一种动态约简的多目标测试用例优先级排序方法

张 娜[1] 徐海霞[1] 包晓安[1] 徐 璐[1] 吴 彪[2]
(浙江理工大学信息学院 杭州 310018)[1] (山口大学东亚研究科 山口 753-8514)[2]

摘 要 针对蚁群算法在求解 MOTCP 问题时存在收敛速度慢、易陷入局部最优等缺陷,提出了一种动态约简的在线指导蚁群信息素更新的多目标测试用例优先级排序方法。该方法引入一种动态约简的思想,首先根据各测试用例覆盖需求的情况,对覆盖有相同需求的初始测试用例集进行初次约简。其次,根据测试用例在执行过程中能否检测出错误以及检测出的错误的严重程度来设计一种测试用例失效度的判别方法,在蚁群每一次迭代后均对未检测出错误的测试用例进行二次约简,以减少下一轮迭代时蚁群需要经过的测试用例数,通过两次约简大幅度缩短排序时间。同时,在蚁群的每次迭代过程中,考虑测试用例的重要度、失效度和实际执行时间 3 个因子对下一轮信息素的影响,设计一种同时在 3 个影响因子下在线指导更新蚁群信息素的方法,使蚁群能够更快更准确地寻找到下一个测试用例。最后,将该方法、传统蚁群排序方法和多目标优化排序方法分别应用于多个开源软件程序进行实验比较。仿真实验结果表明,所提动态约简的在线更新信息素的优先级排序方法在缺陷检错能力以及有效执行时间等性能指标方面均有较大优势,能更早发现等级较高的错误。

关键词 优先级排序,蚁群算法,动态约简,测试用例重要度,测试用例失效度,实际执行时间
中图法分类号 TP311 **文献标识码** A **DOI** 10.11896/jsjkx.181102106

Multi-objective Test Case Prioritization Method Combined with Dynamic Reduction

ZHANG Na[1] XU Hai-xia[1] BAO Xiao-an[1] XU Lu[1] WU Biao[2]
(School of Information Science and Technology, Zhejiang Sci-tech University, Hangzhou 310018, China)[1]
(The Graduate School of East Asian Studies, Yamaguchi University, Yamaguchi-shi 753-8514, Japan)[2]

Abstract Aiming at the shortcomings of ant colony algorithm in solving MOTCP problem, such as slow convergence rate and easy to fall into local optimum, a dynamic multi-objective test case prioritization method for online ant colony pheromone updating was proposed. The method introduces a dynamic reduction idea. Firstly, the initial test case set covering the same requirements is firstly reduced according to the coverage of the requirements by each test case. Secondly, according to whether the test case can detect the error and the severity of the detected error during the execution process, a method for judging the failure degree of the test case is designed. After each iteration of the ant colony, a second reduction is made to the test case in which no error is detected, so as to reduce the number of test cases that the ant colony needs to pass in the next iteration, and the sorting time is greatly reduced by two reductions. At the same time, in the process of each iteration of the ant colony, by considering the influence of the test factor importance degree, the failure degree and the actual execution time on the next round of pheromone, an online ant colony is designed to update the ant colony simultaneously under three influence factors. The pheromone method enables ant colonies to find the next test case faster and more accurately. Finally, this method, traditional ant colony sorting method and multi-objective optimization sorting method were respectively applied to multiple open source software programs for experimental comparison. The simulation results show that the prioritization method of the online update pheromone of the proposed dynamic reduction has great advantages in performance indicators such as defect detection capability and effective execution time, and can detect errors with higher severity at an earlier level.

Keywords Priority sorting, Ant colony algorithm, Dynamic reduction, Test case importance, Test case failure degree, Actual execution time

---

到稿日期:2018-11-15 返稿日期:2019-04-01 本文受国家自然科学基金项目(61502430,61562015),广西自然科学重点基金项目(2015GXNSFDA139038),浙江理工大学 521 人才培养计划项目资助。

张 娜(1977—),女,硕士,副教授,主要研究方向为软件工程、软件测试;**徐海霞**(1994—),女,硕士,主要研究方向为软件测试、智能信息处理;**包晓安**(1973—),男,硕士,教授,主要研究方向为自适应软件、软件测试与智能信息处理,E-mail:baoxiaoan@zstu.edu.cn(通信作者);**徐 璐**(1988—),男,博士,讲师,主要研究方向为智能驾驶;**吴 彪**(1989—),男,博士,主要研究方向为软件工程、软件测试。

1　引言

在整个软件开发周期中,软件研发人员需要经常对软件进行升级和维护,软件代码的改变很大程度上会造成新的软件缺陷,因此回归测试变得必不可少[1-2]。为了降低回归测试阶段软件的研发周期和成本,使软件的变更与计算尽可能高效,测试人员常将测试用例优先级按某种方式排序[3],率先执行优先级高的测试用例。多目标测试用例优先级排序技术[4](Multi-Objective Test Cases Prioritization,MOTCP)可以显著提高测试套件的故障检测率,通过不同的测试用例优先级排序技术来达到成本与效益需求间的平衡[5]。这种排序技术能将特定的测试目标最大化。通过重新排列测试用例的执行顺序来最大程度地提高故障检测能力,是降低回归测试成本的一种有效方法[6]。

蚁群算法在 MOTCP 问题的求解上具有很强的鲁棒性,常被应用于数据挖掘聚类分析、模糊建模、群体智能及无线传感网络等问题的研究,在搜索较优解时具有很强的能力[7-9]。近年来,针对基于蚁群算法的 MOTCP 问题的研究已经有了一些成果。顾聪慧等[10]提出了一种多目标测试用例预优化方法,该方法通过蚁群算法的多次迭代,将蚁群访问测试用例的路径作为测试用例优先级排序标准。Stutzle 等[11]通过对信息素设置最大值、最小值、初始值等措施,来避免算法陷入局部最优。Dorigo 等[12]分别采取了信息素的全局和局部更新方式来提升算法的搜索性能。但是上述在信息素更新上提出的优化方法并没有有效解决 MOTCP 中收敛过慢的问题,当搜索到一定程度时,系统会进入停滞状态,由于没有考虑到信息素缺乏时各个测试用例之间的联系、测试用例与需求之间的联系以及它们的实际执行情况,易陷入局部最优,且盲目的随机搜索会导致搜索时间长、收敛速度慢等问题。

本文针对上述蚁群算法在求解 MOTCP 时出现的问题,提出了一种动态约简的多目标测试用例优先级排序算法。该方法有两次约简过程:第一次约简在算法搜索开始前,根据需求覆盖情况对测试用例进行初始约简;第二次约简在每次蚁群迭代过程中,设计测试用例失效度的判别方法对未检测出错误的测试用例进行二次约简。另外,本文综合考虑了测试用例重要度、失效度以及有效执行时间 3 个因素对信息素的影响,在线指导蚁群信息素的更新,提升蚁群算法的求解精度和收敛速度。该方法能够缩短迭代时间并具有较高的缺陷检测效率。

2　面向 MOTCP 的蚁群算法的相关描述

多目标测试用例优先级排序是按照一定的排序规则,使用非支配排序技术对测试用例序列集合进行多目标排序[13-14],引导算法向更优的方向搜索,从而找到最优解。蚁群算法是受自然界蚂蚁集体觅食行为的启发,模拟蚂蚁协作过程而产生的种群智能优化算法。该算法将若干个蚂蚁觅食过程中走过的路径作为多条解路径,并在运动过程中分泌出一种称为信息素的化学物质来引导其他蚂蚁的觅食,使后面的蚂蚁均朝着信息素浓度较高的路径移动。在求解 MOTCP 问题时,测试用例按照蚁群访问测试用例序列的先后顺序进行排序,通过信息素在测试用例优先级排序路径上的不断积累,来指导蚁群在避免过早陷入局部最优的情况下迅速收敛至全局最优[15]。蚁群算法在 MOTCP 问题应用中的 3 个主要步骤如下。

(1)构造解

在蚁群搜索的初始阶段,随机选择一个测试用例作为初始点,从该初始点出发,根据路径上残留的信息素值和多种影响因素的优化在一定概率下选取下一个访问的测试用例。式(3)表示蚂蚁 k 由测试用例 i 继续访问测试用例 j 的概率,d 为优化目标的数目,N^k 是蚂蚁 k 未走过的测试用例集,参数 α 和 β 分别代表控制信息素 τ_{ij} 和启发信息 η_{ij}^k 的启发因子。

$$p_{ij}^k=\begin{cases}\dfrac{\tau_{ij}^{\alpha}*\prod_{d=1}^{2}[\eta_{ij}^{d}]^{\lambda}d^{\beta}}{\sum_{l\in N^k}\tau_{ij}^{\alpha}*\prod_{d=1}^{2}[\eta_{ij}^{d}]^{\lambda}d^{\beta}}, & l\in N^k\\ 0, & \text{其他}\end{cases}\tag{1}$$

(2)更新最优解集

在整个搜索过程中,蚂蚁每完成一次迭代就对已有的最优解集进行更新,直到找到全局最优解,然后将本次迭代后求得的非支配解集与全局最优解进行支配关系比较。若该解能支配全局最优解,则使用该解替换掉全局最优解,否则不更新最优解集。

(3)更新信息素

蚂蚁完成一次迭代后,依次累加该路径序列中两两测试用例之间的信息素值 $\Delta\tau_{ij}^k$,使用 $1-\rho$ 来表示信息素的消逝程度。非支配解路径的信息素值根据式(2)做调整。

$$\tau_{ij}=(1-\rho)*\tau_{ij}+\Delta\tau_{ij}\tag{2}$$

$$\Delta\tau_{ij}=\sum_{k=1}^{h}\Delta\tau_{ij}^k\tag{3}$$

$$\Delta\tau_{ij}^k=\begin{cases}\dfrac{Q}{L}, & \text{当蚂蚁 } k \text{ 得到的测试用例序列为}\\ & \text{非支配解且经过路径 } ij\\ 0, & \text{其他}\end{cases}\tag{4}$$

其中,$0<\rho<1$,$\Delta\tau_{ij}$ 为一次迭代中路径 $i\rightarrow j$ 上信息素增量的总和,h 代表非支配解的个数,$\Delta\tau_{ij}^k$ 为该次迭代中非支配解路径 $i\rightarrow j$ 上的信息素增量,Q 是常数,L 表示测试用例间的距离。

3　测试用例初始约简

3.1　需求覆盖初始约简

本文在对测试用例优先级排序之前,先考虑测试用例之间的相互关系,利用各用例之间的相互联系对待排序的测试用例进行初始约简。事实上,测试用例间的相互关系对应的是对测试需求的覆盖关系[16]。通常一个测试用例可以覆盖多个需求,而覆盖相同需求的测试用例通常会检测出相似甚至相同的错误,因此,本文在对这些测试用例排序之前,先根据测试用例对测试需求的覆盖情况进行简单约简。

为了更直观地描述该约简过程,现对涉及到的概念及相关变量进行如下表述:

记测试用例集 $T=\{T_1,T_2,\cdots,T_n\}$,测试需求集 $R=\{R_1,R_2,\cdots,R_m\}$,其中,n 和 m 分别代表测试用例和测试需求的个数。

(1)若存在测试用例 T_i 覆盖的测试需求集包含于测试用例 T_j 覆盖的测试需求集,则将测试用例 T_i 视为失效用例,将其从待排序测试用例集中删除,如图 1 所示。

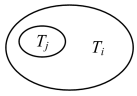

图 1 情景 1 示例

Fig. 1 Example of scenario 1

(2)若存在测试用例 T_n 覆盖的测试需求集包含于多个测试用例集覆盖的测试需求集的并集,则将测试用例 T_n 视为失效用例,并将其从待排序测试用例集中删除,以此类推,如图 2 所示。

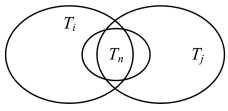

图 2 情景 2 示例

Fig. 2 Example of scenario 2

3.2 失效检测二次约简

在蚁群寻优过程中,直到遍历完全部测试用例时,迭代才会停止。若是在迭代过程中进一步删减待排序的失效测试用例,无疑会缩减蚁群迭代所需的时间。因此,本文提出了一种测试用例失效度的判别方法,在蚁群迭代过程中,判断该测试用例在执行时是否能检测出错误、检测出的错误的严重等级是多少。若发现该测试用例检测出错误,则依据该错误的严重程度加深对信息素的指导;若没有检测出错误,则将该测试用例从待排序测试用例集中删除,以减少下一次迭代蚁群需要经过的测试用例数,缩短算法的寻优时间。具体操作如下:

将每个测试用例的失效度初始值均设为 1,记第 t 次迭代的第 j 个测试用例失效度为 $FC_{c_j|T_t}$,若在当前的迭代过程中,测试用例 c_j 检测出了错误,则根据错误的严重等级将该测试用例的失效度加上相应的严重等级系数 $\Delta\nu$;若测试用例 c_j 未检测出错误,则将该测试用例失效度的值置为 0,并且将失效的测试用例 c_j 从待排序的测试用例集中删除,将此过程删除的测试用例按照测试用例重要度由高到低的顺序排列在蚁群算法得出的序列之后,即最终的排序序列为蚁群算法依次走过的测试用例序列加上二次约简过程中删除的测试用例序列。那么,第 t 次迭代时测试用例 c_j 的失效度判别公式如式(5)所示:

$$FC_{c_j|T_t}=\begin{cases}FC_{c_j|T_{t-1}}+\Delta\nu, & \text{测试用例 } c_j \text{ 检测出错误}\\ 0, & \text{测试用例 } c_j \text{ 未检测出错误}\end{cases} \tag{5}$$

其中,严重等级系数 $\Delta\nu$ 是根据 bug 的严重等级程度而定的,一般分为紧急的、较严重的、一般的和较轻的这 4 个等级,其严重等级系数 $\Delta\nu$ 的值分别对应 1,0.7,0.4,0.1。

4 多目标指导下的蚁群信息素更新策略

蚂蚁之间通过信息素来交流、工作,优质的信息素更新策略可以促进蚂蚁间传递信息,使其更高效地搜索到下一个目标。传统的信息素更新策略是使用一个常量(式(4)中的 Q/L)来提高信息素浓度,没有考虑到测试用例的实际执行情况以及测试用例本身的重要度[17]。本文方法通过引入测试用例失效度、测试用例重要度以及测试用例实际执行时间这 3 个影响因素来在线指导更新信息素,以替代传统的信息素增量。

记测试用例重要度为 I_{c_j},由于每个测试用例覆盖的测试需求不同,而很大程度上测试用例的重要度取决于其所覆盖的测试需求的重要度,因此本文中的测试用例重要度 I_{c_j} 为该用例所覆盖的各个需求的重要度之和与所有需求的总重要度的比值,如式(6)所示:

$$I_{c_j}=\frac{\sum_{r=1}^{p} I_{R_r}}{\sum_{r=1}^{m} I_{R_r}} \tag{6}$$

其中,r 代表第 r 个需求,m 为总的需求数,p 为测试用例 c_j 所覆盖的需求总数。各个测试需求的重要度是由测试人员从需求分析文档中分析而来,一般分为重要、较重要、较不重要和不重要这 4 个等级。与严重等级系数 $\Delta\nu$ 值的划分方法相同,本文将其对应的重要度值划分为 1,0.7,0.4,0.1。

因此,本文中的信息素增量 $\Delta\tau_{ij}^{k}$ 如式(7)所示:

$$\Delta\tau_{ij}^{k}|_{T_t}=\begin{cases}\dfrac{Q*(FC_{C_j|T_t}+I_{c_j})}{Tvector_{c_j}}, & \text{当蚂蚁 } k \text{ 得到的测试用例序列为非支配解且经过路径 } ij,\text{并且测试用例 } c_j \text{ 检测出了错误}\\ 0, & \text{其他}\end{cases} \tag{7}$$

其中,$Tvector_{c_j}$ 表示测试用例 c_j 的实际执行时间,本文使用其倒数形式表示。在蚁群迭代过程中,根据当前迭代的测试用例实际执行情况以及测试用例重要度来在线更新路径节点间的信息素增量。由式(7)可知,当测试用例检测出的错误严重等级越高、测试用例的重要度值越高或者测试用例的有效执行时间越短时,该路径上的信息素增量越大,在下一轮迭代中选择该条路径的可能也就越大,有利于快速得到满足需求的最终路径。

5 仿真实验及结果分析

本文将从两个方面来验证本文所提方法的有效性:1)基于需求的初次约简以及基于测试用例失效度的二次约简方法,是否可以有效缩短整个搜索时间;2)在测试用例的失效度、重要度以及实际执行时间 3 个影响因素下的信息素在线更新策略是否可以准确指导蚁群快速收敛、是否能够提高缺陷检测效率。为了验证本文所提方法在解决 MOTCP 问题时的有效性,本文采用平均缺陷检测率[18](Average Percentage of Failure Detection,APFD)和测试用例序列第一次达到需求全覆盖时执行所需的总时间(Effective Execution Time,EET)作为 MOTCP 问题的两个优化目标,测试用例序列的 $APFD$ 值越大且 EET 值越小,其回归测试的检错效率就越高。$APFD$ 和 EET 的计算公式分别如式(8)、式(9)所示:

$$APFD=1-\frac{TF_1+TF_2+L+TF_m}{nm}+\frac{1}{2n} \tag{8}$$

$$EET=\sum_{i=1}^{M}ET_i \tag{9}$$

其中,m 为检测出的错误数量,n 代表有 n 个测试用例,TF_i 表示检测出第一个错误的测试用例在序列中的位置,M' 表示当达到被测语句全覆盖时需要执行的测试用例数,ET_i 表示测试用例 i 所需的执行时间。

5.1　实验环境设计

本实验选取开源软件程序 Space、MET 以及 Bash 进行实验,这 3 个评测程序经常被用来验证实验有效性,且均可从 SIR 库和 SourceForge 中获得,被测程序的具体相关信息如表 1 所列。本实验采用 VC++语言编写,并在 Visual Studio 2015 中运行,PC 机配置为 Intel®Core™ i5-3210M CPU @ 2.50GHz 2.49GHz,2.00GB RAM,64 位操作系统。

表 1　被测程序的相关信息

Table 1　Relevant information of tested program

被测程序	有效代码行数	函数个数	缺陷个数	50 组测试用例集的平均个数
Space	6173	273	27	948
MET	4953	120	9	83
Bash	19812	556	19	764

本文根据已有的研究方法设置面向 MOTCP 的蚁群算法的相关参数,分别对 3 个被测程序独立重复实验 50 次,蚁群算法中参数组合设置为 $\alpha=1,\beta=5,\rho=0.1,M=32$,其中 M 表示蚂蚁种群大小,将对比实验的最大迭代次数设置为 500。将本文方法与传统蚁群排序方法(传统蚁群排序方法为没有对测试用例集进行两次约简,而且使用传统的蚁群信息素更新策略进行测试用例优先级排序的方法)和多目标优化排序算法(多目标优化排序方法为使用本文方法的信息素在线更新策略,但没有两次约简测试用例)进行实验比较,以对比分析各方法的优劣。

5.2　实验结果对比分析

(1)本文方法与传统蚁群排序方法的信息素更新策略对算法的收敛速度以及迭代时间的对比分析。

本文通过对比不同信息素更新策略下蚁群算法取得的语句覆盖率、EET 值、迭代次数等,来对比两种信息素更新策略下算法的收敛速度和效率。将 50 次实验的平均语句覆盖率、平均 EET、平均迭代次数以及单次迭代的平均运行时间作为对比实验的评价指标。具体的实验数据如表 2 所列。

表 2　两种信息素更新策略下的实验结果对比

Table 2　Comparison of experimental results under two pheromone updating strategies

评测程序	平均语句覆盖率		平均 EET/s		平均迭代次数		单次迭代的平均运行时间/s	
	本文方法	传统方法	本文方法	传统方法	本文方法	传统方法	本文方法	传统方法
Space	0.9931	0.8804	12.9036	91.7489	89.440	46.415	9.4197	11.2365
MET	0.9173	0.9112	7.2103	65.6542	58.414	33.785	5.9844	7.1064
Bash	0.9744	0.8332	19.2491	116.6841	106.142	64.256	4.5757	5.1256

由表 2 可知:1)在 3 个被测程序实验中,本文方法得到的平均语句覆盖率较传统蚁群排序算法均得到了有效提升,其中 Bash 程序实验中本文的平均语句覆盖率提升最多,相比于传统蚁群排序方法提高了 0.1412;2)本文方法下的平均 EET 值相比于传统方法具有更明显的优势,3 组评测程序的平均时间缩短了 78.2405s,这说明本文方法中的两次约简大幅度缩短了蚁群算法迭代所需的时间,信息素在线更新策略也使得蚁群算法对 MOTCP 问题的求解速率得到提升;3)传统的信息素更新策略在程序规模较小的 MET 程序中在 30 多次迭代后达到稳定状态,程序规模较大的 Bash 程序在 64 次左右迭代后达到稳定,而本文方法在接近两倍的迭代次数之后才趋于稳定,传统策略的迭代次数相比于本文方法少。但是,传统蚁群排序方法下的单次迭代平均时间比本文方法长,这说明本文方法在提升算法收敛速度以及缩短整个迭代时间的同时避免了蚁群算法过早地陷入局部最优。

(2)为了更直观地显示本文中 3 个因素指导下的信息素在线更新策略的缺陷检测能力,本文用传统蚁群排序方法、多目标优化排序方法和本文所提方法进行实验对比,测试停止标准均为需求覆盖率达到 100%。

图 3 给出了实验中 3 种排序方法下各评测程序的 $APFD$ 值对比,图 4 为各方法在不同评测程序实验中单位缺陷检测所需要的测试用例数的对比。

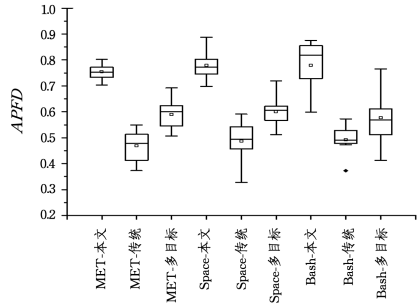

图 3　各方法在不同被测程序中的 $APFD$ 值

Fig. 3　$APFD$ values of each method in different tested programs

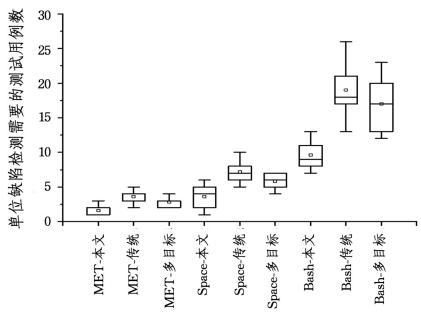

图 4　单位缺陷检测所需的测试用例数

Fig. 4　Number of test cases required for unit defect detection

由图 3 可知,在评测程序 MET 中,本文方法的 $APFD$ 值的最大值比传统蚁群排序方法高 0.254,多目标优先级排序

方法比传统蚁群排序方法高 0.143，本文方法的上下四分位数的值、平均值和中位数均高于多目标优先级排序方法，远高于传统蚁群排序方法；对于评测程序 Space，本文方法的 *APFD* 的平均值比传统蚁群排序方法高 0.316，比多目标优先级排序方法高 0.199；评测程序 Bash 的情况与实验程序 Space 相似，本文方法的各项数值均高于其他两种排序方法，这说明本文方法的改进策略是有效的。由图 4 可知，本文方法下缺陷被检测出时所用的测试用例数在绝大多数情况下都比其他两种排序方法少，在 3 个评测程序中，本文方法所需测试用例数的最大值、平均值、中位数均是 3 种方法中的最低值。

综上可知，本文所提出的动态约简的基于信息素的在线更新策略的多目标优先级排序算法不仅能够缩短时间，而且能够以更少的测试用例检测到更多的、严重等级较高的软件缺陷，具有更高的缺陷检测能力和测试效率。

结束语 本文提出了一种动态约简的基于信息素的在线更新的多目标测试用例优先级排序算法。一方面，该方法根据需求覆盖情况对测试用例进行初始约简，然后又根据测试用例失效度的判别方法对迭代过程中的测试用例进行二次约简，通过两次约简大幅度缩短了蚁群迭代的时间。另一方面，本文综合考虑了测试用例重要度、测试用例失效度和测试用例实际执行时间 3 个因素对信息素的影响，在线指导蚁群信息素的更新，提升了蚁群算法的求解效率以及最终求得的解的缺陷检测能力。该方法的测试效率相较于传统蚁群排序方法和多目标优先排序方法有明显优势。如何更好地利用测试用例之间以及测试用例与需求之间的关系，以及如何更好地优化信息素更新策略将是接下来研究的工作。

参 考 文 献

[1] CHEN X,CHEN J H,YAN X L,et al. A Review of Test Case Prioritization Techniques in Regression Testing[J]. Journal of Software,2013,24(8):1695-1712. (in Chinese)
陈翔，陈继红，鞠小林，等. 回归测试中的测试用例优先排序技术述评[J]. 软件学报，2013，24(8)：1695-1712.

[2] YU H,YANG Y,WANG Y,et al. Priority ordering of regression test cases based on risk analysis[J/OL]. Chinese Journal of Computers,2017:1-19[2019-08-08]. http://kns.cnki.net/kcms/detail/11.1826.TP.20171228.1931.006.html. (in Chinese)
于海，杨月，王莹，等. 基于风险分析的回归测试用例优先级排序[J/OL]. 计算机学报，2017：1-19[2019-08-08]. http://kns.cnki.net/kcms/detail/11.1826.TP.20171228.1931.006.html.

[3] YOO S,HARMAN M. Regression testing minimization,selection and prioritization:a survey[J]. Software Testing,Verification and Reliability,2012,22(2):67-102.

[4] EPITROPAKIS M G,YOO S,HARMAN M,et al. Empirical evaluation of Pareto efficient multi-objective regression test case prioritization[C]// Proceedings of the 2015 International Symposium on Software Testing and Analysis,New York:ACM,2015:234-245.

[5] ZHANG N,LIN Q X,BAO X A,et al. A Prioritization Method for Combined Test Cases Based on OTT Strategy[J]. Computer Engineering and Applications,2018,54(13):41-46,92. (in Chinese)
张娜，林青霞，包晓安，等. 基于 OTT 策略的组合测试用例优先级排序方法[J]. 计算机工程与应用，2018，54(13)：41-46，92.

[6] CHANG L H,MIAO H K,XIAO L. Adaptive Test Case Priority Technology Based on Historical Information[J]. Computer Science,2015,42(9):154-158. (in Chinese)
常龙辉，缪淮扣，肖蕾. 基于历史信息的自适应测试用例优先级技术[J]. 计算机科学，2015，42(9)：154-158.

[7] SINGH Y,KAUR A,SURI B,Test case prioritization using ant colony optimization [J]. ACM SIGSOFT Software Engineering Notes,2010,35(4):1-7.

[8] DORIGO M,STUTZLE T. Ant colony optimization:overview and recent advances[M]// Handbook of metaheuristics. Springer,Cham,2019:311-351.

[9] OLIVAS F,VALDEZ F,CASTILLO O,et al. Ant colony optimization with dynamic parameter adaptation based on interval type-2 fuzzy logic systems[J]. Applied Soft Computing,2017,53:74-87.

[10] GU C H,LI Z,ZHAO R L. ATO-based Test Case Pre-Optimization and Parameter Influence Analysis[J]. Computer Science and Technology,2014,8(12):1463-1473. (in Chinese)
顾聪慧，李征，赵瑞莲. 基于 ACO 的测试用例预优化及参数影响分析[J]. 计算机科学与探索，2014，8(12)：1463-1473.

[11] STUTZLE T,HOOS H. MAX-MIN ant system and local search for the traveling salesman problem [C]// Proceedings of the 1997 IEEE International Conference on Evolutionary Computation. Piscataway,NJ:IEEE,1997:309-314.

[12] DORIGO M,GAMBARDELLA L M. Ant colony system:a cooperative learning approach to the traveling salesman problem [J]. IEEE Transactions on Evolutionary Computation,1997,1(1):53-66.

[13] ZHANG X,YE T,JIN Y. Approximate non-dominated sorting for evolutionary many-objective optimization[J]. Information Sciences,2016,369:14-33.

[14] MASHWANI W K,SALHI A,YENIAY O,et al. Hybrid non-dominated sorting genetic algorithm with adaptive operators selection[J]. Applied Soft Computing,2017,56:1-18.

[15] XING X,SHANG Y,ZHAO R L,et al. A pheromone update strategy for ant colony algorithm for multi-objective test case prioritization[J]. Journal of Computer Applications,2016,36(9):2497-2502. (in Chinese)
邢行，尚颖，赵瑞莲，等. 面向多目标测试用例优先排序的蚁群算法信息素更新策略[J]. 计算机应用，2016，36(9)：2497-2502.

[16] REN H L,ZHANG W,LIANG J A. Optimization of test case set based on ant colony algorithm[J]. Computer Engineering and Applications,2010,46(29):58-62. (in Chinese)
任洪丽，张伟，梁家安. 基于蚁群算法的测试用例集优化方法[J]. 计算机工程与应用，2010，46(29)：58-62.

[17] ZHANG N,YAO L,BAO X A,et al. Multi-objective Optimization Test Case Online Priority Adjustment Strategy[J]. Journal of Software,2015,26(10):2451-2464. (in Chinese)
张娜，姚澜，包晓安，等. 多目标优化的测试用例优先级在线调整策略[J]. 软件学报，2015，26(10)：2451-2464.

[18] CAI K,CHEN T Y,LI Y,et al. Adaptive Testing of Software Components 1[C]// Acm Symposium on Applied Computing. DBLP,2005.