Efficient Allocation of Testing Resources for Software Module Testing Based on the Hyper-Geometric Distribution Software Reliability Growth Model

Rong-Huei Hou & Sy-Yen Kuo

Department of Electrical Engineering
National Taiwan University
Taipei, Taiwan, R.O.C.
Email: sykuo@cc.ee.ntu.edu.tw

Yi-Ping Chang

Department of Business Mathematics
Soochow University
Taipei, Taiwan, R.O.C.

Abstract

Considerable amount of testing resources is required during software module testing. In this paper, based on the Hyper-Geometric Distribution software reliability growth Model (HGDM) we investigate the following two optimal resource allocation problems in software module testing: 1) minimization of the number of software faults still undetected in the system after testing given a total amount of testing resources, and 2) minimization of the total amount of testing resources required given the number of software faults still undetected in the system after testing. Furthermore, based on the concepts of average allocation and proportional allocation, two simple allocation methods are also introduced. Experimental results show that the optimal allocation method can improve the quality and reliability of the software system much more significantly than the simple allocation methods can. Therefore, the optimal allocation method is very efficient for solving the testing resource allocation problem.

1. Introduction

A software system usually consists of many complex software modules, and the software testing phase is divided into three successive stages [1]: module testing, integration testing, and system testing. Considerable amount of testing resources is required during software module testing. These resources are man-power, CPU time, number of test cases, etc. To develop a quality and reliable software system, a project manager has to determine in advance the optimal way to allocate the testing resources for each module [2-4].

In literature, many Software Reliability Growth Models (SRGMs) which describe the relationship between the cumulative number of detected faults and the duration of the testing have been proposed [5-7]. The Hyper-Geometric Distribution software reliability growth Model (HGDM) was first proposed by Tohma *et al.* [8]. A series of studies on the HGDM have been made recently [9-14].

In this paper, based on the HGDM with logistic learning factor [12] we investigate two optimal resource allocation problems in software module testing: 1) minimization of the number of software faults still undetected in the system after testing given a total amount of testing resources, and 2) minimization of the total amount of testing resources required given the number of software faults still undetected in the system after testing. Two efficient and novel optimization algorithms based on the Lagrange multiplier method [15] for the above two problems will be proposed, respectively. Furthermore, based on the concepts of average allocation and proportional allocation, two simple allocation methods, the *average* allocation method and the *proportional* allocation method, will also be introduced, respectively. Experimental results show that the optimal allocation methods can improve the quality and reliability of the software system much more significantly than the simple allocation methods can. Therefore, the optimal allocation methods are very efficient for solving the testing resource allocation problem.

The organization of this paper is as follows. Section 2 briefly reviews the HGDM with logistic learning factor. Based on this model, two optimal resource allocation problems in software module testing are discussed in Section 3. The relationship between the optimal, average, and proportional resource allocation methods is investigated in Section 4. Numerical examples are presented for comparison in Section 5 followed by the conclusions in Section 6.

Notations

M number of software modules

0-8186-7707-4/96 $5.00

m_j, m expected number of initial software faults in module j and the whole system, respectively

t_i the i^{th} test instance, $i = 1, 2, \ldots$ where i represents the order of application

w_i number of faults newly detected or redetected by t_i

q_j, Q amount of testing resources allocated to module j and the whole system during the application of t_k, respectively

$\mathcal{Q}$ $(q_1, q_2, \ldots, q_M)$; vector of q_j

$I(A)$ $\begin{cases} 1, & \text{if } A \text{ occurs} \\ 0, & \text{otherwise} \end{cases}$

$p_{LT,j}, a_j, b_j$ parameters of the HGDM with logistic learning factor in module j

$p_{j,i}$ $p_{LT,j} I(q_j{>}0)/(1{+}e^{-q_j(a_j i+b_j)}), j{=}1,2,\ldots,M$

z_j, Z expected number of software faults undetected in module j and the whole system after the application of $t_1, t_2, \ldots, t_k$, respectively

v_j weighting factor for module j

m_j^* $m_j \prod_{i=1}^{k-1}(1 - p_{j,i})$

$C_{j,k}, C_k$ number of faults in module j and the whole system detected after the application of $t_1, t_2, \ldots, t_k$, respectively

a_j^* $a_j k + b_j,\ j = 1, 2, \ldots, M$

A_j $v_j m_j^* p_{LT,j} a_j^*,\ j = 1, 2, \ldots, M$

B_j $e^{-a_j^* q_j},\ j = 1, 2, \ldots, M$

λ Lagrange multiplier

$L(\mathcal{Q}, \lambda)$ Lagrangean

$\#$ optimal solution of the resource allocation problem

$q_j^{\#}, q_{j,avg}, q_{j,prop}$ amount of testing resources allocated to module j during the application of t_k by the optimal, average, and proportional allocation methods, respectively

$Z_{opt}, Z_{avg}, Z_{prop}$ number of software faults still undetected in the whole system estimated by the optimal, average, and proportional allocation methods, respectively

Assumptions

1. A software system is composed of M modules and each module is tested independently.
2. For each module, faults already detected so far by $t_1, t_2, \ldots, t_{k-1}$ have been collected. Based on these faults, the parameters of each module, m_j, $p_{LT,j}$, a_j, and b_j for $j = 1, 2, \ldots, M$, can be estimated [12].
3. The manager has to decide how to allocate the testing resources to each module during the application of test instance t_k.

2. HGDM with Logistic Learning Factor

In this section, we briefly review the HGDM [8–10]. At the beginning of the test/debug stage, m initial faults are resident in the program. With the application of *"test instances"*, faults can be detected by the testers. The collection of test operations performed in a unit of time (one day, one week, ...) is called a "test instance". At the end of a test instance t_i, each fault will be classified into one of the following two categories, newly detected faults or redetected faults. Some of the faults detected by t_i may have already been detected by the application of $t_1, t_2, \ldots, t_{i-1}$. Therefore, the number of faults newly detected by t_i is not necessarily equal to w_i.

The exact development of the model in terms of mathematical equations has been shown in [8–10]. The mean value function of the HGDM is given by

$$EC_i = m[1 - \prod_{j=1}^{i}(1 - \frac{w_j}{m})] = m[1 - \prod_{j=1}^{i}(1 - p_j)\ , \quad (1)$$

with $EC_0 = 0$ where $p_i = w_i/m,\ i = 1, 2, \ldots$.

Various functions for the learning factor p_i have been proposed in [8–10]. To make the HGDM more realistic and practical, a logistic learning factor based on the S-shaped learning curve was proposed in [12]. The logistic learning factor is

$$p_i = p_{LT} \frac{1}{1 + e^{-u_i(ai+b)}},\ a, b, u_i > 0, 0 < p_{LT} \leq 1, \quad (2)$$

where $u_i > 0$ represents the testing resource consumed in t_i. Note that if there is no testing resource used in t_i (i.e., $u_i = 0$), obviously no faults can be detected (i.e., $w_i = p_i = 0$). Therefore, it is reasonable that Eq.(2) can be modified to be

$$p_i = p_{LT} \frac{I(u_i > 0)}{1 + e^{-u_i(ai+b)}},\ i = 1, 2, \ldots. \quad (3)$$

3. Two Optimal Resource Allocation Problems

To improve the quality and reliability of a software system, the manager of software development has to effectively apportion the testing resources among the modules [2–4]. In this section, based on the HGDM with logistic learning factor, two optimal resource allocation problems in software module testing are discussed.

3.1. Minimizing the number of undetected faults

In this subsection, we study the following problem. Assuming that the total amount of testing resources for software module testing is given, the manager has to allocate these resources to each module to minimize the number of software faults still undetected in the whole system after the application of $t_1, t_2, \ldots, t_k$ [2–4].

Let the amount of resources allocated to module j during the application of t_k be q_j, from Eq.(3) it is obvious that

$$p_{j,k}=p_{LT,j}\frac{I(q_j>0)}{1+e^{-q_j(a_jk+b_j)}},a_j,b_j>0,0<p_{LT,j}\leq 1. \quad (4)$$

Let $C_{j,k}$ denote the number of software faults in module j already detected so far by $t_1,t_2,\ldots,t_k$. From Eqs.(1) and (4), the expected value of $C_{j,k}$ denoted by $EC_{j,k}$ is [8–10]

$$\begin{cases} EC_{j,0}=0, \\ EC_{j,k}=m_j[1-\prod_{i=1}^{k}(1-p_{j,i})], \ j=1,2,\ldots,M. \end{cases} \quad (5)$$

Therefore, the number of software faults still undetected in module j after the application of $t_1,t_2,\ldots,t_k$ can be estimated by Eqs.(4) and (5) as:

$$\begin{aligned} z_j = m_j - EC_{j,k} &= \{m_j\prod_{i=1}^{k-1}(1-p_{j,i})\}(1-p_{j,k}) \\ &= m_j^*(1-\frac{p_{LT,j}I(q_j>0)}{1+e^{-q_j(a_jk+b_j)}}). \quad (6) \end{aligned}$$

Note that m_j^* represents the expected number of software faults still undetected in module j after the application of $t_1,t_2,\ldots,t_{k-1}$. Suppose the total amount of testing resources for module testing is Q, from Eq.(6) this allocation problem can formulated as:

$$\begin{aligned} \text{Min} \quad & \sum_{j=1}^{M} v_j m_j^*(1-\frac{p_{LT,j}I(q_j>0)}{1+e^{-q_j(a_jk+b_j)}}) \quad (7) \\ \text{subject to} \quad & \sum_{j=1}^{M} q_j = Q, \\ & q_j \geq 0, \ \ j=1,2,\ldots,M. \end{aligned}$$

Note that v_j is a weighting factor to represent the relative importance of a fault detected from module j in the future. This problem can be easily rewritten in a more compact formulation (called Problem P1):

$$\begin{aligned} \text{P1}: \ \text{Max} \quad & f(\mathcal{Q})=\sum_{j=1}^{M}\frac{v_j m_j^* p_{LT,j}I(q_j>0)}{1+e^{-q_j(a_jk+b_j)}} \quad (8) \\ \text{subject to} \quad & \sum_{j=1}^{M} q_j = Q, \quad (9) \\ & q_j \geq 0, \ j=1,2,\ldots,M. \end{aligned}$$

Ignoring the constraint $q_j \geq 0$ and using the Lagrange method [15], Problem P1 is equivalent to find the maximum of

$$L_1(\mathcal{Q},\lambda)=\sum_{j=1}^{M}\frac{v_j m_j^* p_{LT,j}I(q_j>0)}{1+e^{-a_j^*q_j}}+\lambda\left(\sum_{j=1}^{M}q_j-Q\right).$$

For convenience, we consider another problem (called Problem $P1'$):

$$\begin{aligned} \text{P1}': \ \text{Max} \quad & f(\mathcal{Q})=\sum_{j=1}^{M}\frac{v_j m_j^* p_{LT,j}}{1+e^{-q_j(a_jk+b_j)}} \\ \text{subject to} \quad & \sum_{j=1}^{M} q_j = Q, \\ & q_j \geq 0, \ j=1,2,\ldots,M. \end{aligned}$$

Ignoring the constraint $q_j \geq 0$ and using the Lagrange method [15], Problem $P1'$ is equivalent to find the maximum of

$$L_2(\mathcal{Q},\lambda)=\sum_{j=1}^{M}\frac{v_j m_j^* p_{LT,j}}{1+e^{-a_j^*q_j}}+\lambda\left(\sum_{j=1}^{M}q_j-Q\right). \quad (10)$$

According to the Kuhn–Tucker conditions [15], the necessary conditions A1–A3 for a maximum of Eq.(10) to exist are as follows:

A1: $\frac{\partial L_2(\mathcal{Q},\lambda)}{\partial q_j}=0, \ j=1,2,\ldots,M.$

A2: $\lambda<0.$

A3: $\sum_{j=1}^{M} q_j = Q.$

For Condition A1, from Eq.(10) we have

$$\frac{\partial L_2(\mathcal{Q},\lambda)}{\partial q_j}=\frac{v_j m_j^* p_{LT,j}a_j^*e^{-a_j^*q_j}}{(1+e^{-a_j^*q_j})^2}+\lambda=0. \quad (11)$$

That is,

$$\frac{A_jB_j}{(1+B_j)^2}+\lambda=0, \ j=1,2,\ldots,M. \quad (12)$$

Since $a_j^*>0$ and q_j **should be larger than or equal to zero** for $j=1,2,\ldots,M$, we have $A_j>0$ and $0<B_j\leq 1$ for $j=1,2,\ldots,M$. Therefore, from Eq.(12) we have $\lambda<0$ (i.e., Condition A2 is satisfied) and

$$\frac{B_j}{(1+B_j)^2}=-\frac{\lambda}{A_j}. \quad (13)$$

Since $0<B_j\leq 1$, from Eq.(13) we can easily obtain

$$0<\frac{B_j}{(1+B_j)^2}\leq\frac{1}{4}, \ j=1,2,\ldots,M. \quad (14)$$

That is,

$$-\frac{A_j}{4}\leq\lambda<0, \ j=1,2,\ldots,M. \quad (15)$$

From Eq.(13), we have

$$\frac{\lambda}{A_j}(1+B_j)^2+B_j=0. \tag{16}$$

Since $0<B_j\le 1$ and $-1/4\le\lambda/A_j<0$, the root of Eq.(16) is

$$B_j=-1-\frac{1-\sqrt{1+4\lambda/A_j}}{2\lambda/A_j}. \tag{17}$$

Therefore, from Eq.(17) we have

$$q_j=-\frac{1}{a_j^*}\ln\left(-1-\frac{1-\sqrt{1+4\lambda/A_j}}{2\lambda/A_j}\right). \tag{18}$$

Let

$$S_L(\lambda) = \sum_{j=1}^{L}\frac{1}{a_j^*}\ln\left(-1-\frac{1-\sqrt{1+4\lambda/A_j}}{2\lambda/A_j}\right) \tag{19}$$

$$= \sum_{j=1}^{L} s_j(\lambda),\ L=1,2,\cdots,M, \tag{20}$$

where

$$s_j(\lambda)=\frac{1}{a_j^*}\ln\left(-1-\frac{1-\sqrt{1+4\lambda/A_j}}{2\lambda/A_j}\right). \tag{21}$$

By simple calculation, we have the following lemma.

Lemma 1.

(i) $s_j(\lambda)$ is decreasing in λ;

(ii) $\lim_{\lambda\to o^-} s_j(\lambda)=-\infty$;

(iii) $\lim_{\lambda\to -A_j/4} s_j(\lambda)=0$, where $s_j(\lambda)$ is given in Eq.(21). □

Since $S_L(\lambda)=\sum_{j=1}^{L}s_j(\lambda)$ and $s_j(\lambda)$ is decreasing in λ, $S_L(\lambda)$ is decreasing in λ for $L=1,2,\cdots,M$. Since $\lambda\ge -A_j/4$ for $j=1,2,\ldots,M$, for convenience we rearrange the indices of A_j's such that $A_1\ge A_2\ge\ldots\ge A_M$. Moreover, from Eq.(21) we have $s_j(\lambda)<0$ for all j, and $S_M(\lambda)$ and $S_{M-1}(\lambda)$ are shown in Figure 1, where $\alpha_M=\max_{\lambda\ge -A_M/4} S_M(\lambda)=S_M(-A_M/4)$ and $\alpha_{M-1}=\max_{-A_M/4>\lambda\ge -A_{M-1}/4} S_{M-1}(\lambda)=S_{M-1}(-A_{M-1}/4)$. To satisfy the constraint $\sum_{j=1}^{M}q_j=Q$ (i.e., Condition A3) and based on Figure 1, the main steps for finding the maximum of Problem P1 is depicted in the following:

Step 1. Rearrange the indices of A_j's such that $A_1\ge A_2\ge\ldots\ge A_M$.

Step 2. (i) If $\max_{\lambda\ge -A_M/4} S_M(\lambda)=\sum_{j=1}^{M}\frac{1}{a_j^*}\ln\left(-1-\frac{1-\sqrt{1-A_M/A_j}}{-\frac{2}{A_j}\frac{A_M}{4}}\right)\ge -Q$, there exists a unique root $\lambda^{\#}$ (here, $\lambda^{\#}\ge -A_M/4$) such that $S_M(\lambda^{\#})=-Q$ and hence we are done. The optimal solution $\mathcal{Q}^{\#}$for Problem $P1'$ is

$$q_j^{\#}=-\frac{1}{a_j^*}\ln\left(-1-\frac{1-\sqrt{1+4\lambda^{\#}/A_j}}{2\lambda^{\#}/A_j}\right), j=1,2,\ldots,M.$$

Since $q_j^{\#}>0, j=1,2,\ldots,M$, the optimal solution for Problem P1 is also $\mathcal{Q}^{\#}$.

(ii) If $\max_{\lambda\ge -A_M/4} S_M(\lambda)=\sum_{j=1}^{M}\frac{1}{a_j^*}\ln\left(-1-\frac{1-\sqrt{1-A_M/A_j}}{-\frac{2}{A_j}\frac{A_M}{4}}\right)<-Q$, we have $\lambda<-A_M/4$ and hence from Eq.(12)

$$\frac{A_M B_M}{(1+B_M)^2}+\lambda<\frac{A_M}{4}-\frac{A_M}{4}=0;$$

that is,

$$\frac{\partial L_2(\mathcal{Q},\lambda)}{\partial q_M}<0. \tag{22}$$

Therefore, $L_2(\mathcal{Q},\lambda)$ is decreasing in q_M. Furthermore, it can be easily shown that $L_1(\mathcal{Q},\lambda)$ is also decreasing in q_M. To maximize $L_1(\mathcal{Q},\lambda)$, let $q_M^{\#}=q_M=0$ (since $q_M\ge 0$) and go to Step 3.

Step 3. (i) If $\max_{-A_M/4>\lambda\ge -A_{M-1}/4} S_{M-1}(\lambda)=\sum_{j=1}^{M-1}\frac{1}{a_j^*}\ln\left(-1-\frac{1-\sqrt{1-A_{M-1}/A_j}}{-\frac{1}{2}\frac{A_{M-1}}{A_j}}\right)\ge -Q$, there exists a unique root $\lambda^{\#}$ (here, $-A_M/4>\lambda^{\#}\ge -A_{M-1}/4$) such that $S_{M-1}(\lambda^{\#})=-Q$ and hence we are done. The optimal solution $\mathcal{Q}^{\#}$ for Problem P1 is

$$\begin{cases} q_j^{\#}=-\frac{1}{a_j^*}\ln\left(-1-\frac{1-\sqrt{1+4\lambda^{\#}/A_j}}{2\lambda^{\#}/A_j}\right), j=1,2,\ldots,M-1; \\ q_M^{\#}=0, \end{cases}$$

(ii) If $\max_{-A_M/4>\lambda\ge -A_{M-1}/4} S_{M-1}(\lambda)=\sum_{j=1}^{M-1}\frac{1}{a_j^*}\ln\left(-1-\frac{1-\sqrt{1-A_{M-1}/A_j}}{-\frac{1}{2}\frac{A_{M-1}}{A_j}}\right)<-Q$, we have $\lambda<-A_{M-1}/4$ and then from Eq.(12) we have

$$\frac{A_{M-1}B_{M-1}}{(1+B_{M-1})^2}+\lambda<\frac{A_{M-1}}{4}-\frac{A_{M-1}}{4}=0;$$

that is,

$$\frac{\partial L_2(Q,\lambda)}{\partial q_{M-1}} < 0.$$

Therefore,

$$\sum_{j=1}^{M-1} \frac{v_j m_j^* p_{LT,j}}{1+e^{-a_j^* q_j}} + \lambda\left(\sum_{j=1}^{M-1} q_j - Q\right)$$

is decreasing in q_{M-1}. Furthermore, it can be easily shown that

$$\sum_{j=1}^{M-1} \frac{v_j m_j^* p_{LT,j} I(q_j>0)}{1+e^{-a_j^* q_j}} + \lambda\left(\sum_{j=1}^{M-1} q_j - Q\right) \quad (23)$$

is decreasing in q_{M-1}. In order to maximize Eq.(23), let $q_{M-1}^{\#} = q_{M-1} = 0$ and go to next step (the subsequent steps are similar to Step 3 to find $\ell^{\#}$ and a root $\lambda^{\#}$ such that $S_{M-\ell^{\#}}(\lambda^{\#}) = -Q$). □

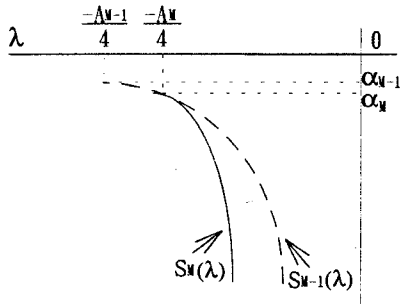

Figure 1. Functions of $S_M(\lambda)$ and $S_{M-1}(\lambda)$.

Based on the above description, we propose an efficient and novel optimization algorithm (called Algorithm 1) to determine the optimal solution for the resource allocation Problem P1.

Algorithm 1:

Step 1. Rearrange the indices of A_j's such that $A_1 \geq A_2 \geq \ldots \geq A_M \geq A_{M+1} = 0$ (here, $A_{M+1} = 0$ is a dummy variable).

Step 2. Let $\ell = 0$.

Step 3. If $\sum_{j=1}^{M-\ell} \frac{1}{a_j^*} \ln\left(-1 - \frac{1-\sqrt{1-A_{M-\ell}/A_j}}{-A_{M-\ell}/(2A_j)}\right) \geq -Q$, (let $\ell^{\#} = \ell$) there exists a unique root $\lambda^{\#}$ such that $S_{M-\ell^{\#}}(\lambda^{\#}) = -Q$ and the optimal solution $Q^{\#}$ is

$$q_j^{\#} = -\frac{1}{a_j^*} \ln\left(-1 - \frac{1-\sqrt{1+4\lambda^{\#}/A_j}}{2\lambda^{\#}/A_j}\right),$$

$j = 1, 2, \ldots, M - \ell^{\#}$; otherwise, let $q_{M-\ell+1}^{\#} = q_{M-\ell+1} = 0$, set $\ell \leftarrow \ell + 1$, and go to Step 3. □

In general, such linearly constrained minimization problem can be solved by the successive quadratic programming method based on the iterative formulation and solution of quadratic programming subproblems [16]. However, this method is complex. On the contrary, in Algorithm 1 the first $\ell^{\#} - 1$ iterations just do simple numerical comparisons. In the $\ell^{\#}$th iteration, since $S_{M-\ell^{\#}}(\lambda)$ is decreasing in λ, there must exist a unique root $\lambda^{\#}$ such that $S_{M-\ell^{\#}}(\lambda^{\#}) = -Q$ and hence the optimal solution $Q^{\#}$ can be obtained. Note that $\lambda^{\#}$ can be easily obtained by simple numerical analysis. Therefore, Algorithm 1 is quite simple, efficient, and novel. By Lemma 1 it can be easily shown that this algorithm always converges in, at worst, $M-1$ steps. The value of the objective function given by Eq.(7) with the optimal solution $Q^{\#}$ is

$$\sum_{j=1}^{M} v_j m_j^* \left(1 - \frac{p_{LT,j} I(q_j^{\#} > 0)}{1+e^{-q_j^{\#}(a_j k + b_j)}}\right). \quad (24)$$

3.2. Minimizing the amount of testing resources

In this subsection, we consider another resource allocation problem. Assuming that the number of software faults still undetected in the whole system after the application of $t_1, t_2, \ldots, t_k$ is less than or equal to Z. The project manager has to allocate an appropriate amount of testing resources to each module during the application of t_k to minimize the total amount of testing resources required throughout module testing [2–4].

From Eq.(6), this resource allocation problem (called Problem P2) can be formulated as:

$$\text{Min} \quad \sum_{j=1}^{M} q_j$$

$$\text{subject to} \quad \sum_{j=1}^{M} v_j m_j^* \left(1 - \frac{p_{LT,j} I(q_j>0)}{1+e^{-q_j(a_j k + b_j)}}\right) \leq Z, \quad (25)$$

$$q_j \geq 0, \; j = 1, 2, \ldots, M.$$

Note that from Eq.(25) we have

$$\sum_{j=1}^{M} \lim_{q_j \to \infty} v_j m_j^* \left(1 - \frac{p_{LT,j} I(q_j > 0)}{1+e^{-q_j(a_j k + b_j)}}\right) = \sum_{j=1}^{M} v_j m_j^* (1 - p_{LT,j}).$$

That is, it is impossible to reduce the number of software faults still undetected in the whole system to $\sum_{j=1}^{M} v_j m_j^* (1 - p_{LT,j})$ after the application of test instance t_k no matter which resource allocation method is used. This phenomenon is the limitation of the HGDM with logistic learning factor model.

Ignoring the constraint $q_j \geq 0$ and using the Lagrange method [15], Problem P2 is equivalent to find the minimum of

$$L_3(\mathcal{Q},\lambda)=\sum_{j=1}^{M} q_j+\lambda\left(\sum_{j=1}^{M} v_j m_j^*\left(1-\frac{p_{LT,j}I(q_j>0)}{1+e^{-a_j^* q_j}}\right)-Z\right).$$

For convenience, we consider another problem (called Problem $P2'$):

$$\begin{aligned} \text{Min} \quad & \sum_{j=1}^{M} q_j \\ \text{subject to} \quad & \sum_{j=1}^{M} v_j m_j^*\left(1-\frac{p_{LT,j}}{1+e^{-q_j(a_j k+b_j)}}\right)\leq Z, \quad (26) \\ & q_j \geq 0,\ j=1,2,\ldots,M. \end{aligned}$$

Ignoring the constraint $q_j \geq 0$ and using the Lagrange method [15], Problem $P2'$ is equivalent to find the minimum of

$$L_4(\mathcal{Q},\lambda)=\sum_{j=1}^{M} q_j+\lambda\left(\sum_{j=1}^{M} v_j m_j^*\left(1-\frac{p_{LT,j}}{1+e^{-a_j^* q_j}}\right)-Z\right). \quad (27)$$

The necessary conditions B1–B3 for a minimum of Eq.(27) to exist are as follows [15]:

B1: $\dfrac{\partial L_4(\mathcal{Q},\lambda)}{\partial q_j}=0,\ j=1,2,\ldots,M.$

B2: $\lambda > 0.$

B3: $\sum_{j=1}^{M} v_j m_j^*\left(1-\dfrac{p_{LT,j}}{1+e^{-a_j^* q_j}}\right)=Z.$

For Condition B1, from Eq.(27) we have

$$\begin{aligned} \frac{\partial L_4(\mathcal{Q},\lambda)}{\partial q_j} &= 1-\lambda\frac{v_j m_j^* p_{LT,j} a_j^* e^{-a_j^* q_j}}{(1+e^{-a_j^* q_j})^2} \quad (28) \\ &= 1-\lambda\frac{A_j B_j}{(1+B_j)^2}=0, j=1,2,\ldots,M \quad (29) \end{aligned}$$

Since q_j should be **larger than or equal to** zero, we have $0 < B_j \leq 1$ for $j=1,2,\ldots,M$. Therefore, from Eq.(29) we have $\lambda > 0$ (i.e., Condition B2 is satisfied) and

$$\frac{B_j}{(1+B_j)^2}=\frac{1}{\lambda A_j}. \quad (30)$$

Since $0 < B_j \leq 1$, from Eq.(30) we can easily obtain

$$\lambda \geq \frac{4}{A_j},\ j=1,2,\ldots,M, \quad (31)$$

and the root of Eq.(30) is

$$B_j=-1-\frac{1-\sqrt{1+4\lambda_j^*}}{2\lambda_j^*}, \quad (32)$$

where $\lambda_j^*=-1/(\lambda A_j)$. Therefore, we have

$$q_j=-\frac{1}{a_j^*}\ln\left(-1+\frac{1-\sqrt{1-4/(\lambda A_j)}}{2/(\lambda A_j)}\right), j=1,2,\ldots,M. \quad (33)$$

Let

$$\begin{aligned} R_L(\lambda) &= \sum_{j=1}^{L} v_j m_j^*\left(1-\frac{p_{LT,j}}{1+e^{-a_j^* q_j}}\right) \quad (34) \\ &= \sum_{j=1}^{L} v_j m_j^*\left(1-p_{LT,j}\frac{2/(\lambda A_j)}{1-\sqrt{1-4/(\lambda A_j)}}\right) \\ &= \sum_{j=1}^{L} r_j(\lambda),\ L=1,2,\cdots,M, \end{aligned}$$

where

$$r_j(\lambda)=v_j m_j^*\left(1-p_{LT,j}\frac{2/(\lambda A_j)}{1-\sqrt{1-4/(\lambda A_j)}}\right). \quad (35)$$

By simple calculation, we have the following lemma.

Lemma 2.

(i) $r_j(\lambda)$ is decreasing in λ;

(ii) $\lim_{\lambda\to\infty} r_j(\lambda)=v_j m_j^*(1-p_{LT,j})$;

(iii) $\lim_{\lambda\to 4/A_j} r_j(\lambda)=v_j m_j^*\left(1-\dfrac{1}{2p_{LT,j}}\right)$, where $r_j(\lambda)$ is given in Eq.(35). □

Since $\lambda \geq 4/A_j$ for $j=1,2,\ldots,M$, for convenience we rearrange the indices of A_j's such that $A_1 \geq A_2 \geq \ldots \geq A_M$. By Lemma 2 we have

$$\begin{aligned} &\sum_{j=1}^{M} v_j m_j^*(1-p_{LT,j}) \leq R_M(\lambda) \leq \\ &\sum_{j=1}^{M} v_j m_j^*\left(1-p_{LT,j}\frac{A_M/(2A_j)}{1-\sqrt{1-A_M/A_j}}\right). \quad (36) \end{aligned}$$

Since $R_L(\lambda)=\sum_{j=1}^{L} r_j(\lambda)$ and $r_j(\lambda)$ is decreasing in λ, $R_L(\lambda)$ is decreasing in λ for $L=1,2,\cdots,M$. Furthermore, from Eq.(35) we have $r_j(\lambda) > 0$ for all j, and from Eq.(36) $R_M(\lambda)$ and $R_{M-1}(\lambda)$ are shown in Figure 2, where $\alpha_M=\sum_{j=1}^{M} v_j m_j^*\left(1-p_{LT,j}\frac{A_M/(2A_j)}{1-\sqrt{1-A_M/A_j}}\right)$, $\alpha_{M-1}=\sum_{j=1}^{M-1} v_j m_j^*\left(1-p_{LT,j}\frac{A_{M-1}/(2A_j)}{1-\sqrt{1-A_{M-1}/A_j}}\right)$, $\beta_M=\sum_{j=1}^{M} v_j m_j^*(1-p_{LT,j})$, and $\beta_{M-1}=\sum_{j=1}^{M-1} v_j m_j^*(1-p_{LT,j})$. To satisfy the

constraint $\sum_{j=1}^{M} v_j m_j^*(1-p_{LT,j}/(1+e^{-a_j^* q_j})) = Z$ (i.e., Condition B3) and based on Figure 2, the main steps for finding the minimum of Problem P2 are depicted in the following:

Step 1. Rearrange the indices of A_j's such that $A_1 \geq A_2 \geq \ldots \geq A_M$.

Step 2. (i) If $\sum_{j=1}^{M} v_j m_j^*(1-p_{LT,j}) \geq Z$, it is impossible to reduce the number of still undetected software faults to Z after the application of test instance t_k.

(ii) If $\sum_{j=1}^{M} v_j m_j^*(1-p_{LT,j}) < Z \leq \sum_{j=1}^{M} v_j m_j^*\left(1-p_{LT,j}\frac{A_M/(2A_j)}{1-\sqrt{1-A_M/A_j}}\right)$, there exists a unique root $\lambda^{\#}$ (here, $\lambda^{\#} \geq 4/A_M$) such that $R_M(\lambda^{\#}) = Z$ and hence we are done. The optimal solution $\mathcal{Q}^{\#}$ for Problem P2 is

$$q_j^{\#} = -\frac{1}{a_j^*}\ln\left(-1+\frac{1-\sqrt{1-4/(\lambda^{\#}A_j)}}{2/(\lambda^{\#}A_j)}\right), j=1,2,\ldots,M.$$

Since $\{(q_1,q_2,\ldots,q_M) : (q_1,q_2,\ldots,q_M)$ satisfies Eq.(25)$\}$ is the subset of $\{(q_1,q_2,\ldots,q_M)$: $(q_1,q_2,\ldots,q_M)$ satisfies Eq.(26)$\}$ and $q_j^{\#} > 0, j = 1,2,\ldots,M$, the optimal solution for Problem P2 is also $\mathcal{Q}^{\#}$.

(iii) If $\sum_{j=1}^{M} v_j m_j^*\left(1-p_{LT,j}\frac{A_M/(2A_j)}{1-\sqrt{1-A_M/A_j}}\right) < Z$, we have $\lambda < 4/A_M$ and hence from Eq.(29) we have

$$1-\lambda\frac{A_M B_M}{(1+B_M)^2} \geq 1-\frac{4}{A_M}\frac{A_M B_M}{(1+B_M)^2} \geq 0;$$

that is,

$$\frac{\partial L_4(\mathcal{Q},\lambda)}{\partial q_M} > 0. \quad (37)$$

Therefore, $L_4(\mathcal{Q},\lambda)$ is increasing in q_M. Furthermore, it can be easily shown that $L_3(\mathcal{Q},\lambda)$ is also increasing in q_M. To minimize $L_3(\mathcal{Q},\lambda)$, let $q_M^{\#} = q_M = 0$ and go to Step 3.

Step 3. (i) If $\sum_{j=1}^{M-1} v_j m_j^*(1-p_{LT,j}) \geq Z - v_M m_M^*$, it is impossible to reduce the number of still undetected software faults to Z after the application of test instance t_k.

(ii) If $\sum_{j=1}^{M-1} v_j m_j^*(1-p_{LT,j}) < Z - v_M m_M^* \leq \sum_{j=1}^{M-1} v_j m_j^*\left(1-p_{LT,j}\frac{A_{M-1}/(2A_j)}{1-\sqrt{1-A_{M-1}/A_j}}\right)$, there exists a unique root $\lambda^{\#}$ (here, $4/A_M > \lambda^{\#} \geq 4/A_{M-1}$) such that $R_{M-1}(\lambda^{\#}) = Z - v_M m_M^*$ and hence we are done. The optimal solution $\mathcal{Q}^{\#}$ for Problem P2 is

$$\begin{cases} q_j^{\#} = -\frac{1}{a_j^*}\ln\left(-1+\frac{1-\sqrt{1-4/(\lambda^{\#}A_j)}}{2/(\lambda^{\#}A_j)}\right), j=1,2,\ldots,M-1; \\ q_M^{\#}=0, \end{cases}$$

(iii) If $\sum_{j=1}^{M-1} v_j m_j^*\left(1-p_{LT,j}\frac{A_{M-1}/(2A_j)}{1-\sqrt{1-A_{M-1}/A_j}}\right) < Z - v_M m_M^*$, we have $\lambda < 4/A_{M-1}$ and then from Eq.(29) we have

$$1-\lambda\frac{A_{M-1}B_{M-1}}{(1+B_{M-1})^2} \geq 1-\frac{4}{A_{M-1}}\frac{A_{M-1}B_{M-1}}{(1+B_{M-1})^2} \geq 0;$$

that is,

$$\frac{\partial L_4(\mathcal{Q},\lambda)}{\partial q_{M-1}} > 0.$$

Therefore,

$$\sum_{j=1}^{M-1} q_j + \lambda\{\sum_{j=1}^{M-1} v_j m_j^*(1-\frac{p_{LT,j}}{1+e^{-a_j^* q_j}})\}$$
$$-\lambda(Z - v_M m_M^*)$$

is increasing in q_{M-1}. Furthermore, it can be easily shown that

$$\sum_{j=1}^{M-1} q_j + \lambda\{\sum_{j=1}^{M-1} v_j m_j^*(1-\frac{p_{LT,j}I(q_j>0)}{1+e^{-a_j^* q_j}})\}$$
$$-\lambda(Z - v_M m_M^*) \quad (38)$$

is increasing in q_{M-1}. To minimize Eq.(38), let $q_{M-1}^{\#} = q_{M-1} = 0$ and go to next step (the subsequent steps are similar to Step 3). □

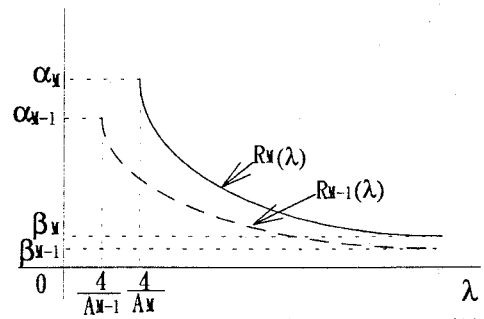

Figure 2. Functions of $R_M(\lambda)$ and $R_{M-1}(\lambda)$.

Based on above description, we propose an efficient and novel optimization algorithm (called Algorithm 2) to determine the optimal solution for the resource allocation Problem P2.

Algorithm 2:

Step 1. Rearrange the indices of A_j's such that $A_1 \geq A_2 \geq \ldots \geq A_M$.

Step 2. Let $\ell = 0$ and $Z^* = 0$.

Step 3. (i) If $\sum_{j=1}^{M-\ell} v_j m_j^*(1 - p_{LT,j}) \geq Z - Z^*$, it is impossible to reduce the number of still undetected software faults to Z after the application of test instance t_k.

(ii) If $\sum_{j=1}^{M-\ell} v_j m_j^*(1 - p_{LT,j}) < Z - Z^* \leq \sum_{j=1}^{M-\ell} v_j m_j^* \left(1 - p_{LT,j} \frac{A_{M-\ell}/(2A_j)}{1-\sqrt{1-A_{M-\ell}/A_j}}\right)$, (letting $\ell^\# = \ell$) there exists a unique root $\lambda^\#$ such that $R_{M-\ell^\#}(\lambda^\#) = Z - Z^*$ and the optimal solution $\mathcal{Q}^\#$ is

$$q_j^\# = -\frac{1}{a_j^*} \ln\left(-1 + \frac{1-\sqrt{1-4/(\lambda^\# A_j)}}{2/(\lambda^\# A_j)}\right), j = 1, 2, \ldots, M-\ell^\#;$$

otherwise, let $q_{M-\ell+1}^\# = q_{M-\ell+1} = 0$, set $Z^* \leftarrow Z^* + v_{M-\ell} m_{M-\ell}^*$, set $\ell \leftarrow \ell + 1$, and go to Step 3. □

In Algorithm 2 the first $\ell^\# - 1$ iterations just do simple numerical comparisons. In the $\ell^\#$th iteration, since $R_{M-\ell^\#}(\lambda)$ is decreasing in λ, there must exist a unique root $\lambda^\#$ such that $R_{M-\ell^\#}(\lambda^\#) = Z - Z^*$ and hence the optimal solution $\mathcal{Q}^\#$ can be obtained. Note that $\lambda^\#$ can be easily obtained by simple numerical analysis. Similar to the discussion in Subsection 3.1, Algorithm 2 is, therefore, also quite simple, efficient, and novel. Furthermore, by Lemma 2 it can be easily shown that this algorithm always converges in, at worst, $M - 1$ steps. The value of the objective function given by Eq.(25) with the optimal solution $\mathcal{Q}^\#$ is

$$\sum_{j=1}^{M} q_j^\# = \sum_{j=1}^{M-\ell^\#} -\frac{1}{a_j^*} \ln\left(-1 + \frac{1-\sqrt{1-4/(\lambda^\# A_j)}}{2/(\lambda^\# A_j)}\right) \quad (39)$$

4. Relationship between Optimal, Average, and Proportional Allocation Methods

In Section 3.1, we discuss the problem of optimal resource allocation if the total amount of testing resources for module testing is specified. In this section, based on the concepts of average allocation and proportional allocation, we introduce two simple resource allocation methods, the *average* allocation method and the *proportional* allocation method, respectively. Furthermore, we also investigate the relationship between these three allocation methods.

A resource allocation method is called an *average* allocation method if the testing resources are allocated to each module evenly. Let $q_{j,avg}$ be the amount of testing resources allocated to module j during the application of t_k by the average allocation method, and then we have

$$q_{j,avg} = \frac{Q}{M}, \; j = 1, 2, \ldots, M. \quad (40)$$

Let Z_{avg} be the number of still undetected faults in the whole system after the application of t_k by the average allocation method, and then from Eq.(6) we have

$$Z_{avg} = \sum_{j=1}^{M} v_j m_j^* \left(1 - \frac{p_{LT,j} I(q_{j,avg} > 0)}{1 + e^{-q_{j,avg} a_j^*}}\right). \quad (41)$$

A resource allocation method is called a *proportional* allocation method if the amount of testing resources allocated to module j is proportional to the number of still undetected faults in module j. Let $q_{j,prop}$ be the amount of testing resources allocated to module j during the application of t_k by the proportional allocation method, and then we have $q_{j,prop}$

$$q_{j,prop} = Q \frac{m_j^*}{\sum_{i=1}^{M} m_i^*}, \quad j = 1, 2, \ldots, M. \quad (42)$$

Note that if $m_1^* = m_2^* = \ldots = m_M^*$, the proportional allocation method is equivalent to the average allocation method. Let Z_{prop} be the number of still undetected faults in the whole system after the application of t_k by the proportional allocation method, and then from Eq.(6) we have

$$Z_{prop} = \sum_{j=1}^{M} v_j m_j^* \left(1 - \frac{p_{LT,j} I(q_{j,prop} > 0)}{1 + e^{-q_{j,prop} a_j^*}}\right). \quad (43)$$

The optimal allocation method is said to be *better* than the average and the proportional methods since $Z_{opt} \leq Z_{avg}$ and $Z_{opt} \leq Z_{prop}$. However, we are also interested in how *much better* the optimal allocation method is. In the following, we investigate the value of Z_{opt}/Z_{prop} under the case $p_{LT,j} = p_{LT}$ for $j = 1, 2, \ldots, M$.

From Eqs.(24) and (43), we have

$$g(p_{LT}) = \frac{Z_{opt}}{Z_{prop}}$$

$$= \frac{\sum_{j=1}^{M} v_j m_j^* (1 - p_{LT} I(q_j^\# > 0)/(1 + e^{-q_j^\# a_j^*}))}{\sum_{j=1}^{M} v_j m_j^* (1 - p_{LT} I(q_{j,prop} > 0)/(1 + e^{-q_{j,prop} a_j^*}))}.$$

With the fact of $Z_{opt} \leq Z_{prop}$ and some manipulations, we have

$$\frac{\partial g(p_{LT})}{\partial p_{LT}} \leq 0. \quad (44)$$

That is, Z_{opt}/Z_{prop} is decreasing in p_{LT}, which means that the larger the p_{LT} is, the smaller the ratio of the number of still undetected faults by the optimal allocation method to that by the proportional method

is. In other words, as p_{LT} increases, the optimal allocation method is *much better* than the proportional allocation method. Furthermore, since p_{LT} represents the skill of detecting faults [12], from Eq.(44) the following can be observed: the better the skill of detecting faults, the better the optimal allocation method (compared with the proportional allocation method).

By the same argument on Z_{opt}/Z_{prop}, we have the property that Z_{opt}/Z_{prop} is decreasing in p_{LT}. That is, the better the skill of detecting faults, the better the optimal allocation method (compared with the average allocation method).

5. Numerical Examples

Consider a software system consisting of 5 modules (i.e., $M = 5$). Suppose there are 4 test instances t_1, t_2, t_3, t_4 (i.e., $k = 5$) applied during module testing so far. For each module, the parameters a_j, b_j, m_j, and $p_{LT,j}$, $j = 1, 2, \ldots, 5$ can be estimated by using the software failure data, and then the estimated value of m_j^* can be obtained. The estimated values of m_j^*, a_j, b_j, $p_{LT,j}$ and the weighting factor v_j are assumed and listed in Table 1. Consequently, the total number of still undetected faults after the application of t_1, t_2, t_3, t_4 is assumed to be 200 (since $\sum_{j=1}^{5} m_j^* = 200$).

Example 1. Minimizing the total number of undetected faults

Suppose the total amount of testing resources Q for module testing during the application of t_5 is $20k$ (i.e., 20,000) man-hours. The manager has to allocate the $20k$ man-hours to the 5 modules to minimize the total number of still undetected faults after the application of t_5. Using the values of m_j^*, a_j, b_j, $p_{LT,j}$, and v_j in Table 1, the optimal solutions $q_j^{\#}$ estimated by Algorithm 1 are shown in Table 1. Consequently, the total number of still undetected software faults after the application of t_5 estimated by Eq.(24) is $Z_{opt} = 115$. That is, the total number of still undetected faults is anticipated to be reduced from 200 to 115 by using the testing resources of $20k$ man-hours. The reduction in the undetected software faults is about 42.5%.

Table 1. Allocated Testing Resources $q_j^{\#}$ for Minimizing undetected Software Faults.
(Q is assumed to be $20k$ man-hours)

Module	m_j^*	a_j	b_j	$p_{LT,j}$	v_j	$q_j^{\#}$	z_j
1	50	0.02	0.1	1.0	1	12.79	3.6
2	45	0.08	0.2	0.6	1	5.17	19.2
3	40	0.2	1	0.3	1	1.76	28.3
4	35	0.8	5	0.03	1	0.28	34
5	30	2.0	10	0.003	1	0	30

$Z_{opt} = \sum_{j=1}^{5} z_j = 115$ (42.5% reduction)

Based on the average allocation method, from Eq.(40) we have $q_{1,avg} = q_{2,avg} = \ldots = q_{5,avg} = 4$ and then from Eq.(41) we have $Z_{avg} = 128$. Furthermore, based on the proportional allocation method, from Eq.(42) we have $q_{1,prop} = 5$, $q_{2,prop} = 4.5$, $q_{3,prop} = 4$, $q_{4,prop} = 3.5$, and $q_{5,prop} = 3$. Therefore, from Eq.(43) we have $Z_{prop} = 125$. Obviously, the optimal allocation method is better than the proportional method and the average method. Furthermore, the optimal resource allocation method is very efficient in this example since the difference of $Z_{prop} - Z_{opt}$ is significant.

In the following, consider the case $p_{LT,j} = p_{LT}$ for $j = 1, 2, \ldots, 5$ and the estimated values of m_j^*, a_j, b_j and v_j are also listed in Table 1. The curves of Z_{opt}/Z_{prop} and Z_{opt}/Z_{avg} vs. p_{LT} are depicted in Figure 3. As $p_{LT} = 0.1$, from Figure 3 we have $Z_{opt}/Z_{prop} = 0.995$ and $Z_{opt}/Z_{avg} = 0.993$. As $p_{LT} = 0.9$, we have $Z_{opt}/Z_{prop} = 0.651$ and $Z_{opt}/Z_{avg} = 0.586$. Obviously, if p_{LT} is smaller, the differences of the number of still undetected software faults among these three resource allocation methods are less significant. Therefore, the average resource allocation method can be employed due to its simplicity if p_{LT} is smaller. On the contrary, if p_{LT} is larger, the optimal resource allocation method should be applied since it reduces the number of still undetected software faults much more significantly. In summary, both curves in Figure 3 are decreasing in p_{LT}; that is, the larger the p_{LT}, the better the optimal allocation method (compared with the average and the proportional allocation methods).

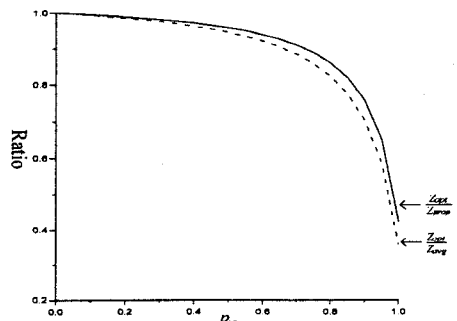

Figure 3. $\frac{Z_{opt}}{Z_{prop}}$ and $\frac{Z_{opt}}{Z_{avg}}$ vs. p_{LT}.

Figure 3 also indicates that the proportional allocation method is better than the average allocation method since $Z_{opt}/Z_{prop} \geq Z_{opt}/Z_{avg}$ for $0 < p_{LT} \leq 1$. This result meets our intuition.

Example 2. Minimizing the total amount of testing resources

Suppose the number of still undetected software faults after the application of t_5 needs to be reduced from 200 to 120. The manager has to determine how much testing resource is to be allocated for each module to minimize the total amount of testing resources. Using the values of m_j^*, a_j, b_j, $p_{LT,j}$, and v_j in Table

1, the optimal solutions $q_j^{\#}$ estimated by Algorithm 2 are shown in Table 2. Consequently, the total amount of testing resources in module testing estimated by Eq.(39) is $14.7k$ man-hours. That is, the amount of testing resources spent during the application of t_5 is anticipated to be $14.7k$ man-hours.

Table 2. Allocated Testing Resources $q_j^{\#}$ for Minimizing Total Testing Resources. (Z is assumed to be 120)

Module	$q_j^{\#}$	z_j
1	9.03	7. 1
2	4.04	20.2
3	1.44	28.6
4	0.19	34.1
5	0.00	30

$\sum_{j=1}^{5} q_j^{\#} = 14.7k$ man-hours

6. Conclusions

In this paper, based on the HGDM with logistic learning factor, we investigate two optimal resource allocation problems in software module testing: 1) minimization of the number of software faults still undetected in the system after testing given the total amount of testing resources and 2) minimization of the total amount testing resources required given the number of software faults still undetected in the system after testing. Two efficient and novel optimization algorithms based on the concept of Lagrange multiplier method for the above two problems are proposed, respectively. In addition, the relationship between the optimal, average, and proportional resource allocation methods is investigated. Experimental results show that the optimal allocation methods are very efficient for solving the testing resource allocation problem. Furthermore, the results show that the better the skill of detecting faults, the better the optimal allocation method.

Acknowledgment. We would like to express our gratitude for the support of the National Science Council, Taiwan, R.O.C., under Grants NSC85-2221-E002-015. Reviewers' comments are also highly appreciated.

References

[1] M. V. Zelkowitz, "Perspectives of software engineering," *ACM Computing Surveys*, Vol. 10, pp. 197–216, 1978.

[2] P. Kubat and H. S. Koch, "Managing test-procedures to achieve reliable software," *IEEE Trans. on Reliability*, Vol. 32, No. 3, pp. 299–303, 1983.

[3] H. Ohtera and S. Yamada, "Optimal allocation & control problems for software-testing resources," *IEEE Trans. on Reliability*, Vol. 39, No. 2, pp. 171–176, 1990.

[4] Y. W. Leung, "Software reliability growth model with debugging efforts," *Microelectron. Reliab.*, Vol. 32, No. 5, pp. 699–704, 1992.

[5] J. D. Musa, A. Iannino, and K. Okumoto, *Software Reliability — Measurement, Prediction, Application*, McGraw-Hill, New York, 1987.

[6] M. Xie, *Software Reliability Modeling*, World Scientific Publishing Company, Singapore, 1991.

[7] M. R. Lyu (ed.), *Handbook of Software Reliability Engineering*, McGraw-Hill, New York, 1996.

[8] Y. Tohma, K. Tokunaga, S. Nagase, and Y. Murata, "Structural approach to the estimation of the number of residual software faults based on the hyper-geometric distribution," *IEEE Trans. on Software Engineering*, vol. 15, No. 3, pp. 345–355, March 1989.

[9] Y. Tohma, H. Yamano, M. Ohba, and R. Jacoby, "The estimation of parameters of the hyper-geometric distribution and its application to the software reliability growth model," *IEEE Trans. on Software Engineering*, vol. SE-17, No. 5, pp. 483–489, May 1991.

[10] R. Jacoby and Y. Tohma, "Parameter value computation by least square method and evaluation of software availability and reliability at service-operation by the hyper-geometric distribution software reliability growth model (HGDM)," *Proc. 13th Int. Conf. Software Engineering*, pp. 226–237, 1991.

[11] T. Minohara and Y. Tohma, "Parameter estimation of hyper-geometric distribution software reliability growth model by genetic algorithms," *Proc. 6th Int. Symposium on Software Reliability Engineering*, pp. 324–329 , 1995.

[12] R. H. Hou, S. Y. Kuo, and Y. P. Chang, "Applying various learning curves to hyper-geometric distribution software reliability growth model," *Proc. 5th Int. Symposium on Software Reliability Engineering*, pp. 7–16, 1994.

[13] R. H. Hou, S. Y. Kuo, and Y. P. Chang, " Optimal release times for software systems with scheduled delivery time based on the HGDM ," *IEEE Trans. on Computers (accepted for publication).*

[14] R. H. Hou, S. Y. Kuo, and Y. P. Chang, "Optimal release policy for hyper-geometric distribution software reliability growth model," *IEEE Trans. on Reliability (accepted for publication).*

[15] M. S. Bazaraa and C. M. Shetty, *Nonlinear Programming: Theory and Algorithms*, Wiley, New York, 1993.

[16] M. J. D. Powell, *A tolerant algorithm for linearly constrained optimization calculations*, DAMTP Report NA17, University of Cambridge, England, 1988.