基于多目标协同进化的测试用例优先排序

石宇楠 李 征 龚 沛
(北京化工大学计算机系 北京 100029)

摘 要 测试用例优先排序是一种有效的降低回归测试开销的技术,通过对测试用例按照其重要程度排序后可获得更高的测试效率。针对传统多目标遗传算法在测试用例优化排序中存在的收敛较慢、易陷入局部最优、缺乏对不同测试准则的综合权衡等缺点,提出一种基于竞争模式的多目标协同进化算法。该方法采用平均代码覆盖率以及平均变异杀死率作为多个约束目标的测试准则来进行适应度度量,提高算法的错误检测率;使用个体绝对适应度与相对适应度对个体生存能力进行评价,衡量个体优秀程度,利用竞争性的协同进化思想加快算法收敛速度;通过剔除"老年"个体控制个体生存周期来避免陷入局部最优问题。同时,在影响算法执行效率的因素方面也进行了一系列的实验,结果表明该算法能够加快收敛速度,加强了局部搜索能力,相对于传统的优化算法来说具有更好的搜索效率和更高的错误检测率,从而验证了算法的有效性和可行性,证明了该算法具有一定的现实意义。

关键词 协同进化,测试用例优化排序,多目标
中图法分类号 TP311 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2015.12.028

Multi-objective Coevolutionary Test Case Prioritization

SHI Yu-nan LI Zheng GONG Pei
(Department of Computer Science and Engineering, Beijing University of Chemical Technology, Beijing 100029, China)

Abstract Test case prioritization is an effective method to significantly reduce costs of regression test. According to some certain aim of test purpose, the main idea of test case prioritization is to rearrange the permutation of the test suite in order to execute the test case with higher priority preferentially. Aiming at the defect of single object genetic algorithm in test case prioritization, such as slow convergence speed, easy to be trapped in local optimum and lacking of the trade-off between multiple testing criteria, a new competitive co-evolutionary approach was adopted to resolve these problems. In this new approach, multi metrics of fitness are used, including the absolute fitness which evaluates the survival ability of an individual and the relative fitness which estimates the number of defeated opponents of each individual. The outstanding individuals who defeat more opponents can join the elite set for further evolution. By eliminating "old" individuals, this approach can control the individual's survival time to avoid the local optimum. To improve the efficiency of error detection, we introduced the average percentage of mutation kill rate as a new multi-objective optimization criterion. Comparing to the classical search algorithm, the co-evolutionary algorithm can improve the search efficiency and local search ability, and experiment verified the validity of the new approach.

Keywords Co-evolution, Test case prioritization, Multi-objective

1 引言

测试用例优先排序技术(Test Case Prioritization, TCP)是回归测试用例维护技术的重要组成部分[1],该技术将测试用例的执行顺序进行优化,使能够发现错误或满足更多约束条件的测试用例优先执行,借此提高整个回归测试过程的效率。从本质上来说,回归测试用例优先排序问题是一个优化问题,国内外科研工作者提出了许多经典优化技术,包括线性规划和限界剪枝等[2,3]。但这些技术会由于决策变量或界限分支的增多使得解空间急速增长,导致算法性能急剧下降。为解决该问题,研究人员提出了基于搜索的优化技术(Search Based Optimization Technique, SBOT)。目前常用的SBOT包括遗传算法(Genetic Algorithm, GA)[4]、粒子群算法(Particle Swarm Optimization, PSO)[5]以及模拟退火(Simulated Annealing, SA)[6]、协同进化算法(Co-evolutionary Algorithm, CA)[7]等,这些算法目前在各个研究领域均得到了广泛的使用,并且取得了良好的效果。

随着回归测试技术的不断发展,测试人员在执行测试用例的过程中通常需要权衡多个客观因素,如测试成本与交付时间等,单一优化目标的进化算法已经无法满足实际需求,因

到稿日期:2015-02-11 返修日期:2015-03-28 本文受2014国家自然科学基金(61170082,61073035),教育部新世纪优秀人才计划(NCET-12-0757),留学回国人员科研启动基金(LXJJ201303)资助。

石宇楠(1990-),女,硕士,主要研究领域为软件测试,E-mail:shiyunan1990@126.com;李 征(1974-),男,博士,教授,CCF高级会员,主要研究领域为软件分析及测试;龚 沛(1990-),男,硕士,主要研究领域为变异测试与错误定位。

此更多的研究者将目光放在了多目标进化算法(Multi-Objective Evolutionary Algorithms,MOEAs)上,包括 NPGA-II[8]、NSGA-II[9]等,这些算法通过群体迭代进化得到一系列的最优解的集合,研究人员可以在权衡实际情况后从中选择出一个较合理的排序。在理想情况下,回归测试的优化目标应为"在较短的时间内发现更多的软件错误",对应的度量准则是软件错误检测率(Average Percentage of Fault Detect,APFD)[10]。但在实际应用过程中,该值在执行测试用例前是未知的,无法直接利用 APFD 来度量该优化目标,而通常采用代码覆盖等作为优化目标,因此本文使用了两种度量准则:一种是平均代码覆盖率(Average Percentage of Statement Coverage,APSC),使得某一个测试用例排序"在较短的时间内覆盖更多的语句";另一种为新度量准则——平均变异杀死率(Average Percentage of Mutation Kill,APMK),即"在较短的时间内杀死更多的变异",从而在一定程度上提高发现错误的能力。

基于软件规模和"测试用例池"不断增大的现状,加之回归测试过程中的多个客观因素的制约,本文重点针对 TCP 问题进行了研究,设计了多目标协同进化的测试用例优先排序方法,采用平均代码覆盖率以及平均变异杀死率作为多个约束目标的测试准则来进行适应度度量,利用竞争性的协同进化思想加快算法的收敛速度;通过剔除"老年"个体以控制个体生存周期,避免陷入局部最优问题。同时,在影响算法执行效率的因素方面也进行了一系列的实证研究,实验结果表明,该算法能够加快收敛速度,加强了局部搜索能力,相对于传统的优化算法来说具有更好的搜索效率以及更高的错误检测率,从而验证了算法的有效性和可行性。

2 相关工作

2.1 测试用例优先排序

根据 Elbaum 和 Rothermel 等人的研究[10],TCP 问题的一般性描述如下。

· 给定:测试用例集 T,T 的全排列集 PT,包含 T 中所有可能的测试用例执行顺序,排序目标函数 f,其定义域为 PT,值域为实数,寻找 $T' \in PT$,使得 $\forall T'' \in PT \wedge T'' \neq T'$ 有 $f(T') \geqslant f(T'')$。

由以上定义可知,将一个特定执行次序作为函数 f 的输入,与排序目标相关的值作为输出,一般来说该输出值越大,排序结果越好。测试用例集的早期缺陷检测速率的最大化是一种常用排序目标,但由于在执行测试用例之前无法提前获取该信息,因此在工程应用中还可以使用其他排序目标进行替代,如代码覆盖率、变异指数和系统可靠性等[12]。

解决 TCP 问题常用的一类算法是贪心算法,包括 total 策略以及 additional 策略[10],但这两类策略得出的结论与最优解相比,仍存在一定差距。Li 等人以 additional 策略为基础,提出另外一种贪心算法——2-Optimal 策略[1],即每一次从候选集中选出两个测试用例,可得到与最优解较为近似的解集。还有一些研究人员利用程序切片来解决 TCP 问题,如 Jeffrey 等人利用基于测试用例的切片[11],输出可能会影响程序结果的语句或分支,并在排序时仅考虑影响这些分支的测试用例。目前也有很多研究人员利用不同的软件度量标准来提高 TCP 技术的有效性,如 Rothermel 等人利用测试用例的 FEP(Fault Exposing Potential)值来指导 TCP 问题[10],该值可以估计测试用例的缺陷检测能力,其计算基于 PIE 模型[12]和变异测试分析。利用一定的变异算子生成语句 s 的不同变异体,计算 $s(s \in P)$ 的 ms 值,计算公式为

$$ms(s,t) = killed(s,i)/mutants(s)$$

其中,$mutants(s)$ 返回一组变异算子应用到语句 s 上得到的变异体数,$killed(s,t)$ 返回测试用例 t 在上述生成的变异体中可以检测出的数量。则覆盖 N 个语句的测试用例 FEP 值为 $\sum_{1}^{N} ms(s,t)$,然后将 FEP 值较高的测试用例优先执行,获得更高的错误检测率。

2.2 多目标协同进化算法

协同进化指的是两个或多个种群在进化过程中相互影响、相互适应的共同进化模式,是生物与环境之间的交互,种群在各自进化的同时影响其他种群的进化,同时也会受到来自其他种群的影响,通过这样的交互,形成了一个相互作用的协同进化系统。

协同进化由 Rosin 和 Belew 等人[13]首次提出,利用两个及两个以上适应度互相关联的种群进行进化。Cartlidge 等根据适应度的不同关联方式,提出了两种协同进化算法的分类,即竞争性的协同进化[14]和合作性的协同进化[15]。目前协同进化算法已经被成功应用到计算机科学的各个领域,包括人工神经网络、模式识别、车间流水线作业调度以及工程优化设计等。

一些研究人员随后提出了多目标的协同进化算法(Multi-Objective Co-evolutionary Algorithm)框架,如 1985 年出现的基于向量评估的 VEGA 多目标进化算法[16],Shang 等人提出的多目标合作式协同进化弧路由规划问题[17],以及 Nguyen 等人提出的用基于协同进化的遗传编程解决车间流水线问题等[18]。目前应用较广泛的多目标优化算法为基于非支配排序的多目标进化算法 NSGA-II(Non-dominated Sorting Genetic Algorithm-II)[9],该算法具有较强的全局搜索能力和较快的收敛速度,可以发现分布良好的 Pareto 最优解集。

3 基于竞争的协同进化算法

在竞争性的协同进化中,种群中个体的适应度值是由该个体与其临时竞争对手在竞争中的表现所决定的。在进化过程中,一般存在两个不相同的种群,即 Host 种群(需要评价其适应度的种群,由学习者组成,简称 H 种群)和 Parasite 种群(被用作评价 Host 的适应度,由部分评价者组成,简称 P 种群)。由于 P 种群的适应度随着竞争结果的不同而进化,H 种群的成功意味着 P 种群的失败。当 P 随着进化的进行克服这种失败缺陷后,会对经过选择、交叉、变异后的 H 带来新的挑战,这样的演化过程会带来"军备竞赛"(Arms Race)的结果[14],使新基因型打败旧基因型,新型的 P 种群个体也会带来进一步的挑战,通过更复杂、更强的生存能力迫使 H 种群对这样的变化产生更好的适应能力。随着进化过程的推进,从理论上讲 P 种群会越来越难被打败,而 H 种群打败 P

种群的能力也会越来越强，这样就达到了一个相互学习、相互刺激的促进过程。图 1 给出了竞争性协同进化算法(Competitive Co-evolutionary Genetic Algorithm，CCGA)框架。

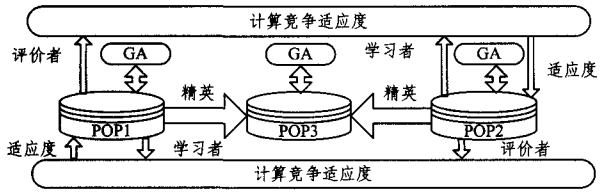


图 1　竞争性的协同进化算法框架

在该框架下，存在着两类个体[19]：

定义 1(评价者，Evaluator)　某轮遗传迭代中从 P 种群中选出的部分个体，是宿主种群临时竞争者，用集合 E 表示。

定义 2(学习者，Learner)　需要计算相对适应度的个体，即处于 H 种群中的个体，用集合 L 表示。

一个学习者能够击败的对手越多，其优秀程度越高，适应度值越高，进入下一轮迭代的可能性越大。通常评价者与学习者处于不同子群体，采用多种群方式实现竞争协同进化。

4　基于协同进化算法的多目标测试用例排序

在利用竞争协同算法进行多目标测试用例排序时，本文使用的多目标协同进化算法(Multi-Objective Competitive Co-evolutionary Genetic Algorithm，MOCCGA)利用 3 类种群进行协同进化，并用两种不同类型的适应度值来评价个体的优秀程度。在保证测试有效性的前提下，同时考虑多个优化准则，通过计算 Pareto 最优解集给出一组可供选择的最优测试用例排序方案。下面是具体技术细节的实现。

4.1　个体设置

测试用例优先排序的任务是对测试用例集中的测试用例顺序进行调整，其解空间应为所有测试用例的全排列的集合。因此在进行测试用例排序时，本文采用序列编码，每个种群的个体都表示一个测试用例的排列序列(即潜在解)，对 N 个测试用例的集合来说，每个个体都是一个长度为 N 的数组，个体上每一个基因位表示一个测试用例编号。

4.2　适应度计算

在进化计算时，选择操作是其基本步骤之一，而选择个体好坏的重要依据就是个体的适应度。本文的适应度值计算主要包括两个部分，一是绝对适应度(Absolute Fitness，AF)，二是相对适应度(Relative Fitness，RF)。在计算前者时，主要考虑的是个体本身的优秀程度，它的值完全由染色体排列决定，即传统遗传算法中的适应度，由算法需要处理的具体问题决定，可以为个体的语句、分支、程序块的覆盖信息或执行开销等；而后者的值由个体能够击败的对手个数及其优秀程度所决定，即竞争性适应度，是个体生存能力的体现，由个体能够打败的评价者的数量和优秀程度来决定，评价者打败的竞争对手越多，该适应度值也越高。

4.2.1　相对适应度(竞争性适应度)

通常计算竞争适应度的方法是将个体在所有竞争回合中的得分直接相加，即简单竞争适应度[20]，如式(1)所示。

$$\forall i\in L, CF_i=\sum_{j\in E, i \text{ defeat } j} 1 \tag{1}$$

在本文使用的“竞争适应度共享”机制中，适应度函数的定义是基于个体之间的(在某些特定准则中)相似度的，将个体的简单适应度与其他个体比较后得到的相似度相除即可。其中，每一个评价者都被学习者当作一个能够击败的独立资源来共享，拥有自己单独的共享适应值，如式(2)：

$$\forall j\in E, N_j=\sum_{k\in L, k \text{ defeat } j} 1 \tag{2}$$

因此学习者的适应度值计算式为：

$$\forall i\in L, CF_i=\sum_{j\in E, i \text{ defeat } j}\frac{1}{N_j} \tag{3}$$

其中，N_j 是种群中能够击败评价者 j 的所有学习者数量。通过这种方式，击败那些较难击败的评价者的学习者可得到奖励，即使这些学习者击败的评价者的数量并不如其他学习者多。当一个学习者在击败那些少有其他学习者能够击败的评价者时，这样的学习者可能具有某些重要的遗传特征，通过奖励这样“不寻常”的具有独特特征的个体来提高整个种群的竞争力[14]。

4.2.2　绝对适应度

在进行测试用例优先排序时，通常将错误检测率作为优化目标，但在执行测试用例之前是无法提前获得该信息的，因此研究人员需要使用其他测试准则进行替代，其中代码覆盖信息是一种较为常用的准则，因为在一般情况下，能够较早达到较高的代码覆盖通常可以较早地发现错误。同时，变异测试也是评估测试用例集充分性的一种有效手段。对被测程序执行符合语法约束的简单代码修改(即变异算子)生成大量变异体，通过杀死更多的变异体来获得更高的检错能力。本文使用了两种覆盖准则作为优化目标，分别是平均代码覆盖率(Average Percentage of Statement Coverage，APSC)和平均变异杀死率(Average Percentage of Mutation Kill，APMK)，其计算公式分别为式(4)和式(5)：

$$APSC=1-\frac{TS_1+TS_2+\cdots+TS_m}{mn}+\frac{1}{2n} \tag{4}$$

$$APMK=1-\frac{TM_1+TM_2+\cdots+TM_{m'}}{m'n}+\frac{1}{2n} \tag{5}$$

其中，m 表示语句个数，m' 表示变异个数，n 表示测试用例个数，TS_i 表示首个可覆盖到第 i 个语句的测试用例在该执行次序中所处的次序，TM_i 表示首个可覆盖到第 i 个变异的测试用例在该执行次序中所处的次序。与 Rothemel 等人提出的 FEP 值不同的是[10]，本文利用 Proteum[14] 提供的所有 uint-level 变异算子生成变异体，每个语句每个算子生成一个变异体，根据测试用例执行是否通过的信息(当测试用例在变异体和原有程序上的执行行为不一致时，则称该测试用例在该变异体上处于 fail 状态，反之为 pass 状态)生成变异覆盖矩阵，以期通过得到较高变异杀死率来提高测试用例检错能力。图 2 给出了测试用例 T_i 在具有 n 个语句的程序上变异覆盖矩阵的基本结构(假设每个语句上生成 5 个变异体)，其中 0 表示该测试用例在该变异体上处于 pass 状态，1 为 fail 状态。

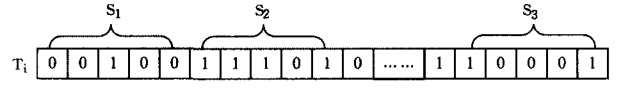


图 2　一个测试用例的变异矩阵结构

4.3 种群设置

在竞争性的协同进化算法框架中存在3类种群，分别是学习者种群、评价者种群以及精英种群。评价者种群(Pop1)，根据其AF值选出部分个体作为评价者，进行遗传操作(选择、交叉、变异)，选择方法为轮盘赌，采用均匀交叉，且设置较高变异率以便在进化过程中产生更多、更新颖的个体特征，较高程度地刺激学习者的进化；学习者种群(Pop2)，在进化过程中根据不同的竞争策略与竞争者计算相对适应度，再结合其RF值进行遗传操作，选择方法为轮盘赌，采用两点交叉，变异率介于评价者种群与精英种群之间；精英种群(Pop3)，每一轮遗传迭代从学习者中选出最优秀的个体(Elite)放入其中，同时也进行遗传操作，选择方法为轮盘赌，采用单点交叉，且具有较小的变异率，以便在进化过程中能够在保留优秀的个体特征的前提下进行局部寻优。

每个种群中的个体的角色在每一轮中必须进行交换，个体的适应度值由个体本身的优良程度以及在与临时竞争对手的竞争中的表现好坏来决定的，一个学习者打败的临时竞争对手越多，其竞争适应度值就越高。这些来自不同种群的个体轮流担任学习者和评价者，轮流给对方施压，刺激对方总体适应度水平的提高。

4.4 学习者与评价者之间的交互模式

在协同进化算法中，个体的适应度由该个体与其他种群中个体之间的相互作用以及自身的优秀程度决定。而种群中个体的适应度的计算由其在每一轮进化迭代中与其竞争种群之间的交互决定。图3给出了一些交互模式。其中，“All VS. Best”计算量虽然小，但易收敛于局部最优解；“All VS. All”模式由于其鲁棒性会产生较优异的解，但却是计算最密集的一个方法，因此并不常用；另一种较为折中的办法是“All VS. Few”模式，其可以在可接受的时间范围内产生较优异的解，本文即采取了这种交互模式。

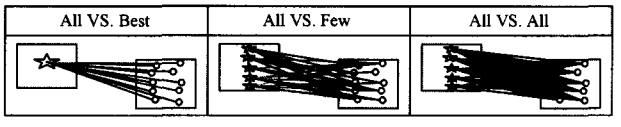


图3 几种不同交互模式

4.5 个体生存周期

在传统遗传算法中，个体生存周期一般与总迭代次数相等。在基于竞争的协同进化算法中，有时会出现由多样性匮乏而导致的过度集中现象。如果竞争对手不具有丰富的多样性，那么提供给对方的进化动力和压力就比较小，因而最后将无法得到具有普遍适应性的进化结果，产生只能打败部分对手的局部最优个体，而不是能够打败全部对手的全局最优个体[21]。解决这一问题的基本思想是保持种群多样性，一个较为典型的方法为上文提到的“竞争适应度共享”，在此基础上，本文提出了限制个体生存周期的思想。

定义3(生存阈值，survival threshold) 总迭代次数的百分比，在该范围内，若个体适应度值没有提高，则杀死该个体。

给每个个体设置年龄属性，如果某个体在连续多代的进化中适应度值并无明显提高，则在该个体年龄超过生存阈值后，以一定的概率“杀死”该贡献小的“老年”个体，重新生成一个新的年轻个体进行替代，克服过度集中现象，并提高种群多样性。

5 实验评测

5.1 被测程序信息

为了验证方法的有效性，本文将两种不同的协同进化算法与NSGA-II在8个被测程序集上进行了比较。这8个被测程序的基本信息如表1所列，集合中的每个文件均包含该程序与其对应测试用例的覆盖信息。

表1 被测程序基本信息

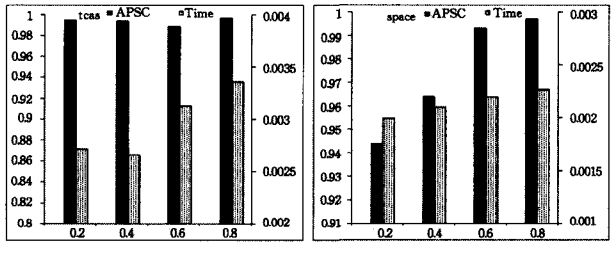
被测程序	语句数	用例池规模	程序版本
Tcas	73	1608	1—41
Totinfo	122	1052	1—23
Schedule	129	2650	1—8
Schedule2	135	2710	无
Printtokens2	198	4115	1—9
Printtokens	209	4130	1—7
Replace	273	5542	无
Space	3827	13550	1—37

5.2 算法效率研究

实验从新算法内部参数和不同算法之间两个方面对算法的有效性进行了研究：对MOCCGA算法来说，实验验证了不同的评价者数量的选取、不同的生存周期、不同迭代次数和种群规模设置对算法性能是否具有一定影响；对算法间有效性比较来说，实验选取了NSGA-II与MOCCGA进行比较，验证在进化相同代数下MOCCGA在多目标的TCP问题上是否能找到更优异的Pareto解集，从而获得更高的APFD值。

5.2.1 评价者占种群比例对算法效率的影响

种群在进行竞争时，选择不同评价者数量对算法进行效率分析，评价者分别占种群总数20%、40%、60%和80%。其中种群中个体数为100，迭代次数为200，生存阈值为50%。图4中左纵坐标轴为APSC值，右纵坐标轴为时间开销，横轴为不同评价者比例数(由于篇幅限制，以space、tcas程序为代表进行分析，下同)。



(a) (b)

图4 不同评价者比例下的APSC值和时间开销

从图4中可以看出，随着评价者数量的不断增长，计算竞争适应度的时间也在增长；在评价者数量达到40%后APSC的增长不明显，经Mann-Whitney秩和检验方法检验后，在评价者个数不同的情况下，APSC的均值具有显著差别，可见不同的评价者选取数量对提高算法效率有一定影响，较多的评价者能够在一定程度上提高算法的鲁棒性。因此在后续计算中本文选择评价者比例为60%的参数，既保证算法有效性，同时时间开销也处于可接受范围之内。

5.2.2 生存阈值对算法效率的影响

设置生存阈值为50%，分别对未包含和包含生存周期限

制的算法进行比较，结果如图 5 所示，图中纵轴为 APSC 值，横轴为不同迭代次数。

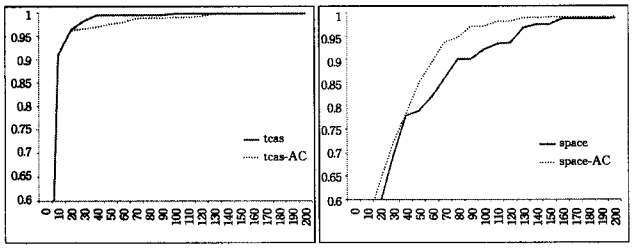
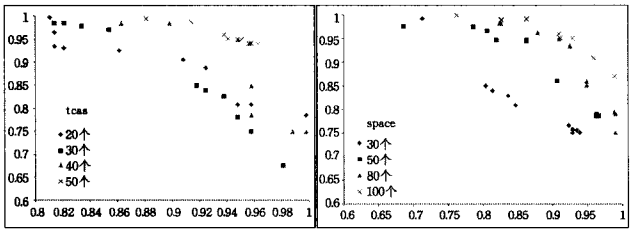


图 5　加入生存周期限制的不同程序收敛情况

从图 5 整体来看，加入生存周期限制后，对规模较大的程序来说(如 space)，可以明显提高种群的迭代速度，更快地收敛到最优解；而对规模较小的程序的收敛速度的提升却不明显，这是因为小规模的程序均能在 60－80 代之间就达到收敛，而生存周期限制的机制要到迭代次数后期才能发挥作用，加之种群的生存过程是随机的，更是在一定程度上削弱了该机制的功能。但对于 space 来说，由于其解空间非常庞大，到迭代后期生存周期限制的机制开始在更多的个体上发挥作用——杀死更多适应度值提高不明显的个体时，会产生更多新的变异个体，加快了新的有利基因的产生。

5.2.3　不同种群数量和迭代数对算法效率的影响(针对多目标)

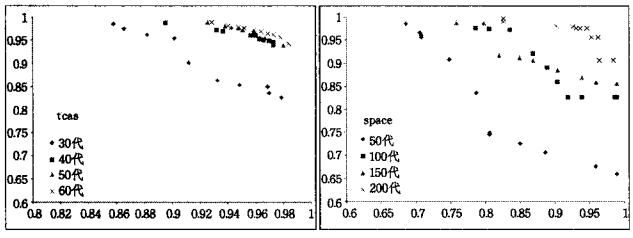
将个体生存阈值设为 50%，评价者数量为 60%，(1)迭代次数相同，种群规模不同的 Pareto 最优解分布情况如图 6 所示；(2)种群规模为 100，不同迭代次数下 Pareto 最优解分布情况如图 7 所示。图中纵坐标轴为 APSC 值，横坐标轴为 APMK 值。



(a)　(b)

图 6　不同种群规模下 Pareto 最优解的分布

种群规模不同的情况下，从图 6 可以看出，对规模较小的程序来说，在种群数量超过 80 之后无显著影响；而对于大规模的程序来说，不同的种群规模对 Pareto 解集具有一定影响，能够在一定程度上对其进行优化。这是由于种群中的个体在生成时采用随机方法，增大了种群规模能够提高个体的多样性，获得分布更加均匀的 Pareto 解集。



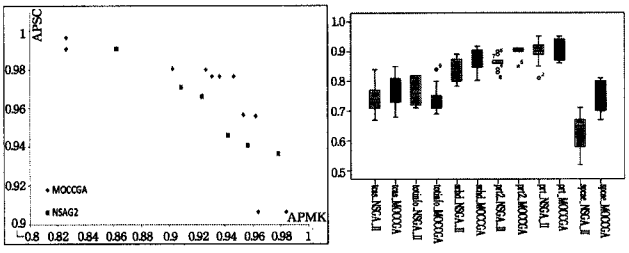
(a)　(b)

图 7　不同迭代次数下 Pareto 最优解的分布

从图 7 可以看出，随着迭代次数的增长，本文得到的非支配解集在不断优化，对于规模较小的程序来说，迭代到 60－80 代后 Pareto 解集会处于一个稳定状态，向前推进的幅度较小；对于规模较大的程序来说，在迭代次数达到 150 代后，Pareto 前沿的推进速度变得缓慢，不再出现明显变化。因此在程序规模一定的情况下，得到稳定的非支配解集所需的迭代次数会稳定在一定的数值上。这是因为在进化前期，由于学习者种群和评价者种群较高的变异率，容易产生新的有利基因，会使得进化过程不断向前推进；而在进化后期，由于大部分个体均收敛到较优的解，新的有利基因产生比较困难，进化也进入尾声，Pareto 前沿的推进速度变慢。

5.2.4　MOCCGA 与 NSGA-II 的比较

为比较 MOCCGA 与 NSGA-II 算法之间的不同，本文计算了两种算法得出的帕累托最优解集和其中每个个体的 APFD 值，并在不同错误版本程序上进行了比较，结果如图 8 所示。



(a)Pareto 解集的分布　(b)APFD 值的分布

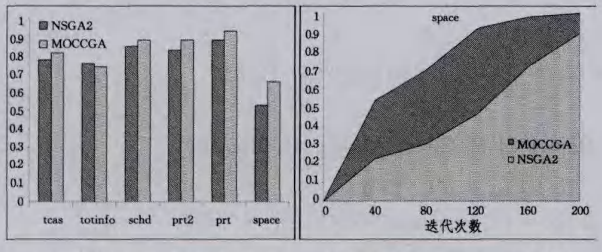
图 8　NSGA-II 和 MOCCGA 算法的比较

图 8(a)为 space 程序使用 NSAG-II 迭代 300 代与 MOCCGA 迭代 200 代分别执行 50 次后的 Pareto 解集的分布，可以看出 MOCCGA 的 Pareto 解集分布范围更广、更均匀，且其中的个体大多都优于 NSGA-II 中的个体，这是因为在多重适应度的刺激选择下，种群的收敛速度大大加快，能够在较少的迭代次数内得到优异的 Pareto 解集。图 8(b)为两种算法得出的 Pareto 解集在不同版本程序上 APFD 值的分布（由于 replace 和 schedule2 程序中错误种类数较多，无法获取变异矩阵信息，因此本文没有给出其 APDF 值信息），选取了两种方法得出的 Pareto 解集中每个个体进行 APDF 值计算，MOCCGA 的总体 APFD 值要优于 NSGA-II，这是因为在 MOCCGA 中加入了竞争机制，且适应度值长期没有提升的个体会被新个体取代，因此发现新的有利基因的能力和短时间内发现最优解的能力较高，能够覆盖更多语句并杀死更多变异体，获得较高的 APFD 值。但对于一定规模的程序来说，MOCCGA 对 APFD 值的提升却不太明显，因为这些程序的测试用例池规模较小，解空间并不特别庞大和复杂，种群之间不需要交换更多的信息，参与更多次竞争也能够在较短的时间内同时收敛到最优，所以 APFD 值的差异不明显。

同时本文也给出了相同迭代次数下的 APFD 值的分布，每隔固定的迭代次数从 Pareto 解集中选取最优个体计算其 APFD 值，对比结果如图 9 所示。

从图 9(a)可看出，对小规模程序来说，不同迭代次数下 APFD 值相差不大；而对于较大规模程序来说，MOCCGA 可以在较快的时间内收敛到最优解。这是因为在相同的实验环境下，MOCCGA 的收敛速度由于竞争机制的使用要高于 NS-

GA-II 的收敛速度，因此在相同的迭代次数下 MOCCGA 可以获得更加优异的 Pareto 解集，APFD 值也会相应增高。从生存周期方面来说，到了进化后期，生存周期的约束加强后，对种群竞争性适应度提高贡献不大的“老年”个体会出现灭亡，新产生的替代个体则有更高的可能性产生更有利的基因，也在一定程度上加快了进化速度。图 9(a)中 MOCCGA 的均值均大于 NSAG-II (totinfo 除外，这是因为对于 totinfo 程序来说，由于其测试用例池规模较小，在随机初始化种群时就很可能已经产生了较优秀的个体，因此进化过程要略快于 MOCCGA)。对于大规模程序如 space 来说，解空间较庞大且可能具有较多局部最优陷阱，NSGA-II 有可能会陷入局部最优，寻优能力也比 MOCCGA 更低，因此较难通过获得更好的 Pareto 解集来达到较高的 APFD 值。



(a)APFD 平均值(迭代次数相同)　(b)不同迭代次数下 space 的 APFD

图 9 不同迭代次数下的 APFD 值

结束语　本文在竞争性的多目标协同进化算法的研究基础上，采用多个适应度度量标准来评价个体优秀程度，加入个体生存周期限制的机制，模拟了 3 个种群在进化上的交互行为，对测试用例优先排序问题在多个优化目标的约束条件下进行研究。实验表明，与采用传统搜索策略的进化过程相比，协同进化可提高搜索全局最优解的有效性和收敛速度，并在一定程度上维持种群多样性；在多目标优化时可以发现分布均匀且较优异的多目标解集，同时获得较高的错误检测率。

参 考 文 献

[1] Li Z, Harman M, Hierons R M. Search algorithms for regression test case prioritization [J]. IEEE Transactions on Software Engineering, 2007, 33(4): 225-237

[2] Zhang L, Hou S S, Guo C. Time-aware test-case prioritization using integer linear programming[C]// Proceedings of the Eighteenth International Symposium on Software Testing and Analysis. ACM, 2009: 213-224

[3] Durillo J J, Zhang Y, Alba E, et al. A study of the bi-objective next release problem [J]. Empirical Software Engineering, 2011, 16(1): 29-60

[4] Srinivas M, Patnaik L M. Genetic algorithms: A survey [J]. Computer, 1994, 27(6): 17-26

[5] AlRashidi M R, El-Hawary M E. A survey of particle swarm optimization applications in electric power systems [J]. IEEE Transactions on Evolutionary Computation, 2009, 13(4): 913-918

[6] Suman B, Kumar P. A survey of simulated annealing as a tool for single and multi-objective optimization [J]. Journal of the Operational Research Society, 2006, 57(10): 1143-1160

[7] Zhou A, Qu B Y, Li H, et al. Multi-objective evolutionary algorithms: A survey of the state of the art [J]. Swarm and Evolutionary Computation, 2011, 1(1): 32-49

[8] Horn J, Nafpliotis N. A niched Pareto genetic algorithm for multi-objective optimization[C]// Proceedings of the First IEEE Conference on Evolutionary Computation, 1994. IEEE World Congress on Computational Intelligence, IEEE, 1994: 82-87

[9] Deb K, Agrawal S, Pratap A, et al. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II [M]// Parallel Problem Solving from Nature PPSN VI. 2000: 849-858

[10] Rothermel G, Untch R H, Chu C. Prioritizing test cases for regression testing[J]. IEEE Transactions on Software Engineering, 2001, 27(10): 929-948

[11] Jeffrey D, Gupta R. Test case prioritization using relevant slices [C]// 30th Annual International. Computer Software and Applications Conference, 2006(COMPSAC'06). IEEE, 2006: 411-420

[12] Delamaro M E, Maldonado J C. Proteum-A Tool for the Assessment of Test Adequacy for C Programs User's guide[C]// Proceedings of the Conference on Performability in Computing Systems. 1996: 79-95

[13] de Jong E D, Stanley K O, Wiegand R P. Introductory tutorial on coevolution[C]// Proceedings of the 2007 GECCO Conference Companion on Genetic and Evolutionary Computation. ACM, 2007: 3133-3157

[14] Rosin C D, Belew R K. New methods for competitive co-evolution [J]. Evolutionary Computation, 1997, 5(1): 1-29

[15] Tan K C, Yang Y J, Goh C K. A distributed cooperative co-evolutionary algorithm for multi-objective optimization [J]. IEEE Transactions on Evolutionary Computation, 2006, 10(5): 527-549

[16] Konak A, Coit D W. Multi-objective optimization using genetic algorithms: A tutorial [J]. Reliability Engineering & System Safety, 2006, 91(9): 992-1007

[17] Shang R, Wang Y, Wang J, et al. A multi-population cooperative co-evolutionary algorithm for multi-objective capacitated arc routing problem[J]. Information Sciences, 2014, 277: 609-642

[18] Nguyen S, Zhang M, Johnston M. Automatic design of scheduling policies for dynamic multi-objective JSS via coevolution genetic programming [J]. IEEE Transactions on Evolutionary Computation, 2014, 18(2): 193

[19] 李碧，林土胜. 基于竞争协同进化的改进遗传算法[J]. 深圳大学学报(理工版)，2009，26(1)：24-29
Li Bi, Lin Tu-sheng. Modified genetic algorithm based on competitive coevolution[J]. Journal of Shenzhen University(Science & Engineering), 2009, 26(1): 24-29

[20] Hillis W D. Co-evolving parasites improve simulated evolution as an optimization procedure [J]. Physica D: Nonlinear Phenomena, 1990, 42(1): 228-234