一种面向测试需求部分覆盖的测试用例集约简技术

顾 庆[1)] 唐 宝[2)] 陈道蓄[1)]

[1)](南京大学软件新技术国家重点实验室 南京 210093)

[2)](西门子 IT 解决方案与服务集团 南京 211100)

摘 要 软件系统开发中频繁面对局部更新和部分缺陷修改,此时需要选择性回归测试;为降低其测试工作量需要解决部分覆盖用例集约简问题.文中基于选择性回归测试形式化定义多目标用例集约简,并设计 HATS 算法解决该问题.HATS 算法遵循启发式贪婪搜索框架,定义权重因子 α 平衡两方面目标:其一是减少用例集规模;其二是避免对无关需求的覆盖.实验结果表明:同现有约简技术和算法相比较,通过适当设置因子 α, HATS 算法能够降低用例集规模、减少对无关需求的覆盖、同时缓解对缺陷检测能力的影响.文中的创新贡献在两方面:其一根据选择性回归测试定义多目标用例集约简问题;其二是设计 HATS 算法以更少的测试用例数量维持缺陷检测效果.

关键词 软件测试;测试用例约简;回归测试;测试需求集;部分覆盖

中图法分类号 TP311 **DOI 号**: 10.3724/SP.J.1016.2011.00879

A Test Suite Reduction Technique for Partial Coverage of Test Requirements

GU Qing[1)] TANG Bao[2)] CHEN Dao-Xu[1)]

[1)](*State Key Laboratory of Novel Software Technology, Nanjing University, Nanjing* 210093)

[2)](*Siemens IT Solutions and Services, Nanjing* 211100)

Abstract During software system development, one has to face with frequent local updates and partial fault corrections, where selective-form regression testing is required. To save its test effort, one needs to solve the partial coverage test suite reduction problem. Under selective-form regression testing, this paper formally defines the multi-objective test suite reduction problem, and develops the HATS algorithm to solve the problem. The HATS conforms to the heuristic greedy search framework, and defines the weighting factor α to balance the two objectives: one is to reduce the test suite size; and the other is to avoid coverage of irrelevant test requirements. The experiment results show that, compared with current test suite reduction techniques and algorithms, with proper setting of the factor α, the HATS can reduce the test suite size, cut down the coverage of irrelevant test requirements, while less compromise the fault detection ability. Original contributions of this paper include two aspects: First defined the multi-objective test suite reduction problem under selective-form regression testing; Second designed the HATS algorithm which used fewer test cases to maintain the fault detection effectiveness.

Keywords software testing; test suite reduction; regression testing; test requirement set; partial coverage

---

收稿日期:2010-01-04;最终修改稿收到日期:2011-03-14.本课题得到国家自然科学基金(60873027,61021062)、国家"八六三"高技术研究发展计划项目基金(2006AA01Z177)、国家"九七三"重点基础研究发展规划项目基金(2009CB320705)资助.**顾 庆**,男,1972 年生,博士,教授,研究领域为软件质量与软件测试. E-mail: guq@nju.edu.cn. **唐 宝**,男,1974 年生,硕士,高级工程师,研究方向为软件质量与软件测试. **陈道蓄**,男,1947 年生,教授,研究领域为分布计算与并行处理.

1 引 言

在软件系统发生变更后或发布新版本前通常需要对软件程序进行回归测试,以确保已有功能或模块没有受到当前变更的负面影响.目前软件系统通常采用迭代式开发,软件升级换代成为惯例,且升级换代的速度不断加快.这使得回归测试成本不断提升.如何在保证检错能力的前提下有效降低回归测试工作量是需要解决的问题.

在软件系统迭代开发过程中,测试用例不断被设计、修改和执行,构成回归测试的备选用例集合.在回归测试中,软件程序被视为测试需求的集合.例如从结构角度可以把程序中的语句视为测试需求,回归测试要求达到和备选用例集相同的语句覆盖率.在通常情况下,对给定的测试需求覆盖要求,备选用例集会存在较大的冗余[1].测试用例集约简技术用于从备选用例集中选择较少数量的测试用例构成约简集,同时达到测试需求覆盖.目前研究者提出多种用例集约简技术,这些技术大致可归为3类:启发式贪婪搜索[1-3]、元启发概率优化[4]以及二进制整数线性规划[5-6].

根据软件系统的更新程度和范围,回归测试可以有两种形式:选择性回归测试和全面回归测试.前者在软件更新范围有限或缺陷尚未全部修正时执行;而后者通常在新版本发布或更新范围较大且全部缺陷修正后执行.相较于后者,选择性回归测试的执行频度更高,更需要控制测试成本.在选择性回归测试中,测试需求集合可分割为两部分:关注需求集和无关需求集.其中关注需求集需要被回归测试用例覆盖;而无关需求集正相反,需要在选择测试用例时避免覆盖以减少测试执行和分析的工作量.现有的测试用例集约简技术尚未考虑这一问题.

本文针对选择性回归测试中测试需求集部分覆盖要求,提出多目标部分覆盖测试用例集约简问题.给出解决该问题的一个启发式贪婪搜索算法——HATS(Heuristic search Algorithm with Three Strategies).基于开源软件系统设计的实验结果表明:同全覆盖测试用例集约简技术相比,HATS算法能够进一步较大程度地减少回归测试用例数量,降低对无关测试需求集的覆盖比率,同时基本保证针对关注需求集的缺陷检测能力.

2 测试用例集约简

2.1 基本概念

描述测试用例集约简问题需要以下基本概念.

定义 1. 备选测试用例集.备选测试用例集T($T=\{t_1,t_2,\cdots,t_m\}$)是针对目标软件系统设计完成的一组测试用例集合.集合中的用例已执行并构成回归测试的备选用例集.

定义 2. 测试需求集.测试需求集R($R=\{r_1,r_2,\cdots,r_n\}$)是测试目标软件系统要求覆盖的测试需求集合.测试需求反应软件系统的基本覆盖单元.如从结构角度一个测试需求可以是一条语句或一个函数(过程或方法);从功能角度一个测试需求可以是一个功能项或一个被测特征.

定义 3. 测试覆盖矩阵.测试覆盖矩阵$\boldsymbol{\Delta}(R,T)$是一个$|R|\times|T|$二进制矩阵,定义用例集T到需求集R的覆盖关系.矩阵元素由式(1)定义:

$$\delta(r_i,t_j)=\begin{cases}1, & t_j\in T \text{ covered } r_i\in R\\ 0, & t_j \text{ ignored } r_i\end{cases} \tag{1}$$

为简便起见,我们用δ_{ij}表示$\delta(r_i,t_j)$,在不引起歧义情况下$\boldsymbol{\Delta}(R,T)$简化表示为$\boldsymbol{\Delta}$.

定义 4. 测试用例的覆盖需求集.测试用例t_j的覆盖需求集$R(t_j)$($R(t_j)\subseteq R$)是一个测试需求集合,其中包含用例t_j覆盖的需求集R中所有测试需求.

定义 5. 测试需求的执行用例集.测试需求r_i的执行用例集$T(r_i)$($T(r_i)\subseteq T$)是一个测试用例集合,其中包含覆盖(执行)需求r_i的用例集T中所有测试用例.

定义 6. 需求集R的代表用例集.需求集R的代表用例集$\gamma(R)$($\gamma(R)\subseteq T$)是一个测试用例集合,其所包含测试用例的覆盖需求集的并集等于(包含)R,即$U_{t\in\gamma(R)}R(t)\supseteq R$.

本文中用例t_j的覆盖需求集$R(t_j)$简化表示为R_j,对应覆盖矩阵$\boldsymbol{\Delta}$的第j列;需求r_i的执行用例集$T(r_i)$简化为T_i,对应矩阵$\boldsymbol{\Delta}$的第i行;不引起歧义的情况下$\gamma(R)$简化为γ.根据上述定义,一个全覆盖测试用例集约简问题可以形式化描述如下:

给定备选用例集T和测试需求集R,给定需求覆盖矩阵$\boldsymbol{\Delta}(R,T)$,令$\forall t_j\in T.R_j\neq\varnothing$且$\forall r_i\in R.T_i\neq\varnothing$.要求寻找最优代表用例集$\gamma(\gamma\subseteq T)$,使得$\gamma$所包含的测试用例数量最少.该问题等价于最小覆盖集问题,是一个NP完全问题[7].

2.2 测试用例集约简技术

研究者已经提出多种解决全覆盖测试用例集约简问题的近似算法.这些算法大致可归为3个类别:启发式贪婪搜索、元启发概率优化以及二进制整数线性规划.

启发式贪婪搜索技术一般一次选择一个(或多

个)局部最佳(如覆盖最大数量的测试需求)的测试用例,排除已经覆盖的测试需求,循环直至所有需求都被覆盖.这类算法中典型的有 Harrold 等人提出的HGS算法[1]以及 Chen 等人提出的 GRE[2-3]算法.

元启发概率优化技术从一个初始的代表用例集(如备选集 T)出发,应用全局概率优化算法推算最优的代表用例集.这类算法中典型的是 Mansour 等人[4]提出的模拟退火算法和混合遗传算法.

二进制整数线性规划 BIP(Binary Integer linear Programming)技术[5-6]将用例集约简目标形式化描述为一个成本函数.优化目标是成本值最小,约束条件是所有测试需求都被覆盖.Black 等人[5]应用 BIP 技术解决一个多目标测试用例集约简问题:最小化选择的测试用例数量,同时保留最多数量的曾检测出缺陷的测试用例.解决 BIP 本身也是一个 NP 完全问题[8].

研究者对不同约简技术的性能做了比较[2,9].其结果是不同类别算法得到的约简集规模基本相当,没有哪个算法能够在所有情况下优于其它算法.其他研究者[10-13]提出针对不同需求类型的特定的约简算法,并同 HGS 算法等做了比较,也得到类似的结论.

3　测试需求集部分覆盖

3.1　问题定义

面对局部范围的软件改动或部分缺陷的修正,选择性回归测试是理想选择.在选择性回归测试下,测试需求集被分割为关注需求集和无关需求集.有如下定义.

定义 7.　关注需求集.关注需求集$CR(CR\subseteq R)$包含本次选择性回归测试中必须覆盖的测试需求,同最近的软件改动和修正相关.

定义 8.　无关需求集.无关需求集包含本次选择性回归测试不需要覆盖的测试需求,同最近的软件改动和修正无关.无关需求集可标记为 $R-CR$.其中"−"操作的定义是:给定集合 A 和 B,$A-B=\{x|x\in A\wedge x\notin B\}$.

根据上述定义,针对测试需求集部分覆盖问题可以形式化定义一个多目标测试用例集约简问题,表述如下:

给定备选用例集 T、测试需求集 R 和关注需求集 CR,给定需求覆盖矩阵 $\boldsymbol{\Delta}$,令 $\forall t_j\in T.R_j\neq\varnothing$ 且 $\forall r_i\in R.T_i\neq\varnothing$.要求寻找关注需求集的最优代表用例集 γ_C,满足

目标 1. γ_C 中测试用例数量最少,即 $\min(|\gamma_C|)$;

目标 2. 无关需求的覆盖数量最少,即

$$\min_{i,j}(|\{r_i|r_i\in R-CR\wedge(\exists t_j\in\gamma_C.\delta_{ij}==1)\}|).$$

3.2　问题阐述

Chen 等人在介绍其 TestTube 工具[14]时提出类似关注需求集的概念,他们的设计是在修正软件缺陷时确定同该缺陷相关的测试覆盖单元集合(即本文的关注需求集),然后根据确定的需求集寻找合适的测试用例.他们没有考虑测试用例集约简问题.

在测试需求集部分覆盖问题中,测试需求集分割为关注需求集和无关需求集.选择测试用例时要求覆盖关注需求集,同时要求避免覆盖无关需求集.这个约简策略基于以下三方面考虑:其一选择测试用例时只要求覆盖部分测试需求,约简集规模相对于全覆盖可以进一步减少,从而降低回归测试工作量和成本.其二避免覆盖无关的测试需求可以有效减少测试分析的工作量,因为没有涉及无关的测试需求和代码;另一方面一些状态为"挂起"的缺陷没有立刻修复,避免覆盖相关需求可以减少额外的用于"关闭"这些缺陷的代码.其三选择性回归测试增加了软件系统的可测试性,尤其是在软件开发的前期,部分组件尚未完成;如果能够避免覆盖和这些组件相关的测试需求,则只需补充简单的测试"桩"就可以实施已完成组件的回归测试.上述思想在西门子公司内部软件开发项目测试中已得到了验证.

4　HATS 算法

4.1　启发式贪婪搜索框架

多目标测试用例集约简问题仍然是一个 NP 完全问题.解决该问题可采用如下的启发式贪婪搜索框架:

1. 采用一种策略从当前备选用例集 T 中选择最合适的一个或多个测试用例;

2. 将所选用例从 T 中移出并置入输出(约简)集;

3. 从关注需求集 CR 中移除已覆盖的测试需求;

4. 重复上述步骤直至 CR 为空集.

步 1 中可采用以下 3 种策略之一:贪婪策略、必选策略和替代策略.

4.2　贪婪策略

针对多目标测试用例集约简,一个简单的贪婪策略可以描述如下.

策略 1.　选择当前最"合适"的测试用例 t_j($t_j\in T\wedge t_j\notin\gamma_C$),覆盖"尽可能"多的关注测试需求,同时覆盖"尽可能"少的无关测试需求.

这里的折中问题是覆盖较多关注需求的测试用例通常也会覆盖较多的无关需求.这引出以下两个指标定义不同方面的覆盖情况.

定义 9. 贡献指标.令当前尚未覆盖的关注需求集合为 cur_CR,测试用例 t_j 的贡献指标(标记为 ς_j)定义为 t_j 所覆盖的关注需求占 cur_CR 的比例.用式(2)表示:

$$\varsigma_j=\sum_{r_i\in cur\_CR}\delta_{ij}/|cur\_CR| \tag{2}$$

定义 10. 损益指标.给定无关需求集 $R-CR$,测试用例 t_j 的损益指标(标记为 υ_j)定义为 t_j 未覆盖的无关需求占 $R-CR$ 的比例.用式(3)表示:

$$\upsilon_j=1-\sum_{r_i\in R-CR}\delta_{ij}/|R-CR| \tag{3}$$

贡献指标反映给定测试用例对关注需求的覆盖情况,而损益指标反映该测试用例对无关需求的避免情况.一个理想的测试用例应该同时具有较高的贡献指标和损益指标,但通常难以兼得.为反映两个指标的权衡,我们定义权重因子 $\alpha(0\leq\alpha\leq1)$,通过加权平均定义以下指标.

定义 11. 效用指标.给定权重因子 α,测试用例 t_j 的效用指标(标记为 ω_j)定义为其贡献指标 ς_j 和损益指标 υ_j 的加权平均.用式(4)表示:

$$\omega_j=\alpha\cdot\varsigma_j+(1-\alpha)\cdot\upsilon_j \tag{4}$$

通过对 α 的不同设定可调整对不同目标的偏重.基于效用指标,贪婪策略可描述如下.

策略 2. 对当前备选集中所有测试用例,计算其效用指标.从中选择具有最大效用指标的测试用例 t_j.

4.3 必选策略和替代策略

为描述这两个策略,我们有以下定义.

定义 12. 必选测试用例.给定当前的备选用例集 T,测试用例 t_j 是测试需求 r_i 的必选测试用例(标记为 ess_i)当且仅当 t_j 是 T 中唯一覆盖 r_i 的测试用例.

定义 13. 替代集.给定当前的关注需求集 cur_CR 和初始关注集 CR,测试用例 t_j 存在替代集(标记为 SUP_j)当且仅当对任意 $t_k\in SUP_j$,其覆盖需求集满足 $R_j\bigcap cur\_CR\subseteq R_k\bigcap cur\_CR$ 并且 $R_j\bigcap(R-CR)\supseteq R_k\bigcap(R-CR)$.

给定关注需求 r_i,其必选测试用例 ess_i(如果存在)是当前唯一覆盖 r_i 的测试用例,据此可以描述必选策略如下:

(1)对当前尚未覆盖的一个关注需求,若其存在必选测试用例,则该用例被选择.

给定备选测试用例 t_j,若 t_j 存在替代集,表示存在备选测试用例 t_k,t_k 覆盖的关注需求包含 t_j 覆盖的关注需求,而 t_k 覆盖的无关需求又被 t_j 覆盖的无关需求所包含.这说明 t_k 可以替代 t_j,据此描述替代策略如下:

(2)对当前一个备选测试用例,若其存在替代集,则该测试用例可以从备选用例集中删除.

上述策略同 GRE 算法中的用例筛选策略类似,但可用于多目标测试用例集约简技术中.针对必选策略和替代策略不难证明有以下定理.

定理 1. 在启发式贪婪搜索框架下,给定尚未覆盖的测试需求 r_i(即 $r_i\in cur\_CR$),若 r_i 存在必选测试用例 ess_i,则 ess_i 必被选入约简集 γ_C.

证明. 因为 ess_i 是当前备选集中唯一覆盖关注需求 r_i 的用例;而关注集中的所有需求必须被约简集 γ_C 中的某个测试用例覆盖;所以有 $ess_i\in\gamma_C$.定理得证. 证毕.

引理 1. 在启发式贪婪搜索框架下,给定当前尚未覆盖的关注需求集 cur_CR,若备选集中测试用例 t_j 存在替代集,则不选择 t_j 而是选择其替代用例(如 t_k)进入约简集 γ_C 将使得新约简集相对于部分覆盖约简目标等同或更优.

证明. t_j 已被选入 γ_C,分两种情况:(1)如果 t_j 和 t_k 同时被选入 γ_C,那么 t_j 属于冗余,直接删除 t_j 会减少 $|\gamma_C|$,从而使新的 γ_C 相对于目标 1 更优;(2)如果 t_k 没有被选入 γ_C,那么将 t_j 替换为 t_k 仍然满足对 CR 中所有需求的覆盖,同时对无关需求的覆盖程度相等或更小,因此新的 γ_C 相对于目标 2 等同或更优.引理得证. 证毕.

定理 2. 如果采用启发式贪婪搜索框架,仅使用必选策略和替代策略就可以得到约简集 γ_C,那么所得到的 γ_C 相对于部分覆盖约简目标最优.

证明. 反证法.假设存在 γ'_C 比 γ_C 更优,那么有 $|\gamma'_C|\leq|\gamma_C|$.按启发式搜索框架,可将 γ_C 中的测试用例按选择顺序排序.不失一般性,令 $\gamma_C=\{t_1,t_2,\cdots,t_{|\gamma_C|}\}$,其中测试用例 t_j 或者是唯一一个覆盖某关注需求的用例;或者经过替代策略筛选后成为某关注需求的必选用例.将 γ'_C 中用例 t'_j 同 γ_C 中用例 t_j 匹配,匹配原则是:或者 $t'_j\equiv t_j$,或者 t'_j 覆盖 t_j 唯一覆盖的需求,直至 γ'_C 为空.匹配后寻找第一个不相同的测试用例.存在 3 种情况:

(1)第一个用例不相同,即 $t'_1\neq t_1$.此时 t_1 必然是 t'_1 的替代用例;根据引理 1,用 t_1 替代 t'_1 所得约简集仍然最优;

(2)第 $j(j>1)$ 个用例不相同,即 $t'_j\neq t_j$.在选定

$\{t_1,\cdots,t_{j-1}\}$后，尚未覆盖的关注需求集构成 cur_CR，其中的关注需求不可能为$\{t_1,\cdots,t_{j-1}\}$所能够替代的测试用例覆盖. 因此 t_j 依然是(或成为)t'_j的替代用例；根据引理 1，用 t_j 替代 t'_j所得约简集仍然最优；

(3) 第 j 个用例不能匹配. 表明存在 t'_k($t'_k\in\gamma'_C\wedge k<j$)覆盖了 t_j唯一覆盖的关注需求. 这说明t'_k相对于t_k覆盖了更多的关注需求，t_k不能成为t'_k的替代用例，从而也不能成为相应需求的必选用例. 这同 γ_C的构成原则矛盾.

由于第(3)种情况不能成立，按第(1)和(2)两种情况得出 γ_C至少和γ'_C相对于部分覆盖约简目标等同，这同假设矛盾. 定理得证.　　证毕.

实际的约简过程可能无法达到理想状态，即在约简过程中存在必选策略无法应用的情况. 此时需要使用贪婪策略选择当前效用指标最高的测试用例，所得到的约简集不再是最优.

4.4　算法描述

HATS 算法遵循启发式贪婪搜索框架，应用必选策略、替代策略和贪婪策略完成部分覆盖测试用例集约简. 算法描述如图 1 所示.

```
算法 HATS
输入：R：测试需求集
      T：备选测试用例集
      CR：关注测试需求集
      Δ：|R|×|T|二进制覆盖矩阵
      α：权重因子
输出：γC：约简集
变量：cur_CR：当前尚未覆盖的关注需求集
      cur_T：当前备选测试用例集
      cur_Select：本次选择的测试用例集
      Rj：用例 tj的覆盖需求集
begin
/*初始化*/
γC=∅; cur_CR=CR; cur_T=T;
while (cur_CR≠∅)
    /*替代策略*/
    for each (tj∈cur_T)
        基于 cur_T 计算 SUPj;
        if (SUPj≠∅) cur_T=cur_T −{tj}; end if
    end for
    cur_Select=∅;
    /*必选策略*/
    for each (ri∈cur_CR)
        查找 essi;
        if (essi≠ null) cur_Select=cur_Select∪{essi}; end if
    end for
    if (cur_Select==∅)
        /*贪婪策略*/
        for each (tj∈cur_T) 计算其效用值 ωj;
        cur_Select={具有 max(ωj)的第 1 个 tj};
    end if
    /*后续处理*/
    for each (tj∈cur_Select) cur_CR=cur_CR −Rj; end for
    cur_T=cur_T −cur_Select; γC=γC∪ cur_Select;
end while
```

图 1　HATS 算法描述

在每一次外层循环(while 循环)中，首先应用替代策略筛除当前备选集(cur_T)中可以被替代的测试用例. 然后针对当前关注需求集(cur_CR)中的每一个(未覆盖)测试需求 r_i，检查其是否存在 ess_i，即是否可以应用必选策略；所选择的必选测试用例构成 cur_Select. 如果必选策略不能应用，则应用贪婪策略选择具有最高效用值(ω_j)的测试用例 t_j构成 cur_Select. 最后将 cur_Select 从 cur_T 中移出并入约简集 γ_C，从 cur_CR 中删除已经覆盖的关注需求. 重复外层循环直至 cur_CR 为空.

考虑算法的性能，根据必选策略和贪婪策略，每次循环至少选择备选集 T 中一个测试用例，该用例至少覆盖并移除关注需求集 CR 中的一个需求. 初始情况下有$|T|$个备选用例和$|CR|$个关注需求，因此外层循环的次数最多为 $\min(|T|,|CR|)$. 循环体包括 4 个部分：替代策略部分复杂度为 $O(|R|\times|T|^2)$. 必选策略部分、贪婪策略部分以及后续处理部分的复杂度均为 $O(|R|\times|T|)$. 累计这 4 个部分的时间复杂度得出外层循环体的时间复杂度为 $O(|R|\times|T|^2)$.

按以上分析，令备选集 T 的用例数量为 m，需求集 R 的需求数量为 n，HATS 算法的时间复杂度应为 $O(\min(m,n)(nm^2))$. 这个复杂度虽然达到了 n^4或 m^4量级，和 HGS 及 GRE 算法相当. 但 HATS 算法的实际性能不至于此. 我们在实践中应用 HATS 算法约简 10^4级别的需求集和 10^2级别的用例集，所需执行时间不超过 10s. 这个时间相对于长达数小时至数天的回归测试时段可以忽略.

5　实例分析

本文的创新性贡献包括两个方面：一是提出部分覆盖用例集约简概念，形式化定义多目标用例集约简问题，在选择性回归测试中进一步减少测试工作量和成本；二是提出 HATS 算法解决该问题，在减少约简集规模的同时避免对无关需求的覆盖. 为验证本文所提概念和技术的有效性，我们设计实验解答以下问题.

问题 1.　同全覆盖用例集约简技术相比较，部分覆盖约简技术能否进一步减少约简集的规模？

问题 2.　对于部分覆盖用例集约简，HATS 算法能否在减少约简集规模的同时有效降低对无关需求集的覆盖比率？

问题 3.　对于部分覆盖用例集约简，约简技术

对关注需求集的缺陷检测率负面影响程度如何？HATS算法能否降低对缺陷检测率的负面影响？

问题 4. HATS算法中权重因子 α 的不同设置对部分覆盖用例集约简问题的两个目标影响如何？

5.1 实验设计

为回答上述问题，我们选择了 3 个中等规模的开源 Java 软件程序. 第 1、2 个为 NanoXML 的不同版本(下载于 http://sir.unl.edu)，我们选择 NanoXML 第 3 和第 5 版作为实验对象. 第 3 个是 JTopas(下载于 Sourceforge)，我们选择其最新版本和相应的用例集作为实验对象. 表 1 列举了这 3 个 Java 程序的相关信息.

表 1 3 个实验对象的信息

标识	名称	接口数	类数	方法数	语句块数	初始用例数量	语句块覆盖
N03	NanoXML v3	6	15	214	847	216	544/847
N05	NanoXML v5	6	18	231	926	216	587/926
J08	JTopas r0.8	15	41	478	1776	220	1249/1776

我们定义语句块为测试需求. Java 语句块是一串没有跳转的语句或表达式集合，视为测试覆盖的基本单元. 这 3 个程序的初始用例集都未能达到 100%语句块覆盖率，这在业界软件系统测试中是一个普遍现象. 即使对一个相对简单的软件程序，达到 100%语句(块)覆盖率也需要较大的成本[15]. 未覆盖的测试需求在实验中被忽略.

在每一个设计的实验中，我们选择一个类中所包含的测试需求(语句块)作为该实验的关注需求集. 这种设计对应的设想是这个类最近被更新或修复了缺陷，是本次选择性回归测试关注的对象. 该设计可以容易地扩展到多个类被同时修改的情况. N03 有 4 个类包含人工植入的缺陷，N06 有 6 个类包含植入缺陷，据此我们设定 10 个对应实验；对于 J08 我们选择 8 个具有高语句块覆盖率的类并设定 8 个对应实验. 表 2 列举了这 18 个实验的相关信息. 其中 CR_{covered} 表示关注需求集中被初始用例集覆盖的关注需求集合.

表 2 18 个设定实验的信息

实验标识		CR	$\|CR\|$	$\|CR_{\text{covered}}\|$
N03	N03-1	StdXMLBuilder	36	32
	N03-2	StdXMLParser	108	86
	N03-3	XMLElement	206	118
	N03-4	XMLException	27	11
N05	N05-1	ContentReader	34	24
	N05-2	NonValidator	98	62
	N05-3	StdXMLParser	119	95
	N05-4	StdXMLReader	104	54
	N05-5	XMLEntityResolver	19	17
	N05-6	XMLUtil	108	82
J08	J08-1	Token	79	65
	J08-2	StandardTokenizer	33	32
	J08-3	TokenizerProperty	66	52
	J08-4	StandardTokenizerProperties	234	181
	J08-5	PatternMatcher	28	22
	J08-6	SequenceStore	83	75
	J08-7	TokenizerException	15	13
	J08-8	CharArraySource	16	16

5.2 约简集规模

首先需要考虑的问题是部分覆盖用例集约简是否能够进一步减少约简集的规模以及 HATS 算法在这方面的性能如何. 我们采用 Matlab(http://www.mathworks.com/)分别实现了 HGS[1]算法、GRE[2]算法、BIP[6]算法和本文提出的 HATS 算法，其中 HGS、GRE 和 BIP 算法可以采用需求集 R 和初始(备选)用例集 T 为输入(分别标记为 HGS_T、GRE_T 和 BIP_T)，以代表全覆盖用例集约简；也可采用关注需求集 CR 和备选集 T 为输入(标记为 HGS_S、GRE_S 和 BIP_S)，以代表单目标部分覆盖用例集约简. HATS 算法通过对权重因子 α 的不同设置运行多个实例，如 $\text{HATS}_{0.5}$ 表示该实例中因子 α 设置为 0.5.

图 2 用箱形图(box-plot)描述了不同算法实例在 18 个实验中运行得到的约简集规模分布. 为便于在不同程序间比较，初始用例集规模被归一化为 100，算法执行得到的约简集规模表示为相应初始集规模的百分比值.

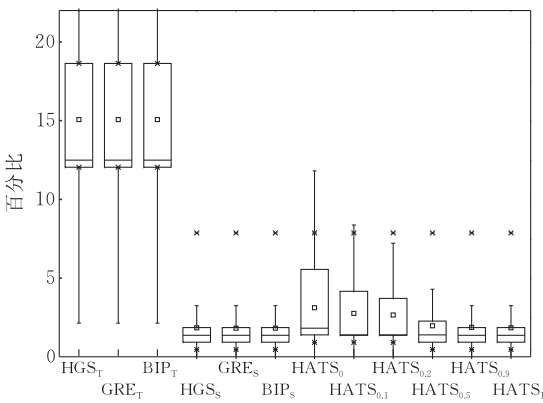

图 2 不同算法得到的约简集规模百分比值

根据图 2 不难看出，同全覆盖用例集约简相比，部分覆盖用例集约简技术能够进一步较大幅度减少约简集 γ_C 的规模. 在一些实验中的降低幅度很可

观，如在 N05-1 和 J08-2 中，部分覆盖约简所得到的最小 γ_C 只包含 1 个测试用例，其原因是相应的关注需求集 CR 在需求集 R 中所占的比例较小.

在所有 18 个实验中，HGS、GRE 和 BIP 算法得到的约简集规模 $|\gamma_C|$ 基本相同，不管输入是需求集 R 还是关注需求集 CR 皆如此；唯一的差异出现在 J08-4，在以 CR 为输入时，HGS_S 得到的约简集比其它两个算法多 1 个用例. 算法 HATS 中 $HATS_1$ 得到的 $|\gamma_C|$ 和其它 3 个全覆盖约简算法以 CR 为输入时得到的值相同，唯一的区别出现在 N05-6，此时 $HATS_1$ 多选了 1 个用例. 在部分覆盖约简算法和实例中，$HATS_0$ 得到的 $|\gamma_C|$ 总是最大，但仍然远小于全覆盖约简算法的结果. $HATS_{0.5}$ 得到的 $|\gamma_C|$ 同 $HATS_1$ 相差不大，在所有的实验中最多不超过 1 个测试用例.

5.3　无关需求集的覆盖率

第 2 个考虑的问题是同单目标部分覆盖约简技术相比，HATS 算法能否进一步降低对无关需求集的覆盖率. 我们以关注需求集 CR 为输入运行 HGS、GRE 和 BIP 算法，并通过 α 的不同设置运行 HATS 的多个实例. 在各实验中我们测量算法所得约简集对无关需求集的覆盖百分比值（标记为 $Cov_{|R-CR|}$），其中初始用例集未覆盖的无关需求被忽略.

图 3 用箱形图描述了各个算法在 18 个实验中所得约简集的 $Cov_{|R-CR|}$ 分布. 可以看出，部分覆盖约简技术可以减少对无关需求的覆盖. 各个算法的 $Cov_{|R-CR|}$ 均值大约在 60%上下，这意味着在平均情况下约有 40%的无关需求可以避免，从而减少回归测试执行和分析的工作量.

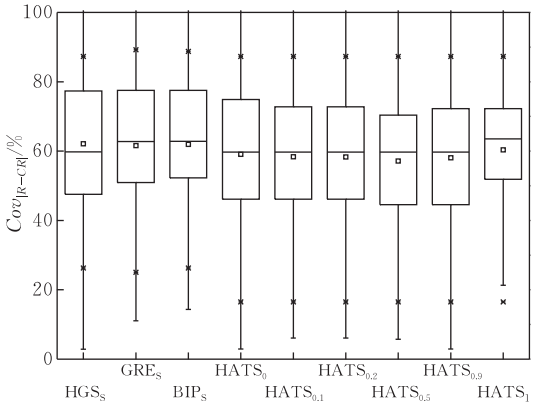

图 3　不同算法所得约简集对无关需求的覆盖百分比值

各算法得到的 $Cov_{|R-CR|}$ 偏差范围普遍较大. 以 CR 为输入的 3 个全覆盖约简算法虽然得到的约简集规模相当，但由于选定的测试用例各不相同，其 $Cov_{|R-CR|}$ 有一定差异. 其中 BIP_S 所得偏差范围相对较小，但在多个实验中 BIP_S 得到最差的 $Cov_{|R-CR|}$ 值. 对于 HATS 算法，$HATS_1$ 得到的 $Cov_{|R-CR|}$ 值和 3 个算法相当，但具有最小的偏差范围. $HATS_0$ 虽然以避免对无关需求的覆盖为唯一目标，但由于选择了较多的测试用例，所得的 $Cov_{|R-CR|}$ 值常常较差. 在绝大多数实验中，$HATS_{0.5}$（平衡考虑减少约简集规模和避免无关需求覆盖）得到了最小的 $Cov_{|R-CR|}$ 值；两个例外是 N05-4 和 N08-4，其最小值都来自 $HATS_1$. 综上所述，在部分覆盖约简技术中，只要权重因子 α 的设置适当，HATS 算法能够进一步减少对无关需求的覆盖比率.

5.4　缺陷检测能力

第 3 个考虑的问题是部分覆盖约简技术对关注需求集相关缺陷检测能力的负面影响以及 HATS 算法能否缓解负面影响. 在 18 个实验中，我们首先测量初始用例集能够达到的针对关注需求集的缺陷检测率. 然后分别运行 HGS、GRE、BIP 和 HATS 算法，前 3 个算法包括全覆盖实例和部分覆盖实例，HATS 算法包括 α 的不同设置. 所得约简集的缺陷检测能力反映为相对初始集缺陷检测率的百分比值（标记为 Ψ_{CR}）. 如此处理的原因一方面是由于初始用例集未达到 100%语句（块）覆盖率，从而也不能达到 100%缺陷检测率；另一方面是便于在不同实验不同关注需求集之间的比较.

我们采用抗变异值（mutation score）反映对关注需求集 CR 的缺陷检测率. 抗变异值指用例集能够检测出变异（mutant）的百分比. 由于实验对象程序由 Java 编写，因此可采用 Jumble 工具（http://jumble.sourceforge.net/）来测量用例集针对给定 Java 类（关注需求集）的抗变异值. 图 4 用箱形图描述了 18 个实验中各个算法所得约简集针对关注需

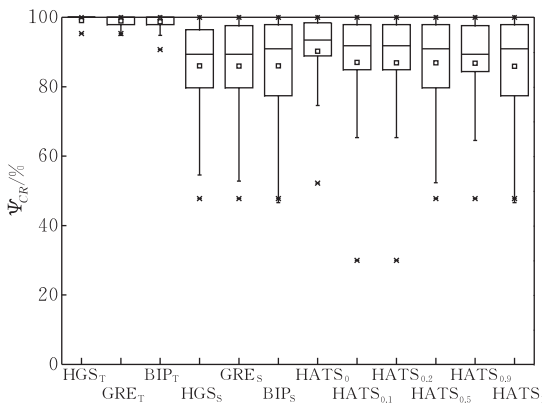

图 4　不同算法所得约简集针对关注需求集的缺陷检测能力

求集 CR 的抗变异值相对初始用例集抗变异值的百分比值 Ψ_{CR}.

由于约简技术减少了测试用例数量,必然会降低针对 CR 的缺陷检测能力.在我们的实验中,全覆盖用例集约简技术对缺陷检测能力的影响不大;其中 HGS_T 在多数情况下表现最好,只在 4 个实验中 Ψ_{CR} 低于 100%,但都在 95%以上;而 BIP_T 则表现出较大的偏差范围.采用部分覆盖约简,由于测试用例数量进一步减少,所得到的 Ψ_{CR} 明显降低;其中实验 N05-1、N05-5 和 N08-3 中降低幅度达到 40%.此时 HGS_S 和 GRE_S 的结果好于 BIP_S,但 3 个算法中没有哪个算法能在所有实验中保持优势.

对于 HATS 算法,$HATS_1$ 的 Ψ_{CR} 值同 HGS_S 和 GRE_S 基本相同.$HATS_{0.5}$ 虽然在约简集规模和无关需求覆盖方面表现较好,但在缺陷检测能力方面较差,所得到的 Ψ_{CR} 值在所有实验中和 $HATS_1$ 相同或优势很小($\leq 5\%$).$HATS_0$ 在所有部分覆盖约简算法和实例中的缺陷检测能力最好,一方面在除 J08-8 之外的所有实验中得到了最大的 Ψ_{CR} 值;另一方面其偏差范围在这些算法中也最小;值得提出的是在 N05-3 和 N05-6 中,其 Ψ_{CR} 值甚至超过了 3 个全覆盖约简算法.综上所述,HATS 算法能够在一定程度上缓解部分覆盖用例集约简对缺陷检测能力的负面影响.

5.5 权重因子 α

第 4 个考虑的问题是 HATS 算法中权重因子 α 对部分覆盖约简问题的两个目标影响如何,包括约简集规模和无关需求覆盖比率.同前述相同,这两个指标分别反映为约简集规模 $|\gamma_C|$ 占初始用例集规模 $|T|$ 百分比值及约简集 γ_C 覆盖无关需求占无关需求总数 $|R-CR|$ 的百分比值 $Cov_{|R-CR|}$.

图 5 用箱形图描述了 $|\gamma_C|$ 百分比值和 $Cov_{|R-CR|}$ 随 α 取值的变化趋势.其中 α 取 0 到 1,间隔为 0.05;各点的期望值用虚线连接.考查 $|\gamma_C|$ 百分比值,不难发现 $|\gamma_C|$ 随 α 增大而单调下降.当 α 取值小于 0.15 时,$|\gamma_C|$ 取值相对较高;此时当 α 增大时,$|\gamma_C|$ 下降较快;在大多数实验中当 α 等于 0.55 时,$|\gamma_C|$ 已达到最小值;当 α 再增大,$|\gamma_C|$ 基本不变或变化很小.在部分实验中,如 N03-3 和 J08-7,$|\gamma_C|$ 保持不变,与 α 取值无关.

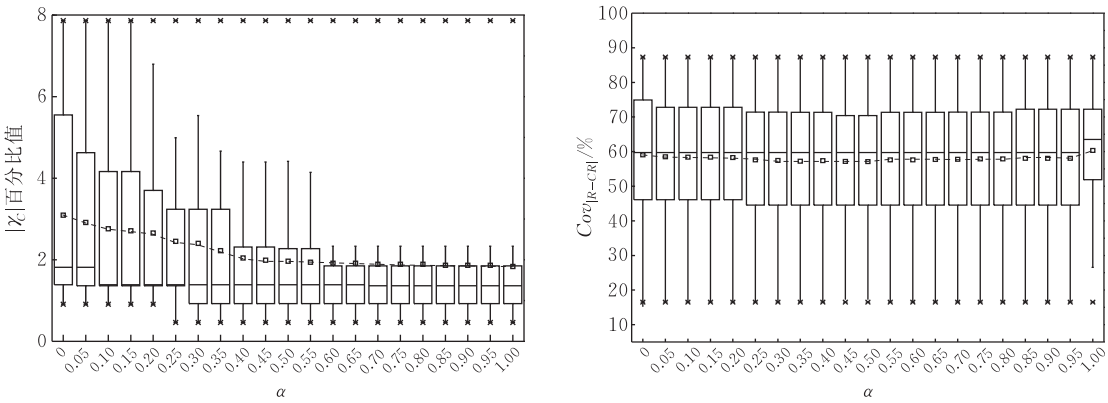

图 5 HATS 算法中权重因子 α 对约简集规模和无关需求覆盖率的影响趋势

考查 $Cov_{|R-CR|}$,$Cov_{|R-CR|}$ 随 α 增大变化比较复杂.在大多数实验中 $Cov_{|R-CR|}$ 随 α 的趋势线呈现盆形,即开始阶段 $Cov_{|R-CR|}$ 随 α 增大而下降;中间阶段达到最小值,此时多数实验中 $Cov_{|R-CR|}$ 在较大的 α 取值范围内保持不变或变化很小;部分实验中 $Cov_{|R-CR|}$ 存在多个最小值阶段;过了盆底阶段后,$Cov_{|R-CR|}$ 随 α 增大缓慢增大;当 α 从 0.95 乃至 0.99 变为 1 时,$Cov_{|R-CR|}$ 通常突然增大,例外情况存在于 N05-4 和 N08-4,当 α 变为 1 时 $Cov_{|R-CR|}$ 反而下降.在 N03-3 和 N03-4 中,$Cov_{|R-CR|}$ 没有变化,其原因是在 α 从 0 到 1 增大的过程中,所得约简集 γ_C 的组成没有变化.

5.6 讨 论

根据实验结果数据,同全覆盖用例集约简相比,部分覆盖约简技术能够进一步降低约简集的规模.通过设置合适的权重因子 α,HATS 算法能够在同时减少对无关需求的覆盖.这可以较大程度降低选择性回归测试的成本和工作量,且后者在软件系统开发过程中需要频繁应用,从而能够降低软件开发成本.

由于部分覆盖约简进一步减少了测试用例数量,缺陷检测能力受到负面影响.按照实验数据,$HATS_0$ 能够有效缓解这一影响,在部分实验中针对关注需求集的缺陷检测能力甚至超过了全覆盖约简.但也存在部分实验,同初始用例集相比,$HATS_0$

所能达到的缺陷检测能力仍有较大差距.这说明非关注需求也需要区别对待,如何识别重要的非关注需求将是我们下一步的工作之一.

我们以 3 个中等规模的开源 Java 软件系统为实验对象,测试需求定义为 Java 的语句块.可以考虑的扩展是添加 C/C++等不同语言编写的软件系统为分析对象,将测试需求定义为其它类型覆盖单元如分支或谓词组合等,以增加实验结果的代表性.另一个需要考虑的问题是实验软件系统的初始(备选)用例集未能达到 100%需求覆盖率,有必要补充新的测试用例;尽管在实践软件项目开发中由于时间和成本的限制,100%测试需求覆盖是难以达成的目标.

6　结　论

选择性回归测试主要用于软件系统局部更新和部分缺陷修正,在当今软件系统开发中需要频繁应用,如何有效完成部分需求覆盖用例集约简从而控制其成本是一个需要考虑的问题.针对该问题的测试需求集被分成两部分:关注需求集和无关需求集.据此本文定义(部分覆盖)多目标测试用例集约简问题:一方面在覆盖关注集的要求下尽量减少约简集规模;另一方面避免对无关集的覆盖,从而减少测试执行和分析的工作量.

本文提出 HATS 算法解决多目标用例集约简问题.HATS 算法遵循启发式贪婪搜索框架,应用 3 个策略:必选策略、替代策略和贪婪策略,以选择当前最合适的测试用例.定义权重因子 α 平衡两方面目标:减少约简集规模同时避免无关需求的覆盖.实验数据表明,同现有约简算法相比较,HATS 算法能够较大幅度减少约简集规模,降低对无关需求的覆盖比率,同时缓解对缺陷检测能力的负面影响.

下一步工作包含两个方面:一方面考虑关注集和无关集的分割准则,提高部分覆盖约简技术针对关注集的缺陷检测能力.另一方面推动 HATS 算法的实践应用,通过更多的软件系统实例验证其有效性,包括面向不同的编程语言、定义新的测试需求类型、提升备选用例集对测试需求的覆盖率等.

参　考　文　献

[1] Harrold M J, Gupta R, Soffa M L. A methodology for controlling the size of a test suite. ACM Transactions on Software Engineering and Methodology, 1993, 2(3): 270-285

[2] Chen T Y, Law M F. A new heuristic for test suite reduction. Information and Software Technology, 1998, 40(5-6): 347-354

[3] Chen T Y, Law M F. Dividing strategies for the optimization of a test suite. Information Processing Letters, 1996, 60(3): 135-141

[4] Mansour Nashat, El-Fakih Khalid. Simulated annealing and genetic algorithms for optimal regression testing. Journal of Software Maintenance: Research and Practice, 1999, 11(1): 19-34

[5] Black Jennifer, Melachrinoudis Emanuel, Kaeli David. Bi-criteria models for all-uses test suite reduction//Proceedings of the 26th International Conference on Software Engineering (ICSE'04). Scotland, UK, 2004: 106-115

[6] Fischer K, Raji F, Chruscicki A. A methodology for retesting modified software//Proceedings of the National Telecommunications Conference. New Orleans, Louisiana, 1981: 1-6

[7] Cormen T H, Leiserson C E, Rivest R L. Introduction to Algorithms. Cambridge, MA: MIT Press, 1990

[8] Karp Richard M. Reducibility among combinatorial problems//Miller R E, Thatcher J W eds. Complexity of Computer Computations. New York: Plenum, 1972: 85-103

[9] Zhong H, Zhang L, Mei Hong. An experimental comparison of four test suite reduction techniques//Proceedings of the 28th International Conference on Software Engineering (ICSE'06). Shanghai, China, 2006: 636-640

[10] Sprenkle S, Sampath S, Gibson E, Pollock L, Souter A. An empirical comparison of test suite reduction techniques for user-session-based testing of Web applications//Proceedings of the 21st IEEE International Conference on Software Maintenance (ICSM05). Budapest, Hungary, 2005: 587-596

[11] Fraser G, Wotawa F. Redundancy based test-suite reduction//Proceedings of the 10th International Conference on Fundamental Approaches to Software Engineering (FASE 2007). Braga, Portugal. LNCS 4422. New York, USA: Springer, 2007: 291-305

[12] Jeffrey Dennis, Gupta Neelam. Test suite reduction with selective redundancy//Proceedings of the 21st IEEE International Conference on Software Maintenance (ICSM05). Budapest, Hungary, 2005: 549-558

[13] McMaster Scott, Memon Atif M. Call-stack coverage for GUI test suite reduction. IEEE Transactions on Software Engineering, 2008, 34(1): 99-115

[14] Chen Yih-Farn, Rosenblum D S, Vo Kiem-Phong. TESTTUBE: A system for selective regression testing//Proceedings of the 16th International Conference on Software Engineering (ICSE-16). Sorrento, Italy, 1994: 211-220

[15] Juristo Natalia, Moreno Ana, Strigel Wolfgang. Software testing and industry needs. IEEE Software, 2006, 23(4): 55-57

GU Qing, born in 1972, Ph. D., professor. His research interests include software quality and software testing.

TANG Bao, born in 1974, M. S., senior engineer. His research interests include software quality and software testing.

CHEN Dao-Xu, born in 1947, professor. His research interests include distributed computing and parallel processing.

Background

The research problem of this paper is test suite reduction, which belongs to the research area of Software Testing. Test suite reduction is needed during regression testing where a smaller set of test cases are selected from maintained test suite built during software development. Here the tested software system is treated as a set of test requirements, which are defined by test coverage units, such as code blocks or branches. Researchers have put forward techniques and algorithms to handle the full-coverage test suite reduction problem. These methods can roughly be divided into three categories: greedy heuristic searches, global probabilistic meta-heuristics, and binary integer linear programming. Performance comparison has been made among these techniques, and no significant difference was found.

Current test suite reduction techniques require selected test cases cover the whole set of test requirements. But under more frequent minor updates and partial corrections, the selective-form regression testing is needed where only a subset of test requirements needs to be covered. This leads to the selective-coverage test suite reduction problem. To solve it, this paper defines the multi-objective test suite reduction problem, and puts forward the HATS algorithm to solve the problem. The HATS conforms to the greedy heuristic search framework, and applies three strategies to select the best-for-now test case(s) in each cycle. The experimental results and practical uses have proved that the HATS can largely reduce the number of test cases needed, lower the coverage of unwanted test requirements, while keep the fault detection abilities against the concerned test requirements.

This work was partially supported by the National Science Foundation of China under grant No. 60873027, the National High Technology Research and Development Program (863 Program) of China under grant No. 2006AA01Z177, the National Basic Research Program (973 Program) of China under grant No. 2009CB320705, and the National Science Foundation of China under grant No. 61021062. Under these projects, our research group planned to develop a series of theories and technologies in Software Testing. The main objective was to enhance the effectiveness of software testing, while save the test effort. We have already published dozens of research papers, and applied for several national patents of invention in this research area. Work of this paper helps to enhance both the effectiveness and the efficiency of regression testing.