UDC 004.4–004.9

PRIHOZHY A.A., ZHDANOUSKI A.M.

GENETIC ALGORITHM OF OPTIMIZING THE QUALIFICATION OF PROGRAMMER TEAMS

Belarus national technical university

Abstract. The partitioning a set of professional programmers into a set of teams when a programming project specifies requirements to the competency in various programming technologies and tools is a hard combinatorial problem. The paper proposes a genetic algorithm, which is capable of finding competitive and high-quality partitioning solutions in acceptable runtime. The algorithm introduces chromosomes in such a way as to assign each programmer to a team, define the team staff and easily reconstruct the teams during optimization process. A fitness function characterizes each chromosome with respect to the quality of the programmers partitioning. It accounts for the average qualification of teams and the qualification of team best representatives on each of the technologies. The function recognizes the teams that meet all constraints on the project and are workable from this point of view. It is also capable of recognizing the teams that do not meet the constraints and are unworkable. The algorithm defines the genetic operations of selection, crossing and mutation in such a way as to move programmers from unworkable to workable teams, to increase the number of workable teams, to ex-change programmers among workable teams, to increase the competency of every workable team, and thus to maximize the teams overall qualification. Experimental results obtained on a set of programmers graduated from Belarus universities show the capability of the genetic algorithm to find good partitioning solutions, maximize the teams' competency and minimize the number of unemployed programmers.

Keywords: *optimization, genetic algorithm, programmer, team, technology, qualification*

Introduction

A genetic algorithm is a meta-heuristic that simulates the process of natural selection [1, 2]. Genetic algorithms are capable of generating high-quality solutions to optimization problems by means of operations such as selection, crossing, and mutation. In a genetic algorithm, chromosomes represent candidate solutions to an optimization problem. Initial population of chromosomes is generated randomly. The main loop describes the evolution process, each iteration of which produces a new generation of chromosomes. The fitness of every chromosome is evaluated. The more fit chromosomes are randomly selected as parents from the current population, and new chromosomes are a result of recombination or mutation of the parent chromosome gens. The algorithm terminates when either a given number of generations has been produced, or a stagnation of population occurs.

Genetic algorithms successfully solve problems in many application fields. Work [3] applies a genetic algorithm to finding a valid and feasible path between two positions of the mobile robot, while avoiding obstacles and optimizing the distance, safety…etc. Work [4] proposes a new prototype of the smart vehicle parking system; a genetic algorithm addresses the issue of scheduling the vehicle to the parking bay. Paper [5] analyzes the genetic algorithm approach for graph coloring corresponding to the university timetable problem; the improvement of the initial solution is exhibited by experimental results. In [6], the authors propose a genetic algorithm for the dynamic airspace configuration; the obtained solutions outperform the existing airspace configurations. Work [7] solves the problem of dataflow pipeline optimization by introducing a genetic algorithm, which performs tuning of optimization heuristics.

Agile technology [8] aims at the flexible software development and finds solutions due to the joint efforts of development teams and customers. Many technological environments use Agile, but it requires further development for distributed programming teams. Agent-based [9] evolutionary optimization methods are capable of performing the management of teams. Work [10] formulated the problem of optimal partitioning of a given set of programmers into teams, and work [11] proposed an approach for the problem solving based on genetic algorithms.

This paper formulates the problem of programmer teams optimization, proposes a genetic algorithm of optimal partitioning a set of programmers into teams, introduces a fitness function for evaluating the quality of solutions found, proposes mechanisms of teams reconstruction based on genetic operations, shows the capability of generating competitive solutions.

Optimizing the size and staff of programming teams on qualification

Let $P = \{p_1,\ldots, p_n\}$ be a set of programmers, and $T = \{t_1,\ldots, t_m\}$ be a set of programming technologies. Let $Qualif(p) \in [0, 1]$ be a qualification level of programmer $p \in P$ with respect to the level of knowledge / competences in technologies of set T in comparison with the maximally feasible level. Qualification $Qualif(p)$ is a weighted sum of qualifications $Qualif(p, t)$ of programmer p on each technology $t \in T$. We use a questionnaire to obtain a value of $Qualif(p, t)$ of each programmer on each technology.

Let the set P of programmers be divided into k teams, which produce a partitioning set $G = \{g_1,\ldots, g_k\}$ such that $g_i \cap g_j = \varnothing$, $i \neq j$ and $g_1 \cup\ldots\cup g_k = P$. Programmers of team $g \in G$ constitute a set P_g. The number of programmers in g is n_g, The overall qualification of team g is $Qualif(g)$. It depends on both n_g and $Qualif(p)$ of each $p \in g$ in sophisticated nonlinear manner. Team qualification $Qualif(g)$ is an integral metric.

Given a partition G, we evaluate the overall qualification of teams of G with a sum of all teams' qualifications:

$$Qualification(G) = \sum_{g \in G} Qualif(g) \qquad (1)$$

Let Ω be a set of all feasible partitions of programmer set P into a set G of teams. The partition G has qualification $Qualification(G)$. It is easy to see, that the cardinality of set Ω grows exponentially of the size of set P. The goal of this paper is to develop a method of finding in Ω a partition, which maximizes the overall qualification:

$$\max_{G \in \Omega} Qualification(G) \qquad (2)$$

Usually we have to solve this task when several constraints on technologies, programmers and teams are given. The constraints as follows are usually associated with a particular programming project.

Constraint 1. It describes a lower level of qualification of programmer $p \in P$ regarding technology $t \in T$ for all programmers and all technologies.

Constraint 2. It describes a lower level of qualification of the best representative of team $g \in G$ regarding technology $t \in T$ for all teams and all technologies.

Constraint 3. It describes a threshold overall qualification of each team $g \in G$ over all demanded technologies.

Genetic algorithm of optimal partitioning a set of programmers into teams

The genetic algorithm (GA) implements a random process of evolution of a population of chromosomes (decomposition solutions) in order to find the best partitioning of the set of programmers into teams. We build a chromosome as a vector of genes that correspond to the programmers:

$$c = (h_1,\ldots, h_i,\ldots, h_n) \qquad (3)$$

where h_i is a gen, which represent a team number of G the programmer i belongs to. It is obviously that $\{h_1\} \cup\ldots\cup \{h_n\} = G$ and $h_1 \cup\ldots\cup h_n = P$. The set $P^i(c)$ of programmers chromosome c assigns to team i is

$$P^i(c) = \{\, j \mid h_j = i,\ j = 1\ldots n \,\} \qquad (4)$$

Therefore, chromosome c completely determines the staff of each team of G.

Since chromosome c describes a partition G, the partition can be characterized by parameter $Qualification(G)$. We consider the parameter as a value of the chromosome fitness function. The goal of GA is to find a chromosome with the maximal value of this function.

Figure 1 depicts a GA flow. Firstly, GA randomly initializes the population of chromosomes (programmer partitions) in such a way as to assign each programmer to an available team. The number of teams varies randomly for each initial chromosome in a certain range. Secondly, GA runs a loop each iteration of which produces a new generation of programmers partitioning by means of such genetic operations as selection, crossover and mutation. It uses the chromosomes of new generation to update the current population.

The selection operation chooses parents according to the rule of roulette wheel for performing crossing and mutation operations. GA uses

additional types of selecting chromosomes to produce the next generation of partitions and to update the population.

The crossover operation performs the recombination of two chromosomes, thus moving randomly selected programmers from one team to other team. The crossover yields two offsprings. As a result, a team can appear which includes no programmer. Such a situation may require the re-enumeration of teams. Additionally it requires facilities that are capable of extending the set of teams.

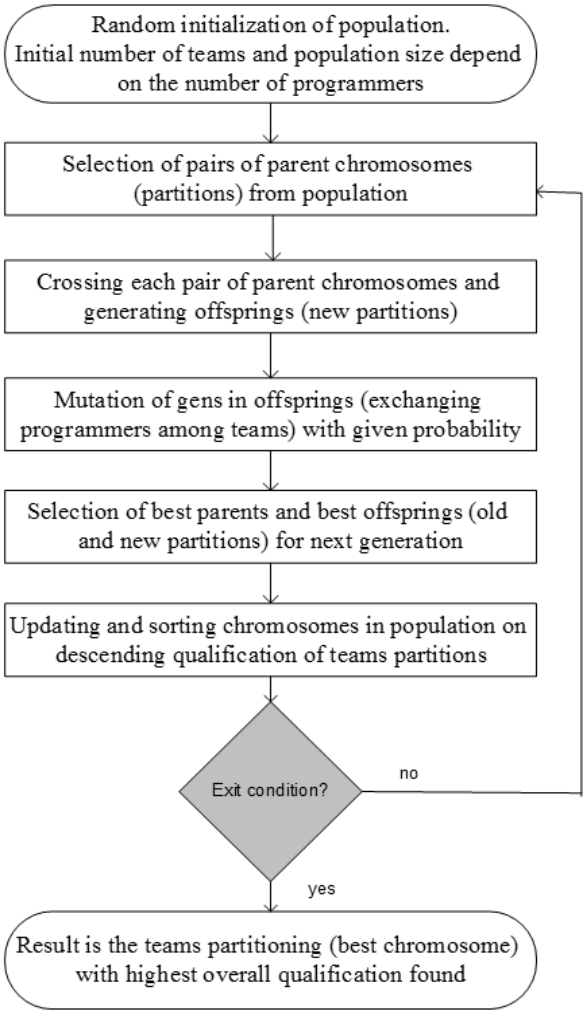

Figure 1 – Genetic algorithm of optimizing the partitioning of programmers into teams

The genetic mutation operation randomly chooses one or more programmers in a chromosome and moves them from one team to another. Each chromosome divides all teams into two classes. The first class includes teams that meet all three constraints described in the previous section. The second class includes teams that fail to meet at least one of the three constraints. We consider programmers of such a team as temporarily unemployed. The evolution process modeled by GA can potentially find a team for each programmer who has obtained a demanded qualification. We represent the loop exit condition via a constraint on the number of loop iterations or a constraint on the GA runtime. The chromosome with the highest value of fitness function is the solution of the optimization problem.

Chromosome fitness function that evaluates the quality of programmers partitioning

A hierarchy of formulas shown in Figure 2 defines a procedure of calculating the value of fitness function that characterizes the overall qualification of partitioning G of programmers into teams. According to (1), the overall *Qualification*(G) is a sum of the qualifications *Qualif*(g), $g \in G$.

The qualification *Qualif*(g) of team g is a weighted team qualification $Qualif^{w}(g)$ if the latter is equal or larger than a threshold qualification RQ^{g}, otherwise we consider team g as unworkable and exclude it from partitioning by zeroing *Qualif*(g). It is reasonable to take the value of RQ^{g} from the range [0.5, 1.0], depending on the requirements of the programming project.

We estimate the weighted qualification $Qualif^{w}(g)$ as the sum of a qualification $Qualif^{best}(g)$ of the best representatives on all technologies with a weight λ, and an average qualification $Qualif^{avg}(g)$ of the team's programmers on all technologies with a weight 1–λ. The weighted qualification can take any value of the range [0, 1]. The larger the value of λ, the higher the importance of the best representatives qualification is. The lower the value of λ, the larger the importance of the average qualification of the team programmers is. The qualification of best representatives describes opportunities for the growth of the average team competency.

We estimate the average qualification $Qualif^{avg}(g)$ of team g, that includes n_g programmers, as an average value of programmers qualification *Qualif*(p) over all programmers $p \in P_g$. The qualification *Qualif*(p) with respect to the level of competency in technologies of set T in relation to the maximum level of competency takes into account a rank of the technology, the competency

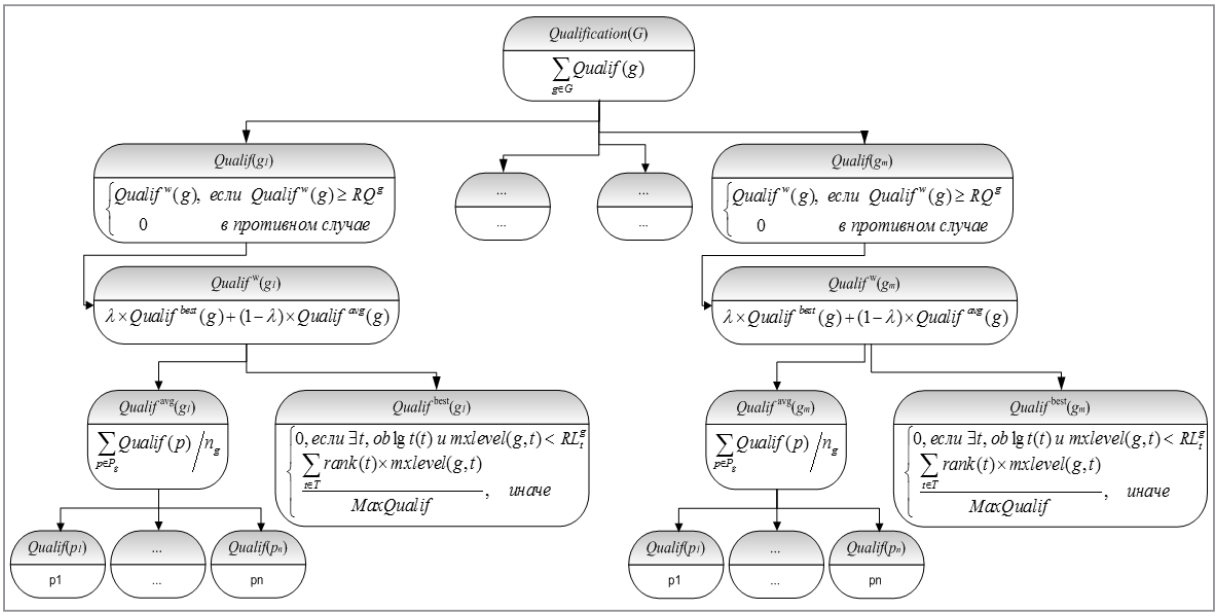

Figure 2 – Calculation of fitness function that evaluates the overall programmer teams qualification

of programmer in the technology, and a threshold value of the competency.

The best representative qualification $Qualif^{best}(g)$ is the most important parameter that characterizes team g, It equals zero if there is at least one mandatory technology for team g, for which the level of qualification of the best representative is less than a threshold value RL^g_t. The justification is that the team is not capable of carrying out projects without highly qualified programmers in key technologies.

Reconstruction of teams by genetic operations

Let us consider the process of crossing and mutation of chromosomes, which represents a repartitioning a set of programmers into teams. To do this, we use two example chromosomes, $c1$ and $c2$ chosen as parents at an iteration of the GA main loop. These chromosomes have the fitness function value of 2.548 and 2.551 respectively. Figure 3 depicts the process of chromosome reconstruction.

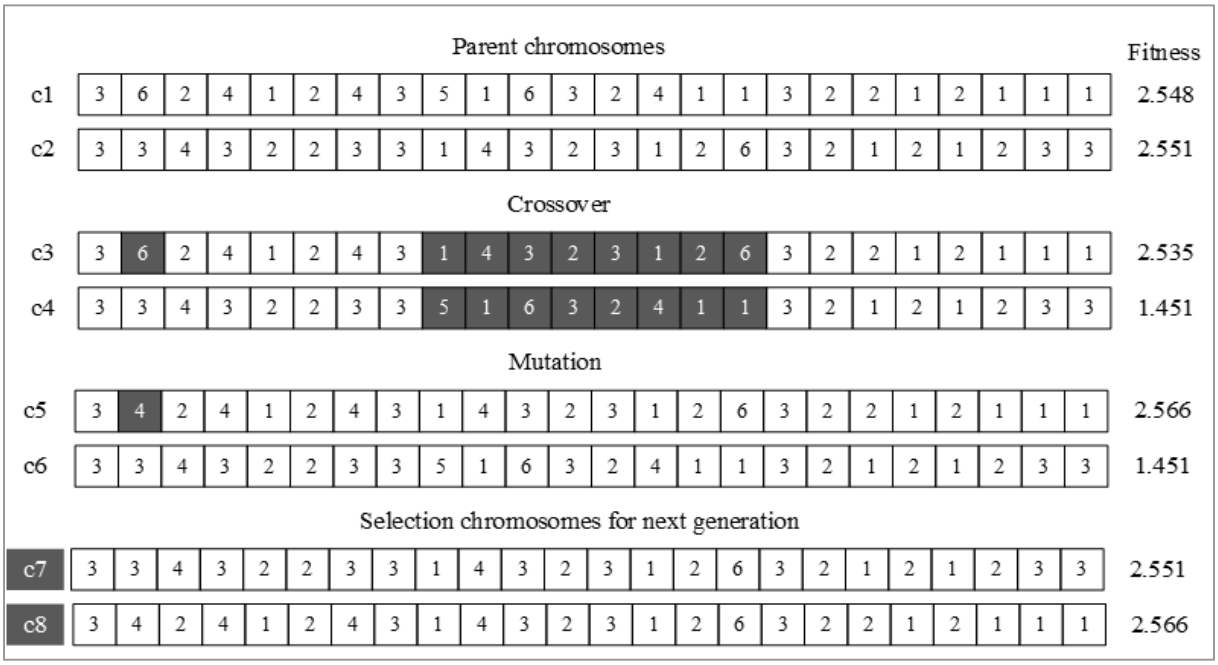

Figure 3 – Application of genetic operations to two parent chromosomes

Crossing consists in breaking two selected parental chromosomes and recombining the resulting chromosomal segments, which give a pair of offspring chromosomes. In the current version of GA, we use a two-point crossover. It divides each of the parental chromosomes into 3 parts (in Figure 3, parts 1 and 3 are in white, and part 2 is in black) and generates on their basis two offsprings *c*3 and *c*4 according to the rules 1–2–1 and 2–1–2 (numbers indicate parents of the parts). The fitness function value of chromosomes *c*3 and *c*4 is 2.535 and 1.451 respectively.

Mutations, i.e. random changes in chromosome gene values are intended to expand the search space when solving the optimization problem. If in a gene of the chromosomes of the initial population only a part of the possible values was generated, then the execution of the operators of crossing and selection cannot produce the values that have dropped out of consideration. The mutation operation is capable of creating new teams of programmers or combining existing ones. It modifies one of the crossover's offsprings with a certain probability. In Figure 3, the first offspring c3 underwent a mutation, as a result the value of second gene was changed from 6 to 4, which means that the programmer, *p*2 was moved from one group to another in chromosome c5. The fitness function of c5 has a value of 2.566. Chromo-some c6 is identical to chromosome c4 as no mu-tation has been performed.

GA selects a best parent chromosome and a best offspring in the next generation. Thus, the chromosome *c*2 with the overall teams' qualification of 2.551 becomes the first chromosome *c*7 that goes to the next generation. The best offspring *c*5 with the overall teams' qualification of 2.566 becomes the second chromosome *c*8 that replaces the worst parent chromosome *c*1 in the next generation.

Figure 4 illustrates the step by step repartitioning of the programmer teams induced by the genetic operations. Team 1 contains programmers not included in the workable teams enumerated starting from two; therefore, we call such a team as unemployed team. This team includes the

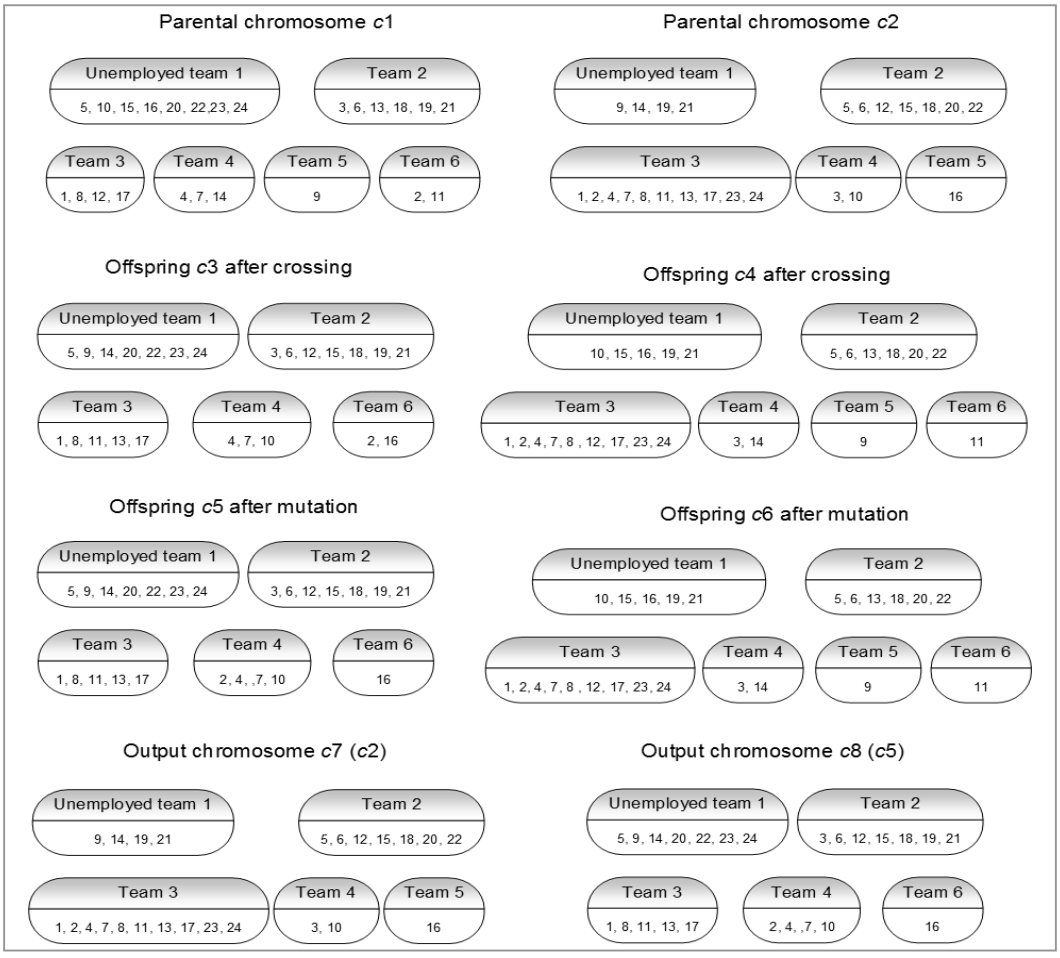

Figure 4 – Reconstruction of programmer teams by genetic operations

representatives of all teams defined by a chromosome the qualification of which do not meet the specified constraints. During functioning, GA may distribute programmers of the unemployed team among workable teams, if this will maximize the overall teams' qualification, and the qualification will be larger than the threshold level.

Experimental results

We have developed a computer program that implements the proposed GA, and have carried out computational experiments on the optimization of partitioning 24 professional programmers into teams. As many as 16 programming technologies used for estimating the qualification of programmers and teams. Questionnaires allowed for obtaining the level of each programmer competency in each of the technologies.

Table 1 reports experimental results obtained on eight runs of GA done at various values of threshold qualification of one team: 0.4, 0.45, 0.5, 0.55, 0.6, 0.65, 0.7, and 0.75. For each run, the key measured parameters are the number of work-able teams and the actual value of the overall teams' qualification. An additional important parameter is the number of programmers involved in the workable teams. The rest programmers are in reserve. The two key parameters allow for the calculation of the average team qualification. This qualification is always larger than the threshold team qualification. The increase of threshold qualification from 0.4 to o.75 with the step of 0.05 has given the number of workable teams of 9, 8, 8, 8, 6, 5, 3, and 2. It has also given the overall qualifications of 5.42, 5.05, 5.04, 5.01, 4.10, 3.48, 2.27, and 1.57 respectively. The number of teams and their overall qualification falls because of more severe requirements to the team qualification. As for the team actual average qualification, it remains al-most the same at the values of threshold qualification from 0.4 to 0.55, and then it grows rapidly reducing the number of teams essentially and increasing the number of unemployed programmers. Table 1 provides detailed information

Table 1. **Partitioning of a set of 24 professional programmers into teams by GA on 16 technologies**

Run	Team qualification constraint	Number of teams (programmers)	Overall qualification	Team average qualification	Teams count	Staff of teams
1	0.40	9 (22)	5.42	0.602	1–9	g_1={6,7,15,17,24}, g_2={11,20,23}, g_3={1,3,5,14},
						g_4={8,19,21}, g_5={18}, g_6={16}, g_7={22},
						g_8={2,12}, g_9={10,13},
					reserve	{4,9}
2	0.45	8 (23)	5.05	0.631	1–8	g_1={8,13,15,21,24}, g_2={2,5,6,9,14}, g_3={3,4,7},
						g_4={12,22}, g_5={10,11}, g_6={16,19},
						g_7={1,17,23}, g_8={18}
					reserve	{20}
3	0.50	8 (23)	5.04	0.630	1–8	g_1={4,11,17,20}, g_2={3,5,14,19}, g_3={7,9,13,22},
						g_4={1,12,15,24}, g_5={2,6,21}, g_6={18},
						g_7={8,23}, g_8={16}
					reserve	{10}
4	0.55	8 (21)	5.01	0.626	1–8	g_1={3,4,6,9,11,17}, g_2={5,19,21,24}, g_3={10,14},
						g_4={8,12,15}, g_5={1,7}, g_6={18},
						g_7={16}, g_8={13,22}
					reserve	{2,20,23}
5	0.60	6 (22)	4.10	0.683	1–6	g_1={2,4,7,9,14,16}, g_2={3,6,10,13,17,21,24},
						g_3={12,22}, g_4={1,5,8,23}, g_5={11,15}, g_6={18}
					reserve	{19,20}
6	0.65	5 (16)	3.48	0.696	1–5	g_1={1,2,4,5,8,14,15}, g_2={12,16,20}, g_3={10,11},
						g_4={19,22,23}, g_5={18}
					reserve	{3,6,7,9,13,17,21,24}
7	0.70	3 (13)	2.27	0.757	1–3	g_1={2,9,10,11,13,14,22,23}, g_2={7,12,15,16},
						g_3={18}
					reserve	{1,3,4,5,6,8,17,19, 20,21,24}
8	0.75	2 (6)	1.57	0.785	1–2	g_1={7,10,11,12,15}, g_2={18}
					reserve	{1,2,3,4,5,6,8,9,13,14,16,17,19,20, 21,22,23,24}

on the number of programmers and on the staff of each workable team. The programmers included in a small team are usually more qualified on average over all technologies. The programmers included in a large team are usually less qualified on aver-age or are highly qualified in restricted set of technologies.

GA has increased the overall teams' qualification within a single run of the algorithm by about 30 % against the best partitioning in the initial random population of chromosomes. It means the genetic operation are an effective facility of searching for an optimal solution.

In order to find an optimal partitioning of a large set of programmers, we develop a parallel version of GA. We use methods of work [12] to create an efficient parallel genetic algorithm for performing on parallel architectures.

Conclusion

Partitioning a set of professional programmers into workable teams is a hard combinatorial problem when the goal is to achieve the maximal overall technological competency while working on an IT project. The genetic algorithm we propose in the paper is capable of finding good solutions of the problem. Depending on the project constraints and on the set of participant candidates, the algorithm finds a preferable number of teams, the optimal size and staff of each team, which maximize the overall teams' qualification and minimize the number of skilled programmers not involved in the project. Experimental results obtained for programmers graduated from Belarus universities show that the proposed genetic operations efficiently recombine promising solutions and exhaustively scan the search space.

REFERENCES

1. **Barricelli, N.A.** Symbio genetic evolution processes realized by artificial methods / N.A. Barricelli // Methodos, 1957, pp. 143–182.
2. **McCall, J.** Genetic algorithms for modelling and optimization / J. McCall // Journal of Computational and Applied Mathematics, Vol. 184, 2005, pp. 205–222.
3. **Lamini, C.** Genetic Algorithm Based Approach for Autonomous Mobile Robot Path Planning / C. Lamini, S. Benhlima, A. Elbekri // Procedia Computer Science, Vol. 127, 2018, pp. 180–189.
4. **Thomas, D., Kovoor B.C.** A Genetic Algorithm Approach to Autonomous Smart Vehicle Parking system / D. Thomas, B.C. Kovoor // Procedia Computer Science, Vol. 125, 2018, pp. 68–76.
5. **Assi, M.** Genetic Algorithm Analysis using the Graph Coloring Method for Solving the University Timetable Problem / Assi, M., Halawi, B., Haraty, R.A. // Procedia Computer Science, Vol. 126, 2018, pp. 899–906.
6. **M. Sergeeva, D. Delahaye, C. Mancel, A. Vidosavljevic.** Dynamic airspace configuration by genetic algorithm / // journal of traffic and transportation engineering 2017; 4 (3): pp. 300–314.
7. **Prihozhy, A.A.** Heuristic genetic algorithm for computational pipelines optimization / A.A. Prihozhy, A.M. Zhdanouski, O.N. Karasik, M. Mattavelli // Doklady BGUIR, 2017, № 1, с. 34–41.
8. **Joshi, S.** Agile Development – Working with Agile in a Distributed Team Environment / S. Joshi // MSDN Magazine, 2012, Vol.27, No.1, pp.1–6.
9. **Müller, J.P., Rao, A.S., Singh, M.P.** A-Teams: An Agent Architecture for Optimization and Decision-Support, Proceedings 5th International Workshop, ATAL'98 Paris, France, July 4–7, 1998, pp. 261–276.
10. **Prihozhy, A.A.** Method of qualification estimation and optimization of professional teams of programmers / A.A. Prihozhy, A.M. Zhdanouski // System analysis and applied information science.– № 2.– 2018.– С. 4–12.
11. **Prihozhy, A.** Genetic algorithm of optimizing the size, staff and number of professional teams of programmers / A. Prihozhy, A. Zhdanouski // Open Semantic Technologies for Intelligent Systems: Research Paper Collection, Issue 3.– Minsk, BSUIR, 2019.– P. 305–310.
12. **Prihozhy, A.A.** Analysis, transformation and optimization for high performance parallel computing / A.A. Prihozhy // Minsk, BNTU.– 2019.– 229 p.

ЛИТЕРАТУРА

1. **Barricelli, N.A.** Symbio genetic evolution processes realized by artificial methods / N.A. Barricelli // Methodos, 1957, pp. 143–182.
2. **McCall, J.** Genetic algorithms for modelling and optimization / J. McCall // Journal of Computational and Applied Mathematics, Vol. 184, 2005, pp. 205–222.
3. **Lamini, C.** Genetic Algorithm Based Approach for Autonomous Mobile Robot Path Planning / C. Lamini, S. Benhlima, A. Elbekri // Procedia Computer Science, Vol. 127, 2018, pp. 180–189.
4. **Thomas, D., Kovoor B.C.** A Genetic Algorithm Approach to Autonomous Smart Vehicle Parking system / D. Thomas, B.C. Kovoor // Procedia Computer Science, Vol. 125, 2018, pp. 68–76.
5. **Assi, M.** Genetic Algorithm Analysis using the Graph Coloring Method for Solving the University Timetable Problem / Assi, M., Halawi, B., Haraty, R.A. // Procedia Computer Science, Vol. 126, 2018, pp. 899–906.

6. **M. Sergeeva, D. Delahaye, C. Mancel, A. Vidosavljevic.** Dynamic airspace configuration by genetic algorithm / // journal of traffic and transportation engineering 2017; 4 (3): pp. 300–314.
7. **Прихожий, А.** Эвристический генетический алгоритм оптимизации вычислительных конвейеров / А. А. Прихожий, А. М. Ждановский, О. Н. Карасик, М. Маттавелли // Доклады БГУИР, 2017, № 1, с. 34–41.
8. **Joshi, S.** Agile Development – Working with Agile in a Distributed Team Environment / S. Joshi // MSDN Magazine, 2012, Vol.27, No.1, pp.1–6.
9. **Müller, J. P., Rao, A. S., Singh, M. P.** A-Teams: An Agent Architecture for Optimization and Decision-Support, Proceedings 5th International Workshop, ATAL'98 Paris, France, July 4–7, 1998, pp. 261–276.
10. **Прихожий, А. А.** Метод оценки квалификации и оптимизация состава профессиональных групп программистов / А. А. Прихожий, А. М. Ждановский // Системный анализ и прикладная информатика.– № 2.– 2018.– С. 4–12.
11. **Prihozhy, A.** Genetic algorithm of optimizing the size, staff and number of professional teams of programmers / A. Prihozhy, A. Zhdanouski // Open Semantic Technologies for Intelligent Systems: Research Paper Collection, Issue 3.– Minsk, BSUIR, 2019.– P. 305–310.
12. **Prihozhy, A.A.** Analysis, transformation and optimization for high performance parallel computing / A.A. Prihozhy // Minsk, BNTU.– 2019.– 229 p.

ПРИХОЖИЙ А. А., ЖДАНОВСКИЙ А. А.

ГЕНЕТИЧЕСКИЙ АЛГОРИТМ ОПТИМИЗАЦИИ КВАЛИФИКАЦИИ ГРУПП ПРОГРАММИСТОВ

Аннотация. Разбиение множества профессиональных программистов на множество команд, когда программистский проект определяет требования к компетенциям в различных технологиях и инструментах программирования, представляет собой сложную комбинаторную проблему. В статье предлагается генетический алгоритм, который способен находить конкурентоспособные и высококачественные решения по разбиению за приемлемое процессорное время. Алгоритм вводит хромосомы таким образом, чтобы распределить каждого программиста в команду, определить состав команд и легко реконструировать команды в процессе оптимизации. Функция приспособленности характеризует каждую хромосому с точки зрения качества разбиения программистов. В ней учитывается средняя квалификация команд и квалификация лучших представителей команд по каждой из технологий. Функция распознает команды, которые удовлетворяют всем ограничениям проекта и являются работоспособными с этой точки зрения. Она также способна распознавать команды, которые не соответствуют требованиям и не являются работоспособными. Алгоритм определяет генетические операции отбора, скрещивания и мутации таким образом, чтобы перемещать программистов из неработоспособных команд в работоспособные, увеличивать количество работоспособных команд, обмениваться программистами между работоспособными командами, повышать компетентность каждой работоспособной команды, и, таким образом, максимально увеличивать общую квалификацию команд. Экспериментальные результаты, полученные на выборке программистов, окончивших вузы Беларуси, показывают способность генетического алгоритма находить хорошие решения для разбиения, максимизировать компетенцию команд и минимизировать количество не работающих программистов.

Ключевые слова: *оптимизация, генетический алгоритм, программист, команда, технология, квалификация.*

Anatoly Prihozhy received his Diploma of Electrical Engineering from the State Polytechnic, Minsk, Belarus in 1975, his PhD degree in computer-aided design from the National Academy of Sciences Minsk, Belarus in 1984, and his Doctor Habilitation degree in computer sciences from Ukraine, Kyiv and Belarus, Minsk in 1999. His research interests include programming, hardware and system description languages, compilers and tools, system-, high- and logic-level computer aided design and optimization of parallel and incompletely specified digital systems

Zhdanouski Arseni is a postgraduate of the Computer and system software department of Belarusian national technical university, and a software engineer at EPAM Systems. His research interests include programming languages and techlogies, and methods of optimization.