doi:10.11887/j.cn.201702027 http://journal.nudt.edu.cn

测试性增长中资源优化配置模型及求解*

赵晨旭[1,2],邱 静[1,2],刘冠军[1,2]
(1. 国防科技大学 机电工程与自动化学院,湖南 长沙 410073;
2. 国防科技大学 装备综合保障技术重点实验室,湖南 长沙 410073)

摘 要:良好的测试性设计对系统维修性具有重要意义,测试性增长试验通过一系列测试性设计缺陷发现和纠正措施,可保证系统测试性指标达到设计要求。针对基于延缓纠正的测试性增长过程中的资源配置问题进行研究,基于增长试验目标是否明确和试验资源是否受限制问题构建资源优化配置模型,并提出一种基于拉格朗日松弛和本地搜索的快速优化算法。仿真结果表明:该模型能够有效指导测试性增长中的资源优化配置问题,所提混合优化方法能够高效、准确地求解整数规划问题。

关键词:测试性设计;资源配置;增长试验;整数规划
中图分类号:TH17;TP30 **文献标志码**:A **文章编号**:1001-2486(2017)02-178-06

Resource allocation problem formulation and solution in testability growth

ZHAO Chenxu[1,2], *QIU Jing*[1,2], *LIU Guanjun*[1,2]
(1. College of Mechatronic Engineering and Automation, National University of Defense Technology, Changsha 410073, China;
2. Science and Technology on Integrated Logistics Support Laboratory, National University of Defense Technology, Changsha 410073, China)

Abstract: Testability is crucial for the enhancement of system maintainability, and testability growth can promote the testability metric of the system to satisfy the design requirements by a series action of identifying and correcting the testability design defects. The test resources allocating problem arising in delay fix based testability growth was studied and modeled, considering that whether there are constrains on the growth object and growth test cost or not. A Lagrangian relaxation algorithm and local search hybrid optimal algorithm were applied to solve this problem. Simulation results show that the proposed model is feasible for the test resources allocating problem in testability growth test, and the hybrid optimal algorithm can promise the effectiveness and accuracy to the integer programming problem.

Key words: design for testability; resource allocation; growth test; integer programming

在系统研制阶段,测试性增长试验是发现并及时改进测试性设计缺陷,保证系统测试性水平达到设计要求最有效的方法,对于对系统测试性有特别要求的各类新研系统更是如此[1]。基于延缓纠正策略的增长试验是测试性增长的重要试验方式之一。延缓修正的一般过程如下:首先开展测试性验证试验[2],在试验过程中系统设计缺陷被不断发现;一次完整的验证试验开展完成后,试验管理者利用试验成败型数据评估当前系统的测试性水平;如果系统现有测试性水平未达到设计要求值,设计师就会对所暴露的缺陷开展一系列的“试验-分析-改进-试验(Test-Analyze-Fix-Test, TAFT)”措施,从而提高系统的测试性水平。

考虑测试的不确定性,系统中每个故障模式的故障检测/隔离概率往往为[0,1]区间内的小数[3],测试性增长的实质即是测试不确定性的减少。系统中各故障模式的故障诊断能力往往呈现不同的水平,对其进行测试性增长时对系统总体故障检测/隔离率的贡献也不一样。有些故障模式诊断方法比较成熟,测试性水平已经较高,继续增长潜力不大;有些故障模式测试性水平较低,但是其属于新型故障模式,在现有技术能力和条件下对其进行设计改进并不能使其测试性水平突增;还有一些故障模式测试性水平不高,但其故障发生概率也相对较小,即使使其测试性水平达到100%,对系统测试性整体水平的提升贡献也不大。因此基于延缓纠正策略的测试性增长试验规

* **收稿日期**:2015-10-08
基金项目:国家自然科学基金资助项目(51175502)
作者简介:赵晨旭(1987—),男,河南郑州人,博士研究生,E-mail:zhao_chenxu@126.com;
邱静(通信作者),男,教授,博士,博士生导师,E-mail:qiujing16@sina.com

划面临一个突出问题:如何将有限的设计改进资源合理地分配[4-6]至每个故障模式,使系统的整体测试性水平得到最大的提升。对于暴露出来的具有测试性设计缺陷的故障模式而言,测试性设计改进成本主要体现在对其实施的 TAFT 阶段数上。因此试验资源配置问题即是如何优化确定对每个故障模式开展的 TAFT 阶段数的问题。该问题是一个整数规划问题。

整数规划问题是运筹学的一个重要分支,如何求解该问题一直是一个重要的研究领域[7]。随着人工智能和计算机技术的发展,涌现出了一批群体智能优化算法,包括遗传算法[8]、分布估计算法[9]、粒子群算法[10]等。这些方法从生物群体智能进化的角度出发,在可行解区域内进行大规模平行搜索,解决了传统整数规划方法单点搜索效率低的问题,对大规模的整数规划问题较为有效[11]。但是这些算法不是确定性算法,对问题的求解不唯一,并且往往只能得到近似最优解,甚至多组解。为了得到整数规划问题的确定性最优解,传统的基于数学规则的搜索算法仍然是最有效的。这些方法包括分支定界法[12]、割平面法[13]、分解算法[14]、松弛算法[15]等。随着求解问题规模的增大,其计算量变得极为可观。

1　问题分析与建模

假设某次测试性验证试验后不可检测/隔离的故障模式种类数为 M,$Cost_i$ 表示分配给故障模式 i 的设计更新总费用。对于每个故障模式而言,设计改进成本主要可以分为三部分:故障注入试验的成本、测试性设计缺陷改进费、支付给设计师和试验者的工资成本[4-6]。

试验成本主要是指试验室场地使用费、试验对象故障件加工采购费,这些费用一般为固定值,不会随着试验的进行而增长。由于暴露出来的设计缺陷往往不能只经一次改进就达到设计目标,如果不在设计改进阶段反复纠正与验证,在后续的测试性验证试验过程中该缺陷仍有极大可能重复出现,这将导致系统测试性指标一直达不到设计要求,从而需要反复开展测试性验证试验,造成不必要的损失。于是每个故障模式都会反复经历 TAFT 的过程,而每阶段试验所需的故障注入次数服从几何分布[1]。令 k_i 表示故障模式 i 按照资源配置得到的 TAFT 阶段数,$Cost_{1i}$ 表示故障模式 i 进行 1 次故障注入试验所需的成本。验证设计改进效果并发现新缺陷的故障注入试验总成本可用式(1)表示。

$$E(Cost_{i\_part1}) = \sum_{j=1}^{k_i} \frac{Cost_{1i}}{\bar{d}_i(j)} \tag{1}$$

其中,$\bar{d}_i(j)$ 为故障模式 i 在第 j 次测试性设计改进更新之前所具有的不可检测 / 隔离概率,其初值可以利用试验成败型数据计算,也可以利用文献[3]中的方法进行经验估计。

一般情况下,随着设计改进的推进,改进的难度越来越大,希望得到更高的检测/隔离效果就必须采用更精密的测量设备和更高级的诊断算法。可以认为测试性设计缺陷改进费随着改进难度的检测/隔离概率的增加和改进量的增大而逐渐增加。另外,随着改进难度的增大,设计更新所耗费的时间必然逐渐增长,成本工资也会相应提高。按照一般工程经验,采用指数模型来描述上述两项成本。

令 $Cost_{2i}$ 表示故障模式 i 进行设计改进时检测/隔离概率提高 1% 所需的成本系数,用故障模式 i 所具有的检测/隔离概率 $1-\bar{d}_i(j)$ 描述改进的难度,用 $\bar{d}_i(j)-\bar{d}_i(j+1)$ 表示第 j 次设计改进提高的故障检测/隔离概率。故障模式 i 在第 j 次设计改进时设计更新所消耗的更新成本为:

$$E(Cost_{i\_part2}) = \sum_{j=1}^{k_i} \{100Cost_{2i}[1-\bar{d}_i(j)]^{\gamma_{1i}} \cdot [\bar{d}_i(j)-\bar{d}_i(j+1)]\} \tag{2}$$

其中,γ_{1i} 为指数函数参数。

由于设计师和试验者的工资通常在试验结束时统一支付,试验结束时故障检测 / 隔离概率提高的总量为 $\bar{d}_i(1)-\bar{d}_i(k_i+1)$。当用 $1-\bar{d}_i(1)$ 表示改进过程的总难度,并令 $Cost_{3i}$ 表示对故障模式 i 进行设计更新需要付给设计师和试验者的额外工资系数时,工资成本为:

$$E(Cost_{i\_part3}) = Cost_{3i}[1-\bar{d}_i(1)]^{\gamma_{2i}} \cdot [\bar{d}_i(1)-\bar{d}_i(k_i+1)] \tag{3}$$

其中,γ_{2i} 为指数函数参数。

于是对故障模式 i 进行测试性设计改进所需的总成本 $Cost_i$ 为:

$$E(Cost_i) = E(Cost_{i\_part1}) + E(Cost_{i\_part2}) + E(Cost_{i\_part3}) \tag{4}$$

对于所有 M 个需要进行设计更新的故障模式,设计更新所需的总成本 $Cost$ 为:

$$E(Cost) = \sum_{i=1}^{M} E(Cost_i) \tag{5}$$

在指数函数中,幂指数的取值决定了函数的形状。根据不可检测 / 隔离概率的取值特点,式(2)和式(3)中指数函数的底数取值范围为 $[0,1]$。令 $f(x)=x^b$,$x\in[0,1]$,则 $f(x)$ 的变化

趋势如图 1 所示。

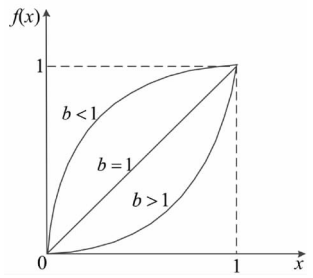

图 1 指数函数变化趋势

Fig. 1 Change trend of exponential function

在测试性增长过程中,成本随着改进目标的变化趋势通常呈现先缓后急,因此式(2)和式(3)中的幂指数通常取值大于 1。

假设系统共有 N 个故障模式,故障率分别为 λ_i,不可检测/隔离概率为 $\bar{d}_i$,其中只有 M 个故障模式需要进行测试性设计更新。于是系统的两类故障模式具有如式(6)所示关系。

$$\begin{aligned}\frac{\sum_{i=1}^{N}\lambda_i\bar{d}_i}{\sum_{i=1}^{N}\lambda_i} &= \frac{\sum_{j=1}^{M}\lambda_j\bar{d}_j+\sum_{k=1}^{N-M}\lambda_k\bar{d}_k}{\sum_{i=1}^{N}\lambda_i}\\ &= \frac{\sum_{j=1}^{M}\lambda_j\dfrac{\sum_{j=1}^{M}\lambda_j\bar{d}_j}{\sum_{j=1}^{M}\lambda_j}+\sum_{k=1}^{N-M}\lambda_k\dfrac{\sum_{k=1}^{N-M}\lambda_k\bar{d}_k}{\sum_{k=1}^{N-M}\lambda_k}}{\sum_{i=1}^{N}\lambda_i}\\ &= \frac{\sigma\sum_{j=1}^{M}\lambda_j+\beta\sum_{k=1}^{N-M}\lambda_k}{\sum_{i=1}^{N}\lambda_i}=p_{\mathrm{req}}\end{aligned} \tag{6}$$

其中:p_{req}为整个系统的测试性增长目标;σ 为进行测试性设计改进的故障模式在改进之后所具有的不可检测/隔离概率加权和;β 为不需要进行测试性设计更新的故障模式的加权不可检测/隔离概率,β 可根据其成败型数据按照现有的系统测试性指标计算方法得到。假设每个故障模式的改进系数为 f_i,那么经历 k_i 次设计改进之后,故障模式 i 的不可检测/隔离概率满足:

$$\bar{d}_i(k_i+1)=\bar{d}_i(1)f_i^{k_i} \tag{7}$$

基于增长试验目标是否明确和试验资源是否受限制,对上述 M 个故障模式进行测试性增长时的资源配置的问题可以分为以下三类。

1.1 问题一

当没有明确的增长试验目标,而试验资源有限时,测试性增长试验资源优化配置问题可以认为是在有限的资源限制下,使系统的测试性指标达到最优的问题,可表述为式(8)所示的优化问题。

$$\begin{cases}\min\ \sum_{i=1}^{M}\dfrac{\lambda_i\bar{d}_i(1)f_i^{k_i}}{\sum_{j=1}^{n}\lambda_j}=\min\sum_{i=1}^{M}\delta_i f_i^{k_i}\\ \text{s.t.}\ \ E(Cost)\leqslant\chi\\ 0\leqslant k_i\leqslant k_{i\_\max}\end{cases} \tag{8}$$

其中,χ 为设计改进资源边界限制条件。

1.2 问题二

当有明确的增长目标,设计更新资源没有限制时,测试性增长试验资源优化配置问题可表述为以设计更新成本最低为目标,以规划的增长目标为限制条件的优化问题,如式(9)所示。

$$\begin{cases}\min\ E(Cost)\\ \text{s.t.}\ \ \sum_{i=1}^{M}\dfrac{\lambda_i\bar{d}_i(1)f_i^{k_i}}{\sum_{j=1}^{n}\lambda_j}\leqslant\sigma\\ 0\leqslant k_i\leqslant k_{i\_\max}\end{cases} \tag{9}$$

另外,限制条件中 $k_{i\_\max}$ 必须满足式(10),否则该问题无解。

$$\sum_{i=1}^{M}\frac{\lambda_i\bar{d}_i(1)f_i^{k_{i\_\max}}}{\sum_{j=1}^{n}\lambda_j}\leqslant\sigma \tag{10}$$

1.3 问题三

当有明确的增长目标时,若试验资源也有限制,则测试性增长试验资源优化配置问题可表述为当试验成本有限时,以增长结果与试验规划目标具有最小偏差为目标进行设计改进的问题,如式(11)所示。

$$\begin{cases}\min\ \left(\sum_{i=1}^{M}\delta_i f_i^{k_i}-\sigma\right)^2\\ \text{s.t.}\ \ E(Cost)\leqslant\chi\\ 0\leqslant k_i\leqslant k_{i\_\max}\end{cases} \tag{11}$$

若在相同成本约束条件下,问题一的最优解满足 $\min\sum_{i=1}^{M}\delta_i f_i^{k_{i\_\mathrm{opt}}}\geqslant\sigma$,则说明在完全消耗试验资源的前提下系统测试性增长极限值不能达到增长要求,问题一的最优解即为问题三的最优解;相反地,若在没有完全消耗试验资源的前提下可以满足系统测试性指标增长要求,即问题三的最优解在满足指标要求的同时,也满足 $\min E(Cost)\leqslant\chi$,则问题二的最优解即为问题三的最优解。因此,问题三的求解可以转化为在问题一和问题二求解结果中选取合适的一组求解结果的方式,三个问题的求解可以简化为两个问题的求解。

2 模型求解

求解整数规划问题最直接的方法就是通过穷

尽搜索枚举出所有可能组合,逐一进行筛选,选出符合限制条件的最优解。然而这种方法是非常耗费计算资源的。拉格朗日松弛算法(Lagrangian Relaxation Algorithm, LRA)是一种常用的求解整数规划的方法,但是由于算法本身的缺陷,LRA 求解结果通常仅为问题最优解的近似解[15]。为了克服这个缺点,针对测试性增长中资源优化配置问题一和问题二提出一种基于传统数学规则的搜索算法。该算法首先利用 LRA 得到问题的近似最优解,然后在该解的基础上利用本地搜索[16]进一步优化求解,从而得到所求问题的确定最优解。

由于问题一和问题二的不同之处仅是限制条件与目标函数的对换,因此下面仅以式(8)所示的问题一为例说明算法。问题二求解过程与问题一的区别在于利用 LRA 求解问题时拉格朗日函数的不同,以及本地搜索时搜索目标的不同,不再做专门阐述。

令 $f(k_i)=E(Cost_i)$,则式(8)可以重构为:

$$\begin{cases}\min\ J_f=\sum_{i=1}^{M}\delta_i f_i^{k_i}\\ \text{s.t.}\ \sum_{i=1}^{M}f(k_i)\leqslant\chi\\ 0\leqslant k_i\leqslant k_{i\_\max}\end{cases}\tag{12}$$

利用拉格朗日乘子 η 松弛约束条件,可得到对偶问题的拉格朗日函数:

$$J_\alpha=\sum_{i=1}^{M}\left[\delta_i f_i^{k_i}+\eta f(k_i)\right]-\eta\chi\tag{13}$$

拉格朗日松弛算法的基本流程如下:

Step 1:初始化,令 $\eta=0$, $\theta=1$, $J_f=+\infty$, $J'_\alpha=0$。

Step 2:利用全局搜索算法找到对偶目标函数取最大值的解 $k^*=\arg\max\{J_\alpha\}$。由于 $\eta\chi$ 为固定值,同时 $\forall k_i$, $[\delta_i f_i^{k_i}+\eta f(k_i)]\geqslant 0$,所以由 $k_i^*=\arg\max\{\delta_i f_i^{k_i}+\eta f(k_i)\}$ 组成的数组即为 k^*,此步的寻优过程不涉及参数组合问题;将 k^* 代入式(13)和式(14)分别计算 J_α 和次梯度 gra。若 $J_\alpha>J'_\alpha$,则令 $J'_\alpha=J_\alpha$。

$$gra=\sum_{i=1}^{M}f(k_i^*)-\chi\tag{14}$$

Step 3:利用如下步骤搜寻式(12)的可行解 kf^*:

1)令 $kf=k^*$。

2)重复下面步骤直至 $\sum_{i=1}^{M}f(kf_i)\leqslant\chi$。对于每个 $i(i=1,2,\cdots M)$,令 $kf'_i=kf_i-1$,计算式(15)的值,从而找到费效比变化最大的故障模式 $j=\arg\min_{i\in\{1,2,\cdots,M\}}\{g_i\}$,并且令 $kf_j=kf'_j$。

$$g_i=\text{abs}\left(\frac{\delta_i f_i^{kf'_i}-\delta_i f_i^{kf_i}}{f(kf'_i)-f(kf_i)}\right)>0\tag{15}$$

3)当 $\sum_{i=1}^{M}f(kf_i)\leqslant\chi$ 时,令 $kf^{*\prime}=kf$,并且计算 $J'_f=\sum_{i=1}^{M}\delta_i f_i^{kf^{*\prime}_i}$;若 $J_f'<J_f$,则令 $J_f=J_f'$, $kf^*=kf^{*\prime}$。

Step 4:为了保证所求问题的收敛性,根据文献[15]研究结果,利用式(16)更新拉格朗日系数 η。

$$\eta=\eta+\theta\frac{(J_f-J'_\alpha)}{\left\|\sum_{i=1}^{M}f(k_i^*)-\chi\right\|^2}gra\tag{16}$$

Step 5:重复 Step 2 ~ Step 4,不断更新 J_f, J'_α 和 kf^* 的值。当 J_α 的取值没有明显的增加趋势时,更新 $\theta=\theta/2$[15]。

Step 6:当次梯度的取值小于1时,算法停止,此时 kf^* 就是近似的最优解。

基于上述近似解,继续利用局部搜索法来精确实现试验资源分配最优化。

首先利用如算法 1 所示的伪代码定义每个自变量 k_i 可行解的取值上界 k_{i_up} 和下界 k_{i_low}。然后在 $k_{i\_\text{low}}\times k_{i\_\text{up}}$ 的空间内通过穷尽枚举进一步优化问题可行解。由于本地搜索的实质为在近似最优解附近进行全局搜索,因此可以保证搜索得到的可行解即为所求问题的最优解。

算法 1　局部搜索空间获取方法

Alg. 1　Method for local search space acquisition

```
For l = 1 :1 :M
    ks_l = kf* + e, e is a vector that e_l = 1, and else equal 0, Using ks_l to calculate the value of Σ_{i=1}^{M} f(ks_li)
    If Σ_{i=1}^{M} f(ks_li) > χ
        For j = 1 :1 :M
            If j = l
                ks_lj = ks_lj
            Else While Σ_{i=1}^{M} f(ks_li) > χ and ks_lj ≥ 0
                ks_lj = ks_lj − 1
            End
        End
    End
    Else While Σ_{i=1}^{M} f(ks_li) ≤ χ and ks_ll ≤ k_lmax
        ks_ll = ks_ll + 1
    End
End
End
For l = 1 :1 :M
    k_l^lower = min{ks_l1, ⋯, ks_lM}
    k_l^upper = max{ks_l1, ⋯, ks_lM}
End
```

3 算法有效性验证

为了验证所提方法的有效性,利用表 1 所示的故障模式,针对优化问题一和问题二分别假设了三种不同规模的优化配置问题,每个示例中需要进行试验资源配置的故障模式分别为 F1 ~ F3, F1 ~ F5, F1 ~ F7。其中问题一的试验资源边界条件 $\chi = 500$,问题二的增长目标 $\sigma = 0.08$。表 2 分别列出了两个问题使用不同方法得到的资源优化配置结果以及计算机资源消耗。所用的方法分别为穷尽枚举法、拉格朗日松弛算法以及本文所提的混合方法(LRA + local search)。问题求解优化过程在计算机上利用 Matlab2011a 进行,计算机配置为 3.19 GHz CPU and 4 GB RAM。

对比表 2 中资源分配结果可知,仅利用拉格朗日松弛算法求解得到的优化结果可能只是近似最优解,并不能保证结果的最优性(如表 2 中粗体所示);而利用混合方法得到的最终结果与穷尽枚举结果一致。结合前文对本地搜索算法的阐述,可以得到以下结论:混合方法在求解结果上达到了较高的精度。对比表 2 中的计算时间以及搜索规模可知,随着问题规模的增大,利用枚举法求解最优解的搜索范围和计算耗时也呈几何状增加;而混合方法在拉格朗日近似最优解邻域进行搜索,可以大大减少搜索范围,从而极大地提高计算效率。

表 1 仿真对象故障模式参数表

Tab. 1 Parameters for failure modes in simulations

	λ_i	$\bar{d}_{i0}$	f_i	$Cost_{1i}$	$Cost_{2i}$	$Cost_{3i}$	γ_{1i}	γ_{2i}	k_{i_max}
F1	0.65	0.1	0.2	1.2	0.2	0.7	8	10	5
F2	0.34	0.8	0.5	0.4	0.4	0.8	12	15	5
F3	0.20	0.2	0.2	1.0	0.9	0.4	5	12	5
F4	0.10	0.8	0.4	0.8	0.7	0.6	9	18	5
F5	0.55	0.6	0.2	2.3	0.1	0.9	10	12	5
F6	0.45	0.6	0.1	1.3	0.7	0.4	16	12	5
F7	0.23	0.6	0.2	2.0	0.5	0.1	2	17	5

表 2 问题求解结果及资源消耗

Tab. 2 Optimization results and the calculation cost

问题	示例	方法	分配结果	总成本	目标值	搜索规模	计算耗时/s
问题一	示例一	枚举法	2,2,2	484.171 6	0.039 3	216	0.025 2
		LRA	2,2,2	484.171 6	0.039 3	—	0.002 9
		LRA + local search	2,2,2	484.171 6	0.039 3	48	0.008 3
	示例二	枚举法	2,0,1,0,2	447.762 3	0.082 7	7776	0.641 4
		LRA	**2,0,2,0,1**	**404.572 0**	**0.087 9**	—	0.008 5
		LRA + local search	2,0,1,0,2	447.762 3	0.082 7	192	0.043 3
	示例三	枚举法	2,0,1,0,2,0,2	494.192 8	0.133 3	279 936	30.382 5
		LRA	**2,0,1,0,1,0,2**	**384.381 6**	**0.147 3**	—	0.009 3
		LRA + local search	2,0,1,0,2,0,2	494.192 8	0.133 3	1400	0.151 7
问题二	示例一	枚举法	2,1,1	343.345 4	0.075 1	216	0.017 4
		LRA	**2,1,2**	**409.966 3**	**0.053 6**	—	0.002 8
		LRA + local search	2,1,1	343.345 4	0.075 1	8	0.003 8
	示例二	枚举法	2,0,2,0,2	514.383 1	0.068 8	7776	0.627 3
		LRA	2,0,2,0,2	514.383 1	0.068 8	—	0.007 6
		LRA + local search	2,0,2,0,2	514.383 1	0.068 8	8	0.011 9
	示例三	枚举法	2,0,2,0,1,1,1	742.593 2	0.078 7	279 936	30.509 5
		LRA	**2,0,2,0,1,1,2**	**760.519 9**	**0.072 8**	—	0.003 6
		LRA + local search	2,0,2,0,1,1,1	742.593 2	0.078 7	32	0.011 5

4　结论

测试性增长资源优化配置是测试性增长试验管理的一个重要组成部分。本文首先分别考虑测试性验证试验成本、设计改进软硬件成本以及人工成本这三部分成本,构造了测试性增长试验成本模型;然后基于增长试验目标是否明确和试验资源是否受限制,提出三个资源配置问题,并分别建立相应的数学模型。为快速求解该问题,提出了混合拉格朗日松弛和本地搜索的混合计算方法;该方法首先利用 LRA 得到问题的近似最优解,然后在近似最优解邻域进行全局搜索,从而获取问题的最优解。该方法融合了 LRA 快速求解和枚举法精确求解的优点,在提高求解精度的同时,极大地提高了求解速度。最后通过不同规模的仿真案例验证了所提方法的有效性。结果表明,所提模型和方法可以有效解决采用延缓纠正措施的测试性增长试验过程中遇到的资源分配问题;同时由于所解决的优化问题具有一般整数规划问题的特点,因此该方法可以有效扩展到类似整数规划问题的求解过程中。

参考文献(References)

[1] Zhao C X, Qiu J, Liu G J, et al. A testability growth model and its application[C]//Proceedings of Autotestcon, 2014: 121 – 128.

[2] 李天梅. 装备测试性验证试验优化设计与综合评估方法研究[D]. 长沙: 国防科学技术大学, 2010.
LI Tianmei. Research on optimization design and integrated evaluation of testability verification test for equipments[D]. Changsha: National University of Defense Technology, 2010. (in Chinese)

[3] 杨述明, 邱静, 刘冠军, 等. 面向装备健康管理的可测性指标研究[J]. 国防科技大学学报, 2012, 34(1): 72 – 77.
YANG Shuming, QIU Jing, LIU Guanjun, et al. Research on testability indexes for equipment health management[J]. Journal of National University of Defense Technology, 2012, 34(1): 72 – 77. (in Chinese)

[4] Zaporozhets A. A short proof of optimality of the bottom up algorithm for discrete resource allocation problems[J]. Operations Research Letters, 1997, 21(2): 81 – 85.

[5] Johnson T J R. An algorithm for the resource-constrained project scheduling problem[D]. USA: Massachusetts Institute of Technology, 1967.

[6] Singh G, Ernst A T. Resource constraint scheduling with a fractional shared resource[J]. Operations Research Letters, 2011, 39(5): 363 – 368.

[7] Sheralih H D, Driscollp P J. Evolution and state-of-the-art in integer programming[J]. Journal of Computational and Applied Mathematics, 2000, 124(1/2): 319 – 340.

[8] Hadj-Alouane A B, Bean J C. Genetic algorithm for the multiple choice integer program[J]. Operations Research, 1997, 45(1): 92 – 101.

[9] Larranaga P, Lozano J A. Estimation of distribution algorithms: a new tool for evolutionary computation[M]. USA: Springer US, 2002.

[10] Laskar I E, Parsopoulos K E, Vrahatism M N. Particle swarm optimization for integer programming[C]//Proceedings of Congress on Evolutionary Computation, 2002: 1582 – 1587.

[11] 杜祜康, 赵英凯. 整数规划问题智能求解算法综述[J]. 计算机应用研究, 2010, 27(2): 408 – 412.
DU Hukang, ZHAO Yingkai. Survey on intelligent optimization algorithms for solving integer programming problems[J]. Application Research of Computers, 2010, 27(2): 408 – 412. (in Chinese)

[12] Tillman F A, Hwang C L, Kuo W. Optimization of system reliability[M]. USA: Marcel Dekker, 1980.

[13] Kelley J E, Jr.. The cutting plane method for solving convex programs[J]. Journal of the Society for Industrial & Applied Mathematics, 1960, 8(4): 708 – 712.

[14] Geoffrion A M. Integer programming by implicit enumeration and bala′s method[J]. SIAM Review, 1967, 9(2): 178 – 190.

[15] Fisher M L. The lagrangian relaxation method for solving integer programming problems[J]. Management Science, 2004, 50(12): 1861 – 1871.

[16] Marc P. General local search methods[J]. European Journal of Operational Research, 1996, 92: 493 – 511.