International Journal of the Physical Sciences Vol. 6(7), pp. 1814-1821, 4 April, 2011
Available online at http://www.academicjournals.org/IJPS
DOI: 10.5897/IJPS11.235
ISSN 1992 - 1950 ©2011 Academic Journals

Full Length Research Paper

Optimization of software engineering resources using improved genetic algorithm

Lu Lu* and Xiuxia Quan

School of Computer Science and Engineering, South China University of Technology, Guangzhou 510006, P. R. China.
E-mail: lul@scut.edu.cn.

Accepted 14 March, 2011

The arrangement of the human and material resources is an important issue in the software development process. To effectively solve the distribution optimization of software engineering resources, an improved genetic algorithm with heuristic feedback is constructed in this work. The proposed approach employs the executive matrix as chromosome which covers the matrix coding chromosome planning and allocation of resources, extracts the heuristic feedback from these obtained solutions, and applies the heuristic feedback to guide the subsequent optimization process. The experimental results suggest that this proposed approach makes a rational distribution of resources and provides a scientific basis for improving the efficiency of software development and the quality of software.

Key words: Genetic algorithms, software engineering, allocation of resources, activity network.

INTRODUCTION

As the emergence of software thoroughly changed our working, studying and life style, there is no doubt that software is the foundation of information society, and consequently becomes one of strategic point of competition among nationalities (Wang and Patel, 2009; Carbone et al., 2008). However, the process of software development exposes various problems, such as the overrun of costs. 'Software crisis' has been influencing the development and management of software, therefore, the improvement of software process has important practical significance to defuse software crisis (O'Hagan et al., 2010).

Software process, which is a representative multi-element nonlinear complicated system, is the sequential set of procedure related to activities, constraints and resources in software life cycle (Distefano et al., 2010; Zhang et al., 2009). As the expansion of software development scale, traditional software resource configuration model has significant limitations. In view of this, scholars nowadays propose plenty of solutions, mainly focusing on utilization of material resources and modeling targeting to cost and construction period in software process (Garcia et al., 2010).

Since software development is a knowledge-intensive activity, the keynote of software engineering modeling is rational distribution of human resources (Zeng, 2009; Santillan et al., 2009). Combining with genetic algorithms, this paper introduces a model for resources distribution of optimal trade-off for cost and construction period that is rationally distributed in human resources. An improved genetic algorithm with heuristic feedback is proposed to effectively solve the distribution optimization of software engineering resources.

Recently, more and more scholars have studied applications of the interaction between evolution and learning (Xing et al., 2008a, 2008b, 2010a). Normally, these approaches keep useful features of previous individuals to improve the performance of current individuals (Xing et al., 2006a, 2007, 2009). In fact, such approaches outperform traditional evolutionary algorithms on several benchmarks (for example, flexible job shop scheduling problem, traveling salesman problem, capacitated arc routing problem) (Xing et al., 2006b, 2010b; Ho et al., 2007; Louis and McDonnell, 2004). In a similar fashion, an improved Genetic Algorithm with Heuristic Feedback (HFGA) is proposed in this work. HFGA extracts some heuristic feedbacks from near-optimal solutions to guide the subsequent evolution.

PROBLEM FORMULATION

In the distribution optimization problem of software engineering resources, the hypothesis can be

summarized as follows:

1) Activity in this paper means an integral part of technology, and it cannot stop once starts.
2) Every activity in network cannot start until all its prepositive activates complete.
3) The cost of project contains: common cost to maintain normal running of project, and development cost of software developers, which points at cost resulted by various level of software developers.
4) The total period of project represents the length of critical path calculated in the condition of determinate period.

Activities

Activity is the fundamental element of optimized model in software engineering, suppose there are a total of N activates and M persons in the software development process.

1) $Node$ denotes the No. of Nodes in the network diagram.
2) T_{di} denotes the period of activity $(i=1,2,\cdots,N)$.
3) T_{di}^{c} denotes the time remainder of activity $(i=1,2,\cdots,N)$.
4) T_{di}^{n} denotes the actual completion time of activity $(i=1,2,\cdots,N)$.
5) $Role_{i,j}$ denotes the i^{th} $(i=1,2,\cdots,N)$ activity can be done by the j^{th} $(j=1,2,\cdots,M)$ person.
6) $RoleNum_i$ denotes the completion persons number of activity $(i=1,2,\cdots,N)$.
7) T_D denotes the planned completion time of engineering project.
8) T_N denotes the actual completion time of engineering project.
9) C_D denotes the planned cost of engineering project.
10) C_N denotes the actual cost of engineering project.
11) $f_T(EM)$ denotes the completion date function of engineering project.
12) $f_C(EM)$ denotes the cost function of engineering project.

According to the order among activities and the probable completion time of each activity, calculate the critical path of network, and the length of it is the value of T_N. The method is as follows: based on the assumption for network, each activity cannot start until all the prepositive activities finished. In accordance with executive matrix EM, find the probable completion time of each activity. The latest time of completion among all the activities is the value of T_N.

The cost of software process contains: management fee and development fee. The former is daily cost C_1 which maintaining software development and cost of administrators C_2. The latter points at cost of device resources C_3 and development cost C_4 of different developers with various abilities and labor-hours.

$$\begin{cases} CN = C_1 + C_2 + C_3 + C_4 \\ C_4 = \sum_{i=1}^{M} c_i T_i \end{cases} \quad (1)$$

Here, c_i and T_i denote the development cost and labor-hour of developers respectively.

Constraints

1) The activity period constraint, it is $T_{di} \le T_{di}^{d} \le T_{di} + T_{di}^{c}$.
2) Process constraints. Activity sequence of software engineering is working sequence which satisfying sequencing. The activity set of procedure j is $p(j) = \{i |$ activity i is the prepositive of activity $j, t_j - t_i \ge T_{di}, i \subseteq p(j), j = 1,2,\cdots,N\}$.
1) Construction period constraint, that is $T_N \le T_D$.
2) Cost constraint, it is $C_N \le C_D$.

Resource problems

A device resource is both a key link in software development process and a ingredient of software development cost. Compare with human resources, it is easily be quantified, not constitute of difficulty of resources distribution. In the circumstances of definite software development, the composition of device resources is relatively fixed, with little space to improve. Moreover, the distribution of device resources cannot be the focus of proposed model.

Rational distribution of human resources is the nucleus of resources distribution in software process. As software is a work of concentration of brain power, the organization, division of labor and distribution of developers is critical

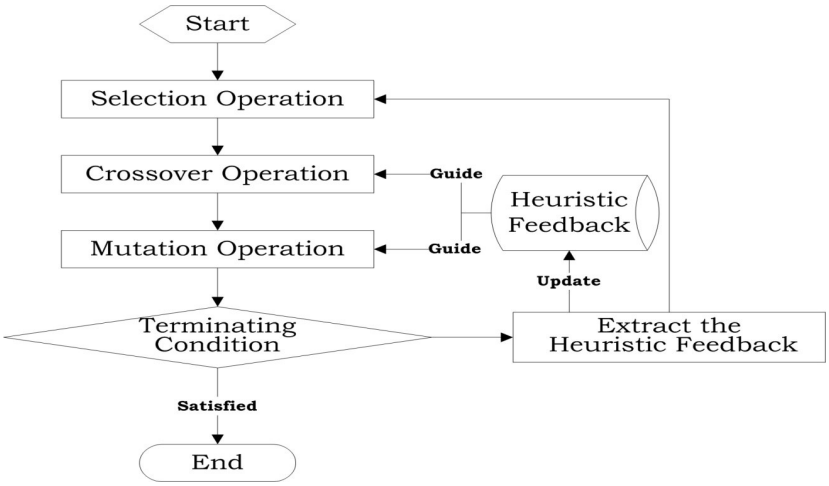

Figure 1. The computational flow of HFGA.

for software process, which is directly influencing whether the software project can be successful or not.

Effort estimation

Effort estimation is the foundation of resources distribution. To realize rational resource distribution, exact effort estimation must be conducted owing that engineering experience only offers qualitative guide of resource distribution. The effort estimation contains two parts: effort estimation for each activity and estimation for working ability of each member. This model adopts the expert judgment method.

The way of expert judgment commonly uses the estimation method in the following equation, in which E_o means Optimistic Effort, E_p represents Pessimistic Effort, and E_m indicates the Most Probable Effort.

$$E = \frac{E_o + 4E_m + E_p}{6} \tag{2}$$

Suppose there are N activities in network, M persons organize software development. As the expert judgment, the effort estimation matrix for various activities of various people is obtained as follows:

$$TM = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}_{m\times n}$$

Here, a_{ij} denotes the probable time of developer i completes activity j.

Objective functions

Cost and construction period is the main attributes of software process. Assume the sum weight of cost and period as optimized objective function, the definition of which is as shown in the following equation. In which, C means cost, T signifies the weight value of construction period x_1, x_2 determined by demand of decision makers, EM is the executive matrix.

$$\begin{aligned} &\max Fitness(C,T) = x_1 * C + x_2 * T \\ &s.t.\begin{cases} C = cost(EM) \\ T = Time(EM) \\ 0 \le x_1 \le 1 \\ 0 \le x_2 \le 1 \\ x_1 + x_2 = 1 \end{cases} \end{aligned} \tag{3}$$

GENETIC ALGORITHM WITH HEURISTIC FEEDBACK

This proposed improved genetic algorithm, Genetic Algorithm with Heuristic Feedback, is characterized by the extraction and application of heuristic feedback in the whole evolution process. In fact, the significant size of the infeasible region in the search space poses the difficulty of finding good quality feasible solutions using the genetic algorithm alone. For this reason, the near-optimal solutions obtained throughout the search are analyzed to extract the heuristic feedback firstly, and then the obtained heuristic feedback is used to guide the subsequent search. The computational flow of HFGA is shown in Figure 1.

Heuristic feedback

The first kind of heuristic feedback is called the activity assignment position which is applied to establish a beneficial order for the given activity. A matrix HF_1 with size $N \times N$ is defined for the activity assignment position, $HF_1(i, j)$ denotes the total number of times of assigning the activity i to the j^{th} position among the near-optimal solutions obtained throughout the search.

The second kind of heuristic feedback is called the activity assignment person which is applied to establish the beneficial person for one given activity. A matrix HF_2 with size $N \times M$ is defined for the activity assignment person, $HF_2(i, j)$ denotes the total number of times of assigning the activity i to the j^{th} person among the near-optimal solutions obtained throughout the search.

Population Initialization

Chromosome, also called individual, is the encoded solution for specific problems. The solutions of activity planning and resource distribution problem are a series of scheme. There remain limitations when traditional binary string encoding describes complicated programming problems, hence, this paper adopts a matrix EM with the $(M+1) \times N$ size as one chromosome.

The chromosome of proposed encoding matrix covers activity planning and resource distribution. The first row of matrix is natural number sequence without repetition from 1 to N, representing initial activity sequence, the second row to $M+1$ row means matching relation between human resource and activity. Unless there exists no confliction among activities which is closely before and after initial planning sequence and personnel distribution, these two activities cannot be executed simultaneously.

Population is the set of chromosomes. In terms of particularity of chromosome structures in this paper, the population initialization is divided into two steps: activity planning initialization and resource distribution initialization. The activity planning section is the natural sequence which satisfies constraint relation, produced by topological sorting algorithm. The resource distribution section is initialized using random generation.

Selection operation

In the proposed HFGA, the Binary Tournament is applied to execute the selection operation. That is, two different chromosomes are randomly selected, and the least-cost one is kept. In order to improve the quality of HFGA, the elitism among the current population is directly copy to the next population, and the population consists exclusively of unique phenotypes and identical solutions are never accepted.

Crossover operation

In the genetic algorithms, crossover operator is a main method for producing new individuals. The chromosome in proposed model is matrix structure, which determines crossover operator cannot adopt traditional character string swap mode. Concrete crossover operation includes the following two steps:

Step 1: Identify the crossover columns of chromosome. Produce a random integer distributed in section $[0, N]$, which is applied to denote the crossover length (such as crossover columns).
Step 2: Determine crossover range of chromosome. Maintain relative sequence of column selected to join crossover operation in chromosome, and then exchange corresponding columns of two chromosomes.

As displayed in Figure 2, suppose crossover column is 3, and crossover range is 3, 4 and 6 respectively. Generate offspring chromosome according to sequence crossover method, so as to solve parallel machine scheduling problem with technology constraints. In virtue of combining the relative and absolute sequence of chromosome genes, sequence crossover method not only ensures the efficiency of solutions, but also effectively solves the constraint of complicated prepositive relation among activities in software process.

In our proposed HFGA, the activity assignment position is applied to guide the crossover operation. The activity assignment position is employed to determine one beneficial position for the given activity. To the activity assignment position matrix displayed in Table 1, if we want to determine the beneficial position for activity 3, then we can obtain the following probabilities, and the beneficial position to activity 3 is decided by a random way with the following probability distribution.

Mutation operation

Mutation operation is classified into two stages, the first is activity planning mutation and the second is resource distribution mutation. On one hand, identifying the mutation position of parent chromosome by the use of random number function, and on the other hand, in the condition of satisfying transposed prepositive constraint relation of activities, reselect planning sequence of mutation position identification. At last, conduct resource distribution mutation, and reselect resource organization for undertaking the activity. The concrete mutation

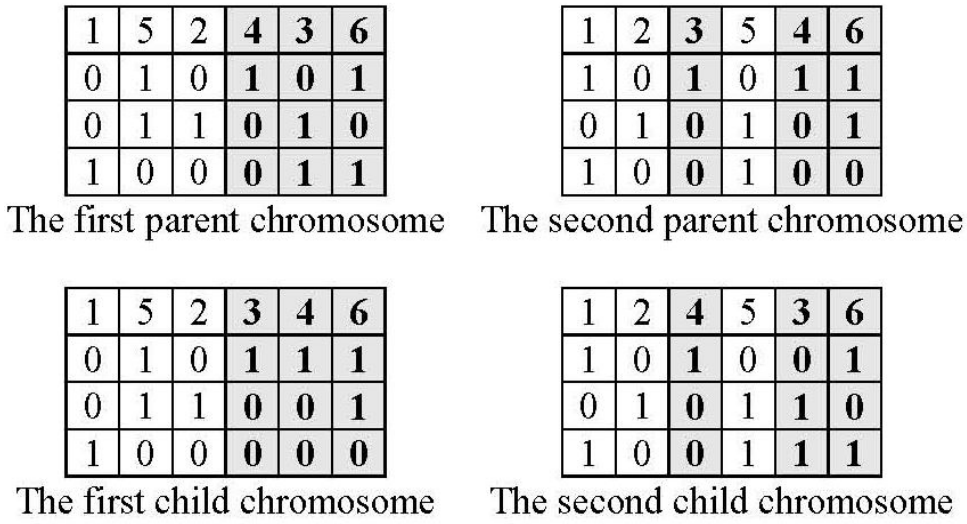

Figure 2. A simple example of crossover operation.

Table 1. An example of activity assignment position.

3	5	4	2	1	0
1	5	6	2	1	0
0	1	2	8	2	2
0	2	2	6	5	1
0	0	4	5	6	0
0	0	2	3	4	6

$$\text{Position } 1: \frac{0}{1+2+\frac{2}{1}+8+2} = 0$$

$$\text{Position } 2: \frac{1}{1+2+\frac{2}{2}+8+2} = 0.07$$

$$\text{Position } 3: \frac{2}{1+2+\frac{2}{2}+8+2} = 0.13$$

$$\text{Position } 4: \frac{2}{1+2+\frac{2}{8}+8+2} = 0.13$$

$$\text{Position } 5: \frac{8}{1+2+\frac{2}{8}+8+2} = 0.53$$

$$\text{Position } 6: \frac{2}{1+2+2+8+2} = 0.13$$

operation is as shown in Figure 3.

In our proposed HFGA, the activity assignment person is applied to guide the mutation operation. The activity assignment person is employed to determine one beneficial person for the given activity. To the activity assignment person matrix displayed in Table 2, if we want to determine the beneficial person for activity 6, then we can obtain the following probabilities, and the beneficial person to activity 6 is decided by a random way with the following probability distribution.

Termination conditions

The HFGA is terminated when one of the following conditions satisfied: the elitism is not improved in the successive SI generations, and the maximum MI generations are exhausted.

EXPERIMENTAL RESULTS

The HFGA was implemented using Visual C++ language, and executed on a personal computer with the 2 GHz processor and 2 GB memory. We established a favorable choice of parameters, as listed in Table 3, by means of systematic experimentation. In this paper, the final experimental results were averaged over 30 trials, and 10 testing instances were randomly produced to validate the performance of our approach.

In order to validate the performance of our HFGA, the standard genetic algorithm (SGA), the intelligent genetic algorithm (IGA) (Xing et al., 2006a) and the multiprogramming genetic algorithm (MGA) (Xing et al.,

1	5	2	**4**	3	6
0	1	0	**1**	0	0
0	1	1	**0**	1	1
1	0	0	**0**	1	1

The original parent chromosome

1	**4**	5	2	3	6
0	**1**	1	0	0	0
0	**0**	1	1	1	1
1	**0**	0	0	1	1

The offspring after activity planning mutation

1	**4**	5	2	3	6
0	**1**	1	0	0	0
0	**0**	1	1	1	1
1	**1**	0	0	1	1

The offspring after resource distribution mutation

Figure 3. A simple example of mutation operation.

Table 2. An example of activity assignment person.

4	2	5	2	4	6
5	7	5	11	6	6
6	6	5	2	5	3

$$\text{Person } 1: \frac{6}{6+6+3} = 0.40$$
$$\text{Person } 2: \frac{6}{6+6+3} = 0.40$$
$$\text{Person } 3: \frac{3}{6+6+3} = 0.20$$

Table 3. A favorable choice of parameters to the HFGA.

Name	Role	Value
P_s	Population size	200
P_X	Crossover rate	0.80
P_M	Mutation rate	0.10
SI	The elitism is not improved in the successive generations	100
MI	The maximum generations	1000

2007) are applied to compare with the HFGA. These different versions of genetic algorithms were implemented using Visual C++ language by us. The final experimental results were summarized in Tables 4 and 5.

The optimal objectives obtained by these four different methods are summarized in Table 4. From the experimental results of Table 4, we can see that, there exists a small gap among these approaches to the small instances, and HFGA largely outperforms to other algorithms to these large instances. In terms of the optimal objective, HFGA is powerful than other different genetic algorithms.

Table 4. The optimal objective obtained by different methods.

SN	N	M	SGA	IGA	MGA	HFGA
1	20	5	82.4	80.2	80.5	79.6
2	20	5	93.3	91.9	91.4	90.1
3	20	5	85.7	85.2	84.1	83.8
4	50	10	212.7	209.5	206.3	201.4
5	50	10	363.5	349.7	336.2	320.5
6	50	10	410.2	398.4	381.6	359.8
7	200	30	780.9	755.8	731.6	701.6
8	200	30	993.6	962.5	933.8	915.7
9	200	30	884.5	857.6	847.2	820.0
10	200	30	956.3	921.4	909.8	873.6

Table 5. The computational time different methods (seconds).

SN	N	M	SGA	IGA	MGA	HFGA
1	20	5	50.2	53.6	52.3	54.2
2	20	5	56.8	57.3	58.5	56.2
3	20	5	61.3	62.8	59.6	58.4
4	50	10	223.6	225.8	220.9	228.5
5	50	10	259.6	261.3	266.1	258.2
6	50	10	288.4	280.9	283.3	286.7
7	200	30	683.5	696.7	662.8	677.5
8	200	30	725.3	716.7	719.8	723.6
9	200	30	810.9	803.5	811.2	799.5
10	200	30	956.1	936.8	944.7	950.2

The computational time of these four different methods are summarized in Table 5. From the experimental results of Table 5, we can see that, there exists a small gap among these approaches to all instances. In fact, the parameters to these algorithms are similar, and the total times of fitness evaluations of these algorithms are similar too. For this reason, the gap between the computational time of these four different methods is small.

In total, the experimental results suggest that this proposed approach makes a rational distribution of resources and provides a scientific basis for improving the efficiency of software development and the quality of software.

CONCLUSIONS

The contribution of this paper can be summarized as follows: An improved genetic algorithm with heuristic feedback is constructed to effectively solve the distribution optimization of software engineering resources. The proposed approach employs the executive matrix as chromosome which covers the matrix coding chromosome planning and allocation of resources, extracts the heuristic feedback from these obtained solutions, and applies the heuristic feedback to guide the subsequent optimization process.

The future research directions can be listed as follows: enhance the performance of HFGA by improving the genetic operators, and employ the HFGA to other practical engineering fields.

ACKNOWLEDGMENT

This paper is supported by the China National Science Fund and the Guangzhou Government Special Fund.

REFERENCES

Carbone P, Buglione L, Mari L (2008). A comparison between foundations of metrology and software measurement. IEEE T. Instrum. Meas., 57(2): 235-241.

Distefano S, Puliafito A, Scarpa M (2010). Implementation of the Software Performance Engineering Development Process. J. Softw., 5(8): 872-882.

Garcia I, Pacheco C, Calvo-Manzano J (2010). Using a web-based tool

to define and implement software process improvement initiatives in a small industrial setting. IET Softw., 4(4): 237-251.
Ho NB, Tay JC, Lai EMK (2007). An Effective Architecture for Learning and Evolving Flexible Job-Shop Schedules. Eur. J. Oper. Res., 179(2): 316-333.
Louis SJ, McDonnell J (2004). Learning with Case-Injected Genetic Algorithms. IEEE Trans. on Evo. Comp., 8(4): 316-328.
OHagan P, Hanna E, Territt R (2010). Addressing the corrections crisis with software technology. Comp., 43(2): 90-93.
Santillan CG, Cruz-Reyes L, Meza E (2009). Improving Distributed Resource Search through a Statistical Methodology of Topological Feature Selection. J. Comp., 4(8): 727-733.
Wang YX, Patel S (2009). Exploring the cognitive foundations of software engineering. Int. J. Soft. Sci. Comp. Intel., 1(2): 1-19.
Xing LN, Chen YW, Cai HP (2006a). An intelligent genetic algorithm designed for global optimization of multi-minima functions. Appl. Math. Comp., 178(2): 355-371.
Xing LN, Chen YW, Shen XS (2006b). A Constraint Satisfaction Adaptive Neural Network with Dynamic Model for Job-Shop Scheduling Problem. Lect. Notes Comput. Sci., 3973: 927-932.
Xing LN, Chen YW, Shen XS (2007). Multiprogramming Genetic Algorithm for Optimization Problems with Permutation Property. Appl. Math. Comp., 185(1): 473-483.
Xing LN, Chen YW, Yang KW (2008a). A hybrid approach combining an improved genetic algorithm and optimization strategies for the asymmetric traveling salesman problem. Eng. Appl. Artif. Intel., 21(8): 1370-1380.
Xing LN, Chen YW, Yang KW (2008b). Double Layer Ant Colony Optimization for Multi-objective Flexible Job Shop Scheduling Problems. New Generat. Comput., 26(4): 313-327.
Xing LN, Chen YW, Yang KW (2009). An Efficient Search Method for Multi-objective Flexible Job Shop Scheduling Problems. J. Intell. Manuf., 20(3): 283-293.
Xing LN, Philipp R, Chen YW (2010a). An Evolutionary Approach to the Multi-depot Capacitated Arc Routing Problem. IEEE Trans. Evo. Comp., 14(3): 356-374.
Xing LN, Chen YW, Wang P (2010b). A Knowledge-based Ant Colony Optimization for Flexible Job Shop Scheduling Problems. Appl. Softw. Comp., 10(3): 888-896.
Zeng Q (2009). Study and implementation of distributed resource management information service. Comput. Eng., 35(4): 72-77.
Zhang S, Wang YJ, Ruan L (2009). Personal Software Process Capability Assessment Method. J. Softw., 20(12): 3137-3149.