a strictly positive operating time. Indeed, if a machine cannot achieve an operation, the operating time T_{ijkm} of the operation O_{ijk} on the machine M_m will be null.

CF Constraints

$$\sum_{c=1}^{nc} z_{mc} = 1 \quad \forall m \tag{4}$$

$$\sum_{m=1}^{nm} z_{mc} \le n_c \quad \forall c \tag{5}$$

Constraints (4) and (5) concern the machine grouping into cells. The first one verifies that all used machines have been grouped. The second one confirms that each cell capacity is not exceeded. The maximum capacity can be different on each cell.

Cost Function

The proposed method is focused on one criterion: the minimization of inter-cellular moves.

$$\phi_{mn} = \sum_{i=1}^{np}\left(\sum_{j=i}^{npr_i} x_{ij}\cdot\left(\sum_{k=1}^{no_{ij}-1}\left(y_{ijkm}\cdot y_{ijkn}\right)\cdot\left(Q_i\cdot T_{ij(k+1)n}\right)\right)\right) \tag{6}$$

$$\Phi_{\text{intracell}} = \sum_{c=1}^{nc}\left(\sum_{m=1}^{nm_c}\sum_{n=1}^{nm_c}\left(z_{mc}.z_{nc}\right).\phi_{mn}\right) \tag{7}$$

$$\text{Cost function} : Max\frac{\Phi_{\text{intracell}}}{\Phi_{\text{Total}}} \tag{8}$$

Equation (6) represents the move between two machines, m and n. It is computed on the basis of the sum of operating time to achieve on machine n for all products coming from machine m. This value can be computed when the first part of the chromosome is completed and the first problem is solved. To compute the equation (7), the second part of the chromosome need to be completed and a valid solution of cell assignment found. The intra-cellular moves into a cell c are the sum of moves between all machines assigned to this cell c. The total intra-cellular move is the sum of intra-cellular moves for each cell. The total moves can be different depending on process and routing choice. To compare two solutions and take into account this difference, the criterion to maximize is the relative intra-cellular moves (8). This criterion is the same as the minimization of the total inter-cellular move.

4. SIGGA Description

4.1. Origins

The genetic algorithms (GAs) are an optimization technique inspired by the evolution process of living organisms (Holland [11]). The basic idea is to maintain a population of chromosomes, each chromosome being the encoding (a description or genotype) of a solution (phenotype) of the problem being solved. The worth of each chromosome is measured by its fitness, which is often simply the objective function value of the search space point defined by the (decoded) chromosome. Falkenauer (Falkenauer, 1998) pointed out the weaknesses of standard GAs when applied to grouping problems, and introduced the GGA, which is a GA heavily modified to match the structure of grouping problems. Those are the problems where the aim is to group together members of a set (i.e. find a good partition of the set). The GGA operators (crossover, mutation and inversion) are group-oriented, in order to follow the structure of grouping problems.

We [26] presented an genetic algorithm in two steps. The used algorithm is based on a semi-simultaneous method. First, a population is initialized by a RP heuristic. This population of chromosome corresponds to the first problem (operation/machine): allocation of operations on specific machines respecting capacity constraints. Next, for each solution, a new genetic algorithm is applied to complete the chromosome with a valid solution for the second problem (machine/cell): machine grouping into cells. When each chromosome owns two solution parts operation/machine and machine/cell, genetic operators are applied to make the

population evolve until a good solution is found. For each modified chromosome, the second GA is applied to reconstruct the complete solution and find the best associated grouping machine/cell.

This method was good but its application was not easy. First, the method was limited to cases studies with alternative routings but with only one process (sequence of machine types). Furthermore, the computation time was considerably long. Indeed, in the principal GA, the secondary GA is called to initialize the second part (machine/cell) of all chromosomes. So there are as many GA as the population size. Next, when the genetic operators are applied, the valid solution of the machine/cell must be computed again. A secondary GA is called to reconstruct all chromosomes after these operator applications. Finally, to have a good optimization and find the best secondary solution (machine/cell) associated to first solution (operation/machine), it was necessary to take into account the same criteria in the principal GA and in all secondary GAs.

The following question can be asked: why solve two problems with two different GAs, if the structure and the criteria are similar?

4.2. Description of SIGGA

The SIGGA (Simultaneous resolution by Grouping Genetic Algorithm) is presented in Fig 1.

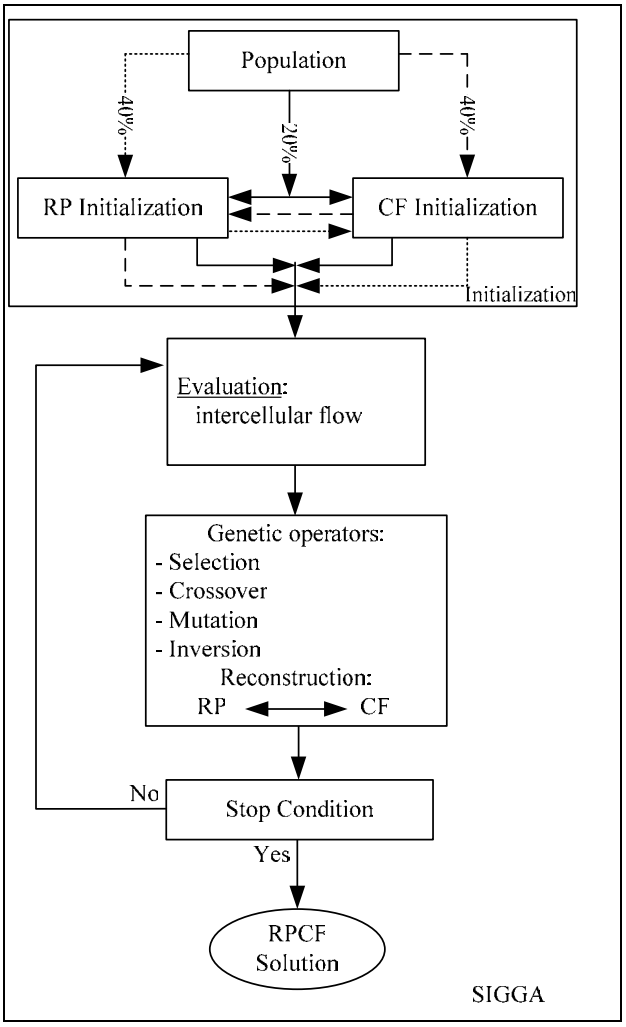

Fig. 1 Adapted SIGGA applied to the cells formation problem

This algorithm is based on a classical GGA. A population of chromosomes is initialized. Each chromosome represents a valid solution to both problems:

- Process selection and the assignment of each operation on a specific machine able to achieve it (Resource Planning Problem: operation/machine)
- Machine grouping into independent cells (Cell Formation Problem: machine/cell).

Both problems are interdependent because groups of the first problem are precisely the objects to group in the second problem. Our objective is to find a "good" solution for both problems (RP, Resource Planning, and CF, Cell Formation).

An initialization phases (section 5.3) generate an initial chromosome population. The heuristics generate only valid solutions respecting all hard constraints defined in section 3.2. After this initialization, each chromosome of the population is evaluated and chromosomes are sorted. The best chromosome is saved. In order to evolve to the best solution, different genetic operators (section 5.4) are applied after a specific selection (tournament strategy). These genetic operators are applied on the complete chromosome to make both problems evolve simultaneously. After the application of genetic operators, both parts of each chromosome are reconstructed (section 5.5). And a new generation is started. The algorithm stops when the stopping condition is reached (section 5.6).

5. Implementation of the SIGGA

5.1. Coding of the Chromosomes

The coding of chromosome is similar to the Grouping Genetic Algorithm (GGA). For the GGA, the standard chromosome of GA is augmented with a group part, encoding the groups on a one gene for one group basis. For example, a chromosome can be encoded as follows:

ADBCEBDE:ADBCE

with the group part written after the semicolon. Numbering the objects from 0 to 5, the object part of the chromosome can be explicitly written

01234567:

ADBCEBDE: ...,

meaning the object 0 is in the group labeled (named) A, 1 and 6 in the group D, 2 and 5 in B, 3 in C and finally 4 and 7 in group E. The group part of the chromosome represents only the groups. Thus

... :ADBCE

expresses the fact that there are five groups in the solution. This chromosome represents the following solution:

A={0}, B={2,5}, C={3}, D={1,6} and E={4,7}.

In fact, the chromosome could also be written in a less visual way as

{0}A{2,5}B{3}C{1,6}D{4,7}E.

On the basis of this encoding, the chromosome is defined with three parts as followed:

- A string 0-1 with the parameter x_{ij}. This solution defines which process is used for each part.
- A grouping encoding to define the allocation of operations into machines. Each group is associated to a specific machine to own specific characteristics as capacity, capabilities to produce operations… But to take

into account the similar machine, and distinguish two same solutions where these duplicate machines are inversed, the grouping encoding is necessary. In function of the solution x_{ij}, all operations are not used and the grouping solution includes some "0" to precise the operation belong to a non used process.

ADB00CBE:ADBCE

- A grouping encoding to define the machine grouping into cell. Each cell can be completed with specific information about its size, location, …

ffgfg:fg

More concretely, let us consider an example. Part 1 is defined by two processes (operation sequences 0-1-2 or 3-4). Part 2 has just one process (operations sequences 5-6-7). The following chromosome represents a possible solution:

ADB00CBE:ADBCE::ffgfg:fg

The chromosome encodes the solution for a double grouping problem where the solution of the first problem, composed by 8 objects/operations and 5 groups/machines, can be written as

A={0}, B={2,6}, C={5}, D={1} and E={7}.

Objects/operations 3 and 4 are not used in this solution. The first process has been chosen for part 1.

The second problem solution, composed of the 5 objects/machines (equivalent to the groups for the first problem) and 2 groups/cells, is:

f={A,B,D}, g={C,E}.

The visual encoding can be written as follows:

{0}A{1}D {2,6}B{5}C{7}E:{A,B,D}f{C,E}g.

This encoding will permit us to apply the genetic operator on the groups rather than the objects.

5.3. Initialization

The objective in both problems is to explore the space without limiting the search. It means that we need to test the solution with all groups and with the not used groups. For example, the best solution can be the one with two non used machines producing the minimal inter-cellular moves.

To have a population sufficiently diversified, the initialization is decomposed as followed:

- 40 % of the population is first initialized by the rand group heuristic (explained here below) applied to the RP problem. The CF problem is then constructed with a specific heuristic oriented to the flow.
- 40 % of the population is constructed conversely with a flow heuristic on the basis of random solution CF.
- 20 % of the population is randomly constructed.

Rand Group Heuristic

Our **Rand Group Heuristic** is based on a first fit with a first random part. The different objects are treated in random order. To authorize the creation of solutions with different number of group, a probability *p* is used to create a new group.

The heuristic is decomposed in several steps:

Step 1. **Verify** if the current object can be used in order to respect the hard constraints with the actual solution in construction (One used process by product (RP), only not empty machine can be grouped (CF)).

Step 2. **Create a new group** with a variable probability equal to

$$p = \frac{MaxGroups - NbGroups}{MaxGroups} \quad (10)$$

where:

- *MaxGroups* = maximum of allowed groups for the problem,
- *NbGroups* = actual number of used groups.

The probability p to create a new group before the assignment of the object in a group decreases with the number of created groups. The number of used groups will not necessary be minimal. Randomly, solutions will not contain the same number of groups except if the capacity requires it.

Step 3. **Find the first group** (disposed in a random order) able to accept the current object in order to respect the hard constraints about capacity or compatibility (Capacity not exceeded (RP and CF), machine able to achieve the operation (RP)). If a group is found, the object is inserted. Otherwise, a new group is created to accept the object.

The algorithm stops when all objects have been tested and inserted if they had to be used.

Flow Heuristic CF

The flow heuristics are based on the precedent Rand Group Heuristic. For **Flow Heuristic CF**, the first step is completed with the compute of the flow matrix ϕ_{mn}. The third step (**Find Group)** is decomposed as followed:

1. **Verify** if the RP is solved. If RP is not solved, go to step 6.
2. **Find** the cell c_{max} with maximum flow ϕ_{max_cell} between current object/machine and the group/cell.
3. **Find** the machine m_{max} with maximum flow ϕ_{max_Mach} between current object/machine and an object/machine not yet assigned.
4. **Assign** the current object to the cell c_{max} if $\phi_{max\_Cell} > \phi_{max\_Mach}$, and if capacity constraints are respected (maximum size of the cell is not reached),
5. **Create** a new cell, if $\phi_{max\_Cell} < \phi_{max\_Mach}$, and if capacity constraints are respected (maximum number of cells is not reached). The current object will be assigned to the new cell with the machine m_{max}
6. **Apply** step 3 of the Rand Group Heuristic, if not group is founded.

Process Heuristic RP

For **Process Heuristic RP**, the flow matrix ϕ_{mn} can not be computed because the operations are not yet allocated to machines. The solution CF proposes a machine grouping into cell, created by the Rand group Heuristic CF. In the first step, the selection of the product process is based on the following computation

The parameter n_{ijc} for process *j* (of part *i*) representing the number of operations that can be achieved in cell *c*. The parameter $\tilde{m}_{ij}$ defines the maximum of operations belonging to the process *ij* that can be achieved in a same cell. The selected process for part *i* will be the one with the maximum value $\tilde{m}_{ij}$. If there are several processes with the same value $\tilde{m}_{ij}$, a draw is made.

$$n_{ijc} = \sum_{m} \sum_{k} y_{ijkm^*} . z_{mc} \quad (11)$$

$$\tilde{m}_{ij} = n_{ij\tilde{c}} \mid \tilde{c} = \max_{c}\left(n_{ijc}\right) \quad (12)$$

$$x_{i\,\tilde{j}} = 1 \mid \tilde{j} = \max_{j}\left(\tilde{m}_{ij}\right) \quad (13)$$

When the process is selected for each part, the heuristic can be applied with the object treated in a random order. The **Find Group** step is decomposed as followed:

1. **Verify** if the CF is solved. If CF is not solved, go to step 4.
2. **Find** a machine m belonging to the cell c able to achieve the current object/operation.
3. **Assign** the current object/operation to group/machine *m* if capacity constraints are respected (maximum charge of the machine is not reached).
4. **Apply** step 3 of the Rand Group Heuristic, if not group/machine is founded.

This heuristic is applied during the initialization phases because the choice of processes is based on the machine grouping

5.4. Genetic Operators

The important point is that the genetic operators will work with the group part of the chromosomes, the standard object part of the chromosomes serving to identify which objects actually form which group. Note in particular that this implies that operators will have to handle chromosomes of variable length with genes representing the groups.

Operators defined here below are applied with a different probability on the first or second part of the chromosome. These probabilities are computed by the program in function of the number of possible solutions for both problems. By this way, if the RP problem includes only few duplicate machines, the genetic operators can be focused on the part CF of chromosomes.

Selection

The tournament strategy is chosen to select the chromosome for the application of operators. The idea is to create an ordered list of the individuals with the best solution always at the top, and the others ordered according to a specific method described below. The upper part of the list will further be used when “good” chromosomes are needed, while the lower part for “bad” chromosomes. An initial set of all the individual identifiers is established. Two identifiers of this set are chosen randomly, and their fitness values are compared. The best of the two - the one corresponding to the individual with the best fitness value - is reinserted into the set, while the other is pushed into the list. This method is repeated until all the identifiers in the set are in the list; this process leads to the construction of an ordered list of individuals, with the best one at the top of the list. The operator can then presume that the chromosomes ranked in the top half will be used as parents for the crossovers and the resulting children will replace the chromosomes in the bottom half.

Crossover

The crossover is applied at each generation. After tournament selection, half chromosomes are crossed and overwrite the worse half.

First two parents are selected. The crossover will be applied on the RP problem and CF problem with a probability of respectively CrossRP % and CrossCF %. In this way, if the sum of CrossRP and CrossCF is greater than 100 % the crossover can be applied simultaneously on both parts.

The circular crossover will fit the following pattern:

1. A crossing site is randomly selected in each parent. Each part of chromosome is represented by a ring. In this case, the crossing site defined by two-point crossover can include interne groups as well as groups located at the extremities of the chromosome.
2. Groups selected by the crossing site of one parent are inserted at the second parent crossing site. At this stage, some objects may appear in more than one group.
3. The objects cannot appear twice in one solution. The new injected objects have the priority. So, the existing groups containing objects that are already in the inserted groups are eliminated. If some groups are empty, they are also removed from the solution.
4. The validity of the solution is verified according to the hard constraints relative to the cell formation problem. The used process can be differed in two parents. Two process of a same product cannot coexist in the solution. Moreover, a specific machine cannot appear twice. Compatibility is tested between inserted groups and existing groups. If the existing groups contain operations belonging to another process or the groups corresponding to an inserted machine, these groups are also eliminated.
5. The objects left aside are reinserted into the solution. It is the reconstruction phases (explained in section 5.5).
6. The other part of the chromosome is adapted and reconstruct if it is necessary.

Here under, an illustration of the crossover is presented. Firs parent is defined in the section of coding.

Parent 1: ADB00CBE:ADBCE::ffgfg:fg

$\{0\}_A\{1\}_D\{2,6\}_B\{5\}_C\{7\}_E$:{A,B,D}$_f${C,E}$_g$.

Parent 2: 000BACAE:BACE::fgg0f:fg

$\{3\}_B\{4,6\}_A\{5\}_C\{7\}_E$:{A,E}$_f${B,C}$_g$.

The crossover is applied to the first part of the chromosome. The crossing sites are defined.

Parent 1: **ADB00CBE:AD|$_1$B|$_2$CE::ffgfg:fg**

Parent 2: 000BACAE:BA|$_2$CE|$_1$::fgg0f:fg

The group **B** containing object 2 and 6 from parent 1 is inserted in parent 2. The inserted group is in bold to distinguish it from the existing groups.

Enfant 1: 00**B**BAC**B**E:BA|$_2$CE|$_1$**B**::fgg0f:fg

$\{3\}_B\{4,6\}_A\{5\}_C\{7\}_E\{\mathbf{2,6}\}_\mathbf{B}$:{A,E}$_f${B,C}$_g$

Inserted group **B** contains object 2 belonging to the first process of part 1. However, the used process in parent 2 was the second one. As the inserted group has the priority, all objects belonging to the second process are eliminated. Objects (6) appearing twice are deleted. The object/operation 0 and 1 belonging to first process and object/machine **B** must be reinserted.

Enfant 1: 00**B**00C**B**E:BACE**B**::fggf:fg {0,1}{**B**}

$\{\}_B\{\}_A\{5\}_C\{7\}_E\{2\}_\mathbf{B}$:{A,E}$_f${B,C}$_g$

Group are adapted, and empty group B is eliminated from the first and second part of the solution.

Enfant 1: 00**B**00C**B**E:CE**B**::00g0f:fg {0,1}{**B**}

$\{5\}_C\{7\}_E\{2,6\}_\mathbf{B}$:{E}$_f${C}$_g$

The leaving objects {0,1} and {**B**} are reinserted by a specific heuristic respecting capability and capacity constraints.

Enfant 1: AD**B**00C**B**E:CE**B**::00g0f:fg {**B**,A,D}

$\{5\}_C\{7\}_E\{2,6\}_{\mathbf{B}}\{0\}_A\{1\}_D$:$\{E\}_f\{C\}_g$

After the reconstruction phases, the heuristic CF is applied to regroup new groups inserted by the crossover or created by the reconstruction heuristic.

Enfant 1: ADB00CBE:CEBAD::ffggf:fg

$\{5\}_C\{7\}_E\{2,6\}_{\mathbf{B}}\{0\}_A\{1\}_D$:$\{A,B,E\}_f\{C,D\}_g$

The same procedure is applied to groups B and A included in the crossing site of parent 2 to be injected in parent 1 at point "$|_1$".

Mutation

The role of a mutation operator is to insert new characteristics into a population to enhance the search space of Genetic Algorithm. But the operator must be defined as the smallest possible modification with respect to the encoding. And this operator must be applied as small as possible to let the population evolve with the crossover (Falkenauer, 1998).

To respect these considerations, the mutation is not applied as classic genetic algorithm with a probability between 1 and 5 %. The mutation is applied when there is no more evolution of the population and the best chromosome of the population is no changed since a specific number of generations (=*AgeBest,* The age of a solution is the number of generations during which the solution is not modified). This number defines the best chromosome age in the population. The mutation is applied when this age is reached. The best chromosome is mutated and replaces the worst one. In the classic genetic algorithm with binary coding, a bit is transformed. With our grouping coding, there is no sense to remove one group of the solution. To apply a modification in the grouping, minimum of two groups must be treated to reassigned differently removed objects.

As defined for the crossover, two parameters, MutRP % and MutCF %, defined probabilities to apply mutation on first and/or second part of the chromosome.

In our case, the mutation is based on two principal aspects:

1. The first idea is to randomly choose a few groups and to remove them from the solution. The objects attached to these groups are then reinserted into the solution.
2. The second idea is specialized for the cell formation problem. It is to force the use of another process for a few products. All operations of selected process are eliminated and the reconstruction phases is applied to the objects belong to the new process.

Thus for instance, the chromosome

ADB00CBE:ADBCE::ffgfg:fg

could be mutated into

A0B000BE:A**0**B**0**E::ffgfg:fg

and reconstructed into

ACB00CBE:A**0**B**C**E::ffg0g:fg

The method used in the reconstruction phases is the same as the one used for the crossover operator.

Inversion

The role of the inversion operator is to propose the same solution to the Grouping Genetic Algorithm, but differently. As pointed out in 6.1, a single solution may have different presentations, and because crossovers work through crossing sites, the way in which a solution is presented influences the crossover operator's results. The groups appearing close to each other in the group element of a chromosome has likely more probability to be chosen and transmitted together to the children (offspring). It is therefore important to include this operator in the Grouping Genetic Algorithms. This operator changes the position of the groups on the chromosome without changing the solution. Thus for instance, the chromosome

ADB00CBE:ADBCE::ffgfg:fg

could be inverted into

ADB00CBE:A**C**B**D**E::ffgfg:fg

The object part of the chromosome stays unchanged. Indeed, the groups are still composed if same objects. Only the position of the representation of the groups is changed. In this case, if groups D and E represent well performing groups, the probability of transmitting both of them during the next crossover is improved after the inversion, since there are closer together.

5.5. Reconstruction

After operators of crossover and mutation, it is necessary to reinsert all the objects not assigned. If only one part of the chromosome is modified, the specialized heuristic (Flow Heuristic RP or CF) is applied to reconstructed the modified part. If both parts are crossed and/or mutated, one part is randomly selected and reconstructed with the Rand Group Heuristic. And the specialized heuristic is applied to reconstruct the second part of the chromosome. The specialized heuristic for part RP of the chromosome is lightly different in this reconstruction phases. The process selection for the chromosome depends on the crossover. In this case, the reconstruction phases for the part RP is based on the computation of maximum flow between the object to insert and the existing cell.

5.6. Stopping condition

A stop condition must determine when a GA stops. When the problem to be optimized is well defined, a particular value of the cost function can be used as exit condition. But such a value is unknown for our grouping problem. Another exit criterion must therefore be defined. The trivial idea used is to run the SIGGA a fixed number of generations. Another possible idea could be to let the SIGGA running until the age of the best solution ever computed is greater than a given threshold.

The number of generations influences the quality of the solution. In fact, the number of generations has a double effect on a Genetic Algorithm (GA):

- When the number of generations increases, the quality of the GA should also increase.
- When the number of generations decreases, the computational time needed also decreases.

The maximum number of generations must always be chosen by balancing these two effects one against the other.

6. Application, case study

The algorithm has been tested with different case studies found in the literature. In this article, we will present the four case studies used by Sofianopoulou [21]. The advantage of these four cases is that they represent a set of the different processes presented in the section 2. The implementation of the SIGGA algorithm for these problems is coded in C++ and run on a Bi-Xeon 3.60Ghz Hyper Threading with 1 Go RAM.

For each problem, the solution is presented in a table where the cell composition is determined by the machines and the products assigned to each cell. When the product is written in parentheses in two cells, there are many moves in each cell. In this case, the product can be assigned

independently to each cell. Furthermore, the selected process plan is defined in parentheses for each product with alternative process plans.

6.1. problem 1

The first problem (P1) is an adaptation from Kusiak [14] to take into account the processing sequence of each part. We consider 5 products and 4 machines. The particularity of this case is the use of alternative routings (alternative machines sequences). Each type of machine is composed by only one machine. The different routings have not necessary the same number of operation as shown by the fifth product. The algorithm solves this problem for a maximum cell size n_{mc}=2.

The solution is presented in table 1. Same solution has been found by Sofianopoulou and Kusiak with a number of inter-cellular moves equal to 0. The selected process for each product is based on Sofianopoulou's notations.

Table 1: Solution of Problem 1

Cell	Machines
1	1, 3
2	2, 4
Selected process for each product	
1(2), 2(2), 3(2), 4(2), 5(2)	

6.2. problem 2

The second problem has the same particularity as the first one except for the size. The algorithm tries to group 20 products and 12 machines into 3 cells. The maximum cell size is equal to 5. The solution to this problem is presented in table 2. The algorithm was run for 500 generations and the corresponding runtime was about 3 seconds.

Table 2: Solution of Problem 2

Cell	Machines
1	1, 4
2	2, 6, 7, 9, 10
3	3, 5, 8, 11, 12
Selected process for each product	
2(2), 5(1), 12(2), 14(2), 17(2)	

The solution contains the same number of inter-cellular moves (29) as the Sofianopoulou's solution (for a total of 65 moves). However, the selected process plans for each product is not the same and the machines are not allocated into the same cell. The principal difference however with Sofianopoulou's solution is the type of moves. In our algorithm, the cell formation heuristic is adapted to preferentially produce unidirectional moves. To simplify moves between cells, it is better to visit cell 1, cell 2 and cell 3 than to visit cell 1, cell 2 and to come back to cell 1. The proposed solution has 13 go-returns and 16 simple moves between cells. Sofianopoulou's solution has the opposite, 16 go-returns, and 13 simple moves.

6.3. problem 3

Problem 3 is composed of 20 products and 14 machines (12 machine types). Two machines are duplicated. Each product is characterized by a unique process sequence. The utilization of duplicate machines implies one or several routings (machine sequence) for each product. As for problem 2, cell size is limited to 5. The algorithm was run for 100 generations in 7 seconds. The solution is presented hereunder.

Table 3: Solution of Problem 3

Cell	Machines
1	1,2, 4, 7, 10
2	3, 5, 6, 11,13
3	8, 9, 12, 14
Selected process for each product	
1(1), 2(1), 3(1), 5(2), 6(2), 8(1), 9(1), 10(1), 11(4), 12(1), 13(1), 15(1), 16(2), 17(2), 18(2), 19(1), 20(1)	

The solution is presented in table 3. The same observations as for the problem 2 can be done. The selected process plans are not equivalent but in this case, the types of moves are equivalents (25 for a total of 65 moves).

6.4. problem 4

Problem 4 is an application of the alternative process plans. Each product is defined by several processes (machine type sequence) and each machine type can contain one or two machines. So the problem will consist to define the preferential process and the preferential routing for each product. This problem is composed by 30 products and 18 machines regrouped into 16 machine types. For this problem, the cell size is limited to 7 machines by cell. The solution presented in table 4 is found with 100 generations in 20 seconds. The best total number of inter-cellular moves found by the algorithm is 32 for a total of 86 moves, less than Sofianopoulou's solution (34).

Table 4. Solution of Problem 4

Cell	Machines
1	1, 2, 11, 13, 14, 15, 18
2	5, 7, 8
3	3, 4, 6, 9, 10, 12, 16
Selected process for each product	
1(1), 2(1), 3(3), 4(2), 6(3), 8(2), 10(2), 11(1), 12(3), 13(1), 16(1), 21(1), 23(1), 24(2), 26(2), 29(5)	

Sofianopoulou needed 100 seconds to solve these two last problems. The presented SIGGA is fast and efficient with all types of data in term of alternative process plans and alternative routings.

7. Summary

In the present paper, an adapted grouping genetic algorithm has been developed to solve a double grouping problem. The algorithm SIGGA can solve simultaneously two grouping problems. It is particularized to the cell formation problem for manufacturing systems. We took into account the three most important production parameters in cell design:

- The process sequence allows to define the flows between machines;
- The production volume allows to compute the real material moves between machines and between cells;
- The use of real alternative process plans allows optimizing the cell formation.

In this paper, the algorithm is applied with a flow criterion to four case study used by Sofianopoulou. These four cases are a good variety of the different processes utilization. The algorithm is efficient with both high and low flexibility cases.

References

[1] Adenso-Diaz, B, Lozano, S. and Racero, J., Guerrero, F. Machine cell formation in generalized group technology. *Computers and industrial engineering*, Vol. 41, No. 2, pp. 227-240, 2001.

[2] Askin, R., Selim, H. and Vakharia, A. A methodology for designing flexible cellular manufacturing systems. *IIE transactions*, Vol. 29, No. 7, pp. 599-610, 1997.

[3] Caux, C. Cell formation with alternative process plans and machine capacity constraints: A new combined approach. *International Journal of Economics*, Vol. 64, pp. 279–284, 2000.

[4] Choobineh, F. A Framework for the Design of Cellular Manufacturing Systems. *International Journal of Production Research*, Vol. 26, No. 7, pp. 1161-1172, 1998.

[5] Defersha F.M. and Chen M. A Comprehensive Mathematical Model for the Design of the Cellular Manufacturing. *Systems. International Journal of Production Economics*, Vol. 103, pp. 767-783, 2006.

[6] Diallo, M., Pierreval, H. and, Quilliot, A.. Manufacturing cells design with flexible routing capability in presence of unreliable machines. *International Journal of Production Research*. Vol. 74, No. 1, pp.175-182, 1993.

[7] Falkenauer, E., 1998. Genetic algorithm and grouping problem. Wiley & Sons.

[8] Filho A.V.G. and Tiberti, A.J. A group genetic algorithm for the machine cell formation problem. *International Journal of Production Economics*. Vol. 102, pp.1-21, 2006.

[9] Goncalves J.F. and Resende M.G.C. An evolutionary algorithm for manufacturing cell formation. *Computers & Industrial Engineering*. Vol. 47. Pp. 247–273, 2004.

[10] Gupta, T. Design of manufacturing cells for flexible environment considering alternative routing. *International Journal of Production Research*, Vol. 31, No. 6, pp. 1259-1273, 1993.

[11] Holland, J. H., 1975. Adaptation in Natural and Artificial Systems. Ann Arbor: University of Michigan Press.

[12] Joines J.A., Culbreth C.T. and King R.E. A Comprehensive Review of Production Oriented Cell Formation Techniques. *International Journal of Factory Automation and Information Management*, Vol. 3, No. 3-4, pp. 225-265, 1996.

[13] Joines J.A., Culbreth C.T. and King R.E. Manufacturing cell design: an integer programming model employing genetic algorithms. *IIE Transaction*. Vol. 28. pp. 69-85, 1996b.

[14] King. Machine-Component grouping in production flow analysis: an approach using a rank order clustering algorithm. *International Journal of Production Research*. Vol. 18. No. 2. pp. 213-232, 1980.

[15] Kusiak, A. The Generalised Group Technology Concept. *International Journal of Production Research*, Vol. 25, No. 3, pp. 561-569, 1987.

[16] Lee, R., Ramakrishna P. and Srikandarajah, C. Tabu search-based heuristics for cellular manufaturing systems in the presence of alternative process plans. *International Journal of Production Research*, Vol. 32, No. 2, pp. 273-297, 1994.

[17] Lei, D. and Wu, Z. Tabu search approach based on a similarity coefficient for cell formation in generalized group technology. *International Journal of Production Research*, Vol. 43, pp. 4035–4047, 2005.

[18] Logendran, R., Ramakrishna P. and Srikandarajah, C. Tabu search-based heuristics for cellular manufaturing systems in the presence of alternative process plans. *International Journal of Production Research*, Vol. 32, No. 2, pp. 273-297, 1994.

[19] Mahdavi I., Javadi B., Fallah-Alipour K. and Slomp J. Designing a new mathematical model for cellular manufacturing system based on cell utilization. *Applied Mathematics and Computation*, Vol. 190, No. 1, Pp 662-670, 2007

[20] Nagi, R., Harhalakis, G. and Proth, J.-M. Multiple routings and capacity considerations in group technology applications. *International Journal of Production Research*, Vol. 28, No. 12, pp. 2243-2257, 1990.

[21] Onwubolu GC. and Mutingi M. A genetic algorithm approach to cellular manufacturing systems. *Computer and industrial engineering*. Vol. 39. pp. 125-144, 2001

[22] Sofianopoulou, S. Manufacturing cell design with alternative process plans and/or replicate machines. *International Journal of Production research*, Vol. 37, No. 3, pp. 707-720, 1999.

[23] Suresh, N.C. and Slomp, J. A multi-objective procedure for labour assignments and grouping in capacitated cell formation problems. *International Journal of Production research*, Vol. 39, No. 18, pp. 4103-4131, 2001.

[24] Uddin, M.K. and Shanker, K.. Grouping of parts and machines in presence of alternative process routes by genetic algorithm. *International journal of production economics*, Vol. 76, No. 3, pp. 219-228, 2002.

[25] Venugopal, V. and Naredran, T. T. A genetic algorithm approach to the machine-component grouping problem with multiple objectives. *Computers and Industrial Engineering*. Vol. 22, pp. 469-480, 1992.

[26] Vin, E., De Lit, P. and Delchambre, A. An integrated approach to solve the cell formation problem with alternative routings. Une approche intégrée pour résoudre le problème de formation de cellules de production avec des routage alternatifs. *Proceedings of 4e Conférence Francophone de MOdélisation et SIMulation*, Vol. 1, pp. 43-50, 2003. Toulouse, France.

[27] Vin, E., De Lit, P. and Delchambre, A. A Multiple Objective Grouping Genetic Algorithm for the Cell Formation Problem with Alternative Routings. *Journal of Intelligent Manufacturing*, Vol 16, No. 2, pp. 189-206, 2005.

[28] Wu T.H., Chang C.C and Chung S.H. A simulated annealing algorithm for manufacturing cell formation problem. *Expert Systems with Applications*. Vol. 34, No. 3, Pp. 1609-1617, 2008.

[29] Yin, Y. and Yasuda, K. Manufacturing cells' design in consideration of various production factors. *International Journal of Production Research*, Vol. 40, No. 4, pp. 885-906, 2005.

[30] Zhao C. and Wu Z. A genetic algorithm for manufacturing cell formation with multiple routes and multiple objectives. *International Journal of Production Research*. Vol. 38, No. 2, pp. 385-395, 2000.