Team formation in engineering classrooms using multi-criteria optimization with genetic algorithms

Jutshi Agarwal
Department of Engineering Education
University of Cincinnati
Cincinnati, Ohio, USA
agarwaji@mail.uc.edu

Emily Piatt
Department of Mechanical Engineering
University of Cincinnati
Cincinnati, Ohio, USA
piatten@mail.uc.edu

P. K. Imbrie
Department of Engineering Education
University of Cincinnati
Cincinnati, Ohio, USA
imbriepk@ucmail.uc.edu

***Abstract* — The research-to-practice paper presents applications of genetic algorithms using multi-criterion optimization to designate student-teams in an academic setting. Teamwork skills are becoming a more desirable trait in industry today and hence more instructors are using teamwork in their engineering courses. ABET also requires engineering degree programs to have student outcomes that provide them with "an ability to function effectively on a team whose members together provide leadership, create a collaborative and inclusive environment, establish goals, plan tasks, and meet objectives." This, however, comes with its challenges because the literature suggests that random or student-selected teams tend to lead to dysfunctional behaviors. Instructor-designated academic teams based on skills, learning personalities, and demographics are known to be more effective in promoting learning and positive interactions. Creating optimal homogeneous or heterogeneous teams based on multiple criteria (e.g., prior knowledge, skills, abilities, psychosocial, demographics) can be a complex and time-consuming task when done manually, especially when large numbers of students are involved.**

This study presents an expansion of previous work that used single-criterion genetic algorithm optimization of teams in a first-year classroom. The research question answered in this study is, *how do teams formed using an algorithm that optimizes multiple criteria with genetic algorithms represent heterogeneity when compared to teams formed manually?* Using a Vector Evaluated Genetic Algorithm (VEGA), optimization of multiple skills is conducted to attain uniform heterogeneity in the classroom. The algorithm uses discrete integer-type representations as a basic unit of team configuration. Self-reported student competency data on three different computational skills were used. Teams were formed for approximately 1300 students enrolled in a first-year engineering design thinking course at a large Midwestern University. Four-member teams were maximized with a minimum number of teams with 3 students in each section. Average skills of the teams are calculated and the standard deviation in class is minimized for each of three skills parallelly. The algorithm also aims to maintain diversity of teams based on gender and ethnicity.

Results presented include visualization of team-configurations and comparison with teams formed manually using the same criteria. The aim of the algorithm is to optimize students into teams that have skills and demographics uniformly distributed across the classroom. Future implications of this study include the potential to use flexible algorithm-based team formations that are built with standard genetic operators so that optimization criteria can be modified by the instructor. Such algorithms can reduce the time needed to manually form teams by a considerable amount and optimize the distribution of skills more uniformly. Steps to further this study will involve investigating whether teams formed using these criteria are more effective.

Keywords—genetic algorithm, multi-criteria optimization, team formation

I. INTRODUCTION

The ABET guidelines require students to demonstrate their ability to function on a team that provides leadership, establishes goals, plans tasks, meets objectives, and fosters an inclusive environment [1]. In general, team-based learning exercises are becoming more common so as to prepare students for environments they will encounter in the workplace. Many teachers utilize team learning with project-based group work. There is evidence that shows collaboration in learning environments shows increased performance from students in terms of problem solving, conceptual understanding, and mental retention [2]. The idea of collaborative learning is based on placing students in groups to work through assignments together and work toward a common learning goal [3].

One common issue with utilizing group-based learning in the classroom is how to assign teams. Most commonly, instructors will assign teams at random or allow students to choose their own groups. This can result in teams having a lack of diversity and unbalanced skillsets. To combat this, instructors can assign teams based on different criteria that are important to the class learning objectives. Research suggests that instructor-assigned teams perform better [4]. Manually assigning teams using criteria-based selection, however, can be a tedious and time-consuming task [5]. This work proposes the use of a genetic algorithm (GA) to create teams in a first-year engineering class given a set of criteria that need to be optimized. The methodology section discusses the details and implementation of the proposed algorithm.

The work presented in this paper focuses on expanding on previous research using genetic algorithms for teaming [6] by applying a Vector Evaluated Genetic Algorithm (VEGA) Method. The VEGA method was first developed when Schaffer [7] modified the selection process of fundamental GA. This modified version was able to adapt the fundamental GA for multiple objectives [8]. Schaffer [7] recognized that, although GAs were already able to outperform other techniques, there was no way to apply the technique to problems in which the fitness function was a vector instead of a scalar. This new method uses a vector version of a survival-of-the-fittest process

to replace the selection portion of the GA. This changed the selection process to a loop that iterated through each of the criteria for the problem and was able to expand the use of GAs to problems that have multiple objectives that need to be met [7]. It must be noted that this study presents an algorithm to optimize teams using certain criteria. The aim is to evaluate the efficiency of the algorithm when compared to manual team formation in classroom. The impact and efficiency of the criteria itself is not evaluated in the study.

II. Problem Formulation

A. Aim

The goal of the algorithm outlined in this paper is to present a genetic algorithm-based system to form teams that are uniformly heterogeneous. The rating of each team is simultaneously based on three different selection criteria; basic programming knowledge of the students, excel knowledge level, and general computer knowledge. Each of these teams must also be diverse in both gender and race. The diversity in gender and in race is attained by avoiding isolation of a single member of a minority group on a team. It has been shown that individuals from a minority group can feel dominated when they are the only minority member on a team [9] [10] [11].

B. Given Data

The data for this problem came from a first year Engineering Design Thinking course at a R1 University. Within this course there are 25-28 sections, each containing 40-60 students each. Each student has a skill score for each of three skills (basic programming knowledge, excel knowledge, and computer knowledge). These skill scores were obtained from normalized values from a self-reported instrument that measured the skills from each student at the start of the semester. The gender and ethnicity data for each student were combined using school records. Specific details about the course structure were elaborated in [6].

III. Previous Work

A. Genetic Algorithms

A genetic algorithm is a search heuristic technique that is based on the process of natural evolution. This process reflects natural selection where the fittest individuals are selected for reproduction to produce offspring for the next generation [12]. This is computed mathematically using initialization, selection, crossover, mutation, and elitism algorithms which will be discussed in the following paragraphs.

Initialization refers to creating a pool of potential solutions through initializing a population of chromosomes made up of genes. The fitness of the solution is calculated based on a function created that is applicable to the problem at hand.

Selection is then used to define which parents will contribute genes to the next generation. The parents are given a probability of being selected based on their fitness values and a random parent whose probability is above the threshold is selected.

Crossover is the process that allows parents to combine in a specified pattern in order to create new combinations of genes in a child chromosome.

Mutation is used to implement the idea of reproduction when a child gene has some different characteristics from their parents.

Elitism is the implementation of natural selection, where the fittest child and parent chromosomes are selected to move on to the next generation.

B. Fundamental GA Approach for Teaming

1) Method

In a previous study by the first author and colleagues [6], a typical GA workflow was used to solve the same problem. The chromosome for this method was an ordered list of students representing a specific team configuration. This list was ordered in a way that the first four students were team one, the second four were team two, and this repeated to maximize the number of four person teams. The remaining genes in the chromosome formed teams with three students.

2) Fitness Function

The fitness function was chosen to be the sum of the standard deviations for each skill in a chromosome. The pseudo-algorithm and history of the fitness function are shown in Fig.1 and 2.

For each skill, i = 1 to 3
  For each team, j = 1 to N_4
    For each member, k = 1 to 4
  $Sum\_skill_{i,j} = Sum\_skill_{i,j} + member\_skill_{i,k}$
    End
  End
  For *each team* $j = N_4 + 1$ *to* N_{tot}
    For *each member* $k = 1$ *to* 3
    $Sum\_skill_{i,j} = Sum\_skill_{i,j} + member\_skill_{i,k}$
    End
  End
    $Std\_dev_i = sqrt(\sum Sum\_skill_{i,j} / (N_{tot}-1))$
End
  Fitness function = $\sum$ Std dev$_i$

Fig. 1. Fitness function from fundamental GA approach [4].

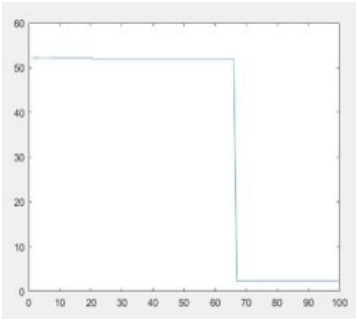

Fig. 2. Fitness function plot from fundamental GA approach [4].

3) Discussion

By using a conventional GA method, teams were able to be formed for the class quickly and easily. This method also had target criteria that was easy to define. The main issue with this method though comes from the fact that it was not designed to handle multiple criteria. It may not be able to detect

differences between high and low values of individual criteria which could lead to the solution not converging as quickly or not being able to find the optimal solution at all in some cases. To solve this problem several other multi-objective optimization algorithms like NASH equilibrium, parallel and distributed genetic algorithms were considered. Given the formulation of the problem and the nature of the three fitness functions, VEGA method was considered to be most appropriate.

C. VEGA Method

The vector evaluated genetic algorithm (VEGA) provides an improvement on the conventional genetic algorithm by changing the selection process. In this method, after a population is defined, the full population is split into subpopulations. The number of subpopulations defined is based on the number of criteria that are being optimized in the scenario. Once the subpopulations are defined, they are each ranked based on the fitness of one criterion. This means that each subpopulation will be optimized on only a single trait. Once the subpopulations are ranked, the lesser solutions of each subpopulation are thrown out to improve the characteristics of the overall population. Each of the filtered subpopulations are then recombined and the GA operators are then applied to create a population to send to the next generation.

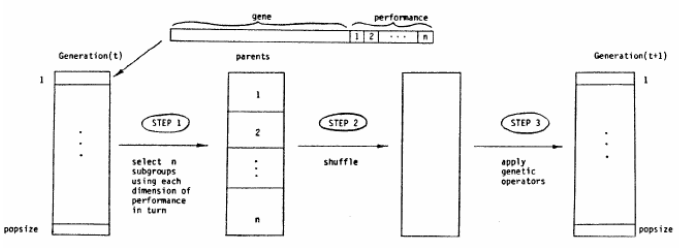

Fig. 3. Vector Evaluated GA approach [7].

IV. METHODOLOGY

A. Algorithm Structure

Inspired by the general structure of VEGA, this method approaches the teaming problem by splitting the general population into three subpopulations based on the three student skills (general programming knowledge, excel knowledge, and computer knowledge). For each subpopulation, the genetic operators of selection, crossover, mutation, and elitism are performed. The subpopulations are then recombined, and the fitness function is calculated for each chromosome. The algorithm tries to find a chromosome in the population where all three fitness functions are at their minimum in the current generation. In the absence of such a solution, the optimal solution is considered to be one with the minimum fitness function for programming skill. The algorithm is depicted in a flow diagram in Fig 4.

B. Selection criteria

A typical VEGA workflow involves applying genetic operators to the full population after the subpopulations are recombined. In this case the GA operators are instead applied on each optimized group. This can result in a process called inbreeding which is when extreme performance is favored by only allowing well performing chromosomes to crossover. This can be an issue for the VEGA method as a whole and not just for the case where GA operators are applied to the subpopulations. Since a typical VEGA method only sends the best of each subpopulation to recombine, the same issue can occur during recombination. Inbreeding can help solutions optimize faster than crossbreeding but only in cases in which there is a utopian solution available. In the case of teaming, there is likely not a utopian solution available, so in some sense the algorithm needs to also favor some less-extreme options since the best solution is likely not the best solution for each individual criterion.

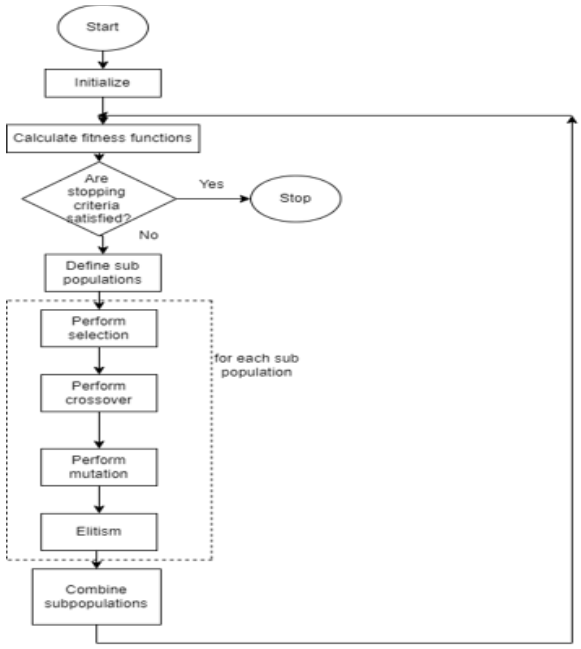

Fig. 4. Multi-Criteria Optimization Algorithm Block Diagram

Favoring non-extreme options is called crossbreeding and it is an important part of making the VEGA method find one optimal result. The algorithm defined in this paper encourages crossbreeding in two ways. The first of these methods is by randomly assigning the possibility of crossing over any member of the team with a consecutive team instead of just crossing two fixed members as was done in previous research [6]. The other way that crossbreeding is encouraged is by choosing to take the top thirty percent and the middle twenty percent of the subpopulation for crossover instead of just the top fifty percent. This ensures that the algorithm does not filter out potentially good solutions just because they are not extreme in one criterion.

C. Fitness Function

The most important adaptation to the fitness function from the conventional GA method was to have a fitness value calculated for each skill instead of combining the skills. This is the central goal of the VEGA as the fitness of each chromosome is now defined as a vector instead of a scalar value. The fitness function first calculates the average skill of each team. Then the standard deviation of the team averages for the skill is calculated. This is repeated for each skill.

D. Crossover

The crossover method for this algorithm was created differently from a typical GA crossover operator. The method used for this crossover technique was implemented to be more like a typical mutation GA operator. This method simulates a group genetic algorithm, and it avoids repeating members in the class and ensures that each child chromosome generates a different team configuration. In essence, the chromosome is essentially made of groups of alleles instead of a single allele in a typical GA. This algorithm performs crossover on all the teams by randomly generating an integer, i between 1 and 4, and swapping the current team's i^{th} member with the next team's i^{th} member. Figure 5 below shows an example of the first iteration of this algorithm with a team of four and a randomly generated integer of three. Figure 6 explains the process using a flow diagram.

Fig. 5. Representation of Teaming Crossover

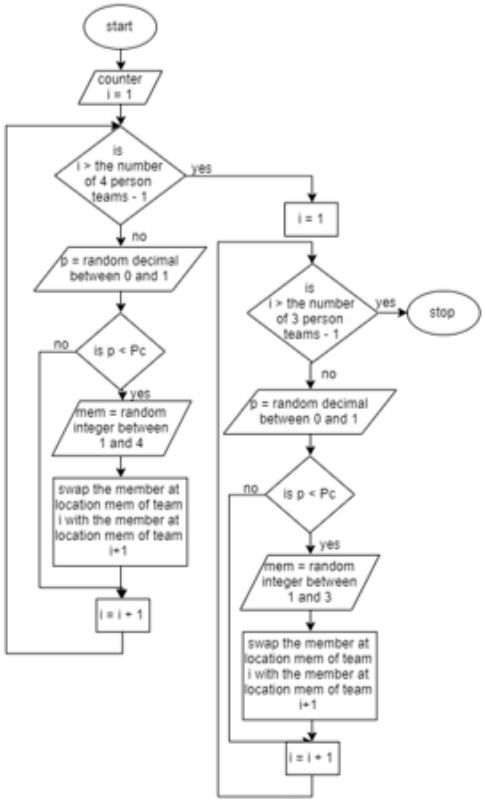

Fig. 6. Flow Diagram for Crossover

E. Mutation

Following the general structure of a vector modified genetic algorithm, this mutation method is implemented similarly to the previously explained crossover algorithm. This algorithm iterates through all the teams by swapping the current team's last member with the next team's first member. The mutation is performed separately on all groups of four before moving on to the groups of three for the sake of the algorithm. Figure 7 display an example of the first iteration of this algorithm where the fourth member of the first team is swapped with the first member of the second team. Figure 8 elaborates the flow diagram for this process.

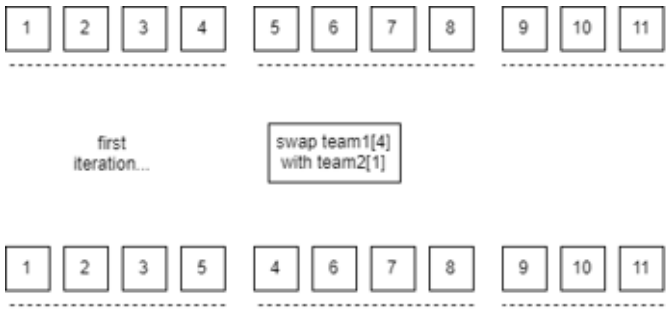

Fig. 7. Representation of Teaming Crossover

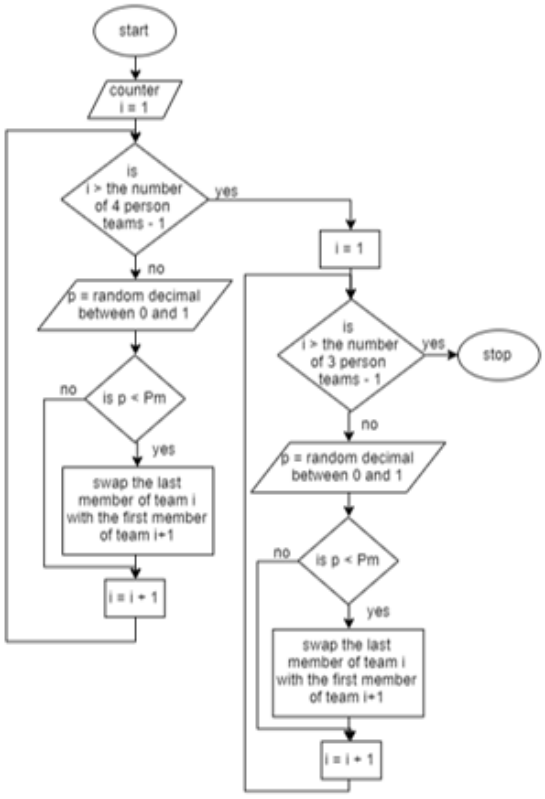

Fig. 8. Flow diagram for mutation

V. Results and Discussion

A. Team configuration

The algorithm generates a sequence of integers, each representing a student ordered in their teams. The user can take this sequence, and assign the first 4 students to team 1, next four to team 2, and so on. An example solution for a class of 47 students is given in Figure 9.

3 43 30 29 6 22 38 39 28 45 8 7 14 37 5 12 9 23 17 31 34 11 35 46 4 40 20 27 44 2 21 18 36 24 33 42 25 47 26 10 41 19 16 15 13 32 1

Fig. 9. Sample representation of teams in optimized configuration

B. Fitness functions optimization over generations

The following result plots show the fitness functions over each generation. It is seen that the Excel and Computer knowledge values fluctuate over generations while programming constantly decreases. This is evidence that algorithm favors minimizing the standard deviation of programming knowledge between teams over Excel and Computer. It should also be noted that the first few generations in each case have a value greater than 1. This corresponds to the configurations where the diversity was not optimized and the solution was penalized. However, the algorithm is able to converge to this optimization fairly quickly. It takes the algorithm about 18 seconds to run for 100 generations with a population size of 1500 chromosomes for a class of 47 students.

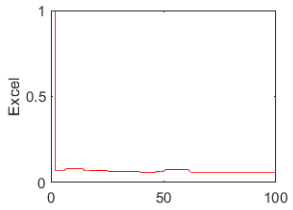

Fig. 10. Standard deviation of Excel Skills vs Generation

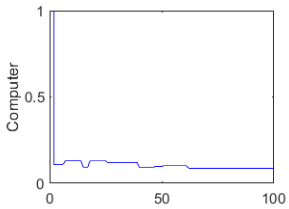

Fig. 11. Standard deviation of Computer Skills vs Generation

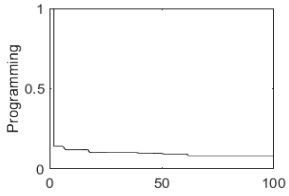

Fig. 12. Standard deviation of Programming Skills vs Generation

C. Comparison with manual assignment of teams

Figure 13 shows the spread of students in a class of 46 students based on their skills. The colors represent the clusters that students can be assigned into. Figure 14, then shows the spread of the team using their average in each skill, when assigned manually. Figure 15 shows the teams' averages in each skill using the algorithm configuration. Each data point in these two figures represent the average skills of each team in the class. It is evident that the algorithm reduces the spread of the skills between the teams when compared to the teams formed manually. This is shown by the points lying in close proximity to each other when compared to the manual assigned teams.

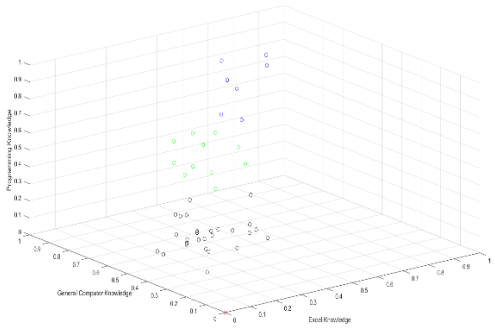

Fig. 13. Clusters showing spread of students in a class

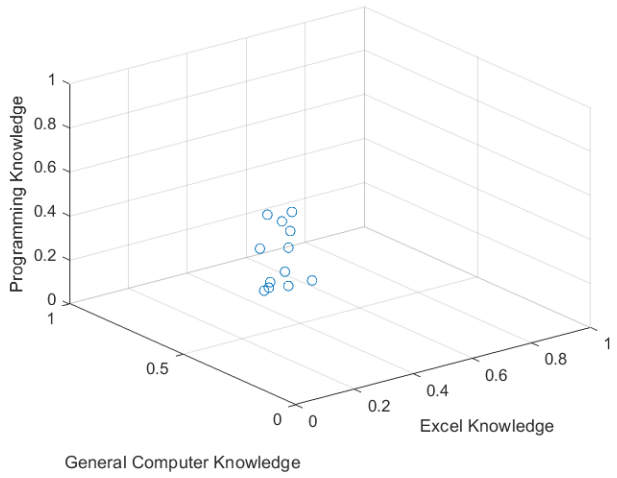

Fig. 14. Spread of average skills of teams generated manually

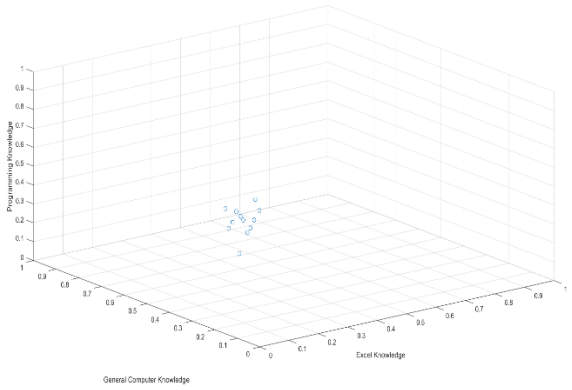

Fig. 15. Spread of average skills of teams generated by algorithm

VI. Conclusion and Future Work

This VEGA inspired method was able to produce better solutions in a shorter amount of time when compared to teams generated manually. In this work, all three skills were optimized simultaneously by implementing multi-criteria optimization. When the algorithm did not reach a utopian solution where each chromosome had a minimum fitness function for each criteria, the programming skill was given preference. This was decided due to a large majority of the course material requiring programming. With this decision, the results still converged within acceptable limits.

To further improve this method, the initialization function could be changed to initializing using clustering. Further cross-breeding between optimized results for each skill after combining into one population could also be implemented. With more cross-breeding, non-extreme options are favored in an attempt to include more potentially good solutions. Future work will involve removing bias in the algorithm to programming skills and introducing optimization based on rankings of fitness function for each of the skills.

The benefit of this algorithm is that it can generate optimal configurations of teams for large number of students in a very short amount of time. Use of standard operators in genetic algorithm also have the potential to use flexible optimization criteria in forming teams in classrooms.

Acknowledgment

The authors would like to thank Ms. Hannah Beals, Dr. Greg Bucks, Mr. Jeremy Hill, and Mr. Bayley King for all their help and support in the completion of this work.

References

[1] Engineering Accreditation Commission of ABET, "Criteria for Accreditation," ABET, Baltimore, MD, 1997.

[2] P. Paredes, A. Ortigosa and R. Rodriguez, "A Method for Supporting Heterogeneous-Group Formation through Heuristics and Visualization," *Journal of Universal Computer Science,* vol. 16, no. 19, pp. 2882-2901, 2010.

[3] S. A. Kalaian, R. M. Kasim and J. K. Nims, "Effectiveness of Small-Group Learning Pedagogies in Engineering and Technology Education: A Meta-Analysis," *Journal of Technology Education,* vol. 29, no. 2, 2018.

[4] S. G. Adams, "Building successful student teams in the engineering classroom," *Journal of STEM Education,* vol. 4, no. 3, 2003.

[5] R. Cavanaugh, M. Ellis, R. Layton and M. Ardis, "Automating the process of assigning students to cooperative-learning teams," in *Proceedings of the 2004 American Society for Engineering Education Annual Conference & Exposition*, Salt Lake City, Utah, 2004.

[6] P. Imbrie, J. Agarwal and G. Raju, "Genetic Algorithm Optimization of Teams for Heterogeneity," in *Frontiers in Education 2020*, Uppsala, Sweden, 2020.

[7] J. Schaffer, "Multiple Objective Optimization with Vector Evaluated Genetic Algorithms," in *Proceedings of the First Int. Conference on Genetic Algortihms*, 1985.

[8] V. Mishra and V. Singh, "Vector Evaluated Genetic Algorithm-Based Distributed Query Plan Generation in Distributed Database.," in *Proceedings of the International Conference on Recent Coognizance in Wireless Communication & Image Processing*, New Delhi, India, 2016.

[9] Academy of Management, "How Adding a Female's Voice Can Improve All-Male Teams," *Summary - Topics - Diversity,* pp. 1-2, 2019.

[10] United Nations Department of Economic and Social Affairs, "Chapter VII : Indigenous peoples and ethnic minorities: marginalization is the norm," in *Report on the World Social Situation 2018*, NewYork, United Nations, 2018, pp. 97-108.

[11] Bonnie Souza; Ken Neubeck, "Marginalized Voices in Eugene: Report on Focus Groupsin Eugene's Communities of Color, Muslim and LGBTQ Communities," Eugene Human Rights Commission, Eugene, 2017.

[12] S. Mirjalili, "Genetic Algorithm," in *Evolutionary Algorithms and Neural Networks*, Springer, 2019, pp. 43-55.