Markov控制转换多模块软件可靠性测试资源动态分配方法

齐 蓓 覃志东
(东华大学计算机科学与技术学院 上海 201620)

摘 要 考虑到软件任务模块执行的不均衡性以及模块级软件可靠性增长测试的具体情况,针对Markov控制转换多模块软件,提出了一种模块级的可靠性测试资源动态优化分配方法,以减少测试代价。与静态资源分配方法相比,本方法更加优化利用测试资源,在确保可靠性指标的前提下可降低总的测试代价。

关键词 软件可靠性,可靠性测试,测试资源分配,马尔科夫链
中图法分类号 TP202+.1 **文献标识码** A

Dynamic Resource Allocation Method of Reliability Testing for Multi-modules Software with Markov Transfer of Control

QI Bei QIN Zhi-dong
(School of Computer Science and Technology, Donghua University, Shanghai 201620, China)

Abstract The task modules have the imbalance characteristic in execution, and to cut back the test cost, a dynamic optimizing allocation method in module-level reliability test was proposed for multi-module software of Markov control transfer, considering the specific situation of the reliability growth test for the module-based software. Compared with the static source allocation methods, the proposed method makes full use of the testing resource optimally, which can reduce the total test cost to meet the reliability target.

Keywords Software reliability, Reliability testing, Testing resource allocation, Markov chain

1 引言

与硬件可靠性技术的迅速发展相比,软件可靠性技术发展滞后,成为计算机系统可靠性进一步提高的瓶颈。而随着计算机系统在安全关键领域日益广泛的应用,为减少软件失效而导致系统失效所带来的灾难性损失,高可靠性软件的可靠性测试方法成为产业和学术界关注的热点[1]。针对软件系统多任务、多模块化发展的现状,我们提出了基于体系结构的软件可靠性测试框架[2]。考虑到模块执行的不均衡性,需要把软件系统可靠性指标分配到各个软件模块,并进行模块级的可靠性增长测试,以期在规定的时间达到该指标;且为了减小测试代价,需要在该阶段进行优化的测试资源分配。

前人对软件测试资源的优化分配已做了大量研究,并取得丰富理论成果。Ohtera H. 和 Yamada S. 在权衡测试资源和软件残留缺陷数后提出相应资源分配方案[3];R. Lyu 等人则着重权衡单应用和多应用系统中测试资源和软件失效率的关系[4]。以上资源优化方案从不同角度说明了资源优化分配的问题,可是并未将软件体系结构考虑在内。Y. S. Dai 等人假设软件模块间类似于硬件的串/并联关系,将遗传算法应用于模块级测试的测试资源优化分配中[5],来解决测试时间资源固定的情况下软件可靠性最大而开销最小的问题。然而,由于软件体系结构的复杂性与特殊性,将软件与硬件的体系结构相等同是不恰当的[6]。J. Rajgopal 和 M. Mazumdar 提出基于 Markov 使用模型的测试资源优化分配方法[7],有效降低了测试用例量;P. Roberto 等人则基于 DTMC 模型建立软件测试资源分配模型[8]。以上方法针对 Markov 使用模型的软件系统进行了测试资源静态优化分配的探讨。但是,各模块在测试过程中表现出的可测试性、可靠性增长情况是在变化的。采取静态资源优化分配方法容易造成测试结束后软件实际残余缺陷数比预期的要多。早在 1997 年,Leung 提出动态资源分配的思想[9],即通过动态调整测试资源分配,达到减小软件模块剩余缺陷方差的目的。

由于软件结构的多样性,本文将针对 Markov 使用模型建立软件可靠性模型,并提出 Markov 使用模型的多模块软件可靠性测试资源动态分配方法。该方法不仅可以根据各软件模块可靠性的提高对系统可靠性影响程度的不同进行有针对性的测试,以实现系统可靠性快速增长,还考虑到在测试过程中各模块表现出的复杂度的动态变化而导致测试难易的不同,通过分阶段动态调整测试资源,来确保测试资源分配的合理性。实验证明,本方法能实现可靠性测试资源的优化分配,有效降低测试代价。

2 测试资源动态优化分配的必要性

2.1 Markov控制转换多模块软件及其可靠性增长

Cheung 于 1980 年首次对 Markov 控制转换多模块软件

到稿日期:2012-12-10 返修日期:2013-03-02 本文受国家自然科学基金(60903160),中央高校基本科研业务费专项基金(11D11209)资助。
齐 蓓(1987—),女,硕士生,主要研究方向为软件可靠性测评,E-mail:qibei.870823@163.com;覃志东 男,博士,副教授,主要研究方向为实时计算与可信性计算。

进行了软件可靠性建模[10]，通过模块间控制传递概率形象地描述了软件模块间的体系结构关系。对于如图 1 所示的 n 模块软件系统，假设各模块可靠性相互独立。用 p_{ij} 表示模块 i 到模块 j 的单步控制传递概率，$p_{ij}\subset[0,1]$。吸收状态 S 和 F 分别表示系统执行成功和失败，软件系统成功执行后控制由模块 i 以概率 p_{iS} 转向吸收状态 S，且控制传递概率满足：

$$p_{iS}+\sum_{j=1}^{n}p_{ij}=1 \tag{1}$$

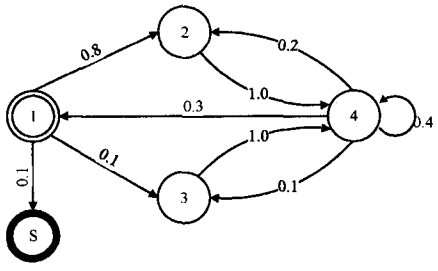

图 1　Markov 软件体系结构图

在软件系统的实际运行中，当软件模块存在的缺陷被触发后就会导致系统运行的失败，使得控制由模块 i 以概率 q_{iF} 转向吸收状态 F。假设软件系统运行一次，模块 i 中缺陷没有被触发的概率(也即模块 i 的运行可靠性)为 r_i，则对于实际运行中的系统，模块 i 控制成功传递到模块 j 的概率 q_{ij} 满足：

$$\begin{cases} q_{ij}=r_i\times p_{ij},i=1,2,\cdots,n;j=1,2,\cdots,n \\ q_{iF}=1-r_i,i=1,2,\cdots,n \\ q_{SS}=q_{FF}=1 \\ q_{ij}=0,\text{其它} \end{cases} \tag{2}$$

因此，可得到在运行中的 n 模块软件系统的运行控制传递矩阵为：

$$Q=\begin{bmatrix} \hat{Q} & C \\ 0 & I \end{bmatrix}_{(n+2)\times(n+2)} \tag{3}$$

式中，$\hat{Q}$ 表示 n 个模块间的单步控制传递概率 $n\times n$ 矩阵 $[q_{ij}]_{n\times n}$，C 表示与吸收状态 S，F 有关的控制传递概率 $n\times 2$ 矩阵。

因此可得到该系统运行可靠性 R_S 为：

$$R_S=\sum_{i=1}^{n}(I_n-\hat{Q})_{1i}^{-1}r_i p_{iS} \tag{4}$$

那么，对于图 1 所示软件系统，由式(4)可得到，当其它软件模块的运行可靠性为 0.999 时，软件系统的运行可靠性 R_S 与模块 i 可靠性 r_i 的关系如图 2 所示。

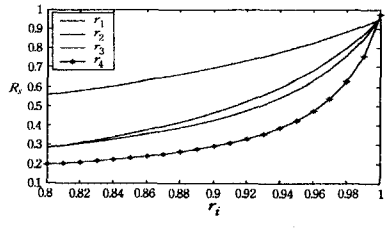

图 2　软件系统对各模块可靠性的敏感性分析图

由图 2 可知，各软件模块可靠性的提高对系统可靠性影响是不同的：提高模块 4 的可靠性可更快实现系统可靠性的增长；相反地，模块 3 可靠性的提高对提升系统可靠性“贡献”最小。所以，为了确保软件系统可靠性，需要按照各模块对系统可靠性影响的重要性把系统可靠性指标分配到软件模块，作为模块级可靠性增长测试的目标。

2.2　动态资源优化分配的必要性

若按照各软件模块对系统可靠性影响的大小粗略为各模块分配可靠性指标和测试资源，那么在模块级可靠性增长测试过程中，最理想的情况是在计划时间 D_0 内，所有模块都能够恰好达到各自的可靠性指标要求。若用线的粗细表示分配的人力资源，那么对于图 1 所示软件系统，该静态测试资源分配和可靠性增长的理想情况如图 3 所示。

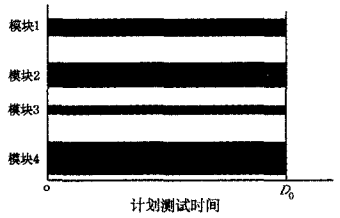

图 3　理想测试情况

然而，在实际的可靠性增长测试的过程中，由于各软件模块自身复杂度以及代码质量的不同，为达到可靠性指标要求的所需测试资源是不同的。静态资源分配后，实际上有的软件模块提前增长到可靠性指标，有的迟于计划时间达到指标，情况如图 4 所示。这样，造成模块级可靠性增长测试超过计划时间 D_0，这是不允许的。

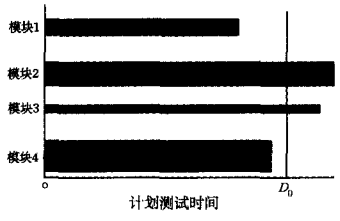

图 4　实际测试情况

为了使软件可靠性测试资源能够被更加有效地利用，并在计划的时间达到可靠性指标，需要在模块级可靠性增长测试时动态地调整资源分配。因此，本文提出动态资源优化分配方法。

3　软件可靠性测试动态资源优化分配方案

3.1　软件可靠性测试资源动态分配思想

假设各软件模块的测试是并行进行的，且各软件模块测试相互独立。通过将计划测试时间分段，并在各阶段开始时根据各模块的失效数据重新估计其可靠性模型参数，再对各模块的测试资源进行重新优化分配。测试资源主要是指将人力资源换算得到的有效测试时间资源。如：有 P_r 名测试人员，设人均工作时间为 8h/人，那么有效测试时间资源就为 $(P_r\times 8)$h。因此可通过调整测试人员数量来实现可靠性测试资源的调整。各模块在计划测试时间内被分配的测试资源示意图如图 5 所示。

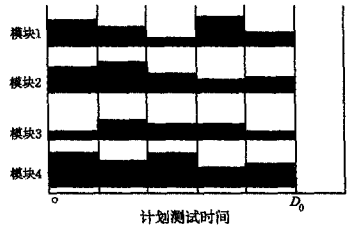

图5 动态资源分配示意图

由图5可知，每个模块被分配的测试资源在各阶段都有所调整。调整的多少主要是由在各阶段开始时所获取的失效数据估计得到的当前软件模块表现出的复杂程度(体现为可靠性模型参数的不同)所决定。测试数据越多，估计得到的软件模块可靠性参数越准确。

若用静态的方法进行可靠性资源优化分配，即只根据模块重要性和初始复杂度信息进行可靠性测试资源分配，而后期不做调整，会导致因参数估计不准确而使资源分配不准确，不仅造成测试结束后软件实际残余缺陷比预期的要多的不良后果[8]，而且会造成测试资源的浪费。对于模块 i，分别用动态分配和静态方法分配得到的测试资源如图6所示，其中 T_i 表示在计划测试时间内模块 i 的累积测试资源量。

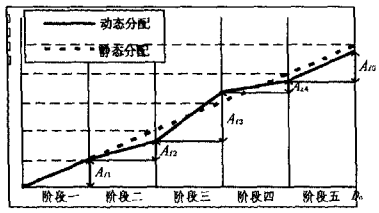

图6 模块 i 累积测试资源示意图

由图6可知，按照静态资源分配方法，所得到的各时间段的测试资源是相同的。而使用动态资源优化分配的方法，得到各时间段内的测试资源 A_{i1}，A_{i2}，A_{i3}，A_{i4}，A_{i5} 是在动态调整的。这样做的优点是可根据各软件模块当前表现出的状况和软件系统自身结构特点动态调整测试资源的分配。运用较少的测试资源可实现系统可靠性的尽快增长，从而使可靠性测试资源能够被更加有效地利用。

3.2 基于 Markov 控制转换的动态资源分配模型

在软件可靠性测试资源优化分配的建模中，既要考虑到软件系统的自身结构特点，又必须考虑各软件的自身复杂度。由于软件体系结构的多样性，这里只针对 Markov 结构软件提出动态分配模型。

3.2.1 基于 Markov 控制转换的软件系统可靠性建模

在可靠性测试过程中，软件可靠性即为经过 T 时长的可靠性增长测试后，软件可以无失效运行 x 时间长度的概率[11]，用 $R_S(x|T)$ 表示。

设软件系统在连续运行过程中，其失效符合泊松分布。用 $N(t)$ 表示到 t 时刻为止，软件系统的累积失效总数。用 $m(t)$ 表示在时刻 t，软件系统单位时间的平均失效个数，则在 $t=T$ 时刻有 $N(t)=m(t)\times T$。因此根据 NHPP 理论，有：

$$P\{N(T+x)-N(T)=k\}=\frac{[m(T+x)-m(T)]^k}{k!}\times \{-[m(T+x)-m(T)]\},k=0,1,2,\cdots \tag{5}$$

因此，软件系统可靠性可表示为：

$$R_S(x|T)\equiv P\{N(T+x)-N(T)=k=0\}=\exp\{-[m(T+x)-m(T)]\} \tag{6}$$

在测试结束后，软件系统的可靠性不再发生变化，即为一定值。若用 $\lambda(t)$ 表示在时刻 t 单位时间内软件的失效个数，即软件的失效密度，且有 $\lambda(t)=\frac{dm(t)}{dt}$，那么式(6)可进一步化为：

$$R_S(x|T)=\exp[-\lambda_S(T)\cdot x] \tag{7}$$

式(7)表示经过 T 时长测试后，软件系统的可靠性。其中，$\lambda_S(T)$ 表示 $t=T$ 时刻系统的失效密度。

那么，对于 n 模块软件系统，若用 $N_i(t)$ 表示其中模块 i 的累积失效总数，用 $m_i(t)$ 表示 t 时刻模块 i 的平均失效数，用 $\lambda_i(t)$ 表示其失效密度，用 T_i 表示在总的测试时间 T 内用于模块 i 测试的时间，且 $T=\sum_{i=1}^{n}T_i$，则对于各模块，有 $N_i(t)=m_i(t)\times T_i$，$i=1,2,\cdots,n$。又由于 $N=\sum_{i=1}^{n}N_i$，因此可得：

$$m(t)\cdot T=\sum_{i=1}^{n}m_i(t)\cdot T_i \tag{8}$$

对式(8)两边同时求导，可得到：

$$\lambda_S(t)=\lambda_1(t)\cdot\frac{T_1}{T}+\lambda_2(t)\cdot\frac{T_2}{T}+\cdots+\lambda_n(t)\cdot\frac{T_n}{T} \tag{9}$$

令 $\pi_i=\frac{T_i}{T}$ 表示模块 i 的执行概率。用 X_i 表示因系统运行一次而触发模块 i 运行的平均次数，其值等于模块 i 作为输入状态的概率 e_i 加上控制从其它模块(或自身)传递到模块 i 的平均次数。即：

$$X_i=e_i+\sum_{j=1}^{n}X_jp_{ji} \tag{10}$$

因此，模块 i 的执行概率 π_i 可由式(11)得到：

$$\pi_i=\frac{T_i}{T}=\frac{T_i}{\sum_{i=1}^{n}T_i}=\frac{u\times T_i\times t_i}{\sum_{i=1}^{n}u\times T_i\times t_i}=\frac{T_i\times t_i}{\sum_{i=1}^{n}T_i\times t_i} \tag{11}$$

式中，t_i 表示模块 i 的单次执行时间，u 表示 T 时间内系统执行的总次数。X_i 可通过解各软件模块的 X_i 方程(即式(10))所得的 n 维方程组得到。

因此，式(9)可化为：

$$\lambda_S(t)=\sum_{i=1}^{n}\lambda_i(t)\cdot\pi_i \tag{12}$$

将式(12)代入式(7)，便可得到软件系统可靠性关于各模块失效密度的函数式：

$$R_S(x|T)=\exp[-\sum_{i=1}^{n}\pi_i\lambda_i(t)\times x] \tag{13}$$

3.2.2 阶段 j 的测试时间资源动态分配模型

若将测试持续期分为 k 个阶段，对于其中任一阶段 $j(1\leqslant j\leqslant k)$，分析其可靠性测试资源的优化分配模型。

在 n 模块软件系统中，对于其中模块 $i(1\leqslant i\leqslant n)$，由 G-O 模型结论[12]知，其失效密度 $\lambda_i(t)$ 与测试时间 T_i 的关系可表示为：

$$\lambda_i(T_i)=a_ib_i\exp(-b_iT_i) \tag{14}$$

式中，a_i 表示模块 i 的缺陷数量，b_i 表示单位时间内缺陷被触发的概率。这两个参数可通过模块 i 的失效数据估计得到：设模块的累积失效个数为 m，及各失效间的时间间隔为 s_p（$p=1,2,\cdots,m$），令 $S_p=\sum_{q=1}^{p}s_q$，则用最大似然估计法可得到 a_i，b_i 值：

$$\begin{cases}a_i=m[1-\exp(-b_iS_m)]\\ b_i=m[\sum_{p=1}^{m}S_p+a_iS_m\exp(-b_iS_m)]\end{cases} \tag{15}$$

将式(14)代入式(13)中，便可得到软件系统可靠性与各软件模块测试时间的关系函数：

$$R_S(T|x)=\exp[\sum_{i=1}^{n}\pi_ia_ib_i\exp(-b_iT_i)\times x] \tag{16}$$

在可靠性测试资源动态优化分配中，为降低总的资源消耗，就要保证在各测试阶段，在达到可靠性指标(R_0,x)即系统无失效运行 x 时长的概率不低于 R_0 的前提下，测试资源要最少。那么，对于阶段 j，软件可靠性测试资源优化分配的问题可描述为如下优化问题：

$$\begin{aligned}&\text{Minimize } T_j=\sum_{i=1}^{n}T_i\\ &\text{Subject to}\\ &R_S(x|T)=\exp[-\sum_{i=1}^{n}\pi_ia_{ij}b_{ij}\exp(-b_{ij}T_{ij})\times x]\geqslant R_0\end{aligned} \tag{17}$$

用拉格朗日法解式(17)，便可得到优化分配的结果 T_{ij}，$i=1,2,\cdots,n,1\leqslant j\leqslant k$。它表示模块 i 根据当前(即阶段 j)表现出的状况应被分配的用于整个测试期的测试资源。若用 A_{ij} 表示模块 i 在阶段 j 所需的测试资源，则有：

$$A_{ij}=T_{ij}/k \tag{18}$$

因此，在整个可靠性测试过程中所需的总测试时间资源为：

$$T=\sum_{i=1}^{n}\sum_{j=1}^{k}A_{ij} \tag{19}$$

3.3 软件可靠性测试资源动态分配方法的实施步骤

首先，根据软件系统的 Markov 控制传递模型，由式(11)求得各模块运行比例系数 π_i 的值，并从开发方获得各软件模块的可靠性模型初始参数值 a_{i1}、b_{i1}（$i=1,2,\cdots,n$）。

然后，将总的测试持续期平均分为 $k(k>0)$段，并按照图 7 所示流程图在软件可靠性测试过程中开展资源动态优化分配。

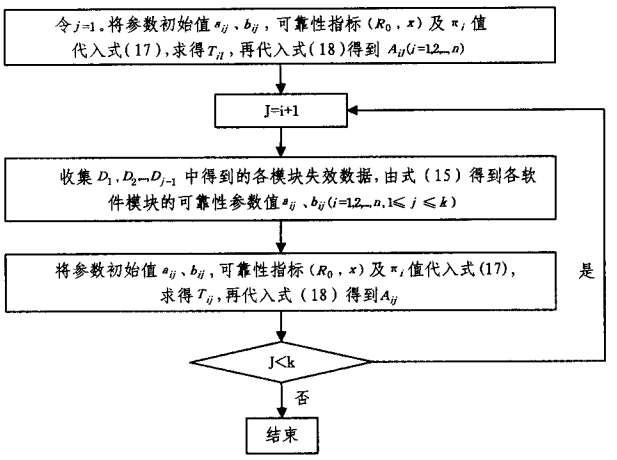

图 7　测试资源分配流程图

4　数值模拟

假设一个由 6 个软件模块组成的软件系统，其 Markov 控制转换模型如图 8 所示。软件的可靠性测试指标为连续无失效运行 100 小时的概率不低于 95%，即$(R_0,x)=(0.95,100)$。

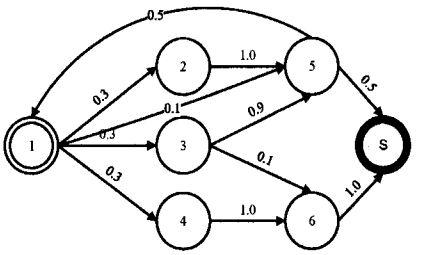

图 8　软件系统结构模型

首先根据图 8，由式(10)、式(11)得到各模块运行比例系数 π_i 的值，如表 1 所列。

表 1　π_i 值

模块	1	2	3	4	5	6
X_i	1.399	0.420	0.420	0.420	0.797	0.462
t_i/s	1.0	1.0	1.0	1.0	1.0	1.0
π_i	0.357	0.107	0.107	0.107	0.204	0.118

假设将软件测试持续期分为 4 个阶段，则按照图 7 所示的流程图，得到此系统的动态资源分配表，如表 2 所列。

表 2　动态资源优化分配的数据

模块	D_1			D_2			D_3			D_4		
	a_{i1}	$b_{i1}/10^{-3}$	A_{i1}/h	a_{i2}	$b_{i2}/10^{-3}$	A_{i2}/h	a_{i3}	$b_{i3}/10^{-3}$	A_{i3}/h	a_{i4}	$b_{i4}/10^{-3}$	A_{i4}/h
1	10.0	2.29	366	10.0	2.29	350	10.0	2.29	350	10.0	2.29	352
2	20.4	3.96	242	12.7	3.96	209	16.7	3.96	223	12.0	3.00	237
3	14.6	5.79	180	19.0	5.79	183	16.0	6.90	159	15.1	6.78	160
4	20.4	3.96	242	11.7	4.80	185	15.3	4.00	217	18.8	4.01	228
5	18.9	4.00	269	22.8	4.00	269	23.5	5.60	218	19.0	5.30	219
6	10.0	2.29	269	6.0	5.29	154	6.9	4.10	180	6.5	5.23	159

从表 2 中 D_1 的各模块的资源分配值可得知：对于模块 2 和模块 3，其在系统中的结构相同但各自的复杂度不同时，模块自身越复杂(表现为 a_i 值越大 b_i 值越小)，则被分得的资源越多；对于模块 1 和模块 6，其各自复杂度相同但在系统中的结构不同时，模块的结构对系统可靠性影响越大(表现为 π_i 值越大)，则被分得的资源越多；对于模块 2 和模块 4，其自身复杂度相同且在系统中的结构也相同，则被分得同样多的资源。同时注意，对于模块 5 和模块 6 的情况，即使模块复杂度和系统中的结构不同，也可能被分得相同的资源。在接下来的 D_2，D_3，D_4 中，一些模块在测试过程中参数发生了变化，而使得资源分配也随之变化。对于模块 1，虽然其在整个测试

(下转第 202 页)

自然科学版,2010,5:538-540
[12] 张铃,张钹. 动态商空间模型及其基本性质[J]. 模式识别与人工智能,2012,25(2):181-185
[13] 李道国,苗夺谦,等. 粒度计算研究综述[J]. 计算机科学,2005,32(9):1-12
[14] Yao Y Y,Zhang N,Miao D Q,et al. Set-theoretic approaches to granular computing[J]. Fundamenta Informaticae,2012,115(2/3):247-264
[15] Yao Y Y,Zhao L Q. A measurement theory view on the granularity of partitions[J]. Information Sciences,2012,213:1-13
[16] 王国胤. Rough 集理论与知识获取[M]. 西安:西安交通大学出版社,2001
[17] Yao Y Y. Stratified rough sets and granular computing[C]// Dave R N,Sudkamp T,eds. Proceedings of the 18th International Conference of the North American Fuzzy Information Processing Society. New York,USA,IEEE Press,1999:800-804
[18] Hu Xiao-hua, Cercone N. Discovery maximal generalized decision rules through horizontal and vertical data reduction[J]. Computational Intelligence,2001,17(4):685-702
[19] Hu Xiao-hua,Cercone N. Learning maximal generalized decision rules via discretization,generalization and rough set feature selection[C]// Proceedings of the 9th IEEE International Conference on Tools with Artificial Intelligence. Newport Beach,CA,USA. IEEE Computer Society,1997:548-556
[20] Shan Ning, Hamilton H J, Cercone N. GRG: Knowledge Discovery Using Information Generalization, Information Reduction, and Rule Generation[C]// Proceedings of the seventh international conference on tools with artificial intelligence. Washington,DC,USA,IEEE Computer Society,1995:372-379
[21] Dong Wei,Wang Jian-hui,Xu Lin,et al. Algorithm of Hierarchical Reduction Based on Rough Entropy[C]// Proceedings of the sixth world congress on intelligent control and automation. 2006:4374-4377
[22] Zhang J,Kang D K,et al. Learning accurate and concise naïve Bayes classifiers from attribute value taxonomies and data[J]. Know-ledge and information systems,2006,9(2):157-159
[23] Feng Q R, Miao D Q, Cheng Y. Hierarchical decision rules mining[J]. Expert Systems with Applications,2010,37:2081-2091

(上接第 165 页)

持续期参数都没有发生变化,但它被分配的资源却在变化。这是由于,在各测试阶段,为保证在可靠性指标的基础上实现资源最少,就要对复杂度较大的模块和在系统结构中影响更大的模块分配更多的测试资源,所以导致了一些模块即使在测试过程中系统结果和模块自身复杂度没有改变,但其被分配的资源却发生了变化。

将表 2 数据代入式(19),可得到用动态资源优化分配方法所需总测试时间资源为 5617h。若用静态资源优化分配法对此软件系统资源优化分配,即将 a_{i1}、b_{i1} 代入式(17)解得的 T_{i1} 值作为优化资源分配的结果,所得到的总测试时间为 6268h。对于此模拟系统,在总的测试时间资源上,动态方法可比静态方法少 651h,较静态方法节约 10.4%的测试资源。总的测试资源比较图如图 9 所示。

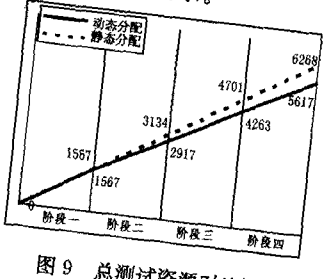

图 9 总测试资源对比图

结束语 基于 Markov 结构的软件可靠性测试资源动态配方法用模块的运行比例系数 π_i 体现软件系统体系结构点,用模块可靠性模型参数体现模块自身复杂度特点。通阶段测试过程中动态调整可靠性模型参数,不仅保证了软可靠性模型的准确性,也使得可靠性测试资源在测试过程根据各软件模块复杂度的变化得到最合理而有效的分配和用。数值模拟结果表明,Markov 控制转换多模块软件可靠试资源动态分配方法可根据软件 Markov 使用结构和模身复杂度合理分配可靠性测试资源,并有效减少总的软可靠性测试时间资源。

参 考 文 献

[1] Lyu Micheal R. Software Reliability Engineering: A Roadmap [A]// Future of Software Engineering,2007[C]. Minneapolis,2007:153-170
[2] 覃志东. 高可信软件可靠性和防危性测试与评价理论研究[D]. 成都:电子科技大学,2005
[3] Ohtera H,Yamada S. Optimal Allocation and Control Problems for Software-Testing Resources[J]. IEEE Trans. on Reliab.,1990,39(2):171-176
[4] Lyu Micheal R, Rangarajan S, van Moorsel Aad P A. Optimal Allocation of Test Resources for Software Reliability Growth Modeling in Software Development[J]. IEEE Transactions on Reliability,2002,51(2):183-192
[5] Dai Y S,Xie M,Poh K L. Optimal Testing-Resource Allocation with Genetic Algorithm for Modular Software Systems[J]. The Journal of Systems and Software,2003,66(1):47-55
[6] 樊林波,吴智,赵明. 基于构件的软件可靠性分析[J]. 计算机科学,2007,34(5):266-269
[7] Rajgopal J,Mazumdar M. Modular Operational Test Plan for Inferences on Software Reliability Based on a Markov Model[J]. IEEE Transactions on Software Engineering,2002,28(4):356-363
[8] Pietrantuo R, Russo S, Trivedi K S. Software Reliability and Testing Time Allocation: An Architecture-Based Approach[J]. IEEE Transactions on Software Engineering,2010,36(3):323-337
[9] Lenug Y W. Dynamic Resource-Allocation for Software-Module Tesing[J]. Journal of Systems and Software,1997,37(2):129-139
[10] Cheung R C. A User-Oriented Software Reliability Model[J]. IEEE Transactions on Software Engineering,1980,6(2):118-125
[11] Yang B,Xie M. A Study of Operational and Testing Reliability in Software Reliability Analysis[J]. Reliability Engineering & System Safety,2000,70(3):323-329
[12] Goel A L, Okumoto K. Time-Dependent Error-Detection Rate Model for Software Reliability and