A New Approach to Collaborative Group Formation

Anurag Sarkar
St. Xavier's College
(Kolkata)

Dibyabiva Seth
St. Xavier's College
(Kolkata)

Kaustav Basu
St. Xavier's College
(Kolkata)

Anal Acharya
Assistant Professor
St. Xavier's College
(Kolkata)

ABSTRACT
This paper implements a tool, referred to as the Automated Group Decomposition Program (AGDP), which divides a class of students into groups, using the k-means algorithm, for the purpose of collaborative learning, and then heterogenizes the groups based on a factor called the degree of heterogeneity (DOH). The tool takes as input two sets of scores and the students' roll numbers and outputs the required groups. The first score set scored students on communication skills, fluency in using computers and group work attitude. This score set was used to generate the homogeneous groups. The second score set scored students on their knowledge of the subject and was used to generate the heterogeneous groups. The tool can be used to generate homogeneous clusters, heterogeneous clusters and a mixture of both. This tool can not only be used by teachers but also by instructors with minimal computer experience wishing to form groups to maximize learning.

Keywords
collaborative learning, k-means, constraint satisfaction, degree of homogeneity, group learning, homogeneous groups, heterogeneous groups.

1. INTRODUCTION
Collaborative learning is a method of learning where two or more people learn together. Each individual in the group works off of their colleagues' knowledge and resources to broaden their own knowledge bases. According to Gerlach (1994), "Collaborative learning is based on the idea that learning is a naturally social act in which the participants talk among themselves. It is through the talk that learning occurs". Interaction is considered to be the integral part of collaborative learning. Participants in a group, with different knowledge levels, interact with each other and try to overcome their own shortcomings. People engaged in collaborative learning are exposed to other viewpoints which may significantly vary from their own. This enables them to come up with ways to defend their viewpoint(s) and ensures that they have a deeper understanding of the subject. Without collaborative learning, once a student takes up a method to tackle some problem, his/her mind becomes sidetracked from alternate solutions. As a result, if that method eventually fails, more often than not, the student fails to accomplish the task. This is where collaborative learning comes into play. In collaborative learning, the students are divided into groups, and each group is assigned a task. The members of a group can discuss the issue and each member can come up with an alternative suggestion to tackle the given problem. They can also point out the faults or problems with their colleagues' ideas and hence such problems can be avoided at an earlier stage. Now, if these groups are made heterogeneous i.e. each group consists of members having somewhat different knowledge bases and resources, then the members benefit even more. They are able to collaborate and learn new things, expand their knowledge bases, and tackle the given problem in an easier way by decomposing the problem into sub-problems among the members. This is absolutely not possible without collaborative learning. Studies ([1],[2],[3],[4],[5],[6],[7],[8]) have shown that collaborative learning is more efficient than individual learning.

In the Automated Group Decomposition Program, the final groups are formed in a heterogeneous manner such that the members of each group can share their experiences and skills in order to help build the skills of one another. The program asks the instructor whether he or she wants to enter the data manually or automatically by uploading files containing the data. If he or she chooses manual entry, the instructor is asked to enter the marks and relevant student information. Else, the instructor is asked to select the files that contain the required data. After this initial phase, the program then asks the instructor to enter the number of groups or clusters that he or she wants to divide the class into. The instructor can also choose whether he or she wants to implement purely homogeneous clusters, purely heterogeneous clusters or a mixture of both. After providing the program with this information, the program then computes and displays the output based upon the choice of the user.

2. RELATED LITERATURE
Graf and Bekele [1] have suggested that "A reasonably heterogeneous group refers to a group where student-scores reveal a combination of low, average and high student-scores. This is justified by the recommendation of Slavin [2] who proposes that students should work in small, mixed-ability groups of four members: one high achiever, two average achievers, and one low achiever." Homogeneous grouping is not a very efficient method of decomposing a number of students into multiple groups because some groups tend to be very strong while other groups may be very weak at performing the given task. As a result, the stronger groups finish the task very easily, while the weaker groups fail to accomplish the goal. Heterogeneous grouping tries to eliminate this drawback and form groups of approximately similar abilities. The stronger members of a heterogeneous group are also able to share their knowledge among the weaker members of the group, thus helping them to learn more effectively. Wang, Lin and Sun [9] have proposed a computer-assisted grouping system called DIANA to form heterogeneous groups exhibiting "internal diversity" and "external balance with other groups". They have performed the grouping based on student data collected via psychological questionnaires. They have then processed this data and used a genetic algorithm (GA) to form the optimal groups. Christodoulopoulos and Papanikolaou [10] have presented a web-based group formation tool which can form both homogeneous and heterogeneous groups automatically, and also allows the learner to negotiate the grouping if needed. They have used the Fuzzy C-Means algorithm to form homogeneous groups, and a standard random selection algorithm to form the heterogeneous groups. Redmond [11] has used a greedy search algorithm to form the optimal groups. He has allowed the students to express their project preferences as well, considering multiple project topics. His

program comprises 32 procedures and functions, spread over 3250 lines of code. Gokhale [12] has undertaken two tests, drill-practice skills and critical thinking skills, to determine whether collaborative learning is more efficient than individual learning. She developed a nine item questionnaire and conducted the study on a total of 48 subjects. Her study shows that collaborative learning is more beneficial for enhancing critical thinking and problem solving skills. Alavi [13] has performed a study to investigate the enhancement of student learning on using a group decision support system (GDSS) in a collaborative learning process. Her study involved 127 MBA students and has determined that the use of a GDSS in collaborative learning leads to higher skill development and self-reported learning.

3. AUTOMATED GROUP DECOMPOSITION PROGRAM

3.1 Research Methodology

In this paper, a method has been devised to implement collaborative learning using the k-means algorithm and the concept of constraint satisfaction to create groups intelligently.

In order to perform the clustering, two sets of scores are used. The first set, referred to as the 'A-scores', is the sum of scores obtained by the students in three categories – communication skills, fluency in computer usage and group work attitude. The second set, referred to as the 'B-scores', is a measure of the subject knowledge possessed by each student. The program has been implemented using the Java programming language.First, a modified version of the k-means algorithm is used to divide the students into k homogeneous clusters where k is supplied by the instructor and the homogenization is based on the A-scores. Then cluster equalization is performed to ensure that each cluster has an equal number of students. Following this, the clusters are heterogenized based on the B-scores using an algorithm that satisfies the degree of heterogeneity (DOH) constraint. The DOH value is defined for each cluster as:

DOH of each cluster = (Number of ranges of B-scores present in that cluster) / Cluster size

where the ranges of B-scores are the ranges 0-10, 11-20, 21-30, …, up to the range where the upper limit is the maximum attainable B-score. This formula has been devised such that, for situations in which the number of members in a cluster is less than or equal to the total number of ranges of B-scores in the data, the maximum possible DOH value for a cluster is 1.0, which represents the situation in which each member of a cluster belongs to a different 10-mark range of the B-scores, and thus, it is impossible to further heterogenize the cluster. For situations in which the number of members in a cluster is greater than the total number of ranges of B-scores, it is impossible to attain a DOH value of 1.0 and the maximum attainable DOH value for each cluster depends on the difference between the cluster size and the total number of B-score ranges.

The following algorithms have been used to form the required groups or clusters.

3.2 Algorithms

The following variables have been used in the algorithms:

ascores ← list of the summation of the marks obtained by the students according to the first 3 parameters

bscores ← list of the marks obtained by the students according to the fourth parameter

roll ← list of the roll numbers of all the students

size ← total number of students present

centers ← stores the centers of the clusters

clusters ← 2D matrix storing the roll numbers of the members of the clusters

k ← number of groups/clusters to be formed

m ← size/k (number of members in each group/cluster)

rangeCount ← 2D matrix in which the element [i,j] stores the number of times a b-score in the jth range appears in the ith cluster

//Create the clusters

3.2.1 Algorithm makeClusters

```
BEGIN
        generateCenters()
        WHILE true, DO
                Clear all the clusters
                LOOP through all the people i = 0 to size-1
                        PERSON = ith person
                        MARKS = marks_obtained_by_PERSON
                        Find the center i such that it is the center closest to MARKS
                        Add PERSON to the ith cluster
                END LOOP
                IF recalculateCenters() is true, then
                        EXIT WHILE
                END IF
        END WHILE
        equalizeClusters()
END
```

//Generate initial cluster centers

3.2.2 Algorithm generateCenters:

```
BEGIN
        Determine the minimum marks from ascores and store it in LO
        Determine the maximum marks from ascores and store it in HI
        Interval = (hi-lo)/k
        Make (lo + Interval/2) as the first center
        LOOP through all centers j = 1 to k-1
                Make cluster j = cluster (j-1) + Interval
        END LOOP
        RETURN
END
```

//Equalize the number of members in every cluster

3.2.2 Algorithm equalizeClusters

BEGIN

LOOP through all clusters i = k-1 to 1

IF cluster i has more than m number of members, then

EXTRA← the extra members of cluster i which are closest to cluster (i-1)

Add all members of EXTRA to cluster (i-1)

END IF

END LOOP

recalculateCenters()

LOOP through all clusters i=0 to k-2

IF cluster i has more than m number of members, then

EXTRA← the extra members of cluster i which are closest to cluster (i+1)

Add all members of EXTRA to cluster (i+1)

END IF

END LOOP

recalculateCenters()

RETURN

END

//Recalculate the centers

3.2.4 Algorithm recalculateCenters:

BEGIN

Flag= true

LOOP through all clusters i = 0 to k-1

NEW_CENTER i = average of all the members of cluster i

IF NEW_CENTER i is different than center i, THEN

Center i = NEW_CENTER i

Flag = false

END IF

END FOR

RETURN Flag

END

//Heterogenize the clusters based on Degree of Heterogeneity

3.2.5 Algorithm makeHet

BEGIN

Calculate the initial degree of heterogeneity (DOH) for each cluster

Determine the frequency of occurrence of each range in each cluster

LOOP through all clusters i = 0 to k-1

LOOP through all clusters j = i+1 to k-1

Record the initial DOH for both clusters i and j

IF DOH of both clusters equals 1.0 THEN

The DOH cannot be optimized further, so break out of inner loop

END IF

Retrieve the b-scores of the students whose roll numbers are in cluster i

Determine the ranges for which there are no scores in the list of b-scores retrieved above. Call it needi

Also, determine the ranges for which there are scores in the list of b-scores above. Call it nonneedi

Do the same for cluster j, to obtain similar lists needj and nonneedj

Find the intersection of needj and nonneedi. This produces the ranges that are needed by j and not needed by i i.e. the ranges that can be transferred from i to j. Call it ITOJ.

Similarly, find the intersection of needi and nonneedj to obtain the ranges that can be transferred from j to i. Call it JTOI.

For ITOJ, determine the range IJ that has the most number of elements in cluster i

Similarly, for JTOI, determine the range JI that has the most number of elements in cluster j

Temporarily transfer an element in the range IJ from cluster i to cluster j

Temporarily transfer an element in the range JI from cluster j to cluster i

Calculate the DOH values for these new temporary clusters

IF the sum of the new DOH values is greater than the sum of the initial DOH values, then

Make permanent the changes to the clusters made above

Set the new DOH value for cluster i as the initial DOH value for cluster i for the next iteration

END IF

END FOR

END FOR

END

//Calculate degree of heterogeneity for a cluster

3.2.6 Algorithm calcDOH:

BEGIN

Initialize a set r to integers 0 to n-1 where n is the number of ranges for the b-scores;

LOOP through all elements b of the cluster for which DOH is to be calculated

Determine the range k that b belongs to

Remove k from set r

END LOOP

Calculate number of ranges of B-scores in the cluster as

(Total number of ranges of B-scores - The number of integers remaining in set r)

Calculate DOH as (the number of ranges of B-scores in the cluster) / (the size of the cluster)

Return DOH

END

//Returns a list of integers where the ith integer is the number of times a number in the ith range appears in the input cluster

3.2.7 Algorithm getRangeCounts:

BEGIN

Initialize a list rc to contain as many zeros as there are ranges

LOOP through all members b of the cluster for which range counts need to be computed

Determine the range k that b belongs to

Increment the kth element in the list rc

END LOOP

RETURN rc

END

3.3 Implementation

The program has been implemented using the Java programming language and the GUI has been implemented using the light-weight Java Swing GUI widget toolkit. When the instructor runs this software, Figure 1 is the first screen which is displayed. By clicking "About", the user can view what the software does. By clicking "Launch Program", Figure 2 is displayed, where the user chooses how he or she shall input the required data. If the "Manual Entry" option is selected, then the user is taken to Figure 3, where he or she manually enters the data. After manual data entry has been completed, the results are displayed (Figure 4). Otherwise, if the user selects the option "Select Files", the software automatically reads data from the selected files and displays output as shown in Figure 5.

In this study, a total of 24 students were divided into 6 groups (each containing 4 members). The scores obtained by the students were given as file-inputs to the program (figure 5), as well as manual inputs to the program (figure 3). The proposed groups were given as output by the program as shown in figure 4 and figure 5. The groups were made as a mixture, that is, homogeneous on the basis of the students' ASCORES and heterogeneous based on the students' BSCORES.

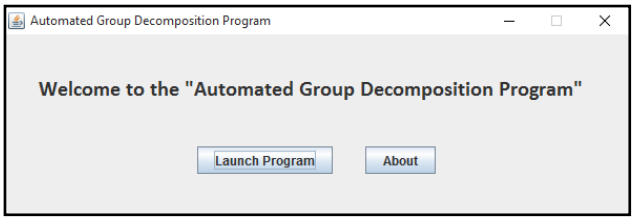

Fig 1: The welcome screen

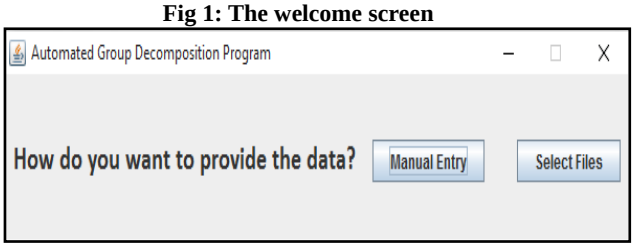

Fig 2: Data-entry option

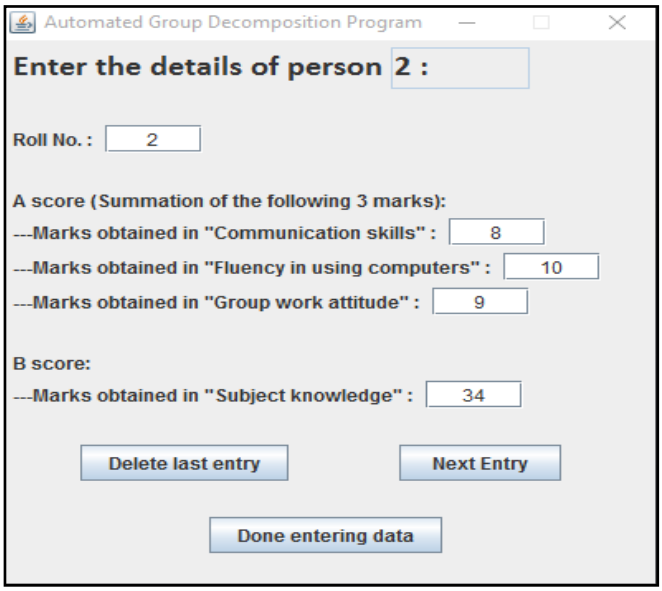

Fig 3: Manual data-entry option

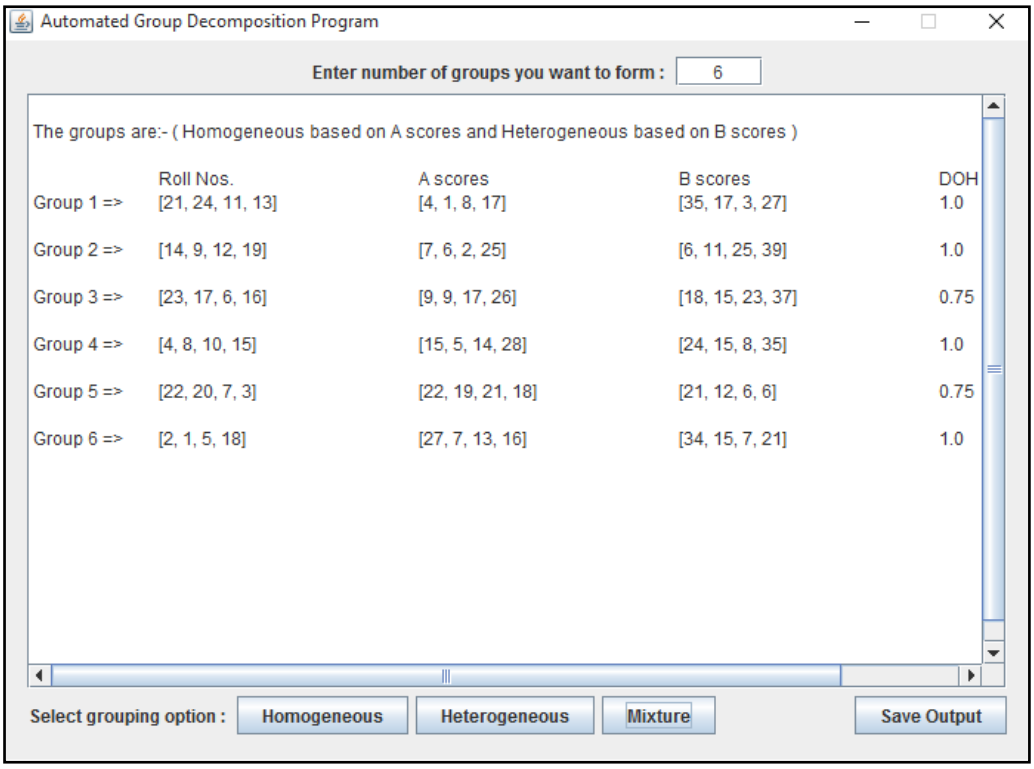

Fig 4: Resulting group decomposition after manual data entry

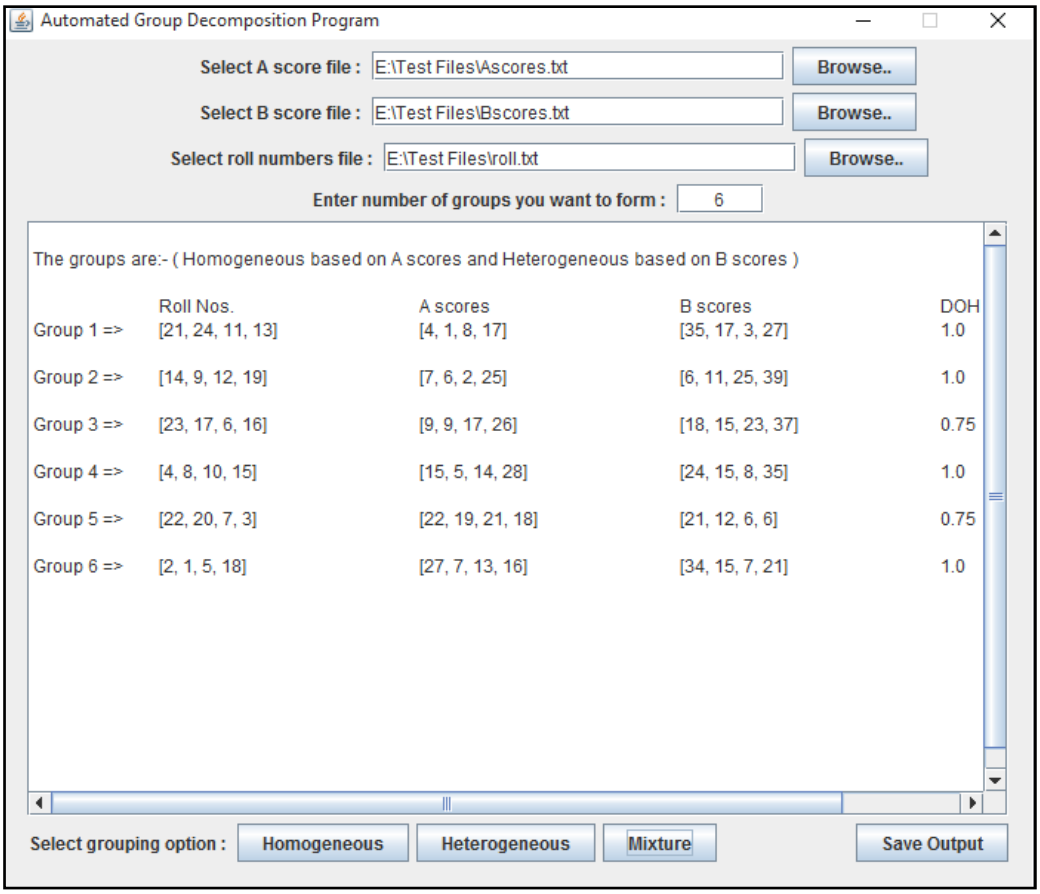

Fig 5: Resulting group decomposition after file selection

4. COMPARATIVE STUDY

In this section, a comparative study has been presented between the AGDP tool that has been developed and three other grouping tools as presented in [9], [10] and [11]. The comparison has been made based on the methodology used, the nature of the experiment, the parameters that were used to group the students and how heterogeneity has been implemented in each system.

4.1 Method

In [9], Wang, Lin and Sun have used a variant of the Random Mutation Hill Climbing (RMHC) algorithm to group students based on "shape". The original RMHC algorithm grouped students based on distances between points denoting individual scores. Here, the authors modified this algorithm to attain higher levels of heterogeneity. Their ultimate aim was to achieve internal diversity and external balance. "DIANA" also uses a genetic algorithm to optimize its groups. Christodoulopoulos and Papanikolaou [10] have used a number of algorithms to create their final group formation tool. They have used the Fuzzy C Means algorithm to calculate homogeneous groups. To compute the heterogeneous groups, they have used the Random Algorithm, which uses randomization as part of its logic. Finally, they employed PHP and MySQL to develop the Group Formation Tool.

In [11], Redmond has used the Pascal programming language to develop the group formation tool. The algorithm is based on greedy search with the aim being to find the student with the tightest time constraint and assign higher priority to said student. AGDP has been developed using the Java programming language. As stated earlier, it takes as input two sets of scores for the students and uses a modified k-means algorithm to form clusters. The clusters are then equalized to ensure that each group has the same number of students and finally heterogenized based on the degree of heterogeneity factor as defined earlier.

4.2 Experiment

"DIANA" [9] was designed to work as follows. Firstly, data was loaded based on student characteristics collected via psychological questionnaires. After determining optimal group size based on instructional and classroom management objectives, teachers could use a report generated by DIANA that listed student characteristic(s) and team numbers for composing heterogeneous groups. Teachers or instructors could load different psychological variables according to task requirements or instructional goals. DIANA can consider a maximum of seven different variables to form groups comprising of 3–7 members.

In [10], Christodoulopoulos and Papanikolaou have implemented a web based Group Formation Tool using PHP. Their experiment rotated around the usage of low complexity algorithms for the homogeneous and heterogeneous grouping.

[11] involves a computer program that aids the assignment of students into groups. The program searches for group assignments that result in compatible schedules and helps to achieve the desirable goal of heterogeneous groups, which leads to more learning for all members of the group.

As stated earlier, AGDP takes two sets of scores as input. The k means algorithm and an equalization process is performed on the first score set. This ensures homogeneous group formation. The second score set is used to perform the heterogeneous group division, which finally ensures collaborative learning. The front end of this tool was developed using Java Swing and has been discussed previously.

4.3 Parameters

In [9], students are grouped based on thinking styles i.e. the methods by which an individual uses his or her intelligence. The authors here have employed three thinking styles for grouping – legislative style thinkers (those who are innovative and like to do things based on their own rules), executive style thinkers (those who like to follow the rules that have been prescribed) and judicial style thinkers (those who do not pay a lot of attention to rules and instead prefer to make judgments and compare ideas based on their benefits and deficiencies).

Christodoulopoulos and Papanikolaou [10] have also performed the grouping based on a maximum of three aspects but have not explicitly specified what those aspects have to be, though they have suggested knowledge level and learning styles as appropriate example aspects that may be used for grouping.

In [11], the grouping has been done based on the schedules of the students to be grouped. The students are asked to rate the different possible time slots from best to worst and are also asked for project preferences. It then uses this scheduling information to assign the students to the different groups, starting by assigning the students with the fewest number of favorable time slots to a group and continuing until all students are assigned to some group.

In AGDP, three parameters are used to form the initial homogeneous groups of students. These are communication skills, fluency in computer usage and attitude towards group work. The sum of these 3 parameters, referred to in this tool as the A-score, is used for forming the initial homogenized and equalized grouping of students. This is followed by heterogenization which is done based on the subject knowledge marks obtained by each student. These marks are referred to as the B-scores.

4.4 Heterogeneity

Owing to the benefits that heterogeneous groups have to offer with respect to collaborative learning, AGDP as well as the three other tools, implement heterogeneity as part of the group formation process. In this section, a comparison, on how heterogenization of the groups has been achieved in the case of each tool, has been made.

In [9], Wang, Lin and Sun used a modified version of the RMHC algorithm to form heterogeneous groups. This algorithm groups students based on the distances between points that denote the individual scores of the students and considers as a single group those students whose scores form a triangular shape. While the original RMHC algorithm focuses on distance between points, the DIANA tool focuses on the similarities in shape of the triangles that are identified during grouping, which results in the formation of heterogeneous groups.

In [10], Christodoulopoulos and Papanikolaou implemented heterogeneity with the help of a random selection algorithm. This algorithm has the benefit of being fast and simple to implement while providing the required amount of heterogeneity as desired by the authors. In this case, a heterogeneous group has been defined to be one that

contains all possible values based on the data that is available. The random selection algorithm works based on the assumption that all the values have the same probability to belong to a certain group.

In [11], the grouping is done based on the preferences of students with respect to the different time slots in which they are able and willing to work. It starts with the student with the fewest preferred time slots and attempts to find a group for that student. It keeps repeating this until all the students have been assigned to groups. This process results in the formation of groups that are more heterogeneous in terms of gender and ethnicity than groups that would be formed by self-selection of the students. However, unlike AGDP and the other tools discussed here, there is no explicit heterogenization in this system based on a specific skill or characteristic of the students.

Finally, in AGDP, to obtain heterogeneous groups, for each iteration of the algorithm, members between any two clusters are exchanged if that exchange results in an increase in the DOH value of at least one of the two clusters, without decreasing the DOH value of either cluster. Thus, the constraint that each iteration of the algorithm must result in an increase in the sum of the DOH values of all clusters is enforced. This results in final groups that are adequately heterogeneous. Conceptually, the goal is to obtain results in which each group has a DOH value of greater than 0.5, which represents an adequate amount of heterogenization. However, explicitly adding this constraint to the program resulted in sub-optimal outputs in which fewer groups had a DOH value of 1.0 and many further optimizations that hadn't been performed by the program were clearly possible. This was because groups that satisfied the constraint of having a DOH value of greater than 0.5 were left untouched even though they could contribute in increasing the DOH value of other groups that were inadequately heterogenized. Removing this constraint, but satisfying the constraint pertaining to the exchanging of cluster members, provided better results since it allowed groups with DOH values of greater than 0.5 to be involved in the process of exchanging members as discussed previously. The details of this heterogenization methodology have been provided in the listing of the makeHet() algorithm in the 'Algorithms' section.

5. CONCLUSION AND FUTURE WORK

In this paper, the Automated Group Decomposition Program (AGDP) is presented which is a tool that can be used to form fully homogeneous, fully heterogeneous or a mixture of homogeneous and heterogeneous groups of students for the purpose of effective collaborative learning. The familiar concepts of k-means clustering and constraint satisfaction are employed to achieve the goal which is to form adequately heterogeneous groups which, as the studies referenced previously have shown, offer many benefits in a collaborative learning environment.

The key to this tool achieving heterogeneity in student groups is defining the 'Degree of heterogeneity' (DOH) value for each cluster. Implementing an algorithm that ensures that each iteration produces an improvement in the overall DOH values of all the clusters led to an increase in the level of heterogeneity as much as could be possible given the various parameters that were used to assess and group the students. Thus, this method of attaining heterogeneous groups strikes a balance in complexity between the more complex RMHC algorithm employed by [9] and the simpler random selection and implicit heterogenization employed in [10] and [11] respectively. This tool also features a simple GUI and can easily be used by any instructor, regardless of their familiarity with computers. Instructors can enter the required information about the students through either manual entry via the application interface or by uploading text files, as is more convenient.

These benefits notwithstanding, the following are possible improvements that can be made to the AGDP tool in the future:

- At present, the parameters used to form homogeneous clusters are fixed and are likely to not be applicable in diverse group learning situations. Thus, the tool could, in the future, allow the instructors to define their own parameters.
- Unlike in [10], where the group formation tool allows students to negotiate which groups they may be assigned to, in this tool, the students have no say in the group formation process. This could be addressed in future versions of the tool by implementing a simple feedback system, for example.
- While Wang, Lin and Sun [9] conducted a comprehensive experiment by employing their tool in a real-world setting, there was less opportunity to test out the effectiveness of this tool in a real classroom group project environment. Given the opportunity, this program can be tested to assign students into groups for an actual academic project and compare their performance with students assigned to random or self-selected groups over the course of an entire college semester.

6. REFERENCES

[1] Graf, S., and Bekele, R., "Forming Heterogeneous Groups for Intelligent Collaborative Learning Systems with Ant Colony Optimization", Lecture Notes in Computer Science Volume 4053, 2006, pp 217-226.

[2] Slavin, R.E, "Developmental and Motivational Perspectives on Cooperative Learning: A Reconciliation", Child Development, Vol. 58, No. 5, Special Issue on Schools and Development (1987) 1161–1167.

[3] Ames, C., and Ames, R. (eds.),"Research on Motivation in Education", Academic Press Inc., Orlando, USA (1985).

[4] Dansereau, D., and Johnson, D., "Cooperative learning. In: Druckman, D., Bjork, R.A.: Learning, Remembering, Believing: Enhancing Human Performance", National Academy Press, Washington, D.C. (1994) 83–111.

[5] Jacobs, G., "Cooperative Goal Structure: A Way to Improve Group Activities", ELT Journal. Vol. 42, No. 2 (1988) 97–100.

[6] Johnson, D.W., and Johnson, R.T., "Cooperative Classrooms. In: Brubacher, M. (eds.): Perspectives on Small Group Learning: Theory and Practice", Rubican Publishing Ind., Ontario (1990).

[7] Krejins, K., Kirschner, P.A., and Jochems, W., "The Sociability of Computer-Supported Collaborative Learning Environments. Educational Technology and Society", Vol. 5, No. 1 (2002) 26–37.

[8] Augustine, D. et al, "Cooperation works! Cooperative Learning can benefit all students", Educational Leadership 7 (1989).

[9] Dai-Yi Wang, Sunny S.J. Lin, and Chuen-Tsai Sun, "DIANA: A Computer-Supported heterogeneous grouping system for teachers to conduct successful small learning groups", Computers in Human Behavior, Volume 23, Issue 4, July 2007, Pages 1997–2010.

[10] Christodoulopoulos, C.E., and Papanikolaou, K.A., "A Group Formation Tool in an E-Learning Context", 19th IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2007), Vol. 2, pp. 117-123, Oct. 2007.

[11] Redmond, M.A., "A Computer Program to Aid Assignment of Student Project Groups", ACM SIGCSE Conference, 2001, pp. 134-138.

[12] Gokhale, A.A., "Collaborative Learning Enhances Critical Thinking", Journal of Technology Education Fall 1995, Vol. 7, Number 1.

[13] Alavi, M., "Computer-Mediated Collaborative Learning: An Empirical Evaluation", Management Information Systems Research Center, University of Minnesota, Vol. 18, No. 2 (Jun., 1994), pp. 159-174.