Multistage Human Resource Allocation for Software Development by Multiobjective Genetic Algorithm

Feng Wen[a,b] and Chi-Ming Lin*[,a,c]

[a]*Graduate School of Information, Production & Systems, Waseda University, Kitakyushu, Japan*

[b]*School of Information Science and Engineering, Shenyang Ligong University, Shenyang, China*

[c]*Center for General Education, College of Management, Kainan University, Taoyuan, Taiwan*

Abstract: Software development is a multistage process. Minimizing the project duration and minimizing the project cost are two objectives of software projects. These two goals are often in conflict with each other. The most important influencing factor of these two objectives is human resource allocation. The best compromised human resource allocation plan based on these two objectives should be provided to the project manager. This is a multistage human resource allocation problem (MHRAP) which belongs to multiple criteria problems. Genetic algorithm is a well-known solving method for multiple criteria problems.

In this paper, we propose a new multiobjective genetic algorithm (moGA). This moGA is based on a new encoding method, named Improved Fixed-length Encoding method. This encoding method is simple and effective for programming. An adaptive-weight fitness assignment mechanism is used to find a Pareto solution set. A factor weight method is proposed to find the best compromised solution from a Pareto solution set. Project managers can assign weight on each objective to decide how to arrange software for the project.

Keywords: Software development, improved fixed-length encoding, multistage human resource allocation problem (MHRAP), multiobjective genetic algorithm (moGA).

1. INTRODUCTION

Currently, software development companies are facing an extremely increasing rise in market demand for software products and services. Software projects usually demand complex management involving scheduling, planning, and monitoring tasks. There is a need to control people and processes, and to efficiently allocate resources in order to achieve specific objectives while satisfying a variety of constraints. Usually, the software project scheduling problem should consider how to allocate human resources to each task.

A Project Scheduling Problem consists of deciding who does what during the software project's lifetime. This is a capital issue in the software project, because the total budget and human resources must be managed optimally in order to result in a successful project. In short, companies are principally concerned with reducing the duration and cost of projects. If there occurs a software project duration extension, fewer people are usually needed to finish the project. Correspondingly, if we want reduce development time, more employees are needed. More person-months units are required as a tradeoff for reducing development duration. And most of the research is concerned with estimating the cost or duration. But in a real software developing environment, the manager faces a problem in deciding how to compromise these two goals to maximize profit. This problem is a multistage human resource allocation problem (MHRAP).

*Address correspondence to this author at the Graduate School of Information, Production & Systems, Waseda University, Kitakyushu, Japan; E-mail: chiminglin@ruri.waseda.jp

In the case of multiple-objectives, there does not necessarily existence a solution that is best with respect to all objectives because of incommensurability and conflict among objectives. A solution may be the best in one objective but the worst in another. Therefore, there usually exists a set of solutions for the multiple-objective case which cannot simply be compared with each other. For such solutions, called no-dominated solutions or Pareto optimal solutions, no improvement is possible in any objective function without sacrificing at least one of the other objective functions.

There is considerable literature focusing on the estimated cost and duration. Some prediction models such as Function Points Analysis [1], Constructive Cost Mode (COCOMO) Models [2-3] and Ordinal Regression Models [4] have been proposed. Artificial neural networks (Ann) are used to produce more accurate resource estimates [5-6]. Among them, the Function Point method measures the developed system by point counts that can be determined relatively early in the development process. It measures software project size by studying external features of the project. For software effort estimation, the counting of function points requires complex training to achieve an objective. But it does not take into consideration the different stages. COCOMO is based on well defined software engineering concepts. The model is simple to apply and can be calibrated for precision. COCOMO offers a more readily adaptable means of developing a tailored model. But it is hard to estimate the cost of the software system at an early stage of the project. Ann is widely used for forecasting problems. But, Ann is apt to converge to the local minimum point during learning. Putnam's Model and Boehm's Model were proposed to optimize resource allocation for software development [7]. But their model cannot solve multicriteria resource allocation.

Evolutionary Algorithms (EA) are being applied to a wide range of optimization, and can offer significant advantages in solution methodology and optimization performance. Genetic algorithms (GA) are one of EA. GA searches from a population of points that can provide globally optimal solutions. In addition, GA uses probabilistic transition rules, and work with a coding of the parameter set. Therefore GA can easily handle the integer or discrete variables.

A genetic method was proposed to solve the software project scheduling problem [8]. However, the different stages of the software project were not considered. Different weights were assigned to different objective values to calculate fitness value. The diversity of solution space was also not considered.

In recent years, EA is also increasingly being developed and used for multiobjective optimization problems. EA provides a framework of using only objective function information for analyzing many multiobjective problem types. Within this framework, optimization techniques can be employed to solve the non-smooth, non-continuous and non-differentiable functions which actually exist in a practical optimization problem. Surveys on such multiobjective Evolutionary Algorithms were given [9-10]. A nondominated sorting-based multiobjective evolutionary algorithm was suggested, that is, nondominated, sorting Genetic AlgorithmII [11].

In this paper, we developed an efficient representation scheme using Improved Fixed-length Encoding method. This method can solve MHRAP effectively. Based on this encoding method, a multiobjective Genetic Algorithm (moGA) applying Adaptive Weight fitness value assignment method is developed. This fitness assignment mechanism helps with finding a set of solutions that are close to the global Pareto set. If we want to get a global Pareto set, it will usually lead to a very long execution time. In our research, we propose moGA to find a near optimal and acceptable solution within a reasonable execution time. Numerical experiments show the effectiveness and the efficiency of our approach by comparing recent research. Then, a distance-based method was proposed to select the best compromised result in a pareto solution set by the managers' preference. Managers can arrange software development by this result. They can thus achieve the optimal trade-off of the two objectives.

The rest of the paper is organized as follows. In Section 2, the software stages and assumptions are described, then a mathematic model of a multiobjective problem is formulated. The proposed moGA is described in Section 3 to solve this problem. In Section 4, an experiment of moGA is given and its results are analyzed. In Section 5, we draw a conclusion from the proposed algorithm.

2. SOFTWARE DEVELOPMENT STAGES AND MATHEMATICAL MODEL

2.1. Cost and Time Efforts and Assumptions

In here, it shows that when the number of employees increases, their rate of effort decreases[7]. One would imagine that with more employees the project duration should decrease sharply. However, the total contribution of employees will increase only slightly when the number of employees increases, mainly because of manpower and time is not interchangeable. There exists a trade-off between person units and development time. In other words, when increasing employee numbers, the average productivity rate is reduced. Here, to analyze the trade-off ratio between personnel unit and development time, we consider the composing of software development process and then make some assumptions for our research.

Usually there are six stages in software projects which includes requirement analysis, architectural design, detailed design, coding, test and maintenance. Among them, the requirement analysis and maintenance stages are discrete time processing and these two stages should involve contact with clients or investors. So we can't use known mathematical methods to estimate the cost and duration of them. Here, we consider the other four stages.

2.2. The Network Model and Mathematical Model

We can reformulate MHRAP as a network model, where limited activity represented by stages (such as jobs or tasks), and the resources gives possible states in each stage (number of workers must be allocated for each stage). The meaning element in network is decision variable x_{ij}. As shown in Fig. (**1**), the inflows of x_{ij} are objective values (w_{1ij}, w_{2ij}, …, w_{pij}, w_{Oij}) when assign j employees to ith stage.

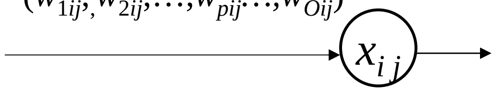


Fig. (1). Illustration of element in network model.

We start by describing the basic model. The network model of MHRAP within multistage is shown in Fig. (**2**). Consider Fig. (**2**), S and T are dummy starting and terminating nodes respectively. The inflow of dummy terminating node T is 0. A path from S to T of this network is one allocation plan for MHRAP.

We consider the MHRAP as multiobjective with minimizing the total cost and minimizing the total duration. In order to calculate these two objective values, we list some assumptions about objective formulation function:

In a pratical software project, there are several kinds of costs in the project process. Here we only consider the cost of employees.

1. A month is the measured unit of software project duration. In a practical enviroment, the software project may be interrupted by some unpredictable event. In order to facilitate our research, we only consider the continuous time process of software projects.
2. Each project has a completion deadline, so we consider that the duration should not extend beyond the deadline.
3. Each project should consider the benefits, so we assume that the total cost should be less than 70 percent of investment.
4. Each stage will allocate at least one employee.

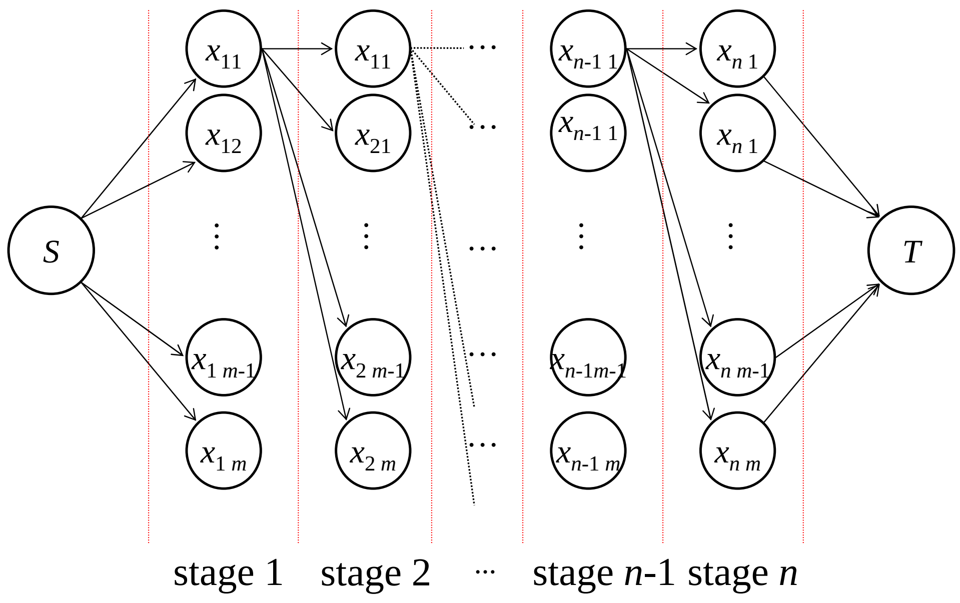


Fig. (2). Model of MHRAP within multistage.

5. In order to be simply calculated, the salary of employees is the same in the same stage.
6. Each software Project includes all stages.
7. For one project, we use the number of person-months required to measure its workload, and we assume that the workload of each stage has been estimated by other algorithms. Here, we only consider how to allocate the human resource.
8. In a practical enviroment, when treating the same task, different employees will use different times to finish his task because of their different abilities. In our research, we assume that if the manager assigns work to different employees, they will finish the task in the same time.

The MHRAP is to assign *m* staff to *n* different projects for maximizing the benefit and minimizing the cost subject to one resource constraint is formulated as a bicriteria integer programming model, which has been proposed in [12]. These two objective values can be minimizing simultaneously [13].

Notations

Indices

i: index of stage, *i* = 1, 2, …, *n*.

j: number of employee, *j*= 1, 2, …,*m*.

Parameters

n: total number of stages in software developing

m: total number of workers

T: maximal duration of the whole project in considered four stages

C: maximal cost consumption

c_{ij} : cost of stage *i* when *j* employee are assigned

t_{ij} : duration of job *i* when *j* employee are assigned

Decision Variables

$$\boldsymbol{x}_{ij} = \begin{cases} 1, \text{ if } j \text{ employees are assigned to stage } i \\ 0, \text{ otherwise} \end{cases}$$

$$\min \quad z_1 = \sum_{i=1}^{n} \sum_{j=0}^{m} t_{ij} \boldsymbol{x}_{ij} \tag{1}$$

$$\min \quad z_2 = \sum_{i=1}^{n} \sum_{j=0}^{m} c_{ij} \boldsymbol{x}_{ij} \tag{2}$$

$$\text{s. t.} \quad \sum_{i=1}^{n} \sum_{j=0}^{m} j\boldsymbol{x}_{ij} \le m \tag{3}$$

$$z_1 \le T \tag{4}$$

$$z_2 \le C \tag{5}$$

$$\sum_{j=0}^{m} \boldsymbol{x}_{ij} = 1 \qquad \forall i \tag{6}$$

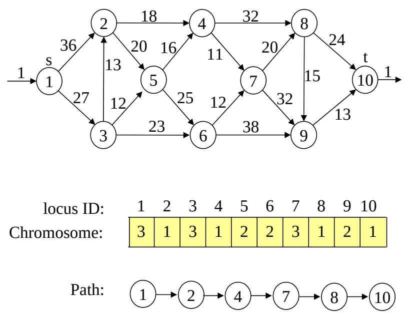
$$\boldsymbol{x}_{ij} = 0 \text{ or } 1 \quad \forall i, j \tag{7}$$

Constraint (3) ensures that we cannot assign the workers more than the total number of workers. Constraint (4) ensures that the total duration of project is less than maximal duration. Constraint (5) ensures that the total cost of the project is less than maximal cost. Constraint (6) ensures that for each job *i* we just can only assign workers for it one time.

3. MULTIOBJECTIVE GENETIC ALGORITHM

3.1. Genetic Representation

The encoding mechanism is fundamental to the GA for representing the optimisation problem's variables. The encoding mechanism depends on the nature of the problem variables. In network problems, recently, to encode a short-


s
t
locus ID: 1 2 3 4 5 6 7 8 9 10
Chromosome: 3 1 3 1 2 2 3 1 2 1
Path: 1 → 2 → 4 → 7 → 8 → 10

Node Id	$\|Suc_i[]\|$	$Suc_i[]$
1	2	{2, 3}
2	2	{4, 5}
3	3	{2, 5, 6}
4	2	{7, 8}
5	2	{4, 6}
6	2	{7, 9}
7	2	{8, 9}
8	2	{9, 10}
9	1	{10}
10	0	Ø

Fig. (3). Example of generated chromosome and its decoded path.

est routing path into a chromosome for GAs, there are various non-string encoding techniques that have been created.

Munemoto *et al.* proposed variable-length chromosomes to represent a routing [14]. But the algorithm requires a relatively large population for an optimal solution due to the constraints on the crossover mechanism, and is not suitable for large networks. Ahn *et al.* developed a variable-length chromosomes [15]. A new crossover operation exchanges partial chromosomes is introduced. But crossover may generate infeasible chromosomes that have loops in the paths. We need to check the feasibility and repair mechanism. It is not suitable for large networks or unacceptable high computational complexity. Inagaki *et al.* proposed a fixed-length chromosome technique [16]. The chromosomes in the algorithm are sequences of integers and each gene represents a node ID that is selected randomly from the set of nodes connected with the node corresponding to its locus number. All the chromosomes have the same length. In their method, some offspring may generate new chromosomes that resemble the initial chromosomes in fitness, thereby retarding the process of evolution.

In this paper, we propose an Improved Fixed-length Encoding method which combines the merit of fixed-length and variable-length chromosome coding method. This encoding method is easy to realize as fixed-length encoding method. As it is known, a gene in a chromosome is characterized by two factors: locus, i.e., the position of gene located within the structure of chromosome, and allele, i.e., the value the gene takes. In this encoding method, a chromosome of the proposed GA consists of sequences of positive integers that are randomly created based on maximum successive set length of each node. Each locus of the chromosome represents an order of a node in a routing path. The gene of the first locus is always reserved for the source node. The length of the chromosome is fixed, and it is the same to the total number of nodes in the network. The value of the gene is used to search the next node in decoding method.

3.2. Improved Fixed-Length Encoding

In the decoding method, the position of a gene is used to represent the node index in the route and its value is used to decide which node will be selected in successive set of current nodes. When destination node occured in successive set, the encoding process is ended. A path can be easily determined by this encoding method. An example of generated chromosome and its decoded path is shown in Fig. (**3**).

We denote the $Suc_i[]$ as the successive set of node i in road network, and $|Suc_i[]|$ is nodes number in successive set of node i. The maximum value of $|Suc_i[]|$ in example network shown in Fig. (**3**) is 3 . All value in gene locus is randomly created between 1 to 3. As shown is Fig. (**3**), the are {3, 1, 3, 1, 2, 2, 3, 1, 2, 1}. Node 1 is original node and 10 is destination node. At the beginning, we try to find out which node should be selected in $Suc_1[]$.

Nodes 2 and 3 are eligible for the position, which can be easily found in $Suc_1[]$. The value in locus 1 is 3, and $|Suc_1[]|$ equals 2. Then, we calculate (2 mod 3) mod 2 = 0. The next node index is $Suc_1[0]$, here is 2. Repeat these steps until we obtain a complete path (1→2→4→7→8→10).

We list the whole encoding process as follows:

```
procedure 1: Improved Fixed-length Encoding
input: number of nodes $n$, array of successive nodes number $Suc[]$
output: $k$th initial chromosome $v_k[]$
begin
select maximum length assign to $j$ in $Suc[]$;
for $i$ =1 to $n$
$v_k[i] \leftarrow$ random$[1, j]$;
output: $k$th initial chromosome $v_k$ []
End
```

Based on the Improved Fix-length Encoding method mentioned before, we present the decoding procedure as follows:

```
procedure 2: Improved Fixed-length Decoding
input: chromosome $v[]$, no. of nodes $n$, origin id $O$, destination id $D$, $Suc[][]$
output: path $P[]$
begin
$P[1] = O$;
for $i$ = 2 to $n$                    // initialize path with zero
$P[i] = 0$;
for $i$ = 1 to $n$-1
$id = P[i]$;
$index = (|Suc[id][]| \% v[i]) \% |Suc[id][]|$; // calculate the index of next node
$P[i+1] = Suc[id][index]$; // add current node into path
if $P[i+1] = D$ then break; // find the destination node
output: path $P[]$
End
```

The trace table of decoding procedure is listed in Table **1**.

Table 1. Trace Table of Decoding Procedure

Iteration	Node ID (i)	$Suc[i]$	$\lvert Suc_i[\,]\rvert$	$Path[\,]$
1	1	{2, 3}	2	1
2	2	{4, 5}	2	1-2
3	4	{7, 8}	2	1-2-4
4	7	{8, 9}	2	1-2-4-7
5	8	{9, 10}	2	1-2-4-7-8
6	10	{Ø}	0	1-2-4-7-8-10

This trace table is used for illustrate the process of decoding process. The data and result is related to Fig. (**3**).

3.3. Genetic Operation

The next step is to generate a next generation population of solutions from those genetic operators: crossover (also called recombination), and mutation.

In the crossover method, two chromosomes are chosen which should have at least one same node index except for source and destination nodes, but there is no requirement that they should be located at the same locus of the chromosome. We illustrate the procedure as follows:

```
procedure 3: Improved Fixed-length crossover
input: $v_1$, $v_2$
output: $v^o_1$, $v^o_2$ //offspring of $v_1$, $v_2$
begin
$P_1 \leftarrow$ decoding($v_1$);
$P_2 \leftarrow$ decoding($v_2$);
if $P_1$ and $P_2$ have one or more same node set ($SetP[]$) then
randomly select index $k$ from the $SetP[]$;
$l_1 \leftarrow$ site in $v_1$ corresponding $SetP[k]$; // $l_1$ is crossover site in $v_1$
$l_2 \leftarrow$ site in $v_2$ corresponding $SetP[k]$; // $l_1$ is crossover site in $v_1$
apply crossover to $v_1$ and $v_2$ and produce offspring $v^o_1$ and $v^o_2$ ;
output: $v^o_1$, $v^o_2$
End
```

Insertion Mutation has been adopted in this paper. In this mutation, a gene is randomly selected and inserted into a position, which is determined randomly.

3.4. Adaptive-Weight Approach

The selection operation is intended to improve the average quality of the population by giving the high-quality chromosomes a better chance to get copied into the next generation. In this paper we use Roulette Wheel selection, which was proposed by Holland [17] to determine selection probability or survival probability for each chromosome proportional to the fitness value [18].

In selection operation, we should design a fitness assignment mechanism. The most important issue is how to design a fitness assignment mechanism when we deal with multiobjective optimization problems. Ho *et al.* [19] use a weighted-sum approach by combining multiple objectives into a single-objective function. However, in order to obtain good solutions using the weighted-sum approach, domain knowledge and large computation costs are required for determining a set of good weight values.

Here we adopt adaptive-weight Genetic Algorithm (awGA), which is an improved adaptive-weight fitness assignment approach proposed by Gen *et al.* [18]. This algorithm considers the disadvantages of weighted-sum approach and Pareto ranking-based approach. This utilizes some useful information from the current population to generate an adaptive weight for each objective, and thereby exerts a search pressure towards the ideal point. To solve multiobjective problems, we first define extreme points of each objective: the maximum extreme point $z^+ \leftarrow \{z_1^{\max}, z_2^{\max}, \ldots, z_q^{\max}\}$ and the minimum extreme point $z^- \leftarrow \{z_1^{\min}, z_2^{\min}, \ldots, z_q^{\min}\}$ in criteria space, where $z_p^{\max}$ and $z_p^{\min}$ $\forall$ $p=1,2,\ldots q$ are the maximum value and the minimum value respectively for pth objective in the current population. We adopted the roulette wheel selection as the supplementary operation to this interactive adaptive-weight assignment approach. The fitness assignment process is shown as follows:

```
Procedure 4: Adaptive-weight fitness assignment
input: chormosome $v_k$
output: fitness value eval($v_k$)
step 1: define two extreme points: the maximum extreme point
        $z^+$ and the minimum extreme point $z^-$ in criteria space
        as follows:
            $z^+ = \{z_1^{\max}, z_2^{\max}, \cdots, z_q^{\max}\}$    $z^- = \{z_1^{\min}, z_2^{\min}, \cdots, z_q^{\min}\}$
        where $z_p^{\max}$ and $z_p^{\min}, \forall p = 1,2,\cdots,q$ are the maximal
        value and minimal value for each objective in the current
        population. They are defined as follows:
$z_p^{\max} = \max\{f_p^k \mid k \in popSize\}$
$z_p^{\min} = \min\{f_p^k \mid k \in popSize\}$
step 2: The adaptive weight for objective p is calculated by
        the following equation:
            $w_p = \frac{1}{z_p^{\max} - z_p^{\min}}$
step 3: Calculate the fitness value for each individual.
            $eval(v_k) = \sum_{p=1}^{q} w_p^k \left(z_p - z_p^{\min}\right), \ \forall k \in popSize$
```

3.5. Overall moGA

The overall procedure for solving MHRAP is outlined as follows:

```
procedure 5: moGA for MHRAP
input: network data (V, A, C), GA parameters //C means offspring set
output: Pareto optimal solution E(t)
begin
t←0;
initialize P(t) by Improved Fixed-length Encoding;
calculate objectives z_p by Improved Fixed-length Decoding;
create Pareto E(P);
fitness eval(P) by adaptive weight approach;
while (not termination condition) do
crossover P(t) to yield C(t) by Improved Fixed-length
crossover;
mutation P(t) to yield C(t) by insertion mutation;
objectives z_p by Improved Fixed-length Decoding;
apply the iterative hill climbing method;
update Pareto E(P, C);
fitness eval(P, C) by adaptive weight approach;
select P(t+1) from P(t) and C(t) by roulette wheel selection;
t←t + 1;
end
output Pareto optimal solution E(t);
End
```

4. EXPERIMENT RESULTS

To evaluate our algorithm, we consider a simple example. We assign 10 to maximal number of employees in each stage, and salaries are different in each stage. The table below shows different salaries in different stages.

In Table **2**, we list the index of software developing stages with different salaries.

Table **3** shows the development duration effect by various combinations of employee number in each stage. This value is based on the history work achievement of employee and the experience of the manager.

To analyze this problem, we reformulate this problem as a network model.

i: index of stage, $i = 1, 2, \ldots, n$.

j: number of employees, $j = 1, 2, \ldots, m$.

t_{ij}: expected duration in each stage with different employee numbers.

c_{ij}: expected cost in each stage with different employee numbers. To calculate c_{ij}, we can multiple employee numbers by salary of different stage. (This value is not appeared in followed figure.)

A path from O to D is a resource allocation solution. We use proposed moGA to find Pareto solution set.

To measure and evaluate the efficency of proposed algorithm, we should find Pareto-optimal set S^* as a criterion set.

In order to make a large number of solutions and make a nearest distance to real Pareto front in Pareto-optimal set S^*, first we calculate the solution sets with special GA parameter settings. The experiments need a long computation time. The results are reference set S^*. Furthermore, we will assign small but reasonable GA parameter settings for comparison experiments.

In this section, the performance of moGA is compared with nsGA-II. We use these parameters in nsGA-II and moGA to find the reference solution set S^*: population size, *popSize* =100; crossover probability, p_C =0.90; mutation probability, p_M =0.90; immigration rate; Stopping criteria: max evolution generations, *maxGen*=10000. Fig. (**6**) shows the reference Pareto solution set.

We compare these two algorithms under these same GA parameter settings: population size, *popSize* =20; crossover probability, p_C =0.70; mutation probability, p_M =0.30; stopping condition, *maxGen* =500. Each simulation was run 20 times to get average result values. We denote S as the solution set of each algorithm. In this paper, the Average Distance $A^D(S)$ is used as a measure which has already used in different moGA studies [20]. $A^D(S)$ is to find an average distance of the solutions of S from S^*. Here, $A^D(S)$ is defined as follows:

Table 2. Index of Salaries in Different Stages

No. of stage (*i*)	**1**	**2**	**3**	**4**
Stage of software development	Architectural design	detailed design	coding	test
Average salary per month	3,800	3,500	2,400	2,200

Table 3. Expected Duration in each Stage with Different Employee Numbers (Month)

i \ *j*	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1	5.22	4.07	3.28	2.72	2.29	1.97	1.71	1.50	1.13	1.19
2	8.35	6.50	5.25	4.35	3.66	3.15	2.74	2.40	1.81	1.90
3	11.48	8.95	7.22	5.98	5.04	4.33	3.76	3.30	2.49	2.62
4	4.18	3.26	2.62	2.18	1.83	1.58	1.37	1.20	0.90	0.95

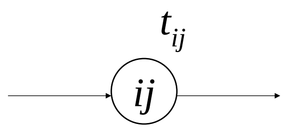


Fig. (4). Element in network model of experiment.

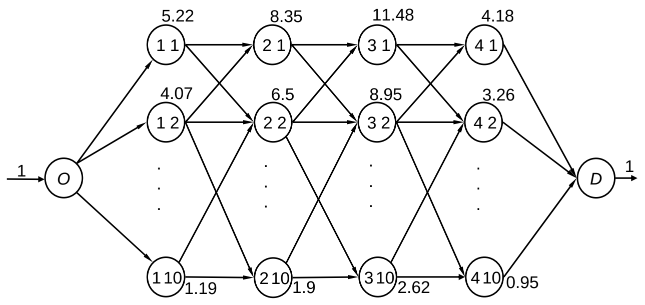


Fig. (5). Network model of test problem.

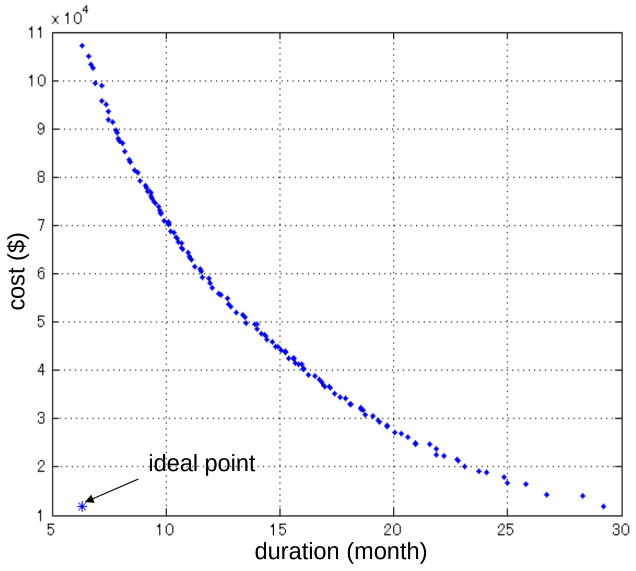


Fig. (6). Reference solutions.

$$\mathbf{A}^{\mathbf{D}}(S) = \frac{1}{|S^*|}\sum_{r\in S^*} d_x \tag{8}$$

where d_x is the distance between a current solution $x \in S$ and a reference solution r in the 2-dimensional normalized objective space, $f_q(x)$ is objective value of qth objective.

$$d_x = \min_r \left\{ \sqrt{\sum_{q=1}^{2} \left(f_q(\boldsymbol{x}) - f_q(\boldsymbol{r})\right)^2} \;\middle|\; r \in S^* \right\} \tag{9}$$

As depicted in Fig. (**7c**) and Fig. (**8**), the Pareto solution of proposed moGA is more close to a reference solution by calculating $A^D(S)$. So, the proposed algorithm is fit for solving MHRAP.

We should find the best compromised solution according to manager's preference. We propose a Factor Weight method in our algorithm. Here we use w_T, w_C as factor weight for duration objective and cost objective respectively, and $w_T + w_C = 1$. All factor weighting should be assigned by manager as his preference.

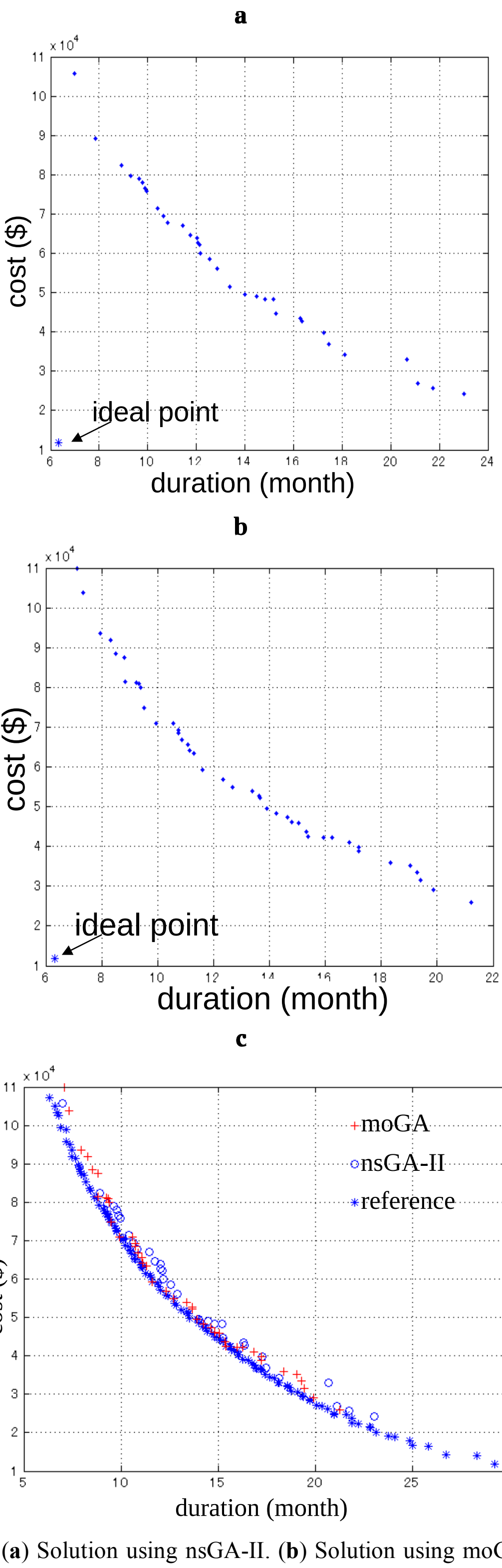


Fig. (7). (**a**) Solution using nsGA-II. (**b**) Solution using moGA. (**c**) Compared reference solution to different algorithm.

We calculate the factor value f_{xl} to decide which route is the best compromised route in Pareto set by factor value function listed bellow.

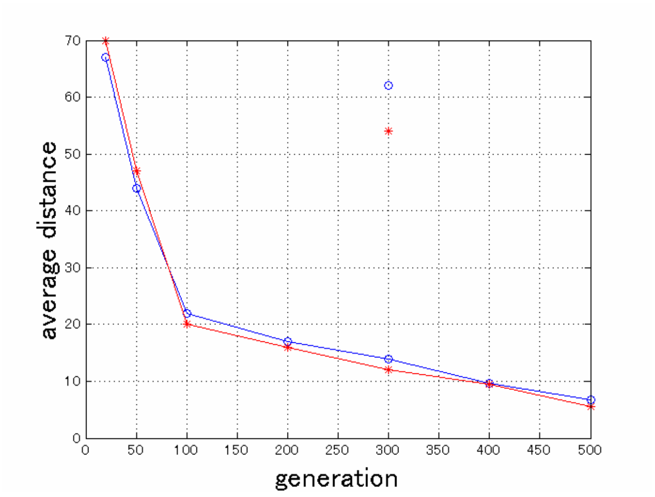


Fig. (8). Compare average distance bewteen two algorithms.

$$f_{xl} = \sqrt{w_{\mathrm{T}}\left(\frac{f_1(x)-f_1(l)}{z_1^{\max}-f_1(l)}\right)^2 + w_{\mathrm{C}}\left(\frac{f_2(x)-f_2(l)}{z_2^{\max}-f_2(l)}\right)^2} \quad (10)$$

Table 4. Comparison of Best Solutions with Different Methods (w_T=w_C=0.5)

Method	The number of Employee in each stage				Total duration (month)	Total cost (US$)
	stage 1	stage 2	stage 3	stage 4		
NSGA-II	4	4	7	5	12.66	57,000
MoGA	2	5	9	4	12.40	55,500

Here, x is the current solution and l is the ideal solution in the 2-dimensional normalized objective space. Using this method, we can get the compromised best solution. When we assign 0.5 to the two weight factors, the best compromised solution of experiment result is shown in Table **4**. After comparing different multiobjective algorithms, the proposed method has been proved to be efficient when solving this problem.

5. DISCUSSIONS AND CONCLUSIONS

In this paper, we proposed an Improved Fixed-length Encoding for designing multiobjective genetic algorithm (moGA) to solve a multi-criteria software project. By this method, we can get a pareto solution set of two objectives that include both project duration and cost of employee. An then, we used a distance method to provide the manager of the software project with the best compromised solution. By comparing different multiobjective algorithms, the proposed method has been proven to be more efficient when solving this problem.

We considered the multi-criteria software development problem with the two conflicting objectives. To solve this problem:

(1) We proposed a new chromosome representation based on Improved Fixed-length Encoding method. In addition, we gave a special decoding and encoding method.

(2) In order to ensure the population diversity characteristic in moGA, we proposed an adaptive-weight fitness assignment approach. Their elements represent that weights are adjusted adaptively based on the current generation to obtain search pressure toward the positive ideal point. It is an effective way when considering the computation time.

(3) To get the best compromised solution in a Pareto set, we proposed a weight factor method to decide which solution best fits a manager's preference.

By the experiment result, with comparison to other algorithms, we can see the efficiency of our proposed method. This efficiency is mainly due to the simple and effective coding method and fitness assignment mechanism.

In our research, the software project is a continuous time process; we have not considered the cooperative factor value when assigning a task to different numbers of workers. In future research, it will be possible to consider the task and cooperative factor value function in each stage, and decide how to allocate employees to reach the best balance of the task. Managers of software companies are more concerned about this problem.

REFERENCES

[1] Albrecht AJ. Measuring application development productivity. Proc IBM Application Development Symposium 1979; pp. 83-92.
[2] Boehm B, Clark B, Horowitz E, Westland C, Madachy R, Selby R. Cost models for future software life cycle processes: COCOMO 2.0. Ann Software Eng 1995; 1: 57-94.
[3] Boehm W, Horowitz E, Madachy R, *et al*. Software cost estimation with COCOMOII. Prentice-Hall Saddle River 2000.
[4] Sentas P, Angelis L, Stamelos I, Bleris G. Software productivity and effort prediction with ordinal regression. J Inform Software Technol 2005; 47(1): 17-29.
[5] Gray AR, MacDonell SG. A comparison of model building techniques to develop predictive equations for software metrics. Inform Software Technol 1997; 39(6): 425-37.
[6] Witting G, Finnie G. Estimating software development effort with connectionist models. Inform Software Technol 1997; 39(1): 369-476.
[7] Ho YC, McDevitt CD. Determination of optimal resource allocation for software development – An application of a software equation. Inform Manag 1990; 18(2): 79-85.
[8] Alba E, Chicano JF. Software project management with Gas. Inform Sci 2007; 177(11): 2380-401.
[9] Fonseca CM, Fleming PJ. An overview of evolutionary algorithms in multiobjective optimization. Evol Comput 1995; 3(1): 1-16.
[10] Horn J. Multicriterion decision making. In: Bäck T, Fogel DB, Michalewicz Z, Eds., Handbook of evolutionary computation. IOP Publishing and Oxford University Press, New York and Bristol, 1997; pp. F1.9:1–F1.9:15.

[11] Deb K, Pratap A, Agarwal S, Meyarivan T. A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans Evol Comput 2002; 6(2): 182-97.

[12] Lin CM, Gen M. Multiobjective resource allocation problem by multistage decision-based hybrid genetic algorithm. Appl Math Comput 2007; 187(2): 574-83.

[13] Khor EF, Lee TH. Multiobjective evolutionary algorithms and applications. London : Springer Science, 2005

[14] Munemoto M, Takai Y, Sato Y. A migration scheme for the genetic adaptive routing algorithm. In Proc IEEE Int Conf Syst Man Cybern 1998; pp. 2774-9.

[15] Ahn CW, Ramakrishna RS. A genetic algorithm for shortest path routing problem and the sizing of populations. IEEE Trans Evol Comput 2000; 6(6): 566-79.

[16] Inagaki J, Haseyama M, Kitajima H. A genetic algorithm for determining multiple routes and its applications. In Proc IEEE Int Symp Circuits Syst 1999; pp. 137-40.

[17] Holland J. Adaptation in natural and artificial systems, University of Michigan Press 1975.

[18] Gen M, Cheng R. Genetic algorithms and engineering optimization. New York: John Wiley & Sons 2000.

[19] Ho SY, Liu CC, Liu S. Design of an optimal nearest neighbor classifier using an intelligent genetic algorithm. Pattern Recog Lett 2002; 23(13): 1495-503.

[20] Deb K. Multiobjective optimization using evolutionary algorithms. Chichester, UK: Wiley 2001.

Received: April 08, 2008 Revised: April 17, 2008 Accepted: May 28, 2008