Zhamri Che Ani et. al. / (IJCSE) International Journal on Computer Science and Engineering
Vol. 02, No. 09, 2010, 3060-3064

A Method for Group Formation Using Genetic Algorithm

Zhamri Che Ani[1], Azman Yasin[2],
Mohd Zabidin Husin[3], Zauridah Abdul Hamid[4]
UUM College of Arts and Sciences
Universiti Utara Malaysia
06100, UUM Sintok, Kedah. Malaysia
[1]zhamri@uum.edu.my, [2]yazman@uum.edu.my, [3]zabidin@uum.edu.my, [4]zauree@uum.edu.my

Abstract— **Due to the increasing of complexity in software projects, group work is becoming more important in order to ensure quality software products can be delivered on time. Thus, in universities, group work is seen as a good preparation for students to enter industry because by working in group, it can reduce the individual workload, improve the ability to manage a project and enhance the problem solving skills. However, due to lack of programming skills especially in Java programming language, most of the students' software project cannot be delivered successfully. To solve this problem, systematic group formation is one of the initial factors that should be considered to ensure that every group consists of quality individuals who are good in programming. This paper presents a method for group formation using genetic algorithm, where the members for each group will be generated based on the students' programming skill.**

Keywords-group formation; genetic algorithm; programming skill

I. INTRODUCTION

Nowadays, due to the increased complexity of Information Technology (IT) projects, many IT organizations especially software industries are shifting away from individual work to group work environment [1]. Group work is becoming more important, because it can reduce individuals' workload and also can be used to support a variety of functions for an organization.

In academic institutions, group work has been seen as a good preparation for students to enter the industry because by working in group, they can improve their ability to manage and solve project problems efficiently. Most courses in a university normally adopt the group structure as a mean for students to share their knowledge, enhance problem solving skills and improve communication skills. However, not all student groups work well [2, 3]. One of the reasons is the groups are not systematically formed. Therefore, group formation is very important as a starting point for the group development and performance [4-6].

There is variety of group formation techniques have been investigated by researchers [7-12]. However, in normal practice, self-selection and random assignment of members are the most popular approaches used in group formation. Unfortunately, these two approaches are not useful in software development group because it will not speed up the development processes. One of the factors that need to be considered in order to produce a quality software product within the given period of time is good programming skill. Therefore, this paper focuses on how to form groups with balanced programming skills to ensure that every group members can complete the software project successfully.

This paper is organized as follows: In section 2 we discuss the method of group formation using genetic algorithm. In section 3 we describe a case study based on the real-world problem. Experiments and result are discussed in section 4 and 5. Section 6 includes conclusion and suggestion for future work.

II. METHOD

Genetic algorithms are search algorithms based on the mechanics on natural selection and natural genetics [13]. It has been applied by many researchers to solve various real-world problems [14]. It was also applied by Wang for solving heterogeneous groups to achieve fairness, equity and flexibility in group formation [15]. However, in this study we focus on group formation based on students' programming skill. The method is divided into two main phases: problem identification and theory building.

A. Problem Identification

Problem identification is the first step to conduct this study. Problem will usually have constraints on certain events that should be identified. In this study, the group formation problem consists of a set of students S = {s1, s2, s3, …., $s_{|s|}$} and a set of groups G = {g1, g2, g3, …, $g_{|g|}$}. The goal is to obtain balanced assignment, where five students in S are allocated to a group G based on programming skill. Programming skill levels are decided based on the examination result of STIA1023 Advanced Programming course, where Java programming language is taught.

B. Theory Building

Theory Building includes the development of methods or models. Chromosomes are typically represented as simple string of data and instruction. In this case, chromosomes

have been chosen to present a solution and students have been chosen as a gene. The algorithm used here was adapted from Goldberg [13, 16] and each step is discussed as below.

1) Initialization: Genetic algorithm begins with an initial population represented by chromosomes. A chromosome is a set of solution from one population. It can be taken and applied to new population. The expectation is that the new population (offspring) will be better than the old one. The offspring will be selected according to the degree of fitness. This solution is repeated until the best solution is achieved. Each chromosome represents a possible solution by sets of parameters. Sets of parameters are identified as genes and consist of fitness score values indicating the success or failure to fulfil all the constraints. Each gene is assigned with a random number to represent the students. For this study, the actual number of chromosome is generated by system prototype based on the total number of students for each class.

Fig. 1 shows the example of initial population of group formation. If there are 35 students in a class and the maximum number of members in a group is 5, then the number of generated groups is 7. In this case, group1 represented by Chromosome(1), group2 represented by Chromosome(2), group3 represented by Chromosome(3), and so forth. Genes are represented by random number, number 1 to 35, where it is equivalent to the number of students in the class. The fitness score of each group is calculated based on the grade of programming skill. The grades are categorized into three categories. Those who score A, A- and B+ (above 3.0) are categorized as good in programming whereas those who got D+, D and F (below 2.0) are categorized as poor in programming. The average grades (between 2.0 and 3.0) are considered as average students in programming.

2) Fitness Evaluation: The fitness of each individual chromosome must be computed when populations of chromosomes are generated. All chromosomes in one generation are evaluated by a fitness function. Each chromosome is compared against all the chromosomes for any constraint violation during the evaluation process. A penalty is given to a chromosome for each of the violated constraint. The penalty score from the constant value is subtracted to reduce the fitness value. The fitness of every chromosome in the population is obtained after the evaluation phase is complete. Fig. 2 shows the algorithm of fitness calculation based on programming skill.

Initial Population:

Chromosome(1) =>	09	34	19	06	07	Fitness: 30000
Chromosome(2) =>	26	13	08	15	25	Fitness: 30000
Chromosome(3) =>	05	23	27	22	31	Fitness: 30000
Chromosome(4) =>	20	35	28	16	01	Fitness: 20000
Chromosome(5) =>	03	11	30	32	24	Fitness: 20000
Chromosome(6) =>	04	33	10	21	12	Fitness: 30000
Chromosome(7) =>	29	02	17	14	18	Fitness: 30000

Total Fitness: 190000

Figure 1. Initial population of group formation.

G = Number of Group
GS = Number of **Good** Student
MS = Number of **Moderate** Student
PS = Number of **Poor** Student
MinGMP = Minimum Number of G|M|P in a group

MinGMP = (GS|MS|PS) / G

Fitness of grade in a group = 10000 if equal or greater than MinGMP; or
Fitness of grade in a group = 0 if less than MinGMP

Total of Fitness = MinGMP^{GS} + MinGMP^{MS} + MinGMP^{PS}

Figure 2. Algorithm of fitness calculation based on programming skill.

3) Reproduction: During reproduction, chromosomes are selected from a combination exists in the population. For Roulette Wheel selection algorithm, the higher fitness value represents the bigger parts of the wheel so that it will have a high probability to be selected several times in reproduction [13]. Crossover is the process in which two chromosomes combine their genetic material to produce a new generation that possesses both their characteristic. Many crossover techniques exist such as the one-point crossover, two-point crossover, 'cut & splice', uniform crossover, half-uniform crossover and others [13]. However, one-point crossover has been selected to be implemented in this case study. One random point is chosen to determine the crossover point. Then all the genes at the crossover point are copied from parents to offspring. As a result, these new chromosomes or offspring share some similar features taken from the parents. The genes after the crossover point are swapped between both parents. One-point crossover can be illustrated as Fig. 3. If the crossover is not applied, offspring are exact copies of parents. The crossover rate in this experiment has been set in range between 75 percent and 95 percent.

Mutation is the process used to maintain genetic diversity from one generation of a population of chromosomes to the next generation. The purpose of mutation is to allow the algorithm to avoid local minima by preventing the population of chromosomes from becoming too similar to

each other. A common example of a mutation operator involves a probability that an arbitrary bit in genetic sequence will be changed from its original state by generating a random variable for each bit. This random variable tells whether or not a particular bit will be modified.

Initially, each chromosome is given the chance to mutate to any sequence after crossover. However in this case study, any changes to the sequence will not affect the fitness value of the population. This is due to the same weight carried by each student in the same group. Therefore, mutation is not applied in the case of group formation.

Parent (1) =>	35	10	26	**07**	**31**
Parent (2) =>	28	30	22	**34**	**15**
Offspring (1) =>	35	10	26	**34**	**15**
Offspring (2) =>	**28**	**30**	**22**	07	31

Figure 3. One-point crossover.

III. A CASE STUDY

In this case, Roulette Wheel Method and students of STIW3053 Real-time programming of first semester 2009/2010 have been selected as the case study, and there are 35 students enrolled in this course. Based on the data obtained in the Real-time Programming class, the distribution of programming skill for STIA1023 Advanced Programming grade is shown in Fig. 4.

The bar graph shows that the distribution of grade for STIA1023 Advanced Programming is not equal; where only two students score A- and five students have B+. Grade B is the highest with seven students, followed by grade C+ with six students. Luckily, none of the students failed this course. The levels of programming skills for these results are summarized in Table 1.

Out of 35 students, 20.0 percent can be categorized as good in programming, 65.71 percent of students are considered moderate and only 14.29 percent are rated as poor. Based on the formula identified earlier, each group should consist of at least one good student and three moderate students.

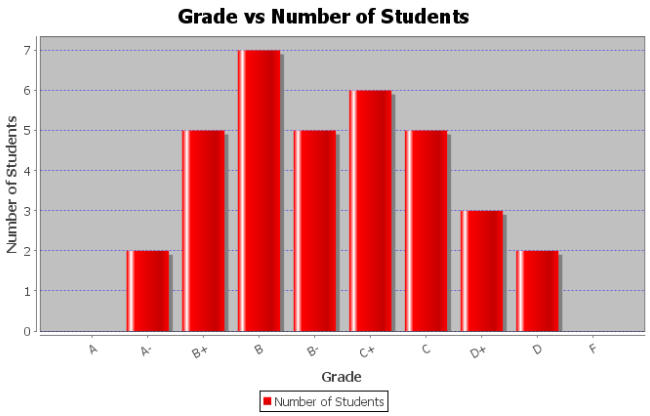

Figure 4. Distribution of grade of Advanced Programming.

TABLE I. LEVELS OF PROGRAMMING SKILL

Grade	Total Student (TS)	Percentage (%)	TS/ Group	Minimum Student(s)
A, A-, B+	7	20.00	1.00	1
B, B-, C+, C	23	65.71	3.29	3
D+, D, F	5	14.29	0.71	0

IV. EXPERIMENTATION

Forming optimal groups can be a time consuming and complex task [17]. To test the feasibility of the algorithm, a system prototype called QuickGroup was developed using Java programming language and several experiments have been conducted using a different set of parameters. Combination of number of generation and crossover rate should be obtained to achieve balanced solution.

For the first experiment, different number of population size has been explored. The sample of population size varied from 200 to 2000. Based on the first experiment, the number of population size that produced the highest fitness score was 1000, followed by 2000 and 1600. The highest fitness score recorded was 200000. Therefore, 1000 will be used as a population size for the next experiment.

For the second experiment, different number of crossover rate has been explored and the sample of crossover rate varied from 75 to 95. Based on the second experiment, the value of crossover rate that produced the highest fitness score was 90, followed by 95 and 75. The highest fitness score recorded was 190000. Therefore, 90 will be used as a crossover rate for the next experiment.

V. RESULT

Based on the experimental result carried out in this research, the best combination of parameters for generating groups in a class in this study is, Population Size =1000, Crossover Rate = 90. The final generation is shown in Fig. 5. To prove that each group has balanced programming skill and adhere to the specification determined earlier in this study, Table II shows the number of students for every level

of programming skill of each group. The result shows that every group consists of at least one good student and at least three moderate students.

To evaluate whether the groups formed using genetic algorithm approach perform better compared to manual group assignment performed in previous semesters, the result of the software products that was given to the students as their project work was checked in this study. The results show that out of seven groups, only one group failed to deliver the software product successfully on time. Our assumption, 85.71% of the groups can write java programs without errors. However, in practice, it is difficult to measure the individual ability systematically [18].

```
Generation: 1000
==========================================
Chromosome(1) => 08 17 29 33 30 Fitness: 30000
Chromosome(2) => 26 13 04 35 07 Fitness: 30000
Chromosome(3) => 19 06 05 22 27 Fitness: 30000
Chromosome(4) => 34 10 25 03 01 Fitness: 30000
Chromosome(5) => 28 09 11 32 24 Fitness: 30000
Chromosome(6) => 21 31 18 12 15 Fitness: 30000
Chromosome(7) => 16 02 14 20 23 Fitness: 30000
Total Fitness: 210000
```

Figure 5. Result of final generation.

TABLE II. RESULT OF GROUP FORMATION

Group	Good	Moderate	Poor
1	1	3	1
2	1	3	1
3	1	4	0
4	1	3	1
5	1	3	1
6	1	3	1
7	1	4	0
Total	**7**	**23**	**5**

VI. CONCLUSION AND FUTURE WORK

This research focused on group formation for IT or Computer Science students where programming skill is the most important criteria that have to be considered in order to form a solid group. In order to form balanced groups in a class, the genetic algorithm approach has been chosen in this study. This approach was applied in the STIW3053 Real-time Programming class, where all 35 students were required to give their previous results of STIA1023 Advanced Programming for the semester 2009/2010.

The results show that the genetic algorithm is a good optimizing method for the group formation. The method used in this study is capable to produce balanced group where each group consists of good, moderate and poor programming skills. In this case, we are hoping that weaker students will learn from stronger students how to solve programming problems while developing software applications. We will continue to enhance this work and our future work includes further systematic analysis of individual group performance.

ACKNOWLEDGMENT

The researchers acknowledge the financial support (Fundamental Research Grant Scheme) received from the Ministry of Higher Education, Malaysia via University Utara Malaysia (S/O Code: 11649).

REFERENCES

[1] J. Brown and G. Dobbie, "Supporting and evaluating team dynamics in group projects," in *Proceedings of the thirtieth SIGCSE technical symposium on Computer science education* New Orleans, Louisiana, United States: ACM, 1999, pp. 281 - 285.

[2] C. Chalmers and R. Nason, "Group metacognition in a computer-supported collaborative learning environment," in *Proceeding of the 2005 conference on Towards Sustainable and Scalable Educational Innovations Informed by the Learning Sciences: Sharing Good Practices of Research, Experimentation and Innovation*: IOS Press, 2005, pp. 35-41.

[3] D. W. Johnson and R. T. Johnson, "Making cooperative learning work," *Theory into practice*, vol. 38, pp. 67-73, 1999.

[4] K. Anewalt, J. A. Polack-Wahl, J. Beidler, and D. L. Smarkusky, "Group projects across the curriculum," *Journal of Computing Sciences in Colleges*, vol. 19, pp. 232-237, 2003.

[5] G. L. Stewart, "A meta-analytic review of relationships between team design features and team performance," *Journal of Management*, vol. 32, p. 29, 2006.

[6] C. E. Christodoulopoulos and K. A. Papanikolaou, "A group formation tool in an e-learning context," in *Proceedings of the 19th IEEE International Conference on Tools with Artificial Intelligence*: IEEE Computer Society, 2007, pp. 117-123.

[7] R. C. Haller, V. J. Gallagher, T. L. Weldon, and R. M. Felder, "Dynamics of peer interactions in cooperative learning," *Journal Engineering Education*, vol. 89, pp. 285-293, 2000.

[8] T. Daradoumis, M. Guitert, F. Gimenez, J. M. Marqu, and T. Lloret, "Supporting the composition of effective virtual groups for collaborative learning," in *Proceedings of the International Conference on Computers in Education*: IEEE Computer Society, 2002, p. 332

[9] I. A. G. Wilkinson and I. Y. Y. Fung, "Small-group composition and peer effects," *International Journal of Educational Research*, vol. 37, pp. 425-447, 2002.

[10] E. Martin and P. Paredes, "Using learning styles for dynamic group formation in adaptive collaborative hypermedia systems," in *Proceedings of the 1st International Workshop on Adaptive Hypermedia and Collaborative Web-based Systems*, 2004, pp. 188-198.

[11] S. Graf and R. Bekele, "Forming heterogeneous groups for intelligent collaborative learning systems with ant colony optimization," in *Proceedings of the 8th International Conference on Intelligent Tutoring Systems*, 2006, pp. 217-226.

[12] T. Lappas, K. Liu, and E. Terzi, "Finding a team of experts in social networks," in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*: ACM, 2009, pp. 467-476.

[13] D. E. Goldberg, *Genetic algorithms in search, optimization, and machine learning*: Addison-wesley Reading Menlo Park, 1989.

[14] E. Falkenauer, "Applying genetic algorithms to real-world problems," *IMA Volumes In Mathematics And Its Applications*, vol. 111, pp. 65-88, 1999.

[15] D. Y. Wang, S. S. J. Lin, and C. T. Sun, "DIANA: A computer-supported heterogeneous grouping system for teachers to conduct successful small learning groups," *Computers in Human Behavior*, vol. 23, pp. 1997-2010, 2007.

[16] D. E. Goldberg and K. Deb, "A comparative analysis of selection schemes used in genetic algorithms," *Foundations of genetic algorithms*, vol. 1, pp. 69-93, 1991.

[17] A. Ounnas, H. Davis, and D. Millard, "A framework for semantic group formation," in *Proceedings of the 2008 Eighth IEEE International Conference on Advanced Learning Technologies*: IEEE Computer Society, 2008, pp. 34-38.

[18] H. Wi, S. Oh, J. Mun, and M. Jung, "A team formation model based on knowledge and collaboration," *Expert Systems with Applications*, vol. 36, pp. 9121-9134, 2009.

AUTHORS PROFILE

Zhamri Che Ani (Corresponding author) is a lecturer at the College of Arts and Sciences, Applied Science Division, Universiti Utara Malaysia, 06010 UUM Sintok, Kedah. Malaysia (e-mail: zhamri@uum.edu.my). His research interest includes software engineering education, real time system and computer-supported heterogeneous grouping system.

Azman Yasin is a senior lecturer at the College of Arts and Sciences, Applied Science Division, Universiti Utara Malaysia, 06010 UUM Sintok, Kedah. Malaysia (e-mail: yazman@uum.edu.my). His research interest includes software engineering education, information retrieval specifically scheduling and timetabling using artificial intelligence techniques.

Mohd Zabidin Husin is a lecturer at the College of Arts and Sciences, Applied Science Division, Universiti Utara Malaysia, 06010 UUM Sintok, Kedah. Malaysia (e-mail: zabidin@uum.edu.my). His research interest includes software oriented architecture, search engine using artificial intelligence techniques.

Zauridah Abdul Hamid is a lecturer at the College of Arts and Sciences, Applied Science Division, Universiti Utara Malaysia, 06010 UUM Sintok, Kedah. Malaysia (e-mail: zauree@uum.edu.my). Her research interest includes software engineering education system and computer-supported heterogeneous grouping system.