Assigning Students to Groups Using General and Context-Specific Criteria

Roland Hübscher

Abstract—Group projects are an important component in many courses. Instructors can allow students to form their own groups or assign them to increase the effectiveness of the group. Most computational tools supporting the assignment of students to class project groups use some general criterion, for instance, maximizing the diversity of the group members. However, frequently, the instructor needs to consider additional context-specific criteria and preferences which force the instructor to figure out the assignment of the students by hand instead of using a software tool. Difficult and time-consuming, this task can easily result in suboptimal assignments. In this paper, a method is introduced that allows the instructor to combine a general criterion and a flexible set of context-specific preferences to describe the type of groups preferred. The heuristic Tabu Search algorithm finds solutions satisfying most preferences.

Index Terms—Computer uses in education, constrained optimization, group formation, coalition formation.

1 INTRODUCTION

Group projects are an important component in many courses, from grade school to university, in classrooms, and when using computer-supported collaborative learning (CSCL) systems. Students collaborating on solving problems within and across groups have many advantages [1]. Compared to large-class settings, students in groups more often explain issues to each other and debate them [2]. The interaction is often cooperative instead of competitive, at least within a group, creating the opportunities to acquire teamwork and collaboration skills and develop interpersonal skills [3]. While most commercial learning management systems (LMSs) provide support for manual group formation only [4], several approaches have been developed in the CSCL community to support automatic group formation [5], [6]. Since students using CSCL systems often work at their own pace as a member of a large learner community, automatic grouping is especially interesting. Some researchers have even automated when learners should transition from an individual learning mode to a group mode [7].

Forming such groups is neither pedagogically nor computationally easy. First, what kinds of students should be in the same group to maximize the group project’s learning effectiveness? Second, once a set of criteria has been decided on, how is an optimal group assignment found?

Mahenthiran and Rouse [8] suggest that the power to assign students to groups should be shared between instructor and students. However, they also make clear that complete power should not be given to students to assure that groups have an appropriate mix of higher and lower skilled students. Yohanand and Revital [9] found that achievement was highest when both instructor and students were in control of forming groups and lowest when neither was in control. Students seem to prefer assignment to groups based on some objective criteria over random or subjective allocation [10], [11]. Chapman and colleagues also suggest that self-selected groups are probably better than random assignment [10].

Some researchers recommend diverse groups so that students can learn from each other within the groups [12], [13], for instance, using reciprocal teaching [14], [15]. Other research suggests that heterogeneous groups are to be preferred because they may result in more creative behavior [16]. Grouping students by ability, especially for reading, results in relatively homogeneous groups that can progress at a similar pace [17]. Groups where each student knew at least one other student, which can be easily achieved by assigning student pairs to groups, were found to do better than randomly assigned groups [8]. Researchers have also found that good students perform better in homogeneous groups, whereas weaker students tend to do better in heterogeneous groups [18], [19].

The research on group assignment is still rather inconclusive. Huxham and Land [20] did not find a clear benefit of random assignment over some scientific assignment, whereas Muller [13] found only a small benefit for groups balanced for some student characteristics over random groups. Some research suggests that the instructor ought to assign the students to groups, because left to themselves, the students tend to build groups with rather unequal skills [21]. However, this seems to apply more to young students and less so for college students. Given all these mixed results, it is rather unclear whether we can refer to a scientific method at all.

Although students can be assigned to groups along quite a few possible dimensions, and although no satisfactory evidence preferring one approach to another exists, computational approaches tend to focus on rather uniform assignment criteria. For instance, some focus on forming maximally diverse groups given multiple criteria for each student [22], [23], whereas others minimize the differences.
between groups [24]. Both approaches tend to result in similar group assignments.

All these approaches seem to assume, at least implicitly, that there is indeed a best way of forming groups. However, what constitutes an effective group depends on the learning goals, student activities, and the teaching method. Groups with high cultural diversity may be useful in certain courses, but not others. The same applies to learning styles, personalities, or physical characteristics. Furthermore, what characteristics an effective group ought to have depend very much on the teaching method used. Are the students encouraged to teach each other, do they have to play specific roles, or are they simply left without any further guidance how to work and learn? Forming effective groups should not be viewed in a pedagogical vacuum. Being well-formed is helpful for a group to succeed, but it is not sufficient.

This paper takes the view that there is no one best optimization criterion for forming groups. What the instructor should be provided with is a toolbox consisting of a variety of criteria that can be selected to assign students to groups.

Optimization criteria need to be considered in the context of the teaching method used, and thus, criteria need to be selected depending on the pedagogical method used. This paper introduces a new criterion, called evenly skilled, for assigning students to a group associated with a teaching method inspired by reciprocal teaching. Similarly, the group forming approach in the CSCL system I-MINDS is normally paired with the Jigsaw method of learning [5]. Furthermore, such general criteria can also be combined, for instance, by preferring homogeneity for some characteristics and heterogeneity for others [25].

General criteria may ignore important factors specific to the learning context. These factors may not be obviously pedagogical in nature, yet may quite strongly influence the expected learning behavior of the groups. It has even been proposed to consider information about learner location, activity, and availability when forming groups [4].

Instructors frequently do not want to rely on a general criterion alone, but would like to use some context-specific preferences to further influence how the groups are composed. Such criteria specific to a learning context can normally not be easily captured by the existing systems’ general assignment criterion. Thus, the presented approach allows for a flexible combination of a general group assignment criterion and criteria specific to the learning context. No matter what criteria are being used, assigning students to groups is a complex problem that cannot be accomplished well without computer support for even modest-sized classes. Classes using CSCL systems are potentially large and the assignment task is even harder.

In this paper, an approach is introduced that provides a means to describe a combination of general and context-specific assignment preferences. These preferences may include such optimizations as maximization of diversity or minimization of differences between groups as mentioned earlier, or making sure that groups have sufficient minimal skills along several dimensions relevant for the project. In addition, they may also include further preferences such as relatively inexperienced students should not be in the smallest groups, certain students should be assigned to the same groups because they tend to work well with each other [8], or some students should not be in the same group because they had negative experiences with each other in groups before. It can easily happen that no solution can satisfy all preferences. Therefore, we consider preferences only and no hard constraints that must absolutely be satisfied. Nevertheless, by weighing preferences differently, we can simulate harder and softer constraints.

All these requirements can be formulated as a constrained optimization problem. Tabu Search [26], a flexible heuristic problem solving method, can be used to solve it. Although the found group assignments may not be always optimal in a purely mathematical sense, they are pedagogically effective, because they also satisfy most learning situation-specific requirements.

Although this paper focuses on the assignment of students to project groups, it can also be used in none-educational settings to assign workers to groups according to a complex set of preferences.

The paper is organized as follows: First, existing computational approaches to the student assignment problem are discussed. Then, the three main contributions are introduced: a new general optimization criterion for groups that is paired with a reciprocal teaching approach, the combination of a general optimization criterion with criteria specific to the learning context, and a Tabu Search algorithm to solve the resulting computational problem. Finally, the results of applying the Tabu Search approach to real classroom data and their implications are discussed.

2 EXISTING APPROACHES

One of the earliest solutions to the group assignment problem was formulated by Dyer and Mulvey [27]. Although they stated that the preferences of faculty and students need to be balanced, they maximized the sum of the preferences of the faculty only. Their optimization function considered the faculty preferences for teaching a certain course subject to the necessary constraints like how many courses somebody teaches and what time slots are available.

The most common approach to assigning students to groups found in the literature is to maximize diversity within groups or minimize differences between groups. These two approaches are “essentially equivalent” [24]. Beheshtian-Ardekani and Mahmood [28] balanced the distribution of student experience by making within-group totals of student background scores as equal as possible. To assign the students, they used a greedy heuristic algorithm which tends to be reasonably efficient but delivers sub-optimal results. Similarly, Weitz and Lakshminarayan [22] created maximally diverse groups by maximizing the differences between all student pairs across all groups. They solved the problem using a heuristic approach based on a solution to scheduling and VLSI design problems. These approaches maximize the sum of group qualities. For the trivial case of two groups, this means that one group may have a high value and the other a low one since the sum of the values is maximized. This can result in homogeneous groups that are balanced by highly diverse groups, thus leading to pedagogically unsound groups. If
one is not careful, pedagogical quality is traded off for mathematical optimality.

Baker and Powell [24] discuss several optimization functions to form maximally diverse groups. As they point out, the assignment problem has the same structure as many other optimization problems found in industry. Although we, therefore, need to take advantage of solutions from those related problems, it is important not to lose track of the actual pedagogical problem of assigning students to groups. Sometimes, the optimization criteria used are quite difficult to justify or understand in pedagogical terms. For instance, DIANA is a system that uses a genetic algorithm “to achieve fairness, equity, flexibility, and easy implementation” [25]. However, how this is achieved is discussed at an implementation level only when describing the specifics of the genetic algorithm. Even though the resulting assignments may be good, it will be difficult to accommodate changes to the requirements and preferences when expressed in pedagogical terms. Preferences always need to be first expressed clearly in pedagogical terms, and then, translated to whatever implementation is used. Modifying the implementation directly without having a pedagogical representation of the changes may lead to pedagogically unclear or even unsound results.

Genetic algorithms have also been used to form heterogeneous, homogeneous, and mixed groups [25]. Heterogeneous groups correspond to maximally diverse groups already discussed, and in homogeneous groups, the students are very similar to each other with respect to the selected characteristics. In a mixed group, the instructor can select which characteristics should be similar and which ones different among the students. Another variation of the diversity-based approach is taken by the squeaky wheel algorithm which finds a grouping of students that attempts to optimize the compatibility of the students with the other students in the same group [29]. The compatibility is based on a rating provided by the students reflecting how much they would like to work with a fellow student; thus, it implements an idea similar to the friend/foe classification.

Agent-based collaborative learning environments have been developed to actively support the learners in collaborating on the Internet [30]. The collaborative learning environment I-MINDS uses an agent-based bidding approach called VALCAM [5] to form groups of learners. The goal of the agents is to assign learners to groups with high expertise and social relationship values.

A variety of student characteristics have been used to assign students to the groups. Huxham and Land [20] used the students’ learning styles, and the groups were formed such that each contained all different learning styles. However, these groups did no better than randomly assigned groups which may be explained by the mixed reputation learning styles have [31]. Personality types similar to Myers-Briggs model [32] were used to assign students to groups without reaching conclusive results [33], [16]. Several approaches use measurements of skill and experience [13], [2] whereas others are open to any characteristics that then can be plugged into their optimization algorithm [22], [24], [23].

**3 Assigning Students with Tabu Search**

It is important that while solving the optimization problem, we do not forget that it is really a student assignment problem we want to solve. The language to describe the preferences needs to be mapped, eventually, to concepts that are meaningful in the domain of assigning students to groups. One needs to avoid sacrificing its naturalness or expressiveness for the sake of an elegant mathematical formulation.

Some of the approaches discussed earlier optimize across all students ignoring the specific groups [27], [28], [22], [24]. For instance, Weitz and Lakshminarayan [22] maximize diversity across all groups by maximizing the sum of all student differences within a group for all groups. Even a mathematically optimal solution may result in some groups being quite homogeneous, if they are balanced out by some highly diverse groups. The approach presented below avoids these problems and adds context-specific criteria.

### 3.1 Tabu Search

Tabu Search is a metaheuristic method that provides a flexible framework for domain-specific approaches to optimization problems [26], [34]. Being a heuristic approach, it does not guarantee finding an optimal solution each time.

The basic idea of Tabu Search is to climb toward a local maximum, and then, to forbid—make tabu—moves similar to the ones that got it to the local maximum, thus making sure that the search is not retracing the same solution again but keeps exploring the space. What differentiates Tabu Search from many of these approaches is its systematic use of memory [35]. It keeps a history of the moves and uses their characteristics to avoid being stuck in a suboptimal solution. Only admissible moves can be chosen, that is, moves that are not tabu or satisfy the aspiration criterion. Therefore, in addition to the typical elements found in other heuristic search methods, Tabu Search also requires the definition of the tabu restrictions and the aspiration criteria. Like most heuristic search methods [36], the representation of a state, moves between states, and an optimization function need also to be specified for the specific problem to be solved.

**BASECTABUSEARCH** shown in Fig. 1 repeatedly selects the locally best move and keeps track of the globally best solution found which is a typical approach of exploratory search algorithms. It finds a goal state S with a high \( f(S) \), that is, it tries to maximize the optimization criterion \( f(S) \).
BASICTABUSEARCH repeatedly generates a subset of the current state \( S \)'s neighborhood \( N(S) \), which is the set of states that can be reached directly from \( S \) with one move.

More specifically, the search starts in an initial state, for instance, a random state or one computed with an efficient greedy heuristic (see step INITIALIZE in Fig. 1). Then, it repeatedly generates admissible moves, applies the best of these moves, and updates the best solution found so far as well as the memory structures to compute a move's tabu status as follows: In step GENERATE, potential moves are checked whether they are tabu and if so, whether they satisfy the aspiration criterion. For instance, the Tabu Search algorithm for the student assignment problem defines a move as swapping any two students between two groups. Making the reasonable assumption that BASICTABUSEARCH tends to make good moves, undoing a move's effect too quickly is prevented with the tabu restriction. For instance, swapped students are forbidden to be moved into another group too soon. This tabu restriction requires that we keep track of the recently moved students in the UPDATE step. Sometimes, the tabu restriction prevents a good move from being made. In such a case, the aspiration criterion can be used to override the tabu restriction. For instance, if a tabu move leads to a new best solution, it should be considered anyway.

Once the set of admissible moves is found, we pick the best move, that is, the move that generates the best neighbor state \( S' \) with a maximal \( f(S') \). It is possible that this state has a lower evaluation than the current state, thus moving away from a possibly local maximum. In the case of several best moves, BASICTABUSEARCH picks one randomly. In step TEST, we keep track of the best move found so far which has two functions. First, it is used to compute the aspiration criterion in step GENERATE, and second, it is the solution we will return at the end of the search.

### 3.2 Assigning Students

The problem-specific elements of the Tabu Search method need to be specified next. A state is defined by the representation of a student assignment as follows: There are \( n \) students and \( m \) groups of size \( k_i \), \( 1 \leq i \leq m \) such that \( k_1 + k_2 + \cdots + k_m = n \). A state \( S \) is the sequence \( (G_1, G_2, \ldots, G_m) \), where the \( G_i \)'s are a partition of the students, that is, the groups containing all the students such that \( G_i \cap G_j = \emptyset \) for all \( i \neq j \).

A move \( i, x \leftrightarrow j, y \) is defined by swapping student \( x \) in group \( i \) with student \( y \) in group \( j \), resulting in \( x \) being in group \( j \) and \( y \) in group \( i \), respectively. The neighborhood \( N(S) \) of a state \( S \) is the set of states that can be generated with one move starting in \( S \).

The initial state is a random assignment of the \( n \) students to the \( m \) groups such that each group has the correct number of students which is set by the instructor. Since only swap moves are possible, the number of students in the groups never changes. A random initial assignment has the advantage that the algorithm can be run multiple times with a different starting point in case a different solution is preferred. In addition, since Tabu Search does not necessarily return the best solution in each run, it can be run a few times and the best solution is returned.

Tabu Search rarely gets stuck on a local optimum because of the tabu restrictions. Thus, the stopping condition is based on when the last time a new best solution was found, that is, when \( S' \) was updated last (step TEST). If the best known solution is not improved for 20 iterations (a value found with a few test runs), the search is terminated and \( S' \) is returned as the final solution.

Three more elements of Tabu Search need to be defined: the evaluation function \( f \), the tabu restriction, and the aspiration criteria. The evaluation function is defined such that it measures the quality of the group assignment. Several different group characteristics are translated into an evaluation function below. But first, attention needs to be given to the tabu restriction and the aspiration criteria without which the search algorithm would be a simple steepest ascent search. The problem with the steepest ascent approach is that it easily gets stuck in suboptimal solutions. Tabu Search addresses this issue with the tabu restrictions and aspiration criteria.

Note that a potential solution \( S' \) is included in the neighborhood only if \( S' \) is not tabu or \( S' \) satisfies an aspiration criterion (step GENERATE). The tabu restriction prevents the search from making or undoing the same moves repeatedly. In our case, this means that it should prevent students from being moved into a group and then right out again. The point is that if it was a good idea to move a student into a group, then it should not be changed right away. This is accomplished by putting students that were moved into a tabu queue. The queue can be kept quite short and a length of five has led to good results for the student assignment problem. Students in the tabu queue are not allowed to be moved. Thus, students participating in a swap move cannot be moved for a few moves. The actual algorithm TABUSEARCHSAP described later uses simple time stamps to simulate the queue.

However, the tabu restrictions might sometimes prevent a good move from being selected. Therefore, an aspiration criterion is used that is satisfied when the value of a potential solution \( S' \) is better than the best solution found so far, that is, when \( f(S') > f(S) \).

The tabu search procedure TABUSEARCHSAP for the student assignment problem is shown in Fig. 2. Given are \( n \) students, the skills and friends/foes preferences, a number \( m \) of groups, and the group sizes \( k_i \) for each group \( G_i \). We assume that the information about the preferences is coded in the optimization function \( f \) as described above, and let \( k = (k_1, \ldots, k_m) \). Then, TABUSEARCHSAP\((n, k, f)\) will find an assignment of the \( n \) students into \( m \) groups of size \( k_1, \ldots, k_m \) trying to maximize \( f \). The tabu queue size \( \delta \) was always set to 5.

One needs to make sure that the tabu queue is properly updated with the swapped students in step UPDATE. Each time, two students are added to the queue and some other students might be pushed off the queue. Another simple way of implementing the queue is by adding a time stamp to each student reflecting when the student was last moved.

We now turn to the evaluation function \( f(S) \) which the TABUSEARCHSAP attempts to maximize. The function \( f(S) \) is a combination of a global criterion \( g(S) \) and a local criterion \( h(S) \), that is, \( f(S) = g(S) + h(S) \). Each student \( i \)

TabuSearchSAP\((n, k, f)\)

1) **INITIALIZE:**
   - let \(G_1, \ldots, G_m\) be a random partition of \(n\) students
   - such that \(|G_i| = k_i\) for all \(i, 1 \leq i \leq m\)
   - \((G_1, G_2, \ldots, G_m) \leftarrow \text{randomPartition}(n)\)
   - initialize current state \(S\), best state \(S^*\), current time \(t_c\),
     and time \(t_e\) when best state was found
   - \(S \leftarrow (G_1, G_2, \ldots, G_m); S^* \leftarrow S\)
   - \(t_c \leftarrow 0; t_e^* \leftarrow t_c\)

2) **GENERATE:**
   - initialize the set of neighbor states
     \(V \leftarrow \emptyset\)
   - find all potential neighbor states
     for all \(i, j\) where \(1 \leq i, j \leq m\) and \(i \neq j\) do
     for all \(x \in G_i\) and \(y \in G_j\) do
       // for all possible moves \(M\)
       \(M \leftarrow (i, x \leftrightarrow j, y)\)
       \(t_{eb} \leftarrow t_{eb} - t_{eb} < \delta\) or \(t_{ec} - t_{ec} < \delta\)
       \(asp \leftarrow f(M(S)) > f(S^*)\)
       if not \(t_{eb}\) or \(asp\) then
         // add state \(M(S)\) to neighbor set \(V\)
       // if \(M\) is not \(t_{eb}\) or the
       // aspiration criterion applies
       \(V \leftarrow V \cup \{M(S)\}\)

3) **SELECT:**
   - let \(S'\) be a best state in \(V\)
   - let \(S'\) \(\in\) \(V\) such that
     \(f(S') \geq f(S'')\) for all \(S'' \in V\)
   - let \(M'\) be the move that generated \(S'\)
   - let \(M'\) \(\text{such that} \ M'(S) = S'\)

4) **TEST:**
   - if \(f(S') > f(S^*)\) then
     // found new best state, keep track of it
     \(S^* \leftarrow S'\)
     // remember when state last improved
     \(t_{ec}^* \leftarrow t_{ec}\)
   
5) **UPDATE:**
   - \(x, y\) are the swapped students
   - let \(x, y\) such that \(M' = i, x \leftrightarrow j, y\)
   - update tabu time for students
     \(t_x \leftarrow t_e; t_y \leftarrow t_e\)
   - update current time and current state
     \(t_c \leftarrow t_c + 1; S \leftarrow S'\)

6) **STOP:**
   - if \(t_c - t_{ec} > 20\) then
     // no further improvement expected
     return \(S^*\)
   - else
     go to step **GENERATE**

Fig. 2. The tabu search algorithm for the student assignment problem.

has \(r\) characteristics \(c_{ij}\), where \(0 \leq c_{ij} \leq 1\) with \(1 \leq i \leq n\)
and \(1 \leq j \leq r\). This function is used to describe the general
assignment criterion like maximizing diversity in groups.

### 3.3 Types of Groups

Several of the approaches discussed earlier attempt to form
maximally diverse groups. First, the optimization function
for this approach is shown. Then, a new optimization
criterion is introduced that addresses the problem of
potentially imbalanced group qualities. In the rest of the
paper, the latter approach will be used.

#### 3.3.1 Maximally Diverse Groups

Many group assignment algorithms have focused on creating
maximally diverse groups where the sum of pairwise
differences of some characteristics is maximized. The
difference between two students \(x\) and \(y\) can be defined as

\[
d(x, y) = \sum_{1 \leq i \leq r} |c_{xi} - c_{yi}|.
\] (1)

The goal of creating maximally diverse groups is then
formulated as adding up the differences for all groups and
all student pairs within each groups, that is,

\[
g(S) = \sum_{1 \leq i \leq m} \sum_{x \in G_i} \sum_{y \in G_j} d(x, y).
\] (2)

The members of the groups \(G_i\) are of \(S\) being represented as
\((G_1, G_2, \ldots, G_m)\). This is the same formulation used by other
authors [22].

#### 3.3.2 Evenly Skilled Groups

One should not assume that groups will work well just
because they are formed properly according to some
criteria. In some group projects, it makes a lot of sense to
encourage the teams explicitly to teach each other relevant
skills and knowledge that some but not all group members
have, consistent with reciprocal teaching [14]. Originally,
reciprocal teaching was used to improve reading compre-
hension where students would take turns in leading the
discussion. In this way, the students took on the role of the
teacher. In a situation where students have a slightly
different skill set, they should help the other students to
learn it. However, this may not come naturally without the
explicit encouragement of the instructor who is not present
in most of the group meetings. Thus, students should be
assigned to teams such that all the relevant skills are
covered. This not only improves the chance of doing a good
project, but it also enables reciprocal teaching.

The even skilled criterion is especially interesting for
skilled, yet diverse student populations. They can be found
interdisciplinary programs such as cognitive science or
human-computer interaction. Other typical situations
matching this student profile are business administration
(MBA) courses and programs helping professionals to find
new careers. The latter has become especially important in
the current economic situation where many jobs disappear
and people have to learn a new profession. It surely would
be useful to take advantage of these students' existing skills
in the classroom.

Table 1 shows a small example with three groups, two
skills \(A\) and \(B\), and four students per group. Each skill is
rated on a scale from 1 to 5 with 5 being best. The last line
displays the maximal rating for each skill and group. For
instance, skill \(A\) gets a rating of 3 for groups 1 and 2,
suggesting that these groups will have problems because nobody is very good at skill $A$. On the other hand, group 3 has a rating of 5 for skill $A$ which means that this group will do fine for skill $A$. Furthermore, if the students indeed teach each other as they were asked to by the instructor at the beginning of the course, the first student in group 3 with the high rating will teach the others. Thus, the other students in that group will improve for this skill probably more than the members in groups 1 and 2. Therefore, the worst value for skill $A$ across all three groups is $C_A = \min(3,3,5) = 3$ and for $B$ it is $C_B = \min(4,5,4) = 4$. These two values are computed by (3). Since we want to have high values for all skills, we maximize the worst skill, that is, $\min(C_A,C_B)$ needs to be maximized. As explained below, a slightly different function shown in (4) is maximized to improve the performance of the search algorithm.

Therefore, we define a new optimization criterion, evenly skilled groups, by maximizing the minimal skill for each group. Assume that we want student groups to have some minimal expertise in designing a system architecture, in user experience design and writing reports. The group skill for each of these individual characteristics is the maximum over all the members’ skills. Since we want to maximize the worst skill of the group, we maximize over the minimal group skill. This avoids the problem of having imbalances that can result from a global sum of differences only as discussed in Section 2. Let $C_j$ be the worst value for skill $j$ across all groups, that is,

$$ C_j = \min_{1 \leq j \leq m} \max_{x \in G_j} c_{x,j}. \quad (3) $$

Then, we ought to maximize over the worst skill, that is, maximize $g(S) = \min_{1 \leq j \leq r} C_j$. However, this is problematic because this function is not smooth, making the search very little information about the quality of the individual groups. For instance, if the skills are rated on an $l$-level Likert scale, $g(S)$ can assume $l$ different values. Considering that $l$ tends to be small, say five or seven, this is not enough information for the search algorithm to differentiate between solutions of somewhat different quality. Therefore, an additional factor is added such that the sum of the values is also considered. This results in up to $l(l-r+1)/l \approx (l-1)r$ more values than without this factor. Although this reintroduces the danger of having imbalanced groups to a minor degree, it contains much more information than the max factor alone. Thus, the actual optimization criterion is

$$ g(S) = \left( \min_{1 \leq j \leq r} C_j \right) \cdot \sum_{1 \leq j \leq r} C_j. \quad (4) $$

### 3.4 Preferences

In reality, an instructor often would like to add some additional preferences to the group creation process. As defined earlier, the function to be optimized is $f(S) = g(S) + h(S)$, where $h(S)$ is the term for the context-specific preferences. For each new preference, a term $h_i(S)$ is specified and $h(S)$ is defined as $h(S) = \sum_i \alpha_i h_i(S)$, where the $\alpha_i$ are parameters to influence what is more important, the general assignment criterion or some of the context-specific preferences. Thus, the function to be maximized is

$$ f(S) = g(S) + h(S) = g(S) + \sum_i \alpha_i h_i(S). \quad (5) $$

Assuming that the maximal values of $h_i(S)$ and $h_j(S)$ are the same, then the preference $i$ expressed by $h_i$ is more important than preference $j$ expressed by $h_j$ if $\alpha_i > \alpha_j$. Thus, preference $i$ has a better chance of being satisfied than $j$ in case of a conflict.

In the current implementation, neither $g(S)$ nor any $h_i(S)$ is scaled to a specific range. The value of $g(S)$ tends to be much larger than the one of any $h_i(S)$, because $r \leq g(S) \leq r^2$ assuming that the $r$ characteristics are rated on a Likert scale with values from 1 to $l$ (see (4)). Thus, satisfying the global criterion $g$ is preferred over satisfying the context-specific criteria $h_i$. Furthermore, the computational results below show that the $\alpha_i$ do not have to be adjusted in too subtle a way.

All preferences are formulated within this framework with one exception, the size of the groups. In its current formulation, TABUSEARCHSAP treats the group size as a hard, built-in constraint because a move in the algorithms is defined as a swap of two students which is a natural and simple approach. The advantage is that this preference is always fully satisfied. The disadvantage is that the size for each group must be specified in advance and is static. In general, this is not a big problem because it is uncommon to use group sizes that vary widely. Alternative approaches to the swap move are possible, for instance, by complementing swaps with simple moves of one student from one group to another [37].

#### 3.4.1 Preferring Friends and Avoiding Foes

Evidence suggests that groups perform better if the students can be in a group with students they like or prefer to work with for some other reason [9], [17]. Assume that we let the students prefer to work with some students and avoid some of the other students. Here, the former will be called friends and the latter foes. $F_{xy}$ represents the preference of student $x$ to be in the same group as student $y$, where $1 \leq x, y \leq n$. Of course, $F_{xy}$ is not necessarily equal to $F_{yx}$:

$$ F_{xy} = \begin{cases} -1, & \text{if } x \text{ wants to avoid } y, \\ 0, & \text{if } x \text{ doesn’t care either way about } y, \\ +1, & \text{if } x \text{ prefers } y’s \text{ group}. \end{cases} \quad (6) $$

Then, the evaluation function can be extended with the following $h_i(S)$. It adds up the $F_{ij}$ for all student pairs in all
the groups punishing the assignment if \( F_{ij} = -1 \) and rewarding it if \( F_{ij} = +1 \):

\[
h_1(S) = \sum_{1 \leq i \leq m} \sum_{x \in G_i} \sum_{y \in G_i} F_{xy}. \tag{7}
\]

A variation of this approach is to change the definition of \( F \) such that being not in the same group is a stronger preference than being in the same group. This can be simply accomplished by replacing \(+1\) with a smaller value, say \(+0.5\), in the definition of \( F_{ij} \) in (6).

Assuming that the friends-and-foes preferences should not influence the assignment too much, \( \alpha_1 \) has to have a relatively small value. Based on the observation that relatively few of these preferences were stated by the students, \( \alpha_1 \) was set to \(1/n\) resulting in \( \alpha_1 h_1(S) < 1 \) for most cases which is much less than \( g(S) \). The value of \( g(S) \) is the worst skill across all groups which tends to be 3 or more (see Tables 3, 4, 5, and 6). This is less arbitrary than it may appear at first. First, the \( \alpha \)s are normally set once for a certain context. Second, the results are quite stable even if the values are changed. What matters is that the relative sizes of the \( \alpha \)s are chosen reasonably.

### 3.4.2 Distributing Subsets of Students

Sometimes, it can be useful to request that a subset of students be evenly distributed over the groups. For instance, in a mixed course, we have a few students that are participating in the course from remote locations via a synchronous connection. One way of dealing with group projects is to have roughly one remote student per project. This can be easily accomplished using the “foes” idea from the previous section.

Assume that \( R \) is a set of students that we want to maximally distribute among the groups. This is achieved by setting the appropriate \( H_{xy} \)s to \(-1.0\) as follows:

\[
H_{xy} = \begin{cases} 
-1, & \text{if } x \in R \text{ and } y \in R \text{ and } x \neq y, \\
0, & \text{otherwise}.
\end{cases} \tag{8}
\]

Then, \( h_2 \) can be defined essentially like \( h_1 \), namely, by

\[
h_2(S) = \sum_{1 \leq i \leq m} \sum_{x \in G_i} \sum_{y \in G_i} H_{xy}. \tag{9}
\]

### 3.4.3 Assigning Students to Specific Groups

Assigning certain students to certain types of groups can be useful. For instance, students can be assigned to larger groups if the students are a bit less experienced or new to the specific academic program. In this case, we simply add a penalty if this condition is violated. Assume that we want to make sure that the students in set \( R \) are assigned to groups with at least \( l \) members. Function \( h^r \) returns \(-1\) if a student \( x \) is assigned to a too small group \( G \) and 0 otherwise:

\[
h^r(x, G) = \begin{cases} 
-1, & \text{if } |G| < l \text{ and } x \in G, \\
0, & \text{otherwise}.
\end{cases} \tag{10}
\]

Then, the definition of \( h_3(S) \) is

\[
h_3(S) = \sum_{1 \leq i \leq m} \sum_{x \in (G_i \cap R)} h^r(x, G_i). \tag{11}
\]

The three functions \( h_1, h_2, \) and \( h_3 \) represent a sample of optimization criteria that come from actual classroom requirements. Of course, this sample is not exhaustive, though it shows how new requirements can be added.

### 3.4.4 More Preferences

These examples show that some preferences occurring in educational contexts can be expressed in a relatively simple way without having to change the algorithm itself. However, can all interesting preferences be expressed in the given framework, and how does an instructor with a mathematically average background express such preferences?

Any preference needs to be translated into a function of the form \( \alpha_i h_i(S) \). The larger \( h_i(S) \), the better the student assignment \( S \) is for this preference. It is advisable to scale \( h_i(S) \) so that it has values in a known range, because this makes it easier to weight it relative to other preferences using \( \alpha_i \). The optimization criteria used or suggested by other group formation approaches are quite easy to formulate in the framework used here. These criteria include heterogeneous groups [13], [21], [16], homogeneous groups [17], mixtures of homogeneous and heterogeneous groups [25], student location and availability [4], meeting capability and gender balance [38], and compatibility of students [29].

However, we should not expect a regular instructor to have to develop the mathematical formulation of the preferences. They need a proper user interface that allows them to formulate the preferences in pedagogical and organizational terms, not mathematical formulas. This is another reason why it is important that all optimization criteria have a clear interpretation in the domain in which the criteria are originally stated. If the only explanation for a group formation approach is an interesting algorithm, we may lack the possibility of providing a useful and usable interface to the end user.

### 4 Results

Assigning students to groups with a computer-supported approach requires us to assess the computational and educational quality of the group assignments. Since good computational results are a prerequisite for assessing the approach’s educational value, we start with the computational results.

#### 4.1 Computational Results

The Tabu Search approach was tested with four classes A–D in graduate-level user-interface design courses. The students were assigned to groups of sizes three and four and had to solve a difficult user interface design problem during a whole 15-week semester. The students in all classes were graduate students between the age of 22 and 61. Most had at least five years of experience in a wide variety of fields including engineering, technical writing, psychology, and history. Thus, they fit the profile mentioned earlier of being diverse but providing some useful skills that not all students yet have. Given that they were in the same course, they had a shared interest in the design of user interfaces.

The design problems given to the groups were too difficult to solve by an individual and team work was essential. The task the students had to design a prototype of
a user interface of some, possibly mobile, computational device. Such a group project requires a set of skills that can be found in some but rarely all students including understanding the latest computer technology, communicating well in written and spoken form, managing a team, and understanding usability criteria. Although most of these design projects could be divided relatively easily into subtasks, the group members had to frequently meet to exchange what they have learned. Such decomposable tasks include researching the audience of the interface, researching potential technologies to be used, and designing some initial prototypes. Some of these subtasks can be very difficult for some of the group members. For instance, an engineer never had interviewed a potential user, or a history major was unfamiliar with technical vocabulary to research the latest gesture recognition technology for very large screens. In these cases, the students in the group were expected to take advantage of the respective experts in the group, not by delegating the subtasks to them but learning from them to do the task. Naturally, there are also activities where it is necessary to meet as a group and discuss the designs and even redesign them together.

Encouraging interaction and teaching each other missing skills and knowledge within a group is important, but does not necessarily happen automatically. Therefore, the students were told to make sure that all group members must understand all subproblems and their solutions by teaching each other the missing skills. This was reinforced by asking the group members for explanations for any of the subproblems.

Class A had 18 students which had to be assigned to groups of size three and four. The input data and parameters are shown in Table 2. The students rated themselves on four skills which were maximized using the “evenly skilled groups” criterion described earlier. The students tend to know each other so that the use of friends and foes seemed to be warranted. However, not much weight was given as discussed earlier and was basically used as a tie breaker. Therefore, the students could list which students they would prefer (friends) to be in their groups and which ones they would rather avoid (foes). Six students chose up to six friends and four up to eight foes. Of course, this information was not made available to other students. The foe preferences were given preferential treatment over the friends by setting $F_{ij}$ as described earlier to the values $-1$ for foes, $+0.5$ for friends, and 0.0 otherwise. Satisfying foe preferences first is based on reports from students that some student pairs had caused frictions within the group and should be avoided in the future. Finally, two subsets of students were spread across all the groups to make sure that they would not all end up in the same project. The second subset $\{3, 15\}$ is redundant because these two students are already in the first subset. However, it should not be the instructor’s task to worry about such redundancies, and furthermore, it strengthens the preference to keep $3$ and $15$ in separate groups since two independent reasons support it. One included students new in the academic program, and the other included remote students who were attending classes via an audio link. Furthermore, new students needed to be assigned to the larger groups with four group members.

Ten runs for class A are shown in Table 3. The number $u$ of runs that found the solution is shown in the first row. The skills column contains the worst rating found across the groups for each skill which was the same for each run. The rating $(5, 3, 3, 3)$ means that no group had a rating lower than 5 for the first skill, and no rating lower than 3 for the other skills. The skills were rated on a five-level Likert scale. The $\alpha$s are the factors used for the various preferences. They had the following values for all runs: $\alpha_1 = 1/n$, $\alpha_2 = \alpha_3 = 1$. Thus, there was no need to adjust the parameters for different classes.

The other columns show the performance for the other preferences as well as the value for $g(S)$, the evenly skilled group optimization criterion. The more friends and the fewer foes, the better. Since in this context, friends are considered less important than foes, the number of foes ought to be zero which is always achieved. The results are very stable and only differ in how many friend preferences could be accommodated. The penalty for having foes in the same group was slightly stronger than not having friends in the same group. This implements the idea that satisfying foe preferences is more important than satisfying friend preferences as argued for earlier. Furthermore, some of these preferences contradict each other and cannot all be satisfied simultaneously. All runs returned solutions where the two groups (remote and new implementation).
students) were appropriately spread across the groups. The preference of putting new students in large groups was also always satisfied.

Class B was similar to A but five instead of four characteristics were used. It had 22 students with five listing up to three friends and three listing up to four foes. The only special preference was to distribute two students and keep them in large groups. The results of 10 runs are shown in Table 4. Also, for this class, the results were quite stable again mostly varying in the number of friend preferences fulfilled. The only exception to that was the run listed last in Table 4 which has one group with a lower minimal skill traded off by many friends. This is not a serious problem and not uncommon with heuristic approaches. Running the search a few times will make sure that one does not end up with such an outlier.

Finally, the same approach was run on two old classes C and D where the students had been assigned manually to the groups trying to find evenly skilled groups, maximizing the number of friends, and minimizing the number of foes. A simple way to evaluate the automatic approach is to compare it to assigning the students manually [39]. Here, it took in the order of 2 hours work using a spreadsheet compared to the 20-40 seconds the Tabu Search algorithm takes. The current implementation is not optimized for speed and implemented in Python, which is an interpreted programming language. It would, therefore, be easy to get a speedup of one or two magnitudes with a compiled language like C++. The complexity of the algorithm can be approximated as follows: Each loop consisting of the GENERATE, TEST, and UPDATE steps has the complexity dominated by the number of neighbor states generated. A good approximation of the number of neighbor states is the number of student pairs (for the swap move) which is \( n(n+1)/2 \), where \( n \) is the number of students. Thus, the algorithm has time complexity of \( O(N(n) \cdot n^2) \), where \( N(n) \) is the number of actual moves made. Test runs show that \( N(n) \approx 1.4n \) for the specific stop criterion and queue length; thus, the approximate complexity of TABUSEARCHSAP is \( O(n^3) \).

Class C had 21 students and not too many friend and foe preferences, whereas class D had 18 students with many friend and foe preferences. The results of 10 runs each are shown in Tables 5 and 6, respectively. The manual approach shown in the “Manual” row was worse both times. With respect to evenly distributed skills, the manual approach was not much worse; however, the computer program found more friends, especially with relatively many additional preferences as for class D.

How good are these really from a computational point of view given that the heuristics used in the Tabu Search implementation are relatively simple? A simple approach is, of course, preferred, but only as long as quality is not sacrificed. Therefore, the results are compared to an upper bound of the solution which is computed as shown in Table 7. The \( m \)th largest value is selected for each individual skill, where \( m \) is the number of groups. This is equivalent to sorting the skills individually, and then, selecting the \( m \)th last row, as shown in Table 7. This is only an upper bound and the optimal solution may be worse. The upper bound for friends and foes is simply the number of friend and foe preferences asserted, respectively. Of course, not all preferences can be completely satisfied because they are contradictory. Some students list many more friends than the group size. Some student \( i \) students in class A.

### TABLE 4
Ten Runs for Class B

<table>
<thead>
<tr>
<th>( u )</th>
<th>Skills</th>
<th>Friends</th>
<th>Foes</th>
<th>Spread</th>
<th>Size</th>
<th>( g(S) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>(4, 4, 5, 4, 4)</td>
<td>4</td>
<td>0</td>
<td>yes</td>
<td>yes</td>
<td>84</td>
</tr>
<tr>
<td>5</td>
<td>(4, 4, 5, 4, 4)</td>
<td>6</td>
<td>0</td>
<td>yes</td>
<td>yes</td>
<td>84</td>
</tr>
<tr>
<td>1</td>
<td>(4, 4, 5, 4, 4)</td>
<td>7</td>
<td>0</td>
<td>yes</td>
<td>yes</td>
<td>84</td>
</tr>
<tr>
<td>1</td>
<td>(4, 4, 5, 4, 4)</td>
<td>9</td>
<td>1</td>
<td>yes</td>
<td>yes</td>
<td>60</td>
</tr>
</tbody>
</table>

### TABLE 5
Ten Runs for Class C

<table>
<thead>
<tr>
<th>( u )</th>
<th>Skills</th>
<th>Friends</th>
<th>Foes</th>
<th>( g(S) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>(4, 4, 4, 3)</td>
<td>3</td>
<td>0</td>
<td>45</td>
</tr>
<tr>
<td>3</td>
<td>(4, 4, 4, 3)</td>
<td>4</td>
<td>0</td>
<td>45</td>
</tr>
<tr>
<td>2</td>
<td>(4, 4, 4, 3)</td>
<td>5</td>
<td>0</td>
<td>45</td>
</tr>
<tr>
<td>Manual</td>
<td>(4, 4, 3, 3)</td>
<td>2</td>
<td>0</td>
<td>42</td>
</tr>
</tbody>
</table>

### TABLE 6
Ten Runs for Class D

<table>
<thead>
<tr>
<th>( u )</th>
<th>Skills</th>
<th>Friends</th>
<th>Foes</th>
<th>( g(S) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>(4, 4, 5, 4, 4)</td>
<td>12</td>
<td>0</td>
<td>84</td>
</tr>
<tr>
<td>3</td>
<td>(4, 4, 5, 4, 4)</td>
<td>13</td>
<td>0</td>
<td>84</td>
</tr>
<tr>
<td>3</td>
<td>(4, 4, 5, 4, 4)</td>
<td>14</td>
<td>0</td>
<td>84</td>
</tr>
<tr>
<td>1</td>
<td>(4, 4, 5, 4, 4)</td>
<td>15</td>
<td>0</td>
<td>84</td>
</tr>
<tr>
<td>Manual</td>
<td>(4, 4, 5, 3, 4)</td>
<td>7</td>
<td>0</td>
<td>60</td>
</tr>
</tbody>
</table>

### TABLE 7
Finding an Upper Bound (UB) of the Skills for Class A

<table>
<thead>
<tr>
<th>Student</th>
<th>Skills</th>
<th>Individually Sorted Skills</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(1, 3, 2, 1)</td>
<td>1 1 1 1</td>
</tr>
<tr>
<td>2</td>
<td>(4, 2, 3, 2)</td>
<td>2 1 1 1</td>
</tr>
<tr>
<td>3</td>
<td>(5, 1, 2, 1)</td>
<td>2 1 2 1</td>
</tr>
<tr>
<td>4</td>
<td>(3, 5, 2, 3)</td>
<td>3 2 2 1</td>
</tr>
<tr>
<td>5</td>
<td>(3, 3, 1, 3)</td>
<td>3 2 2 1</td>
</tr>
<tr>
<td>6</td>
<td>(2, 5, 5, 3)</td>
<td>3 2 2 1</td>
</tr>
<tr>
<td>7</td>
<td>(5, 3, 3, 2)</td>
<td>3 2 2 1</td>
</tr>
<tr>
<td>8</td>
<td>(3, 2, 3, 3)</td>
<td>3 2 2 1</td>
</tr>
<tr>
<td>9</td>
<td>(3, 4, 3, 2)</td>
<td>3 3 3 2</td>
</tr>
<tr>
<td>10</td>
<td>(3, 3, 5, 5)</td>
<td>4 3 3 2</td>
</tr>
<tr>
<td>11</td>
<td>(5, 2, 4, 1)</td>
<td>4 3 3 2</td>
</tr>
<tr>
<td>12</td>
<td>(5, 2, 2, 1)</td>
<td>5 3 3 2</td>
</tr>
<tr>
<td>13</td>
<td>(5, 3, 3, 2)</td>
<td>5 3 3 2</td>
</tr>
<tr>
<td>14</td>
<td>(4, 1, 1, 1)</td>
<td>5 3 3 2</td>
</tr>
<tr>
<td>15</td>
<td>(3, 2, 4, 1)</td>
<td>5 4 4 3</td>
</tr>
<tr>
<td>16</td>
<td>(2, 1, 3, 1)</td>
<td>5 4 4 3</td>
</tr>
<tr>
<td>17</td>
<td>(4, 4, 3, 3)</td>
<td>5 5 5 3</td>
</tr>
<tr>
<td>18</td>
<td>(5, 3, 2, 1)</td>
<td>5 5 5 5</td>
</tr>
</tbody>
</table>

← UB
may list $j$ as a friend, yet $j$ considers $i$ a foe and not both can be made happy.

Nevertheless, looking at the results in Table 8, the results found by the Tabu Search method are good. The best solution found is always optimal with respect to the skills. It always satisfied all foe preferences and many friend preferences.

4.2 Educational Results

As discussed earlier, the various general criteria like homogeneous, heterogeneous, mixed, or evenly skilled groups need to be matched with the learning context including teaching method, type of content, learning goals, and student characteristics. Thus, we don’t need to try to show that one is better than the others independent of the context. For instance, the heterogeneous criterion should be chosen because groups with students who are quite diverse regarding some characteristics tend to result in improved learning. If we want to have diverging views, at least to start with, then using a heterogeneous approach for certain characteristics makes sense. If we are interested in similar students, for instance, with similar interests and similar progress regarding the current topic, then a homogeneous approach is warranted. Similarly, if we have different relevant skills, then the evenly skilled is to be preferred. As mentioned earlier, group assignment criteria need to be chosen dependent on the teaching method. However, these pairings should be viewed as a guidelines only, because the resulting group assignment may not be necessarily optimal due to some, possibly unforeseen, characteristics of the learning situation.

The students in all four classes $A$, $B$, $C$, and $D$ had highly diverse backgrounds including technical writing, engineering, psychology, and history. Although they all were interested in topics related to interface design and human factors, their skill sets were quite different with respect to these topics. Some were excellent writers, some had great programming skills, others yet knew quite a bit about psychology, about usability, and design. Some had several of these skills, but none had all.

Therefore, the evenly skilled group assignment criterion was used and the students were explicitly asked to make sure that everybody will teach each other missing skills. This was further reinforced by requiring that each student could answer any aspect of the project. The experience with this approach has been very positive. The duration of the group projects for all the four classes was about three months. The evenly skilled preference has resulted in several comments from highly skilled students that they indeed had to teach more and from less skilled ones that they indeed benefited from the other students. This is, of course, to be expected if the students are assigned to groups with the evenly skilled criterion. Similarly, empowering students to choose their own teams, as long the groups are still evenly skilled, has also resulted in very few interpersonal problems within groups and positive reactions when the group compositions were announced.

This kind of positive informal evidence may not prove much. However, if we use a teaching method that is known to work for certain student groups and we indeed have such groups, the outcome is bound to be positive. In this case, we know that students teaching each other is a great tool for all students involved, and we also know that the group compositions were good with respect to the evenly skilled criterion. Therefore, the outcome was bound to be positive as the informal evidence shows.

The context-specific preferences tend to be of a somewhat different nature. As their name suggests, they often accommodate some specific local situation that cannot be generalized easily. They can be based on some ad hoc approaches that are nevertheless grounded in long-term teaching experience. Assessing these preferences is difficult because they change in every class, yet they accommodate the real-world needs.

### 5 Conclusion

Although a lot of research into how students should be assigned to groups has been done, the results are still inconclusive at best. In any case, the local learning context of group projects often requires additional preferences that cannot (easily) be dealt with by maximally diverse or other general criteria. Therefore, the approach presented focused on combining a general assignment criterion with a set of context-specific preferences.

A new type of assignment criterion, evenly skilled groups, has been used to reduce the chance for unbalanced groups which is mathematically not a problem, but pedagogically is. It is especially suited for a skilled and diverse student population which is often the case for adult students.

The optimization function was extended to include general criteria as well as context-specific requirements giving the instructor much more flexibility to define what characteristics a good group assignment has. The examples show how just about any preference can naturally be expressed in the presented framework. In fact, in my classes, I use all the types of preferences discussed. All these preferences are used in conjunction with the “evenly skilled groups” assignment strategy with results where all preferences were satisfied except for a few students who all wanted to be in the same group. Although several potentially contradictory preferences are stated, the results were better than those assignments by hand.

A relatively informal criterion for the quality of the results was used: most preferences are satisfied, especially the more important ones, and the optimization criterion...
(e.g., evenly skilled groups) is indeed optimized. The results are indeed always close to an optimistic upper bound. However, since several of the input parameters used to describe the students are rather approximate and subjective, often based on self-evaluation of the students, pushing for a mathematically optimal solution is not that relevant. Under these circumstances, a good solution is just as good as an “optimal” solution.

The next steps will be developing a more efficient version of TABUSEARCHSAP that also scales better and an interface for end users. We can expect 50-fold speedup simply by using a compiled programming language like C++ instead Python resulting in running times of less than 1 second. Test runs with randomly generated classes were run. The classes were generated by generating \( n \) students with uniformly distributed characteristics. Typical numbers for characteristics, friends, and foes were chosen: four characteristics, up to three friends, and at most two foes were selected randomly. These runs suggest that group assignments with up to 100 students could be solved by a C++ implementation in about 1 minute. This is sufficient for assignments with up to 100 students could be solved by a C++ implementation in about 1 minute. This is sufficient for assignments with up to 100 students could be solved by a C++ implementation in about 1 minute. This is sufficient for


differences, up to three friends, and at most two foes were chosen: four characteristics, friends, and foes were selected randomly. These runs suggest that group assignments with up to 100 students could be solved by a C++ implementation in about 1 minute. This is sufficient for most classroom situations but may be wanting for large online communities. Furthermore, switching the implementation language does not improve the \( O(n^3) \) time complexity. Using a smarter move generation strategy that only considers promising moves could possibly improve this bound \[37\].

Showing the educational validity of group formation approaches has been quite challenging in the past, and as a reaction to this situation, a set of metrics has been suggested \[40\] and some interesting data collected \[5\]. However, instead of trying to prove that certain methods—be that homogeneous, heterogeneous, evenly skilled, or yet another criterion—leads to improved learning in certain situations, we can also consider all these approaches, including the set of context-specific preferences defined in this paper, to be part of a tool set for the instructor.

References


Roland Hübscher received the PhD degree in computer science from the University of Colorado at Boulder. He is currently an associate professor at Bentley University, where he teaches in the Human Factors in Information Design program. His research focuses on intelligent user interfaces and algorithms to support students and teachers.