Team formation based on group technology: A hybrid grouping genetic algorithm approach

Luis E. Agustín-Blas, Sancho Salcedo-Sanz *, Emilio G. Ortiz-García, Antonio Portilla-Figueras, Ángel M. Pérez-Bellido, Silvia Jiménez-Fernández

Grupo de Heurísticos Modernos de Optimización y Diseño de Redes (GHEODE), Department of Signal Theory and Communications, Universidad de Alcalá, 28871 Alcalá de Henares, Madrid, Spain

**Abstract**

This paper presents a new model for team formation based on group technology (TFPGT). Specifically, the model is applied as a generalization of the well-known Machine-Part Cell Formation problem, which has become a classical problem in manufacturing in the last few years. In this case, the model presented is especially well-suited for problems of team formation arising in R&D-oriented or teaching institutions. A parallel hybrid grouping genetic algorithm (HGGA) is also proposed in the paper to solve the TFPGT. The performance of the algorithm is shown in several synthetic TFPGT instances, and in a real problem: the formation of teaching groups at the Department of Signal Theory and Communications of the Universidad de Alcalá in Spain.

© 2010 Elsevier Ltd. All rights reserved.

---

1. Introduction

Correct management of their human resources is one of the key factors in the success of many companies and organizations [1–22]. There are many different problems related to human resources that have been previously tackled in the literature: formation of project management teams [10,11], formation of multi-functional work-force teams [15,14], task force teams organized by companies in order to carry out certain tasks [2], or student-teams for improving the learning process [23,24]. The objective in the majority of these problems, in terms of the expected performance of the team, is to select a (some) leader(s) (project manager, task force chief, etc.) and team(s) members from the available staff [10]. Specifically, at teaching or R&D-oriented institutions, such as universities or research institutes, the staff is usually organized in teams at the outset (departments, research groups, teaching groups, etc.), which have an enormous influence on the performance of the entire organization [10]. Unlike the problems mentioned above, in the majority of cases, the organizational teams in this kind of institution do not require a leader: the final objective of grouping the staff is to specialize groups on a given subject resource, allowing all of the members of a given group to have a similar knowledge (skill) of the resource.

Recently, several researchers have dealt with the problem of team formation without leaders in different scenarios, such as multi-functional skill requirements of the teams. In these situations, the makeup of the teams must take into account the personal or technical skills of the staff. In situations with very different staff skill categories, the problem may be difficult, and there is not a unified definition of the model to be used. In [14], a novel approach to this multi-functional version of team formation is presented, and fuzzy planning to match customer’s requirements and engineers’ characteristics (skills or knowledge) is proposed. Different works have also used this fuzzy model, such as [15,13]. In [12], the authors use fuzzy modeling to take into account the different skills of the staff in different tasks and propose a fuzzy decision-support system based on evolutionary programming. In fact, the authors proposed an island model to improve the performance of their approach in the team formation problem.

The purpose of the present paper is to introduce an alternative model to deal with the multi-functional team formation problem, based on group technology problems which arise in manufacturing. Group technology has been defined as “an approach to the organization of work in which organizational units are relatively independent groups, each responsible for the production of a given family of products” [25]. One of the key problems of group technology is the Machine-Part Cell Formation (MPCF) problem [19,17,21,22] in which parts and machines in a given manufacturing process are assigned to different cells, maximizing the machine...
utilization within a cell and minimizing the between-cell movement of parts. This allows manufacturers to group together products with similar processing needs and characteristics, identifying the set of machines needed to process these product families. In this paper, we extend the MPCF to model the multi-functional team formation problem in teaching or similar institutions (Team Formation Problem based on Group Technology (TFPGT), hereafter). We state the TFPGT and propose a hybrid grouping genetic algorithm (HCGA) to solve it. The grouping genetic algorithm was introduced by Falkenauer in [26]. Since then, this approach has been successfully applied to numerous problems [26–31] in very different fields, including the MPCF [18,17]. In this paper, the hybrid-grouping genetic algorithm employed is modified to consider the peculiarities of the TFPGT. First, we introduce a novel encoding of the problem into the grouping genetic algorithm. Using this new encoding, each individual is composed of three parts: the first one is called the assignment part; the second one is called the resource part; and the third one the groups part. The objective function of the TFPGT has been adapted from the MPCF model to consider the peculiarities of the TFPGT. Also, a local search is introduced in the grouping genetic algorithm to improve its performance. A final implementation of the algorithm using simulations in several randomly generated TFPGT instances is conducted in the experimental section (Section 4). Also in this section, a real application in the organization of the teaching at the Department of Signal Theory and Communications, Universidad de Alcalá, Madrid, Spain, is carried out as a final test for the model and algorithm proposed in this paper. Section 5 closes the paper, giving some final conclusions on the work presented.

### 2. Team formation based on group technology: definition of the problem

#### 2.1. Background: the MPCF problem

The MPCF is a classical problem in Group Technology. It is formulated as a block diagonalization problem of a 0–1 machine-part (MP) matrix [17]. Specifically, given $A$ (binary machine-part incidence matrix), an element $a_{ij} = 1$ stands for a machine $j$ needed by part $i$, whereas $a_{ij} = 0$ otherwise. The objective of the problem is to obtain a different matrix $A'$, with columns and rows rearranged in an order corresponding to the groups identified by the MPCF solution. To clarify the problem Fig. 1 shows an example of MP incidence matrix $A$, with 10 machines and 15 parts. Fig. 2 shows the resulting block diagonal matrix $A'$, containing three cells, the first cell contains machines $[1, 7, 10]$ and parts $[2, 7, 10, 11, 12]$, the second cell contains machines $[2, 5, 8]$ and parts $[3, 5, 8, 13, 15]$ and finally, the third cell contains machines $[3, 4, 6, 9]$ and parts $[1, 4, 6, 9, 14]$.

The block structure of the MPCF solution must be evaluated in terms of a quality measure. Several measures to this end have been defined in the literature [3,4,6,7]. The original measure is due to Chandrasekharan and Rajagopolan [3]. It was called grouping efficiency, and is defined as follows:

$$
\eta = q\eta_1 + (1-q)\eta_2
$$

where $q$ is a weighting factor, $\eta_1$ is the ratio of the number of 1s in the diagonal blocks to the total number of 0s in these diagonal blocks, and $\eta_2$ is the ratio of the number of 0s in the off-diagonal blocks to the total number of 0s and 1s in the off-diagonal blocks. An improved measure called grouping efficacy was proposed soon later by Kumar and Chandrasekharan in [4], defined as

$$
\tau = \frac{e - e_0}{e + e_v}
$$

where $e$ is the total number of 1s in a given MP matrix $A$, $e_v$ is the number of voids (0s within diagonal blocks) and $e_0$ is the number of exceptions (1s off-diagonal blocks). Other measures have been proposed for the MPCF, such as the grouping index [5], the group capability index [6] or the doubly weighted grouping efficiency [7].

#### 2.2. Definition of the TFPGT

The definition of the TFPGT tackled in this paper can be stated based on the MPCF. The TFPGT is also a problem of matrix rearrangement, but in this case the matrix represents the knowledge that a given staff member has on the different resources available in the institution. This means that the TFPGT starts from a knowledge (skill) matrix $K$, in which each component $k_{ij}$ stands for the knowledge that the $i$-th staff member, $i=1,\ldots,E$ (E staff members) has on the $j$-th resource, $j=1,\ldots,R$ (R resources). Note that, in this case, matrix $K$ is not a binary matrix, but a matrix of integer numbers, each encoding a level of knowledge (or skill) about a given resource. The objective of the TFPGT is to obtain the

---

1 Here the word resource is used with a general meaning: it can refer for example to a given subject in a University degree, or to the management of a certain type of machinery in a manufacturing company, etc.
diagonalization of matrix $K'$, in order to form several teams of staff members and the corresponding resources, which are optimum in terms of a given measure. An example to clarify the TFPGT is shown in Figs. 3 and 4. Fig. 3 shows an example of knowledge matrix $K$ in a problem with 20 staff members and 15 resources. Fig. 4 shows the final matrix $K'$ obtained after the diagonalization process, in which four teams are defined.

The quality measures defined before for the MPCF problem find difficulties when applied to non-binary matrices. For example, all the elements not 0 out of the diagonal blocks would be treated in a similar way, though they can represent different (integer) numbers. Therefore, a different measure must be defined to tackle the TFPGT

$$F(K) = \frac{\sum_{g=1}^{C} \sum_{r=1}^{R} \sum_{e=1}^{E} X_{rg} Y_{eg} k_{er}}{\sum_{g=1}^{C} \sum_{r=1}^{R} \sum_{e=1}^{E} Y_{eg}}$$

(3)

where $X_{rg}=1$ if resource $r$, $r=1,...,R$, is assigned to the team $g$, $g=1,...,C$, and $X_{rg}=0$ otherwise. $Y_{eg}=1$ if staff member $e$, $e=1,...,E$, is assigned to team $g$, and 0 otherwise.

Note that this objective function is defined as the mean knowledge that the members of a team have about the resources assigned to that team. The maximization of this function leads to a good solution for the TFPGT, but may be incomplete. In order to improve this solution, two hard constraints must be applied to the definition of the TFPGT: first, it is also necessary to ensure that each staff member has a minimum knowledge of the resources in a team:

$$\sum_{g=1}^{C} \sum_{r=1}^{R} X_{rg} Y_{eg} k_{er} \geq s_e, \ \forall e$$

(4)

where $s_e$ is defined as the minimum knowledge that staff member $e$ must have about the resources of a team. Note that this
constraint avoids the assignment of staff members to groups in which they are not productive.

The second constraint to improve the quality of the solutions in the TFPGT is related to the total knowledge in a team about a given resource, which must be larger than a minimum value \( t_r \):

\[
\sum_{g=1}^{G} \sum_{r=1}^{R} X_{rg} Y_{rg} k_{er} \geq t_r, \quad \forall r
\]

(5)

The final statement of the TFPGT is then as follows:

Given an \( E \times R \) knowledge (skill) matrix \( K_e \), a vector \( S \) of minimum knowledge of each staff member, and a vector \( T \) of minimum total knowledge in a team, obtaining a diagonalization \( K' \), such as

\[
\max \left( f(K) = \frac{\sum_{g=1}^{G} \sum_{r=1}^{R} X_{rg} Y_{rg} k_{er}}{\sum_{g=1}^{G} \sum_{r=1}^{R} \sum_{r' \neq r} X_{rg} Y_{rg}} \right)
\]

subject to:

\[
\sum_{g=1}^{G} \sum_{r=1}^{R} X_{rg} Y_{rg} k_{er} \geq t_r, \quad \forall r
\]

(7)

\[
\sum_{g=1}^{G} \sum_{r=1}^{R} X_{rg} Y_{rg} k_{er} \geq s_e, \quad \forall e
\]

(8)

3. A hybrid grouping genetic algorithm with parallelism to solve the TFPGT

The grouping genetic algorithm (GGA) is a class of evolutionary algorithm especially modified to tackle grouping problems, i.e., problems in which a number of items must be assigned to a set of predefined groups (teams, in this case). It was first proposed by Falkenauer [26], who realized that traditional genetic algorithms had difficulties when they were applied to grouping problems (mainly, the standard binary encoding increases the space search size in this kind of problem). Thus, in the GGA, the encoding, crossover and mutation operators of traditional GAs are modified in order to obtain a compact algorithm with very good performance in grouping problems. In this paper, we also include parallelism in the algorithm, in order to improve its performance. This is done by using the well known island model in which several populations are maintained, and individuals are migrated from one population to another on a regular basis.

In the next subsections, we show the main characteristics of the hybrid GGA that we propose. Special attention will be paid to the encoding, genetic operators, and implementation of a local search to improve the quality of solutions, and finally, the island model is considered.

3.1. Problem encoding

The GGA initially proposed by Falkenauer is a variable-length genetic algorithm. The encoding is carried out by separating each individual in the algorithm into two parts: the first is an assignment part that the item (and also the group it is assigned to) is encoded. The second is a group part that defines which groups must be taken into account for this individual. In problems where the number of groups is not previously defined, it is easy to see why this is a variable-length algorithm: the group part varies from one individual to another. In the specific case of the TFPGT, the assignment part is also divided into two subparts: one representing the assignment of staff members to a group and the other representing the resources assigned to that group, i.e., an individual \( e \) has the form \( e = [1, e_{rg}, e_{rg}] \). An example of an individual

<table>
<thead>
<tr>
<th>T</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>SM</td>
<td>1,2,5,7,12,18</td>
<td>3,13,16,20</td>
<td>4,9,17,19</td>
<td>6,8,10,11,14,15</td>
</tr>
<tr>
<td>R</td>
<td>1,4,12</td>
<td>2,6,9,10,15</td>
<td>3,7</td>
<td>5,8,11,13,14</td>
</tr>
</tbody>
</table>

Also, specific teams (T), Staff members (SM) and Resources (R) of the father individual for the example in Section 3.3.

in the proposed GGA for the TFPGT, corresponding to the encoding of the grouping given in Fig. 4 and Table 1 (20 staff members, 15 resources, and 4 groups), is the following:

\[
|1|2|3|4|1|4|3|4|1|2|4|2|3|1|3|2|
|1|2|3|4|2|3|4|2|2|4|2|4|2|1|2|3|4|

3.2. Selection operator

In this paper we use a rank-based wheel selection mechanism, similar to the one described in [17]. First, the individuals are sorted in a list based on their quality (given by Eq. (6)). The position of the individuals in the list is called rank of the individual, and denoted \( R_{i} \), \( i = 1, \ldots, i' \), with \( i' \) number of individuals in the population of the GGA. We consider a rank in which the best individual \( x \) is assigned \( R_{x} = \zeta \), the second best \( y \), \( R_{y} = \zeta - 1 \), and so on. A fitness value associated to each individual is then defined, as follows:

\[
f_{i} = \frac{2 \cdot R_{i}}{\zeta \cdot (\zeta + 1)}
\]

(9)

Note that these values are normalized between 0 and 1, depending on the position of the individual in the ranking list. It is important to note that this rank-based selection mechanism is static, in the sense that probabilities of survival (given by \( f_{i} \)) do not depend on the generation, but on the position of the individual in the list. As a small example, consider a population formed by five individuals, in which individual 1 is the best quality one \((R_{1} = 5)\), individual 2 the second best \((R_{2} = 4)\), and so on. In this case, the fitness associated to the individuals are \([0.33, 0.26, 0.2, 0.13, 0.06]\), and the associated intervals for the roulette wheel are \([0-0.33, 0.33-0.6, 0.06-0.8, 0.81-0.93, 0.94-1]\).

The process carried out in our algorithm consists of selecting the parents for crossover using this selection mechanism. This process is performed with replacement, i.e., a given individual can be selected several times as one of the parents, however, individuals in the crossover operator must be different.

3.3. Crossover operator

The crossover operator implemented in the grouping genetic algorithm used in this paper is a modified version of the initially proposed by Falkenauer [26]. The process to apply this operator has the following steps:

1. First, two individuals are selected, one will be named father and the other mother. Only one individual offspring will be generated from these two individuals.
2. Equalize the offspring to the mother.
3. Select two crossing points in the group’s part of the father.
4. Insert the elements of the father belonging to the selected groups into the offspring, changing the numbers of groups in the offspring to avoid duplicity with the existing ones.
5. Modify the assignment part of the offspring (the staff members and resources coming from the mother) in such a way that duplicity assignments are avoided.
6. Change the role of the father and mother individuals and go to step 2, in order to generate a second offspring.
7. Replace the father and mother individuals in the population by the two offspring generated in this crossover process.

In order to illustrate this procedure, consider the following two individuals. First the father individual (example given by Fig. 4 and Table 1):

\[
\begin{align*}
&1 \ 2 \ 3 \ 4 \ 5 \\
&1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7 \\
&1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7 \\
&1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7 \\
&\vdots \\
&1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7
\end{align*}
\]

and the mother individual:

\[
\begin{align*}
&1 \ 2 \ 3 \ 4 \ 5 \\
&1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7 \\
&1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7 \\
&1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7 \\
&\vdots \\
&1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7
\end{align*}
\]

Tables 1 and 2 show the corresponding assignment of staff members and resources for the father and the mother individuals considered in this example, respectively. Let us consider that the crossover points in the father are point 2 and point 3, which coincide with the selected groups in this case. Therefore, there are two groups selected, groups 2 and 3. Since the mother individual has five groups, the offspring will receive seven groups (five from the mother and two from the father). The elements of the father are inserted in the mother, in this case groups 2 and 3 of the father will receive numbers 6 and 7 in the offspring. The next step is to modify the assignment part of the offspring to avoid duplicities. In Table 3, duplicate assignments have been marked with a line under the number. These assignments are eliminated to produce the final offspring given in Table 4 and also in the encoding form as follows:

\[
\begin{align*}
&1 \ 2 \ 3 \ 4 \ 5 \\
&1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7 \\
&1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7 \\
&1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7 \\
&\vdots \\
&1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7
\end{align*}
\]

3.4. Mutation operator

The mutation operator used in the proposed GGA algorithm is a two-step operator. In the first step, the assignment of a member of the staff to a given group is randomly modified, with a probability \( P_m \). As an example, consider the following case:

\[
\begin{align*}
&1 \ 2 \ 3 \ 4 \ 5 \\
&1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7 \\
&1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7 \\
&1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7 \\
&\vdots \\
&1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7
\end{align*}
\]

where the first staff member is reassigned from group 1 to group 6. In a second step, the operator eliminates, with a small probability, an existing group, that is randomly chosen. For example, consider that in the following individual, we have randomly chosen the group 4 to be eliminated. We mark the group in boldface, whereas the staff members and resources allocated in group 4 are marked in italic.

\[
\begin{align*}
&1 \ 2 \ 3 \ 4 \ 5 \\
&1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7 \\
&1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7 \\
&1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7 \\
&\vdots \\
&1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7
\end{align*}
\]

Then, the group is eliminated, the staff members and resources are randomly reallocated in other groups and the enumeration of groups is modified (group 5 is now 4, 6 is now 5 and so on):

\[
\begin{align*}
&1 \ 2 \ 3 \ 4 \ 5 \\
&1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7 \\
&1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7 \\
&1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7 \\
&\vdots \\
&1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7
\end{align*}
\]

We have implemented an adaptive version of the probability of mutation, that is smaller in the first generations of the algorithm than in the last ones:

\[
P_{m}(k) = P_{m0} + \frac{k}{T_G}(P_{mf} - P_{m0})
\]

where \( P_{m0}(k) \) is the probability of mutation used in a given generation \( k \), \( T_G \) stands for the total number of generations of the algorithm, and \( P_{m0} \) and \( P_{mf} \) are the final and initial values of probability considered, respectively.

3.5. Replacement and elitism

In the proposed GGA, the population at a given generation \( k+1 \) is obtained by replacement of the individuals in the population at generation \( k \), through the application of the selection, crossover, and mutation operators described above. When an elitist schema is also applied, the best individual in generation \( k \) is automatically passed to the population of generation \( k+1 \), ensuring that the best solution encountered so far in the evolution is always kept by the algorithm.

3.6. Repairing of individuals

Once the new offspring population is generated, a repairing procedure is applied to each individual of the new population. This step is necessary because the crossover and mutation

Table 2

<table>
<thead>
<tr>
<th>Mother individual</th>
<th>T1</th>
<th>T2</th>
<th>T3</th>
<th>T4</th>
<th>T5</th>
</tr>
</thead>
<tbody>
<tr>
<td>SM</td>
<td>1,5</td>
<td>10,11,15</td>
<td>6,12,13,16,19</td>
<td>9,20</td>
<td>3,4,7,8,17,18,21,4</td>
</tr>
<tr>
<td>R</td>
<td>1,7</td>
<td>13</td>
<td>4,8,9,15</td>
<td>2,6,10,11</td>
<td>3,14</td>
</tr>
</tbody>
</table>

Table 3

| Middle step of the crossover procedure described in Section 3.3. |
|--------------------------|-----|-----|-----|-----|-----|
| T1  | T2  | T3  | T4  | T5  | T6  | T7  |
| SM  | 1,5,10,11,15 | 6,12,13,16,19 | 9,20 | 3,4,7,8,17,18,21,4 |
| R   | 1,7,13   | 4,8,9,15      | 2,6,10,11 | 3,14 | 5,12 |

T, SM and R stand for Teams, Staff members and Resources, respectively.
operators may generate infeasible solutions due to the following reasons:

- The individual contains groups without staff members or resources in them.
- There exists one or more staff members in a group whose knowledge of all of the resources in the group is less than the predefined minimum $s_r$.
- There exists one or more groups such that its/their total knowledge about a given resource $r$ is less than $t_r$.

In order to solve these points and obtain feasible individuals, the following repairing process is applied to the population:

1. Locate groups without staff members nor resources assigned, and eliminate them.
2. Locate groups without staff members, but with resources assigned. The group is eliminated, and its resources randomly reassigned to the other groups.
3. Locate groups without resources but with staff members. The group is eliminated, and its resources randomly reallocated into other groups.
4. Locate staff members whose knowledge of a given resource $r$ is less than the defined minimum $s_r$. In this case, if there are groups such that this condition is fulfilled, the staff member is reallocated into one of these groups randomly. In case there is not a group in which the staff member’s knowledge about all of the resources in that group is larger than $s_r$, the current group of the staff member is merged with another group, until the condition in $s_r$ is satisfied.
5. Locate resources where the total knowledge of a team for this specific resource is less than the defined minimum $t_r$. If there are groups that can allocate $r$, reassign it randomly. In case there is not a group in which the total knowledge (for resource $r$) of staff members is larger than $t_r$, the current group of the resource is merged with another group, until the condition in $t_r$ is fulfilled.

Let us consider an example, using the final individual obtained after the mutation operator (Section 3.4):

\[
\begin{align*}
&64561212611254156265 \\
&156245625534125123456
\end{align*}
\]

Let us consider, in addition two vectors, containing the different values of $s_r$ and $t_r$ (recall that this is an example with 20 staff members and 15 resources):

\[
\begin{align*}
S = &[2111131212221211102] \\
T = &[5793871016116434]
\end{align*}
\]

In this example we will only show reassignments due to vector $T$ (the process for considering constraints of vector $S$ is similar). In a first step, group 3 must be eliminated since it does not contain staff members. Resource 11, which was assigned to this group, is reassigned to group 1, and the enumeration of the groups is restructured, resulting the following individual:

\[
\begin{align*}
&53451212511243145254 \\
&14523452441312412345
\end{align*}
\]

Table 5 summarizes the total knowledge of each group in each resource given by this individual, considering the knowledge matrix $K$ given in Fig. 3. Note that resources 5 and 12 do not fulfill the constraint given by Eq. (8). Let us suppose that we begin the reassignment in resource 12, first we will reassign this resource to group 2, which has enough knowledge. Regarding resource 5, there is no group with knowledge about this resource, so it is necessary to merge group 3 with another group. If we randomly choose group 1 for merging, the final feasible individual results:

\[
\begin{align*}
&42341212411231134243 \\
&|13421342331212331234
\end{align*}
\]

### Table 5: Example of the repairing procedure given in Section 3.6.

<table>
<thead>
<tr>
<th>Resource</th>
<th>Staff members</th>
<th>Group knowledge</th>
<th>$t_r$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5,7,10,11,15</td>
<td>6</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>3,13,16,20</td>
<td>8</td>
<td>7</td>
</tr>
<tr>
<td>3</td>
<td>1,4,9,17,19</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>4</td>
<td>6,8,12,18</td>
<td>9</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>2,14</td>
<td>5</td>
<td>8</td>
</tr>
<tr>
<td>6</td>
<td>3,13,16,20</td>
<td>9</td>
<td>7</td>
</tr>
<tr>
<td>7</td>
<td>1,4,9,17,19</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>8</td>
<td>6,8,12,18</td>
<td>9</td>
<td>1</td>
</tr>
<tr>
<td>9</td>
<td>3,13,16,20</td>
<td>9</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>3,13,16,20</td>
<td>9</td>
<td>6</td>
</tr>
<tr>
<td>11</td>
<td>5,7,10,11,15</td>
<td>7</td>
<td>11</td>
</tr>
<tr>
<td>12</td>
<td>2,14</td>
<td>9</td>
<td>10</td>
</tr>
<tr>
<td>13</td>
<td>5,7,10,11,15</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>14</td>
<td>6,8,12,18</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>15</td>
<td>1,4,9,17,19</td>
<td>11</td>
<td>4</td>
</tr>
</tbody>
</table>

The underlined values stand for values that do not fulfill the constraint given by Eq. (8), and must be repaired.

### 3.7. Local search

We use a local search procedure to try to find local optima in a close neighborhood of an individual. The local search proposed is based on small modifications of the current individual because they produce an increase of the associated objective function. The local search procedure is applied to every individual in the population and produces, as we will show, a Lamarckian hybrid algorithm, i.e., the chromosome is modified together with its associated objective function. The local search procedure used can be described as follows:

1. For each staff member chosen, the first one randomly reassigns it to the group, providing the maximum fitness function and keeping the individual feasible.
2. For each resource, chosen the first one randomly, reassign it to the group, providing the maximum fitness function and keeping the individual.
3. Go to 1 if there are changes in the individual.

Another example may help to better explain the local search process proposed. Let us consider the feasible solution obtained after the repairing process in the previous section. Its fitness value, calculated by Eq. (6) and the initial matrix $K$ given in Fig. 3, is 2.3378. The local search process implemented is described in Table 6 (one iteration of the local search, only for the reassignment of staff members). The underlined values represent the maximum fitness values obtained up to a point in the search. The boldface values stand for changes carried out in the individual because a better value of fitness has been found. Finally, fitness values between parentheses stand for infeasible individuals. In this example, the local search is started in staff member number 6. Note that there are three changes of staff members from one group to the next one that provide improvements in the fitness values of the individual.
The underlined values represent the maximum fitness values obtained up to a point in the search. The boldface values stand for changes carried out in the individual due to a better value of fitness has been found and finally values between parenthesis stand for infeasible individuals.

Specifically, after this local search process, the following final individual is obtained, with a value of fitness 2.4865:

<table>
<thead>
<tr>
<th>Staff members</th>
<th>Team 1</th>
<th>Team 2</th>
<th>Team 3</th>
<th>Team 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>(2.3784)</td>
<td>2.3378</td>
<td>(2.2000)</td>
<td>(2.3333)</td>
</tr>
<tr>
<td>7</td>
<td>2.3378</td>
<td><strong>2.3649</strong></td>
<td>(2.2667)</td>
<td>(2.3611)</td>
</tr>
<tr>
<td>8</td>
<td><strong>2.4189</strong></td>
<td>2.3649</td>
<td>(2.2603)</td>
<td>(2.3571)</td>
</tr>
<tr>
<td>9</td>
<td>(2.2763)</td>
<td>2.3026</td>
<td>2.3117</td>
<td><strong>2.4189</strong></td>
</tr>
<tr>
<td>10</td>
<td><strong>2.4189</strong></td>
<td>2.2838</td>
<td>(2.2133)</td>
<td>(2.3056)</td>
</tr>
<tr>
<td>11</td>
<td><strong>2.4189</strong></td>
<td>2.4054</td>
<td>(2.2667)</td>
<td>2.3611</td>
</tr>
<tr>
<td>12</td>
<td>(2.3649)</td>
<td><strong>2.4189</strong></td>
<td>(2.2800)</td>
<td>(2.3611)</td>
</tr>
<tr>
<td>13</td>
<td>(2.1781)</td>
<td><strong>2.1781</strong></td>
<td><strong>2.4189</strong></td>
<td>(2.2394)</td>
</tr>
<tr>
<td>14</td>
<td><strong>2.4189</strong></td>
<td>2.3649</td>
<td>(2.2933)</td>
<td><strong>2.3889</strong></td>
</tr>
<tr>
<td>15</td>
<td><strong>2.4189</strong></td>
<td>2.3784</td>
<td>2.3333</td>
<td>(2.3611)</td>
</tr>
<tr>
<td>16</td>
<td>(2.2740)</td>
<td>(2.2740)</td>
<td><strong>2.4189</strong></td>
<td>(2.3380)</td>
</tr>
<tr>
<td>17</td>
<td>(2.2632)</td>
<td>(2.2632)</td>
<td>(2.2338)</td>
<td><strong>2.4189</strong></td>
</tr>
<tr>
<td>18</td>
<td>(2.2973)</td>
<td><strong>2.4189</strong></td>
<td>(2.2667)</td>
<td>(2.3611)</td>
</tr>
<tr>
<td>19</td>
<td>(2.2773)</td>
<td>2.2763</td>
<td>2.2468</td>
<td><strong>2.4189</strong></td>
</tr>
<tr>
<td>20</td>
<td>(2.2603)</td>
<td>(2.2466)</td>
<td><strong>2.4189</strong></td>
<td>(2.3099)</td>
</tr>
<tr>
<td>1</td>
<td>2.3553</td>
<td>2.4079</td>
<td>(2.2727)</td>
<td><strong>2.4189</strong></td>
</tr>
<tr>
<td>2</td>
<td><strong>2.4189</strong></td>
<td>(2.4595)</td>
<td>(2.3200)</td>
<td>(2.4267)</td>
</tr>
<tr>
<td>3</td>
<td>(2.3014)</td>
<td><strong>2.3014</strong></td>
<td><strong>2.4189</strong></td>
<td>(2.3362)</td>
</tr>
<tr>
<td>4</td>
<td>2.2368</td>
<td>2.2368</td>
<td>(2.1948)</td>
<td><strong>2.4189</strong></td>
</tr>
<tr>
<td>5</td>
<td>2.2368</td>
<td><strong>2.4865</strong></td>
<td>(2.3467)</td>
<td>(2.4444)</td>
</tr>
</tbody>
</table>

4. Experiments and results

In order to show the performance of the proposed approach, we consider two types of experiments. First, we tackle a set of randomly generated TFPGT instances, evaluating the performance of the proposed HGGA with and without local search and island model. Second, we present a real application of the TFPGT in the teaching group formation at the Department of Signal Theory and Communications at Universidad de Alcalá, Madrid, Spain. For comparison purposes, we use an adaptation of a greedy two-phases heuristic, which will be described as follows.

### 3.8. An island model to improve the algorithm’s performance

In order to improve the performance of the proposed GGA, an island model is considered [32] to parallelize it. N sub-populations (islands) are taken into account because the evolution in each island is independent, but the migration of good individuals between islands, after a number of generations, is allowed. An example of this process is outlined in Fig. 5 (example with four islands). In fact, we consider an elitist migration model, in which only the best individual in each island migrates. The migration process is as follows:

1. Choose the best individual in each island.
2. Randomly choose the island toward each individual will migrate.
3. Randomly choose an individual in the destiny island and change it by the migrating individual.

### 4.1. A greedy two-phase heuristic for comparison purposes

Since the model described in this paper is novel, there are not specific algorithms described in the literature for comparison purposes. However, an existing algorithm can be adapted to solve the TFPGT, based on a two-phase heuristic [33]. In fact, a version of this heuristic is the one previously used in the Department of Signal Theory and Communications (Universidad de Alcalá) to automatically structure the department into different teaching groups. As mentioned, a similar two-phase approach has been used before in different real assignment problems, such as [34] for a timetabling problem (also in a Spanish University).

Usually, in two-phase approaches, one of the phases or steps is devoted to obtaining a basic solution, and the other one is used to improve its quality or eliminate infeasibilities. In this case, the TFPGT can be solved by using a two-phase heuristic where the first step is used to compute a set of initial groups (usually infeasible), and the second one comes up with a feasible solution to the problem. The description of the algorithm is the following:

Let us suppose that the number of staff members is larger than the number of resources (on the contrary, the heuristic works over the resource instead of staff members, and follows a process completely equivalent to the one following described). In a first phase, each staff member is assigned to the resource for which he/she has the most knowledge. Thus, we obtain a first arranging of staff members in groups (those staff members who share a maximum knowledge in a given resource belong to the same group). In addition, note that it is possible that not all resources are assigned in this first step, so we consider the assignment of the remaining resources to the group with maximum average knowledge about a given resource. Note that, after this final assignment, we obtain a first solution that, in general, will not fulfill the constraints of the TFPGT as defined by Eqs. (8) and (7).

The second phase of the algorithm modifies this initial solution to obtain a feasible one. For this, we consider the merging of pairs of groups in terms of a “gaining function”. For each pair of groups, the gaining function is calculated as follows: for each constraint fulfilled when the two groups are merged, 1 divided by the number of resources or staff members needed to fulfill that constraint is added. If a given constraint is not fulfilled, 0 is added to the gaining function. The pair of groups with the largest gaining function are then merged, and the process is carried out, recalculating the gaining function in each iteration, until all the problem’s constraints are fulfilled. Note that we will always have a solution for this process, since the complete matrix K (one group, without any diagonalization) must be a solution to the problem.
heuristic algorithm.

Table 7
Percentage of knowledge (% of 0s, 1s, 2s, 3s, 4s and 5s in the teams formed (matrix $K$)) obtained in the final solution for each TFPGT instance, using the proposed HGGA without local search and with island model.

<table>
<thead>
<tr>
<th>Instance</th>
<th>Fitness</th>
<th>% of 0s</th>
<th>% of 1s</th>
<th>% of 2s</th>
<th>% of 3s</th>
<th>% of 4s</th>
<th>% of 5s</th>
</tr>
</thead>
<tbody>
<tr>
<td>60 × 40</td>
<td>2.0155</td>
<td>6.61</td>
<td>9.09</td>
<td>13.64</td>
<td>40.00</td>
<td>47.37</td>
<td>60.00</td>
</tr>
<tr>
<td>80 × 60</td>
<td>1.3371</td>
<td>11.29</td>
<td>20.00</td>
<td>28.85</td>
<td>30.77</td>
<td>37.68</td>
<td>54.68</td>
</tr>
<tr>
<td>100 × 80</td>
<td>1.3163</td>
<td>10.67</td>
<td>13.00</td>
<td>24.10</td>
<td>78.12</td>
<td>40.83</td>
<td>50.11</td>
</tr>
<tr>
<td>130 × 100</td>
<td>1.1109</td>
<td>10.50</td>
<td>26.56</td>
<td>28.86</td>
<td>35.67</td>
<td>48.39</td>
<td></td>
</tr>
</tbody>
</table>

Table 8
Percentage of knowledge (% of 0s, 1s, 2s, 3s, 4s and 5s in the teams formed (matrix $K$)) obtained in the final solution for each TFPGT instance, using the proposed HGGA with local search and without island model (but maintaining the same population size).

<table>
<thead>
<tr>
<th>Instance</th>
<th>Fitness</th>
<th>% of 0s</th>
<th>% of 1s</th>
<th>% of 2s</th>
<th>% of 3s</th>
<th>% of 4s</th>
<th>% of 5s</th>
</tr>
</thead>
<tbody>
<tr>
<td>60 × 40</td>
<td>2.1845</td>
<td>5.50</td>
<td>12.12</td>
<td>9.09</td>
<td>52.00</td>
<td>47.37</td>
<td>57.78</td>
</tr>
<tr>
<td>80 × 60</td>
<td>1.4189</td>
<td>10.64</td>
<td>23.33</td>
<td>17.31</td>
<td>30.77</td>
<td>37.68</td>
<td>59.18</td>
</tr>
<tr>
<td>100 × 80</td>
<td>1.3697</td>
<td>10.31</td>
<td>15.00</td>
<td>18.07</td>
<td>56.25</td>
<td>39.17</td>
<td>52.80</td>
</tr>
<tr>
<td>130 × 100</td>
<td>1.1483</td>
<td>10.45</td>
<td>27.34</td>
<td>17.05</td>
<td>31.90</td>
<td>43.43</td>
<td>49.24</td>
</tr>
</tbody>
</table>

Table 9
Percentage of knowledge (% of 0s, 1s, 2s, 3s, 4s and 5s in the teams formed (matrix $K$)) obtained in the final solution for each TFPGT instance, using the proposed HGGA with local search and with island model.

<table>
<thead>
<tr>
<th>Instance</th>
<th>Fitness</th>
<th>% of 0s</th>
<th>% of 1s</th>
<th>% of 2s</th>
<th>% of 3s</th>
<th>% of 4s</th>
<th>% of 5s</th>
</tr>
</thead>
<tbody>
<tr>
<td>60 × 40</td>
<td>2.4067</td>
<td>4.42</td>
<td>15.15</td>
<td>22.73</td>
<td>40.00</td>
<td>31.58</td>
<td>60.74</td>
</tr>
<tr>
<td>80 × 60</td>
<td>1.5907</td>
<td>8.65</td>
<td>15.00</td>
<td>11.54</td>
<td>33.33</td>
<td>42.03</td>
<td>55.81</td>
</tr>
<tr>
<td>100 × 80</td>
<td>1.5338</td>
<td>8.28</td>
<td>19.00</td>
<td>15.66</td>
<td>71.88</td>
<td>42.50</td>
<td>48.77</td>
</tr>
<tr>
<td>130 × 100</td>
<td>1.2264</td>
<td>9.40</td>
<td>32.03</td>
<td>14.73</td>
<td>31.03</td>
<td>40.00</td>
<td>49.92</td>
</tr>
</tbody>
</table>

4.2. Experiments in randomly generated TFPGT instances

In order to test the performance of our approach, we have tackled several artificially generated TFPGT instances, with different numbers of staff members and resources available. The hybrid GGA proposed in this paper has been tested in these instances with and without local search and with and without the island model. Specifically, four TFPGT instances of size (staff members × resources) 60 × 40, 80 × 60, 100 × 80 and 130 × 100 were randomly generated. In order to generate the different knowledge matrices $K$, the following probabilities for the different knowledge levels (0–5) are considered: [0.9 0.015 0.01 0.01 0.015 0.05]. In order to generate vectors $S$ and $T$, it is considered that, in the first instance, at least 1/3 of the total knowledge about a subject must be included in each group (vector $T$), and at least 1/3 of the knowledge of a given lecturer must be used (vector $S$). This factor is changed to 1/3.5 for the second instance, 1/4 for the third instance, and 1/4.5 for the last (the largest) instance considered.

The results obtained in these randomly generated instances are summarized in Tables 7–9. Tables 7 and 8 show the performance of the proposed HGGA with local search but without an island model, and without local search but with the island model, respectively. It is easy to see that the HGGA with local search obtains better results in terms of the objective function in all of the instances than the HGGA without local search but with an island model. The effect of the island model is important when it is added to the local search, as can be seen in Table 9, where the results of the complete algorithm (with local search and the island model) are shown. Note that these results improve the results of the algorithm working only with local search or island model on its own. These tables also show the percentage of knowledge values in the teams formed, with respect to the total knowledge values of the initial matrix $K$. Note that in all cases, the percentage of the minimum level of knowledge in the teams formed (0) is quite reduced (about 2%), whereas the maximum level of knowledge (5) in the teams is over 50% of the total of 5s in the matrix $K$. This means that the algorithm is able to obtain good quality teams out of the initial matrices $K$.

In order to compare the results obtained by the different implementations of the HGGA, we have run the greedy two-phase heuristic described above in these synthetic TFPGT instances. The results are shown in Table 10, with the same structure as previous tables constructed using the results of the HGGA. It is possible to see that the performance of the HGGA is much better than that of the two-phase algorithm even in its simpler implementation (without local search and without island model). Regarding the computational time of the algorithms, the two-phase heuristic is
a very fast, greedy algorithm. It is able to compute the solutions for each of the synthetic instances tackled in less than 2 s. The HGGA computation time is higher than the two-phase heuristic, but it depends on the population size and the number of generations to stop. In one of the tests carried out with 50 individuals in the population and 1000 generations in the HGGA, the computational time varied from approximately 20 s in the smallest instance (60 \times 40) to approximately 100 s in the largest instance. This is still a quite reasonable time for a metaheuristic approach. In addition, note that the computation time in this problem is not a key point, since the optimal assignment is calculated once, and no real-time or on-line calculation is needed.

A graphic example of the best solution found by the proposed HGGA in the synthetic TFPGT instance of size 100 \times 80 is given by Figs. 6 and 8. These figures display the initial matrix $K$ and the final matrix $K'$ obtained for this instance, respectively. The solution found by the two-phase heuristic used for comparison is displayed in Fig. 7. Note that in this case, the two-phase heuristic provides a final matrix $K'$ divided into three groups, whereas the proposed HGGA introduces seven groups in the diagonalization matrix $K'$. On the other hand, Fig. 9 shows the evolution of the percentage of

---

**Fig. 7.** Best solution found for the synthetic 100 \times 80 instance using the greedy two-phases heuristic.

**Fig. 8.** Best solution found for the synthetic 100 \times 80 instance using the proposed hybrid GGA.

**Fig. 9.** Evolution of the percentage of 0s, 1s, 2s, 3s, 4s and 5s values within the formed groups (hybrid GGA) in the synthetic 100 \times 80 instance.
0s, 1s, 2s, 3s, 4s and 5s values within the formed groups, also for the 100 × 80 instance. It is easy to see that the proposed HGGA evolves towards solutions with a low percentage of 0s (no knowledge) in the groups. This point is important in real cases because it makes the teams more effective. This behavior will be also found in the real application, in the next section.

4.3. A real application in a Spanish University

In order to test the performance of the proposed algorithm in a real case, the formation of teaching groups in the Department of Signal Theory and Communications at Universidad de Alcalá, Madrid, Spain, has been tackled. In this case, the staff members are the lecturers in the department, and the resources of the TFPGT are the courses taught in the department; in addition, the teams are called problem groups in this case. Specifically, there are 61 lecturers and 65 courses given by the department. The objective is to organize the department in teaching groups (including lecturers and courses), allowing the lecturers in the same group to have the maximum possible knowledge of the courses in that group.

Each lecturer in the department has some knowledge about the different courses, so a knowledge matrix $K$ can be constructed.
as is shown in Fig. 10. The “knowledge” in this case is related to the percentage of a subject that a lecturer is able to teach without investing months in preparation. If the whole subject can be completely taught by a lecturer, then a knowledge of 5 is given. The value of knowledge 0 means that the lecturer is not specialized in that course, and a lot of work would be necessary for him/her to teach the course. Note that in this real case the structure of the matrix \( K \) is quite different from the one obtained in the randomly generated case. This is due to the specialization of lecturers in similar subjects. Once the matrix \( K \) has been defined, vectors \( T \) and \( S \) are calculated, similarly to the synthetic case, i.e., at least a fourth of the total knowledge about a course must be included in each group (vector \( T \)) and at least a fourth part of the knowledge of a given lecturer must be used (vector \( S \)).

The proposed HGGA has been applied to this specific problem, and its performance compared with that of the greedy two-phase approach described in Section 4.1. The results obtained by the two-phase heuristic are shown in Fig. 11. On the other hand, Fig. 12 and Table 11 show the results obtained by the proposed HGGA. Specifically, it is possible to compare Figs. 11 and 12, which show the obtained matrix \( K \) using the two-phase heuristic and the HGGA, respectively. Note that the two-phase heuristic tends to form larger groups than the HGGA (all of the lecturers are divided into 10 groups using the greedy two-step heuristic and in 15 groups if we consider the HGGA). Note, however, that the division into groups generated by the proposed HGGA algorithm better uses the available knowledge, so it obtains a better value in terms of objective function than the two-phase algorithm (2.2730 with the HGGA and 1.0916 with the two-phase heuristic).

Table 11 shows a summary of the solution displayed in Fig. 12.

### Table 11
Specific composition of the groups formed in the real application at Universidad de Alcalá using the proposed HGGA.

<table>
<thead>
<tr>
<th>Group</th>
<th>Lecturers</th>
<th>Subjects</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>24,28</td>
<td>6,7,19</td>
</tr>
<tr>
<td>2</td>
<td>8,17,37,42</td>
<td>52</td>
</tr>
<tr>
<td>3</td>
<td>12</td>
<td>61</td>
</tr>
<tr>
<td>4</td>
<td>9</td>
<td>55</td>
</tr>
<tr>
<td>5</td>
<td>4,26,52,60</td>
<td>47,63</td>
</tr>
<tr>
<td>6</td>
<td>19,39,56</td>
<td>18,23,30,31,35,45,65</td>
</tr>
<tr>
<td>7</td>
<td>6,27,29,35,41,44,45</td>
<td>26,36,37,39,59,60</td>
</tr>
<tr>
<td>8</td>
<td>20,21,23,40,49,50,51,53</td>
<td>10,14,17,27,28,40,50</td>
</tr>
<tr>
<td>9</td>
<td>1,13,22,33,46,47,59</td>
<td>1,2,3,4,11,12,24</td>
</tr>
<tr>
<td>10</td>
<td>11,32,36</td>
<td>41,42,46</td>
</tr>
<tr>
<td>11</td>
<td>5,15,16,25,38,61</td>
<td>5,32,33,34,44,48,49,51,54,62</td>
</tr>
<tr>
<td>12</td>
<td>10</td>
<td>25,53</td>
</tr>
<tr>
<td>13</td>
<td>7,14,30,31,34,55</td>
<td>16,34,36,57,58,64</td>
</tr>
<tr>
<td>14</td>
<td>18,43,48,54,57,58</td>
<td>8,9,13,15,22,38</td>
</tr>
<tr>
<td>15</td>
<td>2,3</td>
<td>20,21,29</td>
</tr>
</tbody>
</table>

Fig. 13. Mean and Max. fitness evolution produced by the hybrid GGA algorithm in the real problem at Universidad de Alcalá.

Fig. 14. Evolution of the percentage of 0s, 1s, 2s, 3s, 4s and 5s values within the formed groups (hybrid GGA) in the real problem at Universidad de Alcalá.
obtained with the HGGA, with the numbers assigned to lecturers and subjects.

Figs. 13 and 14 show the evolution of the best solution obtained by the proposed HGGA approach. Fig. 13 shows the evolution of the maximum and mean fitness with generations in the algorithm. As previously mentioned, the final (best) fitness value obtained was 2.2730. Fig. 14 shows the evolution of the percentage of 0s, 1s, 2s, 3s, 4s and 5s values within the formed groups. Note that the percentage of knowledge 5 is larger than any other of the groups. The final percentages obtained are 3.87, 21.74, 30.91, 38.00, 64.58 and 74.59, respectively.

5. Conclusions

In this paper, we have proposed a novel model for the team formation problem based on group technology (TFPGT). Specifically, the Machine-Part Cell Formation (MPCF) problem is established as starting point, and the model is defined as a generalization of the MPCF. The proposed model considers different skills of staff members and sets two hard constraints related to the minimum total knowledge about a resource in a team and to the minimum knowledge that a given staff member must have about the resources of a team. These hard constraints imply that there exist unfeasible solutions to the problem, and guarantee a minimum of quality in the obtained assignment. A measure to evaluate the goodness of a given assignment is also defined in this paper for the TFPGT. In the paper a hybrid-grouping genetic algorithm with parallelization is proposed to solve the problem. The details of the algorithm, such as encoding, genetic operators, repair procedures and a local search application have been fully described in the paper. In order to compare its performance, an alternative algorithm based on a greedy two-phase heuristic was implemented. The performance of both approaches has been tested in several TFPGT synthetic instances and in a real application, the formation of teaching groups at the Department of Signal Theory and Communications, Universidad de Alcalá, Spain.

References