UDC 004.421.4:005.8
Original Scientific Paper

Using Coevolution Genetic Algorithm with Pareto Principles to Solve Project Scheduling Problem under Duration and Cost Constraints

Budylskiy Alexandr Victorovich *alexandrvictor88@mail.ru*
Faculty of Informational Technologies and Communications
Astrakhan state technical university, Astrakhan, Russia

Kvyatkovskaya Irina Yuryevna *i.kvyatkovskaya@astu.org*
Faculty of Informational Technologies and Communications
Astrakhan state technical university, Astrakhan, Russia

Abstract

This article considers the multicriteria optimization approach using the modified genetic algorithm to solve the project-scheduling problem under duration and cost constraints. The work contains the list of choices for solving this problem. The multicriteria optimization approach is justified here. The study describes the coevolution approach together with Pareto principles, which are used in the modified genetic algorithm. We identify the mathematical model of the project-scheduling problem. We introduced the modified genetic algorithm. The article includes the example.

Keywords: project, project management, project scheduling, multicriteria optimization, coevolution, Pareto principles, genetic algorithm

1. Introduction

Project scheduling is an integral part of project management in IT. Decisions made in this stage can influence on the whole project. Since the incorrect decision-making in resource allocation can lead to breach of project term. The reduction of the project cost and duration are the critical goals for many IT companies. Some companies often employ developments only for project development. Each developer requires fixed wage and as a rule, the better developer requires the greater wage.

The project-scheduling task under duration-cost trade-off is a typical task of multi-objective optimization. The article proposes a solution of this problem by means of the modified genetic algorithm proposed by Goldberg [3] using coevolution [10] and Pareto Principles. The following things should be done to reach the solution:

1. Formulate the multi- objective optimization task in general form.
2. Formulate the project-scheduling task.
3. Consider the ability of using problem solution alternatives.
4. Describe the modified genetic algorithm for project scheduling task.

2. Multi-objective optimization

The multi-objective optimization [7] is the tuple <x, z, g>:

$$\begin{cases} x = \{x_1, ..., x_n\} \in X \\ z(x) = \{z_1(x), ..., z_k(x)\} \\ g(x) = \{g_1(x), ..., g_m\} \end{cases} \tag{1}$$

x – Decision variable vector.
X– Solution space.
z –Objective functions (optimality criterions).
g- Constraints.

For multiple-objective problems [9], the objectives are generally conflicting, preventing simultaneous optimization of each objective hence, optimizing x with respect to a single objective often results in unacceptable results with respect to the other objectives. The Pareto principals [7] are using for these purposes

If all objective functions are for minimization, a solution x is said to dominate another solution y (x ϕ y), if and only if:

$$\forall i, z_i(x) \le z_i(y) \wedge \exists i, z_i(x) < z_i(y) \tag{2}$$

A solution is called Pareto optimal, if it is not dominated by another solution in the search space. The list of all non-dominated solutions is calls the Pareto optimal set and it forms the Pareto Front.

The main goal of a multi-criteria optimization algorithm is to determine solutions in the Pareto optimal set. However, it is very difficult to identify the whole Pareto optimal set due to its size. It is also computationally infeasible for many cases. Therefore, a practical approach is to determine the approximate solution (the best-known Pareto set) that represent the Pareto optimal set as much as possible. With these concerns in mind, a multi-objective optimization approach should achieve the following three conflicting goals:

1. The approximate set should be as close as possible to the real Pareto front. It should be a subset of the Pareto optimal set.
2. Solutions in the set should be uniformly distributed over of the Pareto front.
3. Approximate solutions should cover the whole spectrum of the Pareto Front.

3. Mathematical model

According to PMBOK [1] a project – a temporary undertaken to create a unique product, service or result. We will understand the project P as undertake characterized by a set of goals Z, constraints C that includes a set of tasks and a set of resources R. Thus, a project is defined as a tuple:

$$P = \langle Z, C, W, R \rangle \tag{3}$$

The project purposes are defined according to the project optimality criterions. The objective functions are presented as:

$$Z = \{z_T, z_C\} \tag{4}$$

The minimal duration and the minimal cost are two main goals in this case:

$$\begin{cases} z_T \to \min \\ z_C \to \min \end{cases} \tag{5}$$

Every project has the following constraints as a rule:

- The total duration T0.
- The total cost C0.

The tasks set is the constraint too. To simplify the calculation of the project cost, the cost will only include the personnel salaries, excluding other costs (rent of space, public service, etc). Thus, under the constraint we will understand the follows:

$$C = \langle T0, C0 \rangle \tag{6}$$

The set W includes tasks, which represent an unit of work. Each task consists of laboriousness, which is expressed, in unit of time.

$$\begin{cases} W = \{w_i\}, i = \overline{1, N} \\ |W| = N \\ w \in W \end{cases} \tag{7}$$

Each project is defined as a directed acyclic graph G= (V, E) in which V is the set of nodes and E is the set of the arcs showing the relationships between tasks. There are four types of relationships:

FS – finish-start, in a finish-to-start dependency, the second task in the relationship cannot begin until the first task finishes;

FF – finish-finish, the first task be finished, in order for the second task to finish. The second task can finish any time after the first task finishes;

SS – start-start, the first task has begun, in order for the second task to begin;

SF – start-finish, the second task in the relationship cannot finish until the first task starts.

To determine the relationships between the tasks we use first-order logic, we introduce the functions and relations:

$$\begin{cases} FinishStart(\mathrm{w_i}, w_j) \Rightarrow Start(w_i) \geq Finish(w_j) \\ FinishFinish(\mathrm{w_i}, w_j) \Rightarrow Finish(w_i) \geq Finish(w_j) \\ StartStart(\mathrm{w_i}, w_j) \Rightarrow Start(w_i) \geq Start(w_j) \\ StartFinish(\mathrm{w_i}, w_j) \Rightarrow Finish(w_i) \geq Start(w_j) \end{cases} \quad (8)$$

Start – the function returning the date of commencement of the task,

Finish – the function returning the date of completion of the task,

FinishStart – the FS relationship,

FinishFinish – the FF relationship,

StartStart – the SS relationship,

StartFinish – the FS relationship,

The set R includes human resources.

$$\begin{cases} \mathrm{R} = \{\mathrm{r_j}\}, \mathrm{j} = \overline{1, \mathrm{M}} \\ |\mathrm{R}| = M \\ r \in R \end{cases} \quad (9)$$

Each resource r_j has specified skills to perform tasks and receive the salary c_j. To calculate the duration of the task w_i by assignment the resource r_j to it we introduce the function **Duration**. To simplify the calculation of the tasks duration we use the correspondence matrix **B={b_{ij}}**. The size of this matrix is N*M. Each element of the matrix represents the value of the task run-time by the resource. Thereby:

$$\mathrm{Duration}(\mathrm{w_i}, r_j) = b_{ij} \quad (10)$$

In addition, only one task can be assigned to only one resource. We introduce the flag for this purpose

$$\mathrm{x_{ij}} = \begin{cases} 1, \text{if the task i is assigned to the resource j,} \\ 0, \text{if not.} \end{cases} \quad (11)$$

Then the following condition should be satisfied for any task:

$$\forall \mathrm{i}, \mathrm{i} = \overline{1, \mathrm{N}}, \sum_{j=1}^{M} x_{ij} = 1 \quad (12)$$

Thus, the project duration and cost can be calculated as:

$$\max(\mathrm{Start}(\mathrm{w_i}) + \sum_{j=1}^{M} \mathrm{Duration}(\mathrm{w_i}, r_j) * x_{ij}) \quad (13)$$

$$\sum_{i=1}^{N} \sum_{j=1}^{M} \mathrm{Duration}(\mathrm{w_i}, r_j) * x_{ij} * c_j \quad (14)$$

The objective function can be represented as:

$$\begin{cases} \max(\mathrm{Start}(\mathrm{w_i}) + \sum_{j=1}^{M} \mathrm{Duration}(\mathrm{w_i}, r_j) * x_{ij}) \to \min \\ \sum_{i=1}^{N} \sum_{j=1}^{M} \mathrm{Duration}(\mathrm{w_i}, r_j) * x_{ij} * c_j \to \min \end{cases} \quad (15)$$

The constraints can be represented as (8) and:

$$\begin{cases} \max(\text{Start}(\text{w}_\text{i}) + \sum_{j=1}^{M} \text{Duration}(\text{w}_\text{i}, r_j) * x_{ij}) < T_0 \\ \sum_{i=1}^{N} \sum_{j}^{M} \text{Duration}(\text{w}_\text{i}, r_j) * x_{ij} * c_j < \text{C}_0 \end{cases} \quad (16)$$

4. Alternatives

If we consider only one parameter such as duration or cost, we have one-criterion optimization. The evolutionary algorithms show oneself to advantage for this problem:

- The bee's algorithm [5].
- The ant colony algorithm [8].
- The genetic algorithm [2].

The evolutionary algorithms works well with multi-objective optimization compared to the pointed algorithms such as simulated annealing [6] and tabu search. The methods provides the following advantages:

- More accurate search.
- Finding a set of solutions instead of a single.
- As a rule, the methods are independent to the basic solution.

The classical approach to solve a multicriteria optimization problem is to assign a weight to each normalized objective function so that the problem is converted to a one-criterion optimization problem with a scalar objective function [2]. This approach greatly simplify and expedite search theoretically, but computed solutions are often not optimal.

It is also possible to reduce a multicriteria optimization problem to a one-criterion optimization problem by random selection of criteria that will be evaluated the alternative. The modification of genetic algorithm VEGA (Vector Evaluated Genetic Algorithm) includes this approach.

These approaches are easy to implement and computationally as efficient as a single objective genetic algorithm. The major drawback of objective switching is that the population tends to converge to solutions, which are very superior in one objective, but very poor at others.

Using of the Pareto principles excludes this drawback. Today the most popular methods to solve multicriteria optimization problem is using of the modified genetic algorithms that take into account the Pareto principles. These methods provide complex analysis of the whole criteria spectrum simultaneously.

This approach has one disadvantage its performance. The possible methods to increase the algorithm performance are reducing of the search space and using parallel computing. The coevolutionary algorithms [4] can solve this problem. a coevolutionary algorithm is an evolutionary algorithm (or collection of evolutionary algorithms) in which the fitness of an individual depends on the relationship between that individual and other individuals As a rule there is a splitting of optimization problem into smaller components thereby, the search space reducing occurs. Each of the components responsible for optimizing single criteria, then the computation can be performed in parallel streams, better using computational capabilities of the machine. After the calculation of the fitness values of all species in population the process of cooperation or competition among different populations is used.

5. Solution method

At first, we should identify the chromosome definition or the coding system for the solution representation. The vector p of the N size will represent solution which elements are integers identifying the specific resource. Thus:

$$\forall \text{i}, 1 \le \text{p[i]} \le \text{M} \quad (17)$$

The index of this vector represents the task number or identifier, so if j = p [i], that means that the task i has been assigned resource j. Each chromosome defines the schedule which has its own duration and cost.

Because the scheduling has one collective goal (executing of the project within the time limit and budget) the cooperative type of coevolution is suggested here. As it is two-criterion optimization problem, the whole problem is divided into two subproblems: the first minimizes the project duration, the second – the cost. Each of the subcomponents uses the genetic algorithm. Selection of individuals requires the fitness values of individuals. Total fitness value of individual is determined according to internal and external fitness values. The internal fitness value is calculated by the value of optimized criteria. For the first component, it is duration, for the second – cost. Next, it should be determined how well the individual from the one population of one the component cooperates with the individual of another population of another component.

There are mainly three such attributes: the sample size, selective bias, and credit assignment for potential interactions during fitness assessment. Interaction sample size determines number of collaborators/competitors from each population to use for a given fitness evaluation. Interaction selective bias determines the degree of bias of choosing a collaborator/competitor. Interaction credit assignment determines the method of credit assignment of a single fitness value from multiple interaction-driven objective function results. We will use the population count as the sample size. Thus, the selective bias determines the whole population. The total fitness value is calculated by the formula:

$$f(\mathrm{p}) = f_{\mathrm{int}}(p) * f_{\mathrm{ext}}(p) \tag{18}$$

p – Individual

$f_{\mathrm{int}}(p)$ - Internal fitness value individual

$f_{ext}(p)$ - External fitness value of individual.

The internal fitness value is calculated by the formula:

$$f_{\mathrm{int}}(p) = \frac{1}{\mathrm{value(p,criteria)}} \tag{19}$$

Criteria – Optimization criteria. Criteria has two possible values: time, cost.

Value (p, criteria) –Value of the criteria of individual p, see 13 for time and 14 for cost.

External fitness values are calculated using the following formulas:

$$\begin{cases} f_{ext}(p_{\mathrm{ki}}^{\mathrm{T}}) = \dfrac{\sum_{\mathrm{j}}^{\mathrm{N}} cooperate\,(p_{\mathrm{ki}}^{\mathrm{T}}, p_{\mathrm{kj}}^{\mathrm{C}}, \mathrm{cost})}{L} \\ f_{ext}(p_{\mathrm{ki}}^{\mathrm{C}}) = \dfrac{\sum_{\mathrm{j}}^{\mathrm{N}} cooperate\,(p_{\mathrm{ki}}^{\mathrm{C}}, p_{\mathrm{kj}}^{\mathrm{T}}, time)}{L} \end{cases} \tag{20}$$

k – Population generation (number).

L – Population size

p_k^T - The individual of the population in generation k, which optimizes time.

p_k^C - The individual of the population in generation k, which optimizes cost.

cooperate (p1,p2, criteria) – Cooperation value of two individuals in different population of the same generation, return one of the possible values: 1(true), 0(false)

The cooperation function is calculated according to:

$$cooperate(p_{\mathrm{i}}, p_{\mathrm{j}}, \mathrm{criteria}) = value(p_{\mathrm{i}}, criteria) \leq value(p_j, criteria) \tag{21}$$

The external fitness value for individual i in generation k is greater, the more individuals from other population of the same generation k, which optimize one criteria, greater than the value of non-optimized criterion of individual i.

The crossover, mutation and ellitism operations do not differ from the corresponding operations in the case of single-criterion optimization. Selection of the individuals is based of the total fitness value. Because we are facing multicriteria optimization problems with the approach of the Pareto principles, we need a specific ranking strategy based on the classification of candidate solutions. The iteration count can be considered as a stop condition of algorithm.

Because we are facing multicriteria optimization problem at the end we should construct Pareto front of non-dominated solutions.

6. Example

As an example, the problem has been solved with the following parameters:

- N= 20
- M= 10
- Population size – 100
- Iteration count – 50
- Mutation – 5 %
- $T_0 = \infty$
- $C_0 = \infty$

The data about resources R are shown in the table 1.

Resource number	1	2	3	4	5	6	7	8	9	10
Salary	7	8	9	2	4	6	7	5	8	7

Table 1 Resources list

The tasks data W are shown in table 2.

Task number	1	2	3	4	5	6	7	8	9	10
Predecessors				1, 3	2, 3	1, 2, 3		1, 2, 3		1, 2, 3, 4, 5, 6
Task number	11	12	13	14	15	16	17	18	19	20
Predecessors	1, 3, 4, 6, 7	1, 2, 4, 5, 6, 7	4, 5, 7, 8, 9	1, 3, 7, 10, 11, 13	1,3	4,5, 11, 12, 15	2, 3, 4, 5, 6, 9, 14, 16	1, 2, 3, 7, 9, 11	1, 3, 7, 12, 13, 14, 16, 18	1, 4, 11, 15

Table 2 Tasks list

The relationship FS was considered in this sample for simplicity. The correspondence matrix B is shown in table 3

Resource/ Task	1	2	3	4	5	6	7	8	9	10
1	8	10	4	3	2	6	2	9	7	4
2	9	9	3	7	10	2	8	2	10	3
3	10	3	10	4	5	10	7	2	2	1
4	3	2	8	5	4	10	7	1	10	6
5	4	5	5	3	8	5	9	8	3	7
6	7	1	6	2	3	8	9	8	7	8
7	7	4	10	1	1	5	2	6	3	5
8	5	1	5	8	3	10	5	10	5	4
9	9	2	7	5	2	2	8	4	6	4
10	7	5	9	8	9	1	3	2	1	6

11	5	5	7	4	9	1	8	6	5	5
12	4	3	3	10	5	2	4	10	7	7
13	8	4	3	9	8	10	10	9	1	5
14	3	10	1	3	5	9	6	6	7	3
15	3	8	8	10	6	2	4	2	2	5
16	6	5	10	8	8	4	1	9	10	9
17	5	3	7	10	10	2	10	6	6	10
18	6	6	8	5	6	10	6	4	6	3
19	1	7	8	8	6	7	9	1	9	8
20	3	8	3	9	3	10	1	7	1	2

Table 3 Period of execution of tasks according to resource

As a result, two solutions were found:

Schedule#1. Total Duration – 18, total Cost – 204 and allocation:

Task1: Begin: 0; End: 3; Resource4. Task2: Begin: 0; End: 3; Resource10.
Task3: Begin: 3; End: 7; Resource4. Task7: Begin: 7; End: 8; Resource4.
Task9: Begin: 0; End: 2; Resource2. Task4: Begin: 4; End: 5; Resource8.
Task5: Begin: 8; End: 11; Resource4. Task6: Begin: 11; End: 13; Resource4.
Task8: Begin: 4; End: 5; Resource2. Task15: Begin: 5; End: 7; Resource8.
Task10: Begin: 7; End: 8; Resource6. Task11: Begin: 8; End: 9; Resource6.
Task12: Begin: 7; End: 11; Resource7. Task13: Begin: 7; End: 8; Resource9.
Task14: Begin: 13; End: 16; Resource4. Task16: Begin: 11; End: 12; Resource7.
Task18: Begin: 11; End: 15; Resource8. Task20: Begin: 11; End: 12; Resource9.
Task17: Begin: 15; End: 18; Resource2. Task19: Begin: 15; End: 16; Resource8.

Schedule#2. Total Duration – 13, total cost – 211 and allocation:

Task1: Begin: 0; End: 2; Resource7. Task2: Begin: 0; End: 2; Resource6.
Task3: Begin: 0; End: 1; Resource10. Task7: Begin: 2; End: 4; Resource7.
Task9: Begin: 0; End: 2; Resource2. Task4: Begin: 2; End: 3; Resource8.
Task5: Begin: 2; End: 5; Resource9. Task6: Begin: 2; End: 4; Resource4.
Task8: Begin: 2; End: 3; Resource2. Task15: Begin: 5; End: 7; Resource9.
Task10: Begin: 5; End: 6; Resource6. Task11: Begin: 6; End: 7; Resource6.
Task12: Begin: 7; End: 9; Resource6. Task13: Begin: 7; End: 8; Resource9.
Task14: Begin: 7; End: 10; Resource4. Task16: Begin: 7; End: 8; Resource7.
Task18: Begin: 7; End: 11; Resource8. Task20: Begin: 8; End: 9; Resource7.
Task17: Begin: 11; End: 13; Resource6. Task19: Begin: 11; End: 12; Resource1.

Dynamics of cost and duration changes is presented in table 4.

Iteration	T optimization		C optimization		Iteration	T optimization		C optimization	
	T	C	T	C		T	C	T	C
0	31	571	37	494	**26**	13	235	19	274
1	38	516	39	471	**27**	13	217	19	302
2	35	432	34	487	**28**	13	239	20	275
3	34	415	31	472	**29**	13	243	18	271
4	29	362	35	435	**30**	13	223	18	258
5	28	384	31	466	**31**	13	219	18	253
6	26	358	28	456	**32**	13	220	18	262
7	25	407	29	388	**33**	13	225	18	253
8	21	354	27	432	**34**	13	220	18	238

9	21	359	27	367	**35**	13	219	18	228
10	20	261	22	339	**36**	13	219	18	226
11	16	335	22	369	**37**	13	219	18	224
12	17	319	23	332	**38**	13	220	18	224
13	15	289	23	332	**39**	13	219	18	222
14	15	280	22	348	**40**	13	219	18	222
15	14	293	21	311	**41**	13	219	18	222
16	14	293	21	311	**42**	13	212	18	222
17	14	298	20	313	**43**	13	217	17	218
18	14	245	19	330	**44**	13	217	18	217
19	14	231	19	341	**45**	13	211	18	211
20	14	266	19	329	**46**	13	211	18	211
21	13	217	20	297	**47**	13	209	18	211
22	13	237	20	278	**48**	13	209	17	212
23	13	248	19	302	**49**	13	209	18	204
24	13	214	19	291	**50**	13	211	18	204
25	13	235	20	254					

Table 4 Time history of optimization criteria

This table show the dynamics of criteria optimization. The iteration#50 shows that it were found two solutions (schedules) which compose the Pareto-optimal set. There are two solutions (one for each optimization function or subcomponent of program) in each iteration (row in table). The solutions represented in rows have the best fitness values in their generations for each subcomponent. Because of using mutation there are some leaps, for example in the iteration#0 the min value of duration for subcomponent optimizing duration is 31 while for the iteration#1 it is 38.

Different launchings of the program give different results

Launch	Iteration count	Mutation rate	T optimization		C optimization	
			T	C	T	C
1	75	5%	14	225	18	210
2	100	5%	14	200	16	190
3	125	5%	12	200	15	182
4	150	5%	12	180	15	156
5	175	5%	12	158	15	150
6	75	25%	15	184	16	182
7	100	25%	14	186	15	170
8	125	25%	15	160	17	150
9	150	25%	17	149	17	149
10	175	25%	14	148	15	147

Table 5 Results depending on different input

As it is shown in the table 5 the results of algorithm depends on iteration count and mutation ration. Of course using of more iteration gives better result. The genetic algorithm gives different generations and as a result different solutions while mutation rate gives locale optimum escape.

7. Conclusion

This article contains information about how to solve the problem of duration-cost trade-off in project scheduling using the modified genetic algorithm that takes into account the coevolution and Pareto Principles The proposed two-criterion optimization can be extended by adding additional objective functions such as minimization of risks or maximization of quality Specific alternative strategy rankings can be replaced by another. Besides, it is possible to use this approach in other evolutionary algorithms: the ant colony algorithm and the bee's algorithm as an example. Using of this method can greatly simplify the work of the project manager in scheduling and recruitment of staff for project teams.

References

[1] A Guide to the Project Management Body of Knowledge 5-th edition. Project management institute, 2013.

[2] Abdullah Konak, David W. Coit. Multi-objective optimization using genetic algorithms: A tutorial. Reliability Engineering & System Safety In Special Issue - Genetic Algorithms and Reliability, Vol. 91, No. 9. (September 2006), pp. 992-1007.

[3] Ashish Ghosh, Satchidananda Dehuri. Evolutionary Algorithms for MultiCriterion Optimization: A Survey. International Journal of Computing & Informational Science-Vol 2. No. 1, April 2004. Available at: http://www.ijcis.info/Vol2N1/38-57S.pdf (accessed 15 June 2013). – pp 38-57.

[4] Carlos A. Coello, Gary B Lamount. Evolutionary algorithms for solving Multi-Objective problems 2-nd edition. – Springer, 2007 – pp. 144-168.

[5] D.T.Pham, A.Ghanbarzadeh, E.Koç, S.Otri, S.Rahim, M.Zaidi. The Bees Algorithm – A Novel Tool for Complex Optimisation Problems. 2006. Available at: https://svn-d1.mpi-inf.mpg.de/AG1/MultiCoreLab/papers/Pham06%20-%20The%20Bee%20Algorithm.pdf (accessed 15 June 2013).

[6] Franco Busetti. Simulated annealing overview. 2003. Available at: http://163.18.62.64/wisdom/Simulated%20annealing%20overview.pdf (accessed 15 June 2013).

[7] Matthias Ehrgott. Multicriteria Optimization 2-nd edition.–Springer, 2005.–pp. 1-20.

[8] Michael Morin,Luc Lamontagne. The Ant Search Algorithm:An Ant Colony Optimization Algorithm for the Optimal Searcher Path Problem with Visibility. 2010. Available at: http://www.academia.edu/1144152/The_Ant_Search_Algorithm_An_Ant_Colony_Optimization_Algorithm_for_the_Optimal_Searcher_Path_Problem_with_Visibility. (accessed 15 June 2013).

[9] Polumordvinova Anna Olegovna, Kvyatkovskaya Irina Yurievna. Information system of the search of the optimal administrative decision. Vestnik astrakhnaskogo gosudarstvennogo tekhnicheskogo universiteta, No 2, 2009. – pp. 61-64.

[10] R. Paul Wiegand. An Analysis of Cooperative Coevolutionary Algorithms, 2003. Available at: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.109.1816&rep=rep1&type=pdf (accessed November 23, 2013).