Management of Software Project using Genetic Algorithm

Milena Karova[1], Nevena Avramova[2], Ivaylo Penev[3], Yulka Petkova[4]

Abstract –**This paper presents a heuristic method – genetic algorithm to solve the Project Management Problem. The problem is complex, NP complete. The objectives are to minimize the project duration and to minimize the project cost. The constraints are that each task must be performed by at least one person and every person must have a set of knowledge. The algorithm must define the degree of dedication of each employee. The Genetic Algorithm (IGAPM) proposes a binary chromosome encoding, single crossover, two types of selection and flip-bit mutation.**

Keywords – **Project Management Problem, Genetic Algorithm, chromosome, fitness function, project cost, constraints.**

I. INTRODUCTION

The software projects consist of interrelated activities with a set of necessary skills to perform them. The activities should be performed as much as possible at lower costs and less overlap utilizing available resources (staff with skills). Project activities and resources need to be organized in such a way that the project duration and costs are minimized and the project quality is maximized.

The presented algorithm uses as instance an implementation of software project. Project activities steps are: specification (making specifications according to customer requirements), programming (individual characteristics), architecture (defining the system architecture) and interface testing. The knowledge as a resource that is not quantified a number of related activities and employees. Any employee involved in the project implementation has a set of knowledge (skills), enabling its participation in the project. The project cost includes the employee's salaries. These are direct costs. Indirect costs of a project, such as licenses, rent, equipment and more are not included in the project.

II. THE PROJECT SCHEDULING PROBLEM

A. Definition of the Project Problem

The software project consisting of five activities Ak, which are common in projects of this type. The resources to the project are employee with skills. The links between the activities are presented in Fig.1.

The skills (knowledge) necessary for the project was presented as a set: knowledge = {C #, MySQL, PHP, Network, Design, PMR}. PMR (project management roles) is the allocation of roles in project management.

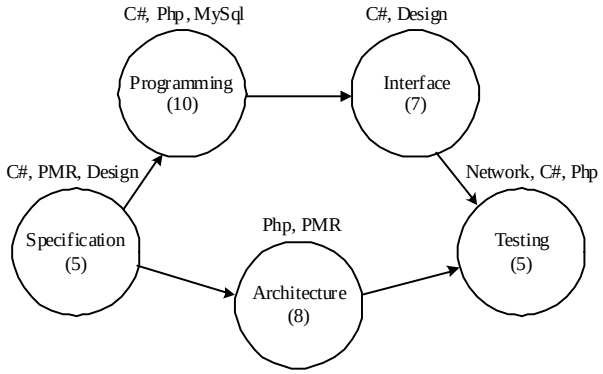

Fig. 1. Software Project Activities

Define the project P_V = {Ak1, Ak2, Ak3, Ak4, Ak5} [Table 1].

TABLE I
PROJECT ACTIVITIES

Activities	Name Ak_i^{title}	Duration (days) Ak_i^{t}	Activities precedents Ak_i^{pred}	Skills Ak_i^{know}
Ak1	Specification	5	no	C#, PMR, Design
Ak2	Programming	10	Specification	C#, Php, MySql
Ak3	Architecture	8	Specification	Php, PMR
Ak4	Interface	7	Programming	C#, Design
Ak5	Testing	5	Interface, Architecture	Network, C#, Php

There is a set of people working on the project P_V = {E1, E2, E3, E4, E5} [Table 2]

The duration of activity tj in the project P_V depends on Akj dedication-part-time percent of employees (which is dependent on knowledge) [1]. The duration and the other activities of the project are: $t_1 = 3$, $t_3 = 4$, $t_4 = 3$ and $t_5 = 2$. The activities supplying on critical path are determined by the

[1]Milena Karova is with the Department of Computer Science and Technologies at Technical University of Varna, 1 Studentska str, Varna 9010, Bulgaria, E-mail: mkarova@ieee.bg.

[2]Nevena Avramova is a master student with the Department of Computer Science and Technologies at Technical University of Varna, 1 Studentska str, Varna 9010, Bulgaria, E-mail: nevenka@abv.bg

[3]Ivaylo Penev is with the Department of Computer Science and Technologies at Technical University of Varna, Email: ivailopenev@yahoo.com

[4] Yulka Petkova is with the Department of Computer Science and Technologies at Technical University of Varna Email: jppet@abv.bg

critical path method. For P_V project there are: specification, programming and interface testing.

B. Evaluation Functions

The project duration is determined by the critical path. Many of the activities involved in the critical path t_{cr} [Eq.1] denote by CPA (Critical Path Activities) [4].

$$t_{cr} = \sum_{j \in CPA} \frac{Ak_j^t}{\sum_{i=1}^{Z} m_{ij}} \quad (1)$$

TABLE II
PROJECT STAFF

Employee	Name E_i^{name}	Salary (leva) E_i^{sal}	Project Dedication (%) E_i^{comm}	Knowledge E_i^{know}
E_1	Ivan Ivanov	1100	100	C#, MySql, Php
E_2	Georgi Vasilev	900	100	Php, Network
E_3	Gergana Kancheva	1100	50	Php, PMR, Design
E_4	Veselin Georgiev	900	100	Design, PMR, Network
E_5	Emilia Avramova	1000	100	C#, MySql

$$t_{pr} = t_{All} - t_{cr} \quad (2)$$

$$t_{All} = \sum_{j=1}^{A} \frac{Ak_j^t}{\sum_{i=1}^{Z} m_{ij}} \quad (3)$$

$$Oc_M = u_{cr} * t_{cr} + u_{cost} * S_{cost} + u_{pr} * t_{pr} \quad (4)$$

Eq.2 is an overlap time. Eq.3 is the full project duration. Eq.4 is the fitness function. u_{cr}, u_{cost} and u_{pr} are weight parameters and m_{ij} is a employee's part-time of the project activities.

III. THE GENETIC ALGORITHM IMPLEMENTATION

A. Algorithm Description

IGAPM (Implementation Genetic Algorithm for Project Management) is the realization of genetic algorithm for software project management. The algorithm is a part of a group planning algorithms and it is defined as a tool for planning projects (in particular, and software projects). The application provides the ability to manage projects so that project will be completed at - short term and / or minimal cost and with minimal overlapping activities.

IGAPM provides: an interface enabling the user to implementation of projects and their saving in format XML; GA management through its basic and additional parameters to obtain the optimal solution for each run of the GA (the decision to submit via charts) and removal of most - good fitness function.

IGAPM application is realized by programming language C# development environment and Visual Studio. Net 2008. For plotting graphs is using MS Chart VisualStudioAddOn.exe. The implementation of the algorithm realizes the critical path method based on the literature [4].

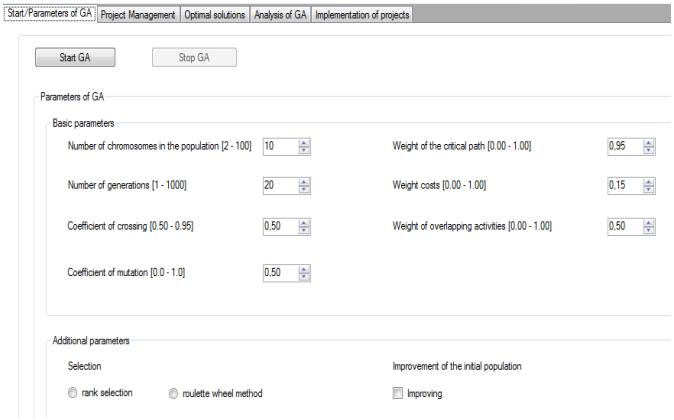

Fig. 2. IGAPM Interface

B Algorithm Pseudo code

Start
size = size_generation, i = 0
Create a population *P (i)* based on input data: *Knowledge Employees* P_V.
Evaluation of the population *P (i),* using parameters u_{cr}, u_{cost} and u_{pr}.
Repeat while (*i <size_generation*) or (interrupted by the user)
Selection (*sel, P (i)*); result *parent_1* and *parent_2*.
Crossover (*parent_1, parent_2*); result *child_1* and *child_2*
rnd = random number in the range 0 to 100.
if (*rnd > Pm*) **Mutation** (*child_1, mut probability*) end if
Evaluation *child_1 child_2*, using parameters u_{cr}, u_{cost} and u_{pr}.
Addition *child_1 and child_2* in *P (i).*
i = i + 1;
Repeat
End
Output at - optimal solution R

C. Chromosome encoding (matrix presentation)

IGAPM proposes matrix presentation (M) of chromosome. The value of m_{ij} gene is represented by one of eight values [1], approximately evenly distributed in the interval [0,1]. So each gene is described by three bits. The meanings of the bits are examined from left to right. The first bit in the unit ie 100 presented positive value 0.56 in the second bit in the unit ie 010 is positive 0.28, third in a bit that has value 001 is rounded to 0.16. If one bit is zero, therefore the value that represents is zero. The binary gene 000 is set to 0 and a gene with a record 111 is set to 1. Each gene is able to adopt the following actual values: 0, 0.16, 0.28, 0.44, 0.56, 0.72, 0.84 and 1. These values are approximate percentage of employee's project part-time (performance of Ei). Akj activity and project managers are responsible for employee's supervising in the organization. On Fig. 3 it is shown a sample format of chromosome M.

TABLE III
CHROMOSOME EXAMPLE

M	Specification	Programming	Architecture	Interface	Testing
I.Ivanov	0,44	0,84	0,28	0,84	0,84
G.Vasilev	0	0,16	0,84	0	1
G.Kanaheva	0,16	0,28	0,44	0	0,16
V.Georgiev	0,56	0	0,56	1	0,56
E.Avramova	0,28	1	0	0,16	0,28

The size of the chromosome can be represented by size | Ak | * | EW | * 3. IGAPM used chromosome composed of genes with a record length of size three. Binary record length 3 provides sufficient values to describe the percentage of employee's dedication. The binary format allows diverse describe of employee's project part-time. The bit increasing will significantly increase the solving time.

D. Initial Population, Selection, Crossover and Mutation

Creating the initial population:
Start
Size_population = population size
Establish population *P(0)* by the number of chromosomes in the population size_population (set by the user interface)
while (*size_population!* = 0) repeat
Create new chromosome corresponding to any restrictions
Normalization of chromosome
Calculation of fitness function for chromosome Oc_M
if (one or more activities have assigned staff)
$Oc_M = Oc_M * 100$
end if
The new chromosome is added to *P(0)*
end repeat
End
Output: population *P(0)*

The main purpose of selection is to obtain suitable parents for crossover, to get children with a good fitness function. IGAPM implementations are realized two methods of selection, determined by the user interface. Both methods select two parents for crossover. The methods of selection are: rank selection and method of "roulette."

For crossover IGAPM uses (two parents and two children) one - positional crossover (k = 1) that supports the rules for chromosome's composing. The probability of crossover is determined by the user interface.

IGAPM uses a mutation in Invert bit (Flip Bit). In random it turns one bit of the gene in the chromosome. The mutation is applied in random order on the chromosome and is applied for each gene from the choose line. The mutation probability (Pm) is determined by the user interface.

The last step of each generation is performing update of the population. IGAPM uses method for removing the poorest chromosomes. To the next population of children O(I) only good individuals are transferred. The chromosomes with poor fitness function are removed [3].

The program is completed by two ways [2]. One is when the algorithm reaches the maximum number of generations, set by the user and the second way is forcibly stopped by the user.

IV. TESTING AND RESULTS

There are a variety of methods for evaluating and experimenting with the Genetic Algorithms. There is no uniform methodology for testing and evaluation. Most studies include subjective evaluations based mainly on tasks that they solve. The standard approach exists. It analyzes and evaluates the Genetic Algorithm, changing the following parameters: number of generations, number of chromosomes in the population, crossover and mutation probability, type of selection and improvement of the initial population.

The behavior of the GA is defined on Fig. 5 and Fig.6. For finite number starts with a constant configuration of genetic parameters and genetic operators, the variations of the values of fitness function are relatively constant.

The analysis is based on the different number of GA generations. Experiments were performed with common parameters: number of chromosomes in the population = 10, crossover rate Pc = 0.65, mutation rate Pm = 0.55, weight of the critical path $u_{cr} = 0.95$, weight cost $u_{cost} = 0.15$ and weight of overlapping activities upr = 0.5. Additional parameter is the type of selection - by rank.

TABLE IV
RESULTS- NUMBER GENERATIONS/ FITNESS FUNCTION

Generations	Best fitness	Worst fitness	Average fitness	Project cost
20	17,35	23,75	19,157	1058
150	17,39	26,1	17,801	1065
400	**17,11**	27,57	17,262	1018
600	**17,51**	23,52	19,03	**1009**
1000	17,25	25,05	17,44	**1041**

The user interface allows for changing the number of generations in the interval [1, 1000]. The number of generations is one of the important factors defining the duration of algorithm.

Table IV shows that the increasing the number of generation has limit. The great number of generation couldn't improve a good value of fitness function. It depends on the specificity of current problem (activities' number, number of employees and etc.)

Fig. 3 shows the actual duration as result of IGAPM for 20 generations. The project duration is less than the initial (given duration). The project cost is 1058.

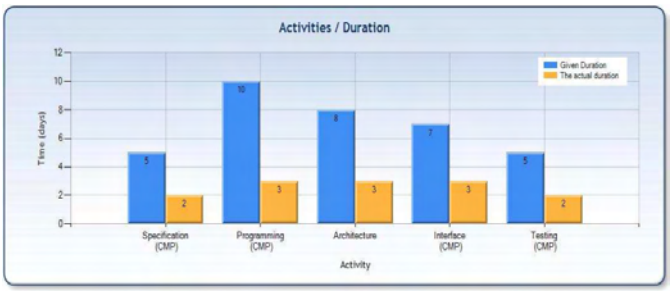

Fig. 3. Given Duration/ The Actual Duration (20 generations)

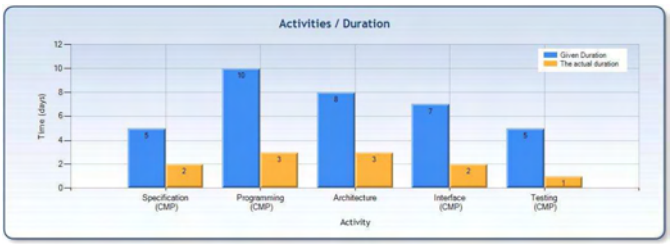

Fig. 4. Given Duration/ The Actual Duration (600 generations)

Fig. 4 is the graphical view of the best result of Tabl. 4. The project cost is 1009.

The evolution process of genetic algorithm is shown on charts Fig.5 and Fig. 6. Genetic algorithm does not allow increasing of the fitness function. It eliminates bad chromosomes with higher fitness function. This ensures a reduction of the fitness function over time.

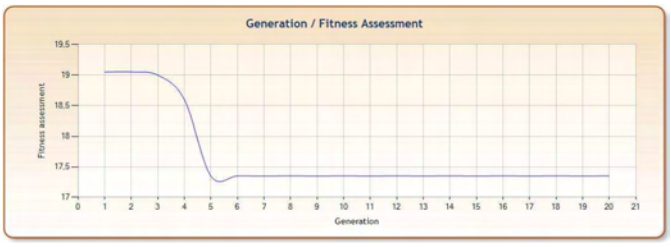

Fig. 5. Generations/ Fitness function (20 generations)

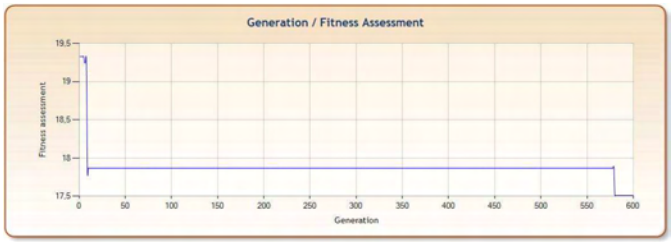

Fig. 6. Generations/ Fitness function (600 generations)

The Figures show that regardless of the number of generations the function finds its optimal solution for a small number of generations and their further increase is not necessary. The role of mutation or other factors is important in order to escape the algorithm from local minimum [Fig. 6].

Standard approach to research and evaluate the genetic algorithm does not exist. There is no exact number of algorithm executions. The optimal result depends on the specific task, on use of genetic operators and genetic parameters and on the number of generations.

The complexity of the algorithm can be calculated by considering the worst case |Ak| = |Ew|. If n = |Ak| therefore chromosome has n^2 genes. Any operation performed on the chromosome was assumed to be performed per unit time. If you refer to s_0 population size and number of generations is *i*. Complexity is Eq.5.

$$O\left(\sum_i s_i * n^2\right) \quad (5)$$

V. Conclusion

Using Genetic Algorithm (GA) for resolving Project Management Problem is one of the advanced applications of heuristic algorithms for automation. The GA is a good decision for resolving the conflicts (resources and costs, project duration and employee's knowledge).

The transformation of the actual requirements into chromosomes is a hard phase and the important factor for GA success. It has observed that the optimal results can be achieved if the GA is trained in further.

The future work involves a testing of the algorithm with various instances to improve the dominance of the project resources on genetic algorithm evolution process.

References

[1] E. Alba, J. Francisco Chicano, Software project management with Gas, Information Sciences 177, An International Journal, pp. 2380-2401, 2007

[2] S. Barthelemy, Principe general des algorithmes genetiques, www.barth.netliberte.org, 2000

[3] H. Sonke, A Self-Adapting Genetic Algorithm for Project Scheduling under Resource Constraints, Naval Research Logistics, John Wiley&Sons Inc, pp49:433-448, 2002

[4] http://www.pmipr.org/html/Presentations/ The Critical Path Method.pdf