基于蚁群算法的多路径覆盖测试数据生成

廖伟志,夏小云,贾小军
(嘉兴学院数理与信息工程学院,浙江嘉兴 314001)

摘 要: 为了提高多路径覆盖测试数据的生成效率,研究了一种基于蚁群算法的多路径覆盖测试数据生成方法. 首先给出蚁群算法的一种改进方法,该算法以蚂蚁对生成测试数据的重要性作为蚂蚁状态转移和蚂蚁路径变异的依据,以引导更多蚂蚁穿越小概率节点,提高测试数据生成效率. 其次,根据改进的蚁群算法分别提出了基于单信息素表和多信息素表的多路径覆盖测试数据生成方法. 在基于多信息素表的方法中,每条目标路径的信息素表均被用于其它路径测试数据的求解,而且蚁群算法运行一次即可求解多条目标路径的覆盖测试数据. 最后对所提出方法的有效性和复杂度进行了理论分析. 实验结果表明,与其它方法相比,基于多信息素表的测试数据生成方法能够有效地生成多路径覆盖测试数据.

关键词: 测试数据生成; 蚁群算法; 多路径; 路径覆盖; 蚂蚁珍贵度
中图分类号: TP301 **文献标识码:** A **文章编号:** 0372-2112 (2020)07-1330-13
电子学报 URL: http://www.ejournal.org.cn **DOI:** 10.3969/j.issn.0372-2112.2020.07.011

Test Data Generation for Multiple Paths Coverage Based on Ant Colony Algorithm

LIAO Wei-zhi, XIA Xiao-yun, JIA Xiao-jun
(*College of Mathematics Physics and Information Engineering, Jiaxing University, Jiaxing, Zhejiang* 314001, *China*)

Abstract: In order to improve the generation efficiency of multipath coverage test data, a novel method is proposed based on ant colony algorithm (ACO). Firstly, an improved ACO is developed. The importance of an ant to generate test data is considered as a factor for ant state transfer and path mutation. As a result, more ants are guided to traverse small probabilities node and the efficiency of test data generation is improved. Secondly, according to the improved ACO, test data generation of multipath coverage based on single pheromone table and multiple pheromone tables are proposed. In a multiple pheromones table based approach, the pheromone table of each target path is also used to generate test data for other target path, and the test data of multiple paths are generated by running ACO only once. Finally, the effectiveness and complexity of the proposed method are analyzed theoretically. Experimental results show that test data generation based on multi-pheromone tables can effectively generate multipath coverage test data compared with other methods.

Key words: test data generation; ant colony algorithm; multiple paths; path coverage; valuableness of ants

1 引言

软件自动测试是减少软件测试费用的有效途径之一. 自动生成满足指定准则的测试数据是自动测试的关键问题,许多研究人员已经提出了各种测试数据自动生成方法. 至今,基于搜索的软件测试方法已成为自动软件测试领域的一个研究热点. 所谓基于搜索的软件测试方法,实质上就是把软件测试问题转换为优化问题并通过搜索算法生成测试数据的方法. 作为经典的搜索算法,遗传算法(GA)首先于 1992 年被应用于测试数据的自动生成[1]. 之后,包括模拟退火算法(SA)、蚁群算法(ACO)、粒子群优化算法(PSO)、差分进化算法(DE)等搜索算法也被应用于自动软件测试领域[2~5].

路径覆盖测试的目的是生成测试数据,使得程序的每条可行路径至少执行一次. 许多软件测试问题都

收稿日期:2018-09-13;修回日期:2019-04-16;责任编辑:孙瑶
基金项目:国家自然科学基金(No. 61703183, No. 61773410);浙江省公益技术应用研究计划(No. LGG19F030010, No. LGG20F010010);嘉兴市公益性研究计划(No. 2018AY11008)

可以归结为路径覆盖测试数据生成问题. 根据算法运行一次是否能生成多条目标路径的测试数据,生成路径覆盖测试数据的搜索算法可归为两类:一类搜索算法每运行一次只能生成穿越一条路径的测试数据,该方法称为单路径法;另一类搜索算法每运行一次则能生成多条路径的测试数据,该方法称为多路径法. 若通过单路径法来实现多条路径测试数据的生成,则须多次运行搜索算法. 显然,用单路径法来求解多路径覆盖测试数据的效率并不高. 因此,如何设计有效的多路径覆盖测试数据生成算法,使得算法运行一次便能有效地实现多条路径测试数据生成已成为路径覆盖测试方向的一个重要问题.

作为一种新兴启发式算法,蚁群算法是一个增强型学习系统,具有分布式计算、易与其它算法相融合、鲁棒性强等优点,在诸多领域应用均取得比较满意的结果[6]. 目前,蚁群算法已应用于软件测试的各个领域,包括路径覆盖测试[6,7]、结构测试[8~10]、变异测试[11]、统一建模语言(UML)状态图测试[12,13]、面向对象软件一致性测试[14]、基于状态的软件测试[15]、基于分支覆盖准则测试[16,17]等. 与其它搜索算法相比,基于蚁群算法的软件测试,尤其是基于蚁群算法的路径覆盖测试研究成果较少,所得的研究结果也是初步的,如何运用蚁群算法有效地实现软件测试仍需做深入的研究.

本文研究基于蚁群算法的多路径覆盖测试数据生成问题. 首先给出一种评价蚂蚁对生成测试数据重要程度的方法. 然后提出一种改进的蚁群算法,该算法以蚂蚁珍贵度作为蚂蚁状态转移和蚂蚁路径变异的依据,以引导更多蚂蚁穿越小概率节点,提高测试数据生成效率. 同时,在算法中对蚂蚁路径实施变异操作以增加蚂蚁路径搜索的多样性. 基于改进的蚁群算法,分别提出了基于单信息素表和多信息素表的多路径覆盖测试数据生成方法. 理论分析与实验结果均表明,本文方法能够有效地生成多路径覆盖测试数据.

2　相关工作

作为有效的问题求解方法,搜索算法已被广泛应用于路径覆盖测试数据生成领域. Bueno P M S 和 Jino M[18],Watkins A 和 Hufnagel E M[19]先后给出了基于遗传算法的路径覆盖测试数据生成方法. 为了进一步提高算法效率,许多学者对算法提出了各种改进. Yeresime 和 Santanu[20]在对分支函数进行改进基础上设计了新的适应值计算方法,然后给出了利用遗传算法生成路径覆盖测试数据的方法. 谢晓园和徐宝文[21]针对基于遗传算法的路径覆盖测试数据生成问题,提出了三种适应值函数,并通过实验验证了所提出方法的有效性. Mei 和 Wang 等[22]提出了一种称为蜂王演化遗传算法实现了路径覆盖测试数据的生成. 张岩等[23]通过固定被穿越子路径对应的输入分量实现交叉和变异操作的减少,使测试数据生成速度得到有效提高. 张岩和巩敦卫[24]以个体对生成穿越目标路径测试数据的贡献作为权重调整个体适应值,使稀有数据在后续进化中得到保留,从而进一步提高了路径覆盖测试数据的生成效率. 田甜等[25]用遗传算法实现了并行程序路径覆盖测试数据的自动生成. 巩敦卫和任丽娜[26]则研究了利用已有测试数据和遗传算法生成回归测试数据以覆盖目标路径的方法. 夏春艳等[27]提出一种基于节点概率的路径覆盖测试数据进化生成方法. 李爱国等[28]和 Narmada N 等[29]则先后提出了基于粒子群算法的路径测试数据生成方法. 史娇娇等[30]提出一种约简的自适应粒子群优化算法并应用于测试数据的自动生成. 廖伟志[31]在提出一种路径自动分割的基础上利用人工鱼群算法给出了一种路径覆盖测试数据的生成方法. 傅博[6]通过构建被测程序输入空间到蚂蚁路径网络的映射模型把路径覆盖测试数据生成问题转换为蚂蚁路径搜索问题,然后通过蚁群算法实现路径覆盖测试数据的生成,实验结果表明该方法要优于模拟退火遗传算法,是一种适合路径覆盖测试数据生成的启发式算法. Bidgoli A M 等[7]提出一种改进的蚁群算法和粒子群算法并基于这些算法生成覆盖路径的测试数据.

然而,上述文献所提出的算法均是针对单路径覆盖测试数据生成问题,即算法每运行一次只能求解一条目标路径的测试数据,而不能同时得到多条路径的测试数据. 针对此问题,研究人员提出了多路径覆盖测试数据求解方法,以实现算法运行一次就能得到多条路径的测试数据. Ahmed 和 Hermadi[32]提出了一个基于遗传算法的多路径测试数据生成方法,该方法把寻找每条路径的测试数据问题转化一个目标优化问题,进而把多路径覆盖测试数据生成问题转换为多目标优化问题,最后通过遗传算法实现多目标优化问题的求解. 巩敦卫等[33]提出一种自适应分组的大规模路径覆盖测试数据进化生成方法. Gong 等[34]首先根据目标路径的相似性对目标路径进行分组,每组路径测试数据生成问题为一个子优化问题,然后基于遗传算法实现多路径测试数据的生成. Maragathavalli P 等[35]提出了一种利用已被测试程序的测试数据生成多路径覆盖测试数据的方法. Li A G 等[36]在构建新适应值函数的基础上,利用粒子群优化算法(PSO)实现多路径覆盖测试数据生成. Nie P 等[37]通过定义测试用例优化计算资源及测试路径间的信息交换的决策矩阵提出了多路径 PSO 测试数据生成算法. Wang S T 等[38]将所有粒子均自动划分为若干个子粒子群,每个子粒子群用于生成一条目标路径的测试数据. Han 等[39]则提出了改进的多路径 PSO 测试数据生成算法,与其它基于 PSO

的多路径测试数据生成方法不同,该方法通过定义不同的适应值函数用于评价个体最佳位置和全局最佳位置从而使算法的效率得到了进一步提高. Yao 等[40]首先建立了一个多路径覆盖测试数据生成的数学模型,然后提出基于个体共享的多种群遗传算法实现该数学模型的求解. Zhu 等[41]则提出了一种改进的多目标路径覆盖测试数据生成方法,该方法对目标路径的分组策略进行了改进,该策略使得在同一组路径有共同的约束条件从而加快了算法的收敛性,提高了算法的效率. 巩敦卫和张岩[42]将被测程序表示成一棵二叉树,对目标路径采用赫夫曼编码方法表示成二进制,然后用遗传算法生成多条目标路径的测试数据. Zhang 等[43]则结合缩小输入变量空间和遗传算法实现多路径覆盖测试数据生成.

3 蚂蚁珍贵度

研究表明,穿越目标路径节点的数据个数与该节点被穿越的概率成正比,即穿越大概率节点的数据个数多,而穿越小概率节点的数据个数少[24]. 因此,在使用蚁群算法生成测试数据时,穿越小概率节点的蚂蚁对目标路径测试数据的生成更重要,需要引导更多蚂蚁穿越这些小概率节点,从而更有效地实现测试数据的生成. 为此,本文首先给出蚂蚁珍贵度概念,用以评价蚂蚁对生成测试数据的重要程度.

设 n_j为程序 P 的目标路径 P_1上的一个节点($1\leqslant j\leqslant M$,其中 M 为路径 P_1的节点总数),用 C_{ij}表示蚂蚁 ant_i 穿越节点 n_j的程度,若蚂蚁 ant_i对应的测试数据所穿越路径 $P(ant_i)$包含了 n_j,则 C_{ij}按如下公式计算:

$$C_{ij}=\frac{\min(K_j,K_{ij})}{K_j} \tag{1}$$

其中 K_j为节点 n_j在目标路径上出现的次数,K_{ij}为节点 n_j在路径 $P(ant_i)$中出现的次数.

设 I_{ij}为蚂蚁 ant_i对于目标路径 P_1上节点 n_j($1\leqslant j\leqslant M$)的珍贵度,且有

$$I_{ij}=\left(m-\sum_{i=1}^{m}C_{ij}\right)\cdot C_{ij} \tag{2}$$

其中 m 为蚂蚁数目. 根据式(2),如果节点被蚂蚁对应的测试数据穿越得越少,则穿越了该节点的蚂蚁对该节点的珍贵度就更高. 在下一节给出的蚂蚁状态转移和变异操作中,珍贵度越高的蚂蚁走过的路径被选择和变异的概率越大,从而能引导蚂蚁穿越小概率节点,提高测试数据生成效率.

设 $I(ant_i,P_k)$为蚂蚁 ant_i对于目标路径 P_k的珍贵度,且有

$$I(ant_i,P_k)=\sum_{j=1}^{M}I_{ij} \tag{3}$$

其中 M 为路径 P_k的节点数.

4 基于 ACO 的多路径覆盖测试数据生成方法

4.1 蚂蚁路径网络模型[6]

要利用蚁群算法生成覆盖路径的测试数据,则需要把程序的输入变量空间映射成蚂蚁寻找食物的路径网络. 本文采用文献[6]的蚂蚁路径网络模型,其主要思路为:设程序的输入变量为 $x_1,x_2,\cdots,x_n$,其二进制编码字长分别为 $N_1,N_2,\cdots,N_n$,令 $N=\sum_{k=1}^{n}N_k$,则输入变量空间所映射的蚂蚁路径网络图由 N 层节点构成,每层包含两个节点,其中第 1 层节点 b_1,b_{N+1}对应于变量 x_1二进制编码最高位的 0,1 两个状态,第 2 层节点 b_2,b_{N+2}对应于变量 x_1二进制编码次高位的 0,1 两个状态,……,第 N 层节点 b_N,b_{2N}对应于变量 x_n二进制编码最低位的 0,1 两个状态. 网络图有向边集合 E 共有 $4N$ 条边,且 $E=\{(b_i,b_{i+1}),(b_i,b_{N+i+1}),(b_{N+i},b_{N+i+1}),(b_{N+i},b_{i+1}),(b_N,b_1),(b_N,b_{N+1}),(b_{2N},b_1),(b_{2N},b_{N+1})\mid 1\leqslant i\leqslant N-1\}$. 基于上述蚂蚁路径网络图,从第一层到第 N 层节点间的有向边构成一条路径,该路径对应输入变量 $x_1,x_2,\cdots,x_n$的一个二进制编码,也就是对应于 $x_1,x_2,\cdots,x_n$的一个取值. 这样,覆盖程序路径 P_i的测试数据生成问题就转化成了如何在蚂蚁网络路径图中找到一条路径,该路径对应的 $x_1,x_2,\cdots,x_n$数据能穿越程序路径 P_i.

4.2 蚁群算法改进

4.2.1 蚂蚁信息素计算

信息素是蚂蚁选择路径的依据. 相似路径的测试数据也有相似性. 因此若蚂蚁走过的网络路径所对应数据穿越的程序路径与目标路径相似度越大,则应使蚂蚁在该条路径上释放的信息素越多. 为此,用来反映路径相似程度的适应值函数的优劣将直接影响信息素的计算,而信息素计算的合理性则决定着蚂蚁算法的收敛速度及能否找到最优解. 层接近度是路径相似程度适应值函数计算的主要依据之一. 传统层接近度计算方法仅考虑个体所穿越路径与目标路径相同节点个数所占的比例. 为了更准确地描述层接近度,本节给出一种新的层接近度计算方法. 该方法不仅考虑个体穿越路径与目标路径相同节点个数,而且还考虑了这些相同节点所在位置相同的个数.

设程序 P 的目标路径为 P_k,其节点个数用 $|P_k|$ 表示. 设蚂蚁 ant_i走过的网络路径 $b_{i_1}\rightarrow b_{i_2}\rightarrow\cdots\rightarrow b_{i_h}\rightarrow\cdots\rightarrow b_{i_{N-1}}\rightarrow b_{i_N}$对应于数据 y_i,而 y_i穿越的程序路径为 $P(y_i)$. 统计 $P(y_i)$与 P_k相同的节点个数,记为 $\alpha(y_i)$,即 $\alpha(y_i)=|\{n_j\mid n_j$同时为路径 $P(y_i)$和 P_k上的节点$\}|$. 记 $\beta(y_i)$为$P(y_i)$与 P_k相同节点所在位置相同的个数,即

$\beta(y_i)=|\{n_j|\ n_j$为路径$P(y_i)$的第m个节点、n_k为路径P_k的第m个节点且$n_j=n_k\}|$. 将$P(y_i)$与P_k的层接近度用$ap_level(i)$表示,本文提出的层接近度计算方法如式(4).

$$\text{ap\_level}(i)=\frac{\alpha(y_i)}{|P_k|}+\frac{\beta(y_i)}{\alpha(y_i)} \tag{4}$$

设蚂蚁ant_i走过的网络路径$b_{i_1}\to b_{i_2}\to\cdots\to b_{i_k}\to\cdots\to b_{i_{N-1}}\to b_{i_N}$对应于数据$y_i$,则蚂蚁$ant_i$在该网络路径上的各条边所释放的信息素$\Delta\tau(b_{i_{j1}},b_{i_{j2}})$如式(5).

$$\Delta\tau(b_{ij_1},b_{ij_2})=\text{ap\_level}(i) \tag{5}$$

其中$j_1,j_2\in[1,N]$且$j_1<j_2$.

设(b_k,b_{k+1})为蚂蚁网络路径的有向边,$\Delta\tau^{(t-1)}(b_k,b_{k+1})$为蚁群算法第$t-1$次迭代所有蚂蚁在有向边$(b_k,b_{k+1})$增加的信息素,$\tau^{(t-1)}(b_k,b_{k+1})$为第$t-1$次迭代有向边$(b_k,b_{k+1})$的信息素,则第$t$次迭代有向边$(b_k,b_{k+1})$的信息素$\tau^{(t)}(b_k,b_{k+1})$如式(6).

$$\tau^{(t)}(b_k,b_{k+1})=\rho\times\tau^{(t-1)}(b_k,b_{k+1})+\Delta\tau^{(t-1)}(b_k,b_{k+1}) \tag{6}$$

其中ρ为残留系数.

4.2.2　基于珍贵度的状态移动规则

在蚁群算法中,蚂蚁如何选择路径是算法能否有效求解的关键. 文献[6]根据适应度的评价结果,对蚂蚁经过的路径按适应度的大小释放信息素,然后根据信息素大小来决定蚂蚁从当前节点应转移到下一层的哪个节点. 在该规则下,并没有考虑保护那些穿越小概率节点的蚂蚁. 针对此问题,本文在蚂蚁选择移动方向时,不仅依据路径上的信息素大小,还要依据蚂蚁的珍贵度大小,从而能有效引导蚂蚁穿越小概率节点.

设$I(ant_i,P_i)$为蚂蚁ant_i对路径P_i的珍贵度,与ant_i对应的测试数据$\boldsymbol{X}$的蚂蚁路径为$b_{i_1}\to b_{i_2}\to\cdots\to b_{i_k}\to\cdots\to b_{i_{N-1}}\to b_{i_N}$,则该蚂蚁路径上每一条有向边对于程序路径$P_i$的珍贵度大小均为$I(ant_i,P_i)$. 若用$I_i(b_{i_k},b_{i_{k+1}})$表示蚂蚁$ant_i$对有向边$(b_{i_k},b_{i_{k+1}})$的珍贵度,则有

$$I_i(b_{i_k},b_{i_{k+1}})=I(ant_i,P_i) \tag{7}$$

其中$1\leqslant k\leqslant N-1$.

设蚂蚁路径网络图由N层节点构成,若蚂蚁所在的当前节点$b_j'=b_k$,则蚂蚁ant_i在第t次迭代将按如下规则转移到下一个节点b_{j+1}':

(1)若$1\leqslant k\leqslant N-1$则

$$b_{j+1}'=\begin{cases}b_{k+1}, & \tau^{(t)}(b_k,b_{k+1})\geqslant\tau^{(t)}(b_k,b_{k+1+N})\text{且}r\leqslant q_0\\ b_{k+1}, & I_i^{(t)}(b_k,b_{k+1})\geqslant I_i^{(t)}(b_k,b_{k+1+N})\text{且}r\leqslant q_0\\ b_{k+1+N}, & \text{其他}\end{cases} \tag{8}$$

(2)若$1+N\leqslant k\leqslant 2N-1$则

$$b_{j+1}'=\begin{cases}b_{k+1}, & \tau^{(t)}(b_k,b_{k+1})\geqslant\tau^{(t)}(b_k,b_{k+1-N})\text{且}r\leqslant q_0\\ b_{k+1}, & I_i^{(t)}(b_k,b_{k+1})\geqslant I_i^{(t)}(b_k,b_{k+1-N})\text{且}r\leqslant q_0\\ b_{k+1-N}, & \text{其他}\end{cases} \tag{9}$$

(3)若$k=N$或$k=2N$则

$$b_{j+1}'=\begin{cases}b_1, & \tau^{(t)}(b_k,b_1)\geqslant\tau^{(t)}(b_k,b_{1+N})\text{且}r\leqslant q_0\\ b_1, & I_i^{(t)}(b_k,b_1)\geqslant I_i^{(t)}(b_k,b_{1+N})\text{且}r\leqslant q_0\\ b_{1+N}, & \text{其他}\end{cases} \tag{10}$$

其中,$\tau^{(t)}(b_k,b_{k+1})$为有向边(b_k,b_{k+1})在第t次迭代时的信息素大小,按式(6)计算. $I_i^{(t)}(b_k,b_{k+1})$为蚂蚁ant_i在第t次迭代对有向边(b_k,b_{k+1})的珍贵度,其大小按式(7)计算. q_0为算法的一个参数且$0\leqslant q_0\leqslant 1$,$r$为0到1之间的随机数.

4.2.3　蚂蚁路径交叉与变异

为了增加蚂蚁路径搜索的多样性以便于算法跳出局部最优路径,在蚂蚁选择路径后对路径进行交叉和变异操作.

(1)蚂蚁路径交叉

设蚂蚁数目为m,则蚂蚁路径共有m条. 在1到$\left\lfloor\frac{m}{2}\right\rfloor$之间随机生成整数$t$,从$m$条路径中选择$t$对路径.

设交叉概率为p_c,随机生成一个交叉点CPoint,其中$1<\text{CPoint}<N$,不妨设$\text{CPoint}=h$. 接着在$[0,1]$之间产生一个均匀分布的随机数r. 若$r<p_c$,则将这两条路径位于交叉点右半部的路径进行交换,得到两条新路径用以替代原来的路径.

(2)蚂蚁路径变异

设蚂蚁ant_i的路径为$b_{i_1}\to b_{i_2}\to\cdots\to b_{i_k}\to\cdots\to b_{i_{N-1}}\to b_{i_N}$,需变异的节点数目为$c$. 在1到$N$之间随机生成$c$个不同的整数,不妨设为$j_1,j_2,\cdots,j_c$,假设路径的第$j_n(1\leqslant n\leqslant c)$层节点为$b_{i_k}$,第$j_n+1$层节点为$b_{i_{k+1}}$,则按如下方法对$b_{i_{k+1}}$进行变异:

$$b_{i_{k+1}}=\begin{cases}b_{i_{k+1}}', & r<1-\dfrac{I(ant_i,P_i)}{\sum\limits_{k=1}^{m}I(ant_k,P_i)}\\ b_{i_{k+1}}, & \text{否则}\end{cases} \tag{11}$$

其中$b_{i_{k+1}}'$为第j_n+1层$b_{i_{k+1}}$的兄弟节点,r为$[0,1]$的随机数,$1-\dfrac{I(ant_i,P_i)}{\sum\limits_{k=1}^{m}I(ant_k,P_i)}$为变异概率. $\sum\limits_{k=1}^{m}I(ant_k,P_i)$为所有蚂蚁对路径$P_i$珍贵度的总和. 不难看出,变异概率由蚂蚁珍贵度决定,珍贵度大的变异概率小,珍贵度小的则变异概率大.

4.2.4 信息素残留系数

为了提高算法的搜索能力和收敛速度,需要动态调整信息素残留系数. 在算法搜索前期,采用较大的残留系数,使蚂蚁能够在更多的路径中搜索,从而增强了蚂蚁算法的全局搜索能力. 随着循环次数的增加,残留系数应该要逐步减小. 因为在路径搜索后期,残留系数过大会导致收敛速度过慢. 设初始残留系数为 ρ_0,最大循环次数为 $T_{\max}$,第 t 次迭代所采用的残留系数 $\rho(t)$ 按下式计算:

$$\rho(t)=\rho_0\times\left(1-\frac{t-1}{T_{\max}}\right) \tag{12}$$

4.3 基于单信息素表的多路径测试数据生成

设程序 P 的多路径集 $\Omega=\{P_1,P_2,\cdots,P_m\}$ 且程序的输入空间为 $D(\varphi)$,则多路径覆盖测试数据生成问题可描述成如下:在输入空间 $D(\varphi)$ 找到测试集 $\{\boldsymbol{X}_1,\boldsymbol{X}_2,\cdots,\boldsymbol{X}_m\}$,$1\leqslant i\leqslant m$,使得 $P(\boldsymbol{X}_i)=P_i$,其中 $P(\boldsymbol{X}_i)=P_i$ 表示 $\boldsymbol{X}_i$ 穿越了程序 P 的路径 P_i. 为此,可根据蚂蚁信息素表运行蚁群算法逐一找到各条目标路径的测试数据,其求解模型如图 1 所示. 可见,在该模型下要求解 m 条路径的测试数据则蚁群算法需运行 m 次且每次运行蚁群算法需建立一个信息素表. 同时,在求解某条路径的测试数据时只能根据一张信息素表来寻找最优解. 为此,把该方法称为基于单信息素表的多路径覆盖测试数据生成算法,简称 ACO-S 算法.

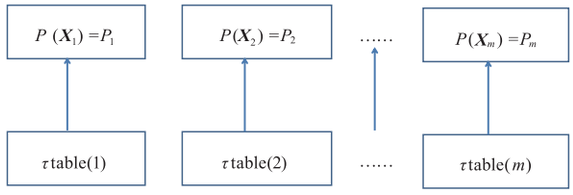

图1 基于单信息素表的多路径测试数据求解

根据图 1 的求解模型,下面给出基于单信息素表求解多路径覆盖测试数据生成算法的主要步骤,具体见算法 1.

算法 1 基于单信息素表的多路径测试数据生成算法(ACO-S 算法)

Step1 构造测试程序 P 输入变量 $x_1,x_2,\cdots,x_n$ 的蚂蚁路径网络.

Step2 输入路径 $P_1,P_2,\cdots,P_m$ 且令 $i=1$.

Step3 初始化蚁群参数和最大迭代次数 $T_{\max}$,放置 a 只蚂蚁到路径网络上的各个节点,给每条有向边设置初始信息素浓度 τ_0 并构造信息素表 $\tau\text{table}(i)$. 每条有向边对路径 P_i 的珍贵度初始值为 0. 迭代计数变量 k 初值为 1.

Step4 各只蚂蚁根据信息素表 $\tau\text{table}(i)$ 和蚂蚁状态移动规则确定蚂蚁所选择的路径,并对路径实施交叉和变异操作.

Step5 对蚂蚁所选择的路径进行解码得到变量 $x_1,x_2,\cdots,x_n$ 的值,记为 $\boldsymbol{X}$ 且 $\boldsymbol{X}=(x_1,x_2,\cdots,x_n)$.

Step6 若 $P(\boldsymbol{X})=P_i$ 即 $\boldsymbol{X}$ 穿越路径 P_i,则保存 $\boldsymbol{X}$ 且转至 Step12,否则转至 Step7.

Step7 根据式(3)计算各只蚂蚁对路径 P_i 的珍贵度并按式(7)修改蚂蚁对每条有向边的珍贵度.

Step8 根据式(5)计算各只蚂蚁在其走过的网络路径上所释放的信息素大小.

Step9 根据式(6)对蚂蚁路径网络上的各条边进行信息素更新.

Step10 残留系数更新:$\rho=\rho_0\times\left(1-\frac{k-1}{T_{\max}}\right)$.

Step11 $k=k+1$. 若 $k<T_{\max}$,则转至 Step4,否则转至 Step12.

Step12 $i=i+1$. 若 $i<m$,则转至 Step3,否则转至 Step13.

Step13 输出各条路径的测试数据.

Step14 算法结束.

算法 1 中 a 大于 0 但不能大于蚂蚁网络路径结点数. 算法的 Step3 到 Step11 是蚁群算法的核心,其功能是通过蚁群算法来求解穿越一条路径的测试数据,其最大的迭代次数是 $T_{\max}$. 因此,在 ACO-S 算法中,要求解多条路径的测试数据则需多次执行 Step3 到 Step11 的操作.

4.4 基于多信息素表的多路径测试数据生成

在基于单信息素表的多路径求解模型中,m 条路径测试数据的问题被分解为优化 m 个问题且每优化一个问题需要运行一次蚁群算法. 通过该方式求解多路径覆盖测试数据存在两个主要问题:(1)不能利用求解某一目标路径测试数据的信息素去求解其它目标路径的测试数据;(2)不能一次运行蚁群算法同时实现多条目标路径测试数据的生成,这势必大大增加蚁群算法的迭代次数.

为了使蚁群算法运行一次就能得到多条目标路径的测试数据,本节提出一种基于多信息素表的多路径测试数据求解方法. 与基于单信息素表求解模型不同,该方法在求解某条路径的测试数据时不仅依据该路径的信息素表,而且还可依据其它路径的信息素表. 该方法在每次迭代过程中随机选择信息素表,所有程序路径可根据所选中信息素表寻找测试数据. 基于多信息表的多路径测试数据求解模型如图 2 所示.

基于如图 2 的模型,下面给出基于多信息素表的多路径测试数据生成算法(简称 ACO-M 算法),具体步骤见算法 2. 该算法的主要思路如下:

对于程序 P 的 m 条目标路径,在 ACO-M 算法中放置共同的 a 只蚂蚁. 每条路径各有一个信息素表,m 条路径则有 m 个信息素表. 每次迭代,蚂蚁选择路径所根据的信息素表从 m 个信息素表中随机选择. 若有目标路径的测试数据已找到,则在后续迭代中不再更新其信息素表,否则将根据蚂蚁所选择的路径对各个信息素表进行更新. 多信息素表的使用和更新流程如图 3 所示.

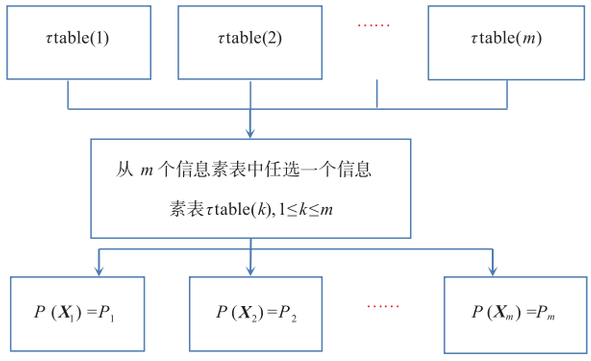

图2　基于多信息素表的多路径测试数据求解

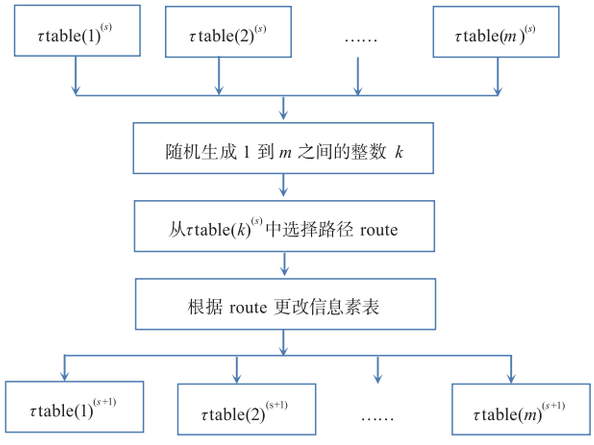

图3　多信息素表操作流程

算法 2　基于多信息素表的多路径测试数据生成算法(ACO-M 算法)

Step1　构造测试程序 P 输入变量 $x_1,x_2,\cdots,x_n$ 的蚂蚁路径网络.

Step2　输入路径集 $\Omega=\{P_1,P_2,\cdots,P_m\}$.

Step3　初始化蚁群参数和最大迭代次数 $T_{\max}$,放置 a 只蚂蚁到路径网络上的各个节点,迭代计数变量 k 初值为 1.

Step4　构造 m 个初始信息素表:τtable(1),τtable(2),…,τtable(m). 各只蚂蚁对每条有向边的珍贵度初始值为 0.

Step5　产生随机数 $j,1\leqslant j\leqslant m$.

Step6　各只蚂蚁根据信息素表 τtable(j) 和蚂蚁状态移动规则确定蚂蚁所选择的路径,并对路径实施交叉和变异操作.

Step7　对蚂蚁所选择的路径进行解码得到变量 $x_1,x_2,\cdots,x_n$ 的值,记为 $\boldsymbol{X}$ 且 $\boldsymbol{X}=(x_1,x_2,\cdots,x_n)$.

Step8　$i=1$.

Step9　若 $P(\boldsymbol{X})=P_i$,则保存 $\boldsymbol{X}$ 且 $\Omega=\Omega-\{P_i\}$.

Step10　$i=i+1$.

Step11　若Ω为空集,则转至 Step18,否则转至 Step12.

Step12　若 $i\leqslant m$,则转至 Step9,否则转至 Step13.

Step13　根据式(3)计算各只蚂蚁对Ω中各条路径的珍贵程度并按式(7)修改蚂蚁对每条有向边的珍贵度.

Step14　根据式(5)计算各只蚂蚁在其走过的网络路径上释放的信息素大小.

Step15　根据式(6)对Ω中的每条路径 P_h,更改信息素表 τtable(h).

Step16　残留系数更新:$\rho=\rho_0\times\left(1-\frac{k-1}{T_{\max}}\right)$.

Step17　$k=k+1$. 若 $k<T_{\max}$,则转至 Step5,否则转至 Step18.

Step18　输出各条路径的测试数据.

Step19　算法结束.

根据所给算法,可知 ACO-M 算法与 ACO-S 算法的主要不同之处包括以下三点.

(1)在 ACO-S 算法中,只能通过一张信息素表来寻找每条路径的测试数据,而且用来寻找某路径测试数据的信息素表不能被寻找其它路径的测试数据所用. 而在 ACO-M 算法中,蚂蚁选择路径时所根据的信息表是从多张信息素表中随机选择,确保了求解某一路径测试数据时均能用到其它路径的信息素表.

(2)在 ACO-M 算法中,所有路径测试数据的查找只需 a 只蚂蚁;而用 ACO-S 算法求解每条路径的测试数据时均在路径网络上放置 a 只蚂蚁,因此寻找 m 条路径测试数据则需 $m\times a$ 只蚂蚁.

(3)在 ACO-M 算法中,只需运行蚁群算法一次就能同时求解多条路径的测试数据. 因此,若目标路径有 m 条且蚁群算法最大迭代次数为 $T_{\max}$,则 ACO-M 算法最大迭代次数为 $T_{\max}$. 而 ACO-S 算法要求解 m 条路径则需运行 m 次蚁群算法,因此最大迭代次数可达 $m\times T_{\max}$.

5　算法性能分析

本节从理论上对 ACO-S 算法和 ACO-M 算法的有效性和复杂度进行分析和比较.

5.1　算法有效性分析

设目标路径为 $P_1,P_2,\cdots,P_m(m>1)$,τtable(i) 为对应于求解目标路径 P_i 测试数据的信息素表,Y_i 为通过 τtable(i) 成功找到 P_i 测试数据时蚁群算法已经迭代的次数. 显然 Y_i 是一个随机变量,不妨用 $P\{Y_i\leqslant s\}$ 来表示在算法迭代次数 s 内找到 P_i 测试数据的概率. 设 $Y_{ij}(i\neq j)$ 通过 τtable(j) 成功找到 P_i 测试数据时蚁群算法已经迭代的次数. 设在 τtable(i) 中找到覆盖路径 P_j 测试数据的概率为 γ_j,则 $P\{Y_{ij}=s\}=(1-\gamma_j)^{s-1}\gamma_j,s=1,2,\cdots,T_{\max}$,其中 $T_{\max}$ 为算法的最大迭代次数.

如果用 ACO-S 算法去求解 $P_1,P_2,\cdots,P_m(m>1)$ 测试数据,设在迭代次数 s 内找到 P_i 测试数据的概率表示为 $P\{Y_i\leqslant s\}$,在迭代次数 s 内找到所有 m 条目标路径测试数据的概率为 $F_1(s)$. 由于找 $P_1,P_2,\cdots,P_m$ 每条目标路径的测试数据均是独立运行一次蚁群算法,因此有

$$F_1(s)=P\{Y_1\leqslant s\}\times P\{Y_2\leqslant s\}\times\cdots\times P\{Y_m\leqslant s\}=\prod_{i=1}^{m}P\{Y_i\leqslant s\}\tag{13}$$

若采用 ACO-M 算法生成多路径覆盖测试数据,m

条路径有 m 张信息素表,因此通过每张信息素表都有可能找到每条路径的测试数据,因此在迭代次数 s 内找到路径 P_i 测试数据的概率为 $P\{\min(Y_{1i},\cdots,Y_{mi})\leqslant s\}$,根据概率知识和文献[40]的计算方法可知

$$\begin{aligned}&P\{\min(Y_{1i},\cdots,Y_{mi})\leqslant s\}\\&=1-P\{\min(Y_{1i},\cdots,Y_{mi})>s\}\\&=1-(1-P\{Y_i\leqslant s\})\times(1-\gamma_i)^{s(m-1)}\end{aligned}\tag{14}$$

所以在迭代次数 s 以内找到所有 m 条目标路径测试数据的概率

$$F_2(s)=\prod_{i=1}^{m}(1-(1-P\{Y_i\leqslant s\})\times(1-\gamma_i)^{s(m-1)})\tag{15}$$

由于 $(1-\gamma_i)^{s(m-1)}<1$,所以有

$$(1-P\{Y_i\leqslant s\})\times(1-\gamma_i)^{s(m-1)}<(1-P\{Y_i\leqslant s\})\tag{16}$$

$$1-(1-P\{Y_i\leqslant s\})\times(1-\gamma_i)^{s(m-1)}>1-(1-P\{Y_i\leqslant s\})\tag{17}$$

于是有

$$F_2(s)>\prod_{i=1}^{m}(1-(1-P\{Y_i\leqslant s\}))\tag{18}$$

所以有

$$F_2(s)>F_1(s)\tag{19}$$

因此,ACO-M 算法找到全部路径测试数据的概率要大于 ACO-S 算法,即 ACO-M 算法比 ACO-S 算法更有效.

5.2 算法复杂度分析

对于 ACO-S 算法,假设最大迭代次数为 $T_{\max}$,设 Y_i 为通过信息素表 τtable(i) 成功找到路径 P_i 测试数据时蚁群算法已经迭代的次数,Y 为找到所有 m 条路径测试数据需要的迭代次数,根据事件的独立性可知如下式子成立.

$$Y=Y_1+Y_2+\cdots+Y_m\tag{20}$$

根据概率知识可知 Y 的期望值如式(21),即

$$E_1(Y)=E(Y_1)+E(Y_2)+\cdots+E(Y_m)\tag{21}$$

容易知道 $Y_1,Y_2,\cdots,Y_m$ 服从同一分布,所以 $E(Y_1)=E(Y_2)=\cdots=E(Y_m)$,于是有

$$E_1(Y)=m\times E(Y_i)\quad i=1,2,\cdots,m\tag{22}$$

设 Y 为 ACO-M 算法找到所有路径测试数据时算法的迭代次数,显然有 $Y=\min_{1\leqslant i,j\leqslant m}\{Y_{ji}\}$. 于是,若 ACO-M 算法的最大迭代次为 $T_{\max}$,则 Y 的期望值

$$\begin{aligned}E_2(Y)&=\sum_{s=1}^{T_{\max}}(s\times P\{Y=s\})\\&=\sum_{s=1}^{T_{\max}}(s\times(P\{Y\leqslant s\}-P\{Y\leqslant s-1\}))\end{aligned}\tag{23}$$

由于 $0\leqslant P\{Y\leqslant s\}\leqslant 1,0\leqslant P\{Y\leqslant s-1\}\leqslant 1$,所以有

$$P\{Y\leqslant s\}-P\{Y\leqslant s-1\}\leqslant 1\tag{24}$$

因此有

$$E_2(Y)\leqslant\sum_{s=1}^{T_{\max}}s\tag{25}$$

所以,当算法最大迭代次数为 $T_{\max}$,ACO-M 算法找到所有路径测试数据的平均迭代次数不超过 $1+2+\cdots+T_{\max}$.

比较 $E_1(Y)$ 和 $E_2(Y)$,不难知道 $E_1(Y)$ 大小与目标路径数目 m 成正比. 目标路径数目越大,则 ACO-S 算法的平均迭代次数也越大. 而 $E_2(Y)$ 的最大值不超过 $1+2+\cdots+T_{\max}$,即 ACO-M 算法的最大平均迭代次数只与算法的最大迭代次数有关而与目标路径数目无关.

6 实验分析

为了说明本文所提出方法的有效性,本节对包括三角形分类程序在内的四个程序进行测试,并与文献[6]的蚁群算法(简称 ACO)、文献[39]基于粒子群的多路径覆盖测试数据生成方法(简称 PSO-M)及文献[40]基于遗传算法的多路径覆盖测试数据生成方法(简称 GA-M)进行实验对比. 实验环境:windowsXP 操作系统、机器主频为 2.5GHz、内存为 3GB. 采用 Python3.2 语言编程.

6.1 三角形分类程序的多路径测试数据生成

三角形分类程序 P 的伪代码见程序 1. 该三角形分类程序共有 40 条路径,其中可行路径 19 条. 该组实验以 19 条可行路径为目标路径,在相关数据范围寻找覆盖所有可行路径的测试数据. 各个方法的适应值函数只考虑层接近度而不考虑分支距离,这将增加成功找到测试数据的难度,从而能更好地体现各种方法的优劣性. 每种方法的最大迭代次数均为 5000. PSO-M 方法主要参数:粒子群个数为 50,加速因子为 2,最大速度和最小速度分别为 8 和–8. GA-M 方法主要参数:种群个数为 200,交叉概率和变异概率分别为 0.95 和 0.8. 为了和文献[6]的 ACO 方法做比较,ACO-S、ACO-M 方法和文献[6]的 ACO 方法主要参数一致,其中蚂蚁个数为 30,初始信息素残留系数为 0.9,交叉概率为 0.9,ACO 变异概率为 0.9. 比较这 5 种方法的平均运行时间、平均成功率、平均迭代次数和成功找到所有路径测试数据的次数. 其中,平均运行时间、平均迭代次数分别为 30 次运行时间及迭代次数的平均值. 每次运行的成功率等于找到测试数据的路径数除以所有可行路径数,平均成功率则为 30 次运行成功率的平均值. 相关实验数据对比见图 4 ~ 图 9.

程序 1 三角形分类伪代码

输入:三角形三条边 a,b,c
输出:分类信息

```
if(a >=b) 交换 a,b 的值
if(a >=c) 交换 a,c 的值
if(b >=c) 交换 b,c 的值
if(a+b< =c) 显示“不是三角形”
else{
  if(a = =b){
    if(b = =c) 显示“等边三角形”
    else 显示“等腰三角形”}
  else{
    if(b = =c) 显示“等腰三角形”
    else 显示“普通三角形”}
}
```

从实验结果可以看出:

(1)从平均成功率来看,本文提出的 ACO-S、ACO-M 方法和文献[6]的 ACO 方法在各种数据范围内的平均成功率均为 100%. GA-M 方法在各种数据范围内的平均成功率都低于 100%. 而 PSO-M 方法在[0,1023],[0,2047],[0,4095],[0,8191],[0,16383],[0,32767]内的平均成功率为 100%,而在更大的数据范围[0,65535]和[0,262143]内,PSO-M 方法的平均成功率分别为 99.8% 和 73.3%. 平均成功率对比见图 4.

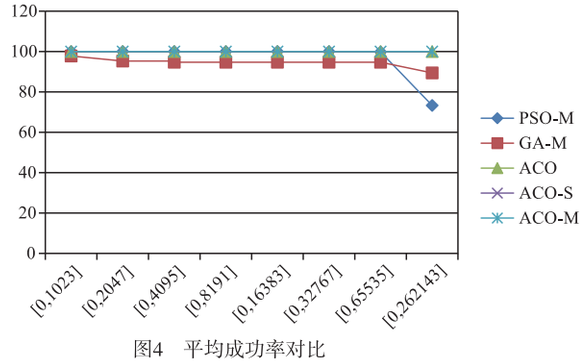

图4　平均成功率对比

(2)从成功找到所有路径测试数据的次数来看,在 30 次实验中,ACO-S、ACO-M 方法和 ACO 方法在各种数据范围内每次均能成功找到全部路径的测试数据. PSO-M 方法在[0,1023]、[0,2047]、[0,4095]、[0,8191]、[0,16383]、[0,32767]、[0,65535] 和 [0,262143]的数据范围成功找到所有路径测试数据的次数分别为 30、30、30、30、30、30、29、0. 当数据范围达到[0,262143]时,PSO-M 方法在每次运行中已经不能成功找到所有路径的测试数据. GA-M 方法在[0,1023]、[0,2047]和[0,4095]数据范围内分别有 19 次、7 次和 1 次成功找到所有路径的测试数据,在其它数据范围内均不能成功找到所有路径的测试数据. 各种方法成功找到所有路径测试数据次数对比见图 5.

(3)从平均迭代次数来看,本文提出的 ACO-S 和 ACO-M 方法的平均迭代次数均低于其它三种方法,其

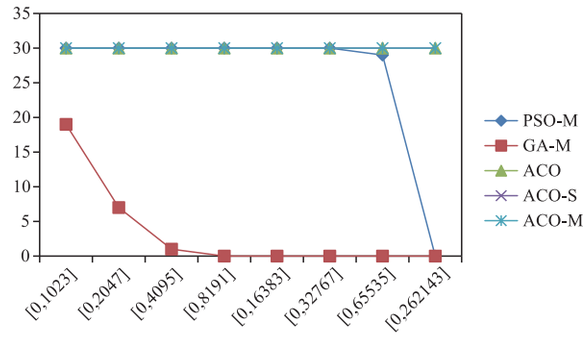

图5　成功找到所有路径测试数据次数对比

中 ACO-M 方法在各种数据范围内的平均迭代次数均明显少于其它方法的平均迭代次数. 5 种方法的平均迭代次数对比见图 6.

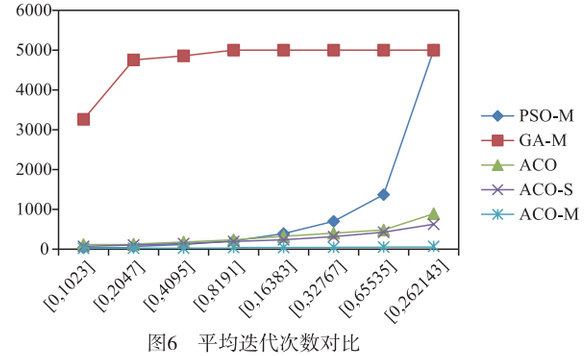

图6　平均迭代次数对比

(4)各种方法的平均运行时间对比见图 7. 结果显示,本文提出的 ACO-S 和 ACO-M 方法的平均运行时间也低于其它三种方法. 其中平均时间最低的是基于多信息素表的 ACO-M 方法,该方法在各个数据范围内的平均运行时间仅分别为 ACO-S 平均运行时间的 48.5%、35.0%、28.2%、29.7%、22.6%、18.7%、17.3% 和 15.3%,为 ACO 方法的 33.1%、29.2%、23.4%、22.9%、14.4%、15.2%、15.3% 和 11.3%,为 PSO-M 方法的 47.5%、22.6%、15.5%、23%、10.2%、8.9%、5.9% 和 4.1%. 而仅为 GA-M 方法的 1.0%、0.48%、0.54%、0.69%、0.42%、0.31%、0.32%、

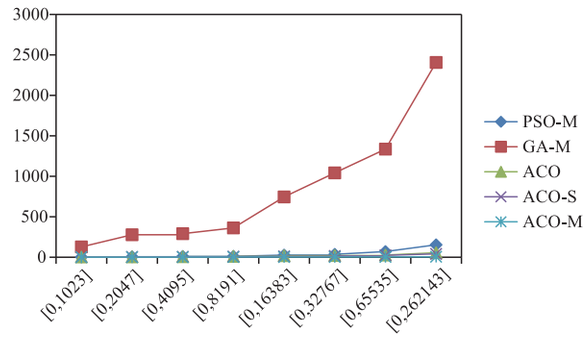

图7　平均运行时间对比

0.28%.随着数据范围的扩大,ACO-M 方法的平均运行时间明显少于其它方法.

(5)在大数据范围[0,524287]、[0,1048575]、[0,2097151]、[0,4194303]内,GA-M、PSO-M、ACO、ACO-M 方法的平均成功效率对比见图 8. PSO-M 在各个范围的平均成功率不超过 40%,GA-M 方法则分别为 84.2%、78.9%、73.7%、68.4%,在大数据范围内 PSO-M 和 GA-M 方法都难于生存穿越等腰、等边三角形路径的测试数据. ACO 和 ACO-M 方法的平均成功率都是 100%,但 ACO-M 所用的平均时间明显比 ACO 的少,两者时间对比见图 9.

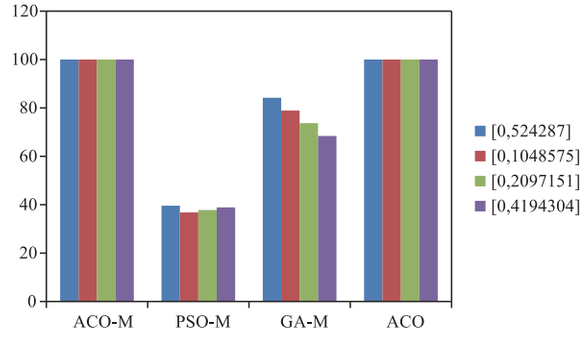

图8 大范围数据成功率对比

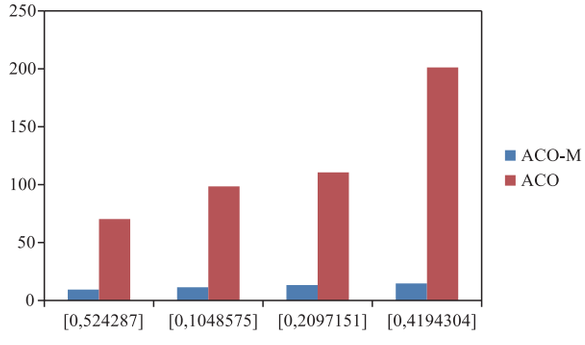

图9 AOC-M和ACO平均时间对比

总体来看,对于三角形分类程序,本文提出的 ACO-M 方法寻找多路径覆盖测试数据的效率明显优于其它方法,数据范围越大其优势越明显.

6.2 其它程序的多路径测试数据生成

为了进一步说明本文 ACO-M 方法的有效性,下面对另外三个经典程序的部分目标路径进行实验,并和 PSO-M、GA-M 和 ACO 方法进行比较. 各方法实验参数:所有方法的最大迭代次数均为 10000;GA-M 算法的种群为 200,交叉概率 0.9,变异概率 0.3;PSO-M 算法粒子群个数为 50,加速因子为 2,最大速度和最小速度分别为 8 和–8;蚁群算法蚂蚁数为 30,传统 ACO 方法交叉概率和变异概率采用文献[6]的参数,均为 0.9;ACO-M 方法交叉概率为 0.9.

6.2.1 最大公约数程序的测试数据生成

求解最大公约数 gcd 程序[40]使用辗展相除法,所考察目标路径分别为循环体执行 5 次到 15 次的 11 条路径,寻找在[0,4095]范围内穿越这些路径的测试数据.

几个方法的平均成功率和平均时间对比见图 10. 结果显示 ACO-M 方法成功率为 100%,而平均运行时间最少.

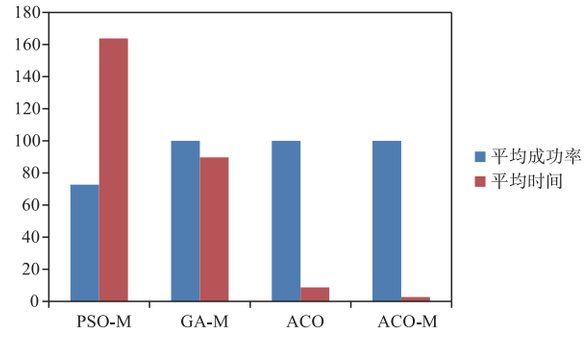

图10 gcd程序实验数据对比

6.2.2 日期统计程序的测试数据生成

日期统计 sumday 程序[26]用来求解某年某月某日是该年的第几天. 考察年份为闰年且循环体执行 1 次到循环体执行 11 次的 11 条路,比较 PSO-M、GA-M、ACO 和 ACO-M 方法在[0,1023]、[0,2047]、[0,4095]范围内生成穿越这些路径测试数据的效率. 所比较的 4 种方法在 3 个数据范围内的成功率都为 100%. 其中,PSO-M 方法用时最少,第二少为本文的 ACO-M 方法,用时最多为传统的 ACO 方法,平均时间对比如图 11. 在 sumday 程序这组实验,除了 PSO-M、ACO 效率比其它方法都好.

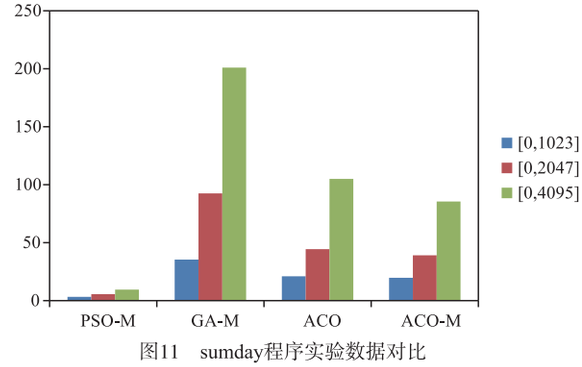

图11 sumday程序实验数据对比

6.2.3 数制转换程序的测试数据生成

数制转换程序 dtox 的伪代码见程序 2,是一个有循环嵌套的程序,其功能是把整数 a 转换成 2 到 b 进制数并保存,最后输出 c 进制的数,b 的最大值为 9. 本组实验选出 11 条可行路径为目标路径,在[0,255]、[0,511]内生成穿越这些路径的测试数据. 生成该程序目

标路径测试数据的难度比其它 3 个程序要大.

程序 2　数制转换伪代码

```
输入: 整数 a,b,c
    输出:输出 a 的 c 进制数或错误信息
    若 b<2 或 b>9 则输出错误,程序结束
    d=1
    while(d<=b)
      e=a
      while(e!=0)
        求 a 的 d 进制数并保存
      d=d+1
    若 c>=2 且 c<=b 则输出 a 的 c 进制数
```

实验结果对比见图 12 和图 13. 结果显示,在 4 种方法中,PSO-M 的成功率最低,在[0,255]、[0,511]内成功率仅分别为 52.7% 和 50.9%. GA-M 为 98.2% 和 76.4%,ACO 为 94.5% 和 90.9%. ACO-M 方法成功率为 100% 和 94.5%. 结合各种方法的平均时间,本文方法 ACO-M 方法效率要优于其它方法.

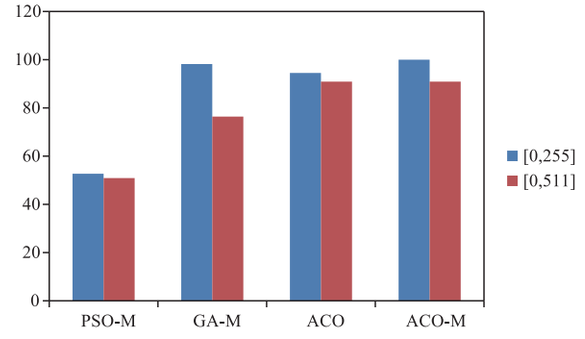

图12　dtox程序实验成功率对比

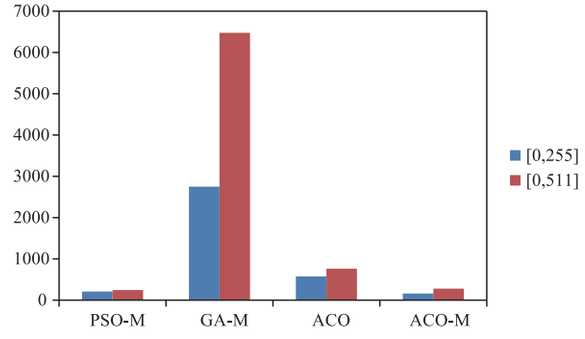

图13　dtox程序实验时间对比

6.3　ACO-M 算法参数分析

最后来考察本文提出的 ACO-M 算法参数选取对算法运行性能的影响. 以三角形分类程序为例,分别考察蚂蚁数目选取和初始残留系数选择对算法运行性能的影响.

(1)当数据范围为[0,8191],根据蚂蚁网络路径的构建方法可知,投放的蚂蚁数目最多只能为 78. 在初始信息素残留系数为 0.9,交叉概率为 0.9 情况下,蚂蚁数分别为 10、20、30、40、50、60、70 情况下 ACO-M 运行 30 次找到所有路径测试数据的平均时间分别为 4.12s、3.25s、2.53s、2.01s、1.79s、1.63s、1.94s、1.95s,对比结果见图 14. 数据范围为[0,65535]的对比结果见图 15. 实验数据表明,当蚂蚁数目太小时寻找测试数据的平均时间值明显要高于蚂蚁数目大的情况,但当蚂蚁数目达到一定数量时平均时间的差距则并不明显.

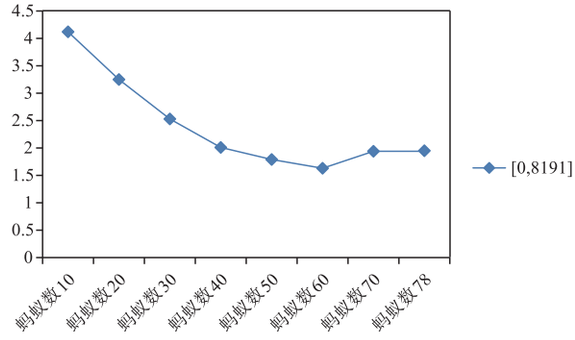

图14　[0, 8191]不同蚂蚁数平均运行时间对比

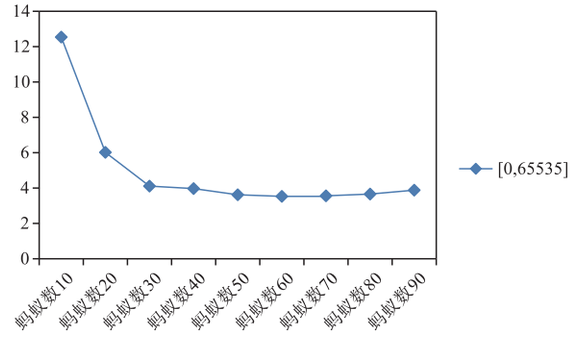

图15　[0, 65535]不同蚂蚁数平均运行时间对比

(2)在蚂蚁数为 60,交叉概率 0.9,而残留系数为 0.2、0.6、0.7、0.8、0.9 和 0.95 情况下,在数据范围[0,2047]、[0,8191]、[0,16383]、[0,32767]、[0,65535]内测试数据的平均时间对比见图 16. 结果显示,当残留系

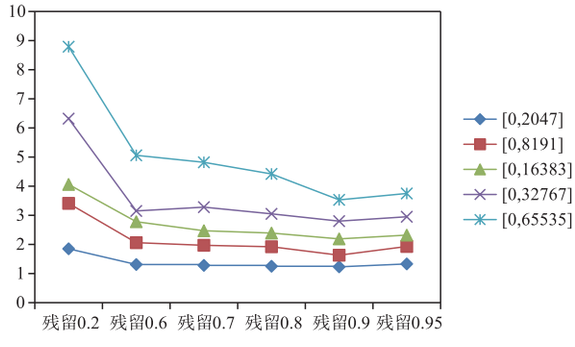

图16　不同残留系数平均运行时间对比

数为0.9和0.95时,寻找测试数据的平均时间比其它残留系数的少.而当残留系数为0.2时,寻找测试数据的平均时间则明显比其它情况多.

7 结论

本文研究一种新的基于蚁群算法的多路径覆盖测试数据生成方法.为此,首先提出了一种评价蚂蚁珍贵度的方法,进而给出基于珍贵度的蚂蚁路径选择和变异操作以引导更多蚂蚁穿越小概率节点.基于改进的蚁群算法,本文提出了基于单信息素表的多路径测试数据生成方法(ACO-S方法)和基于多信息素表的多路径测试数据生成方法(ACO-M方法).与其它基于蚁群算法的测试数据生成方法相比,ACO-M方法有两个显著的特点:(1)每条目标路径的信息素表均可作为求解其它路径测试数据的依据;(2)一次运行蚁群算法可同时求解多条目标路径的覆盖测试数据.理论分析与实验结果表明,本文提出的ACO-M方法是一种有效的多路径覆盖测试数据生成方法.

目前,基于蚁群算法的软件测试研究还处于起步阶段,仍有许多问题值得研究.下一步将探讨如何利用蚂蚁在搜索多路径测试数据过程中所反映出的综合信息来改进蚁群算法从而进一步提高多路径测试数据的生成效率.另外,本文在ACO-M算法中信息素表的选择是随机的,是否存在其它选择方式使得算法效率更高也是一个需要进一步探讨的问题.

参考文献

[1] XANTHAKIS S, ELLIS C, SKOURLAS C, et al. Application of genetic algorithms to software testing[A]. Proceedings of the Fifth International Conference on Software Engineering and its Applications[C]. Toulouse, France: ACM, 1992. 625 – 636.

[2] LATIU G I, CRET O A, VACARIU L. Automatic test data generation for software path testing using evolutionary algorithms[A]. Proceedings of the Third International Conference on Emerging Intelligent Data and Web Technologies [C]. Bucharest, Romania: IEEE, 2012. 1 – 8.

[3] SURI B, SINGHAL S. Literature survey of ant colony optimization in software testing[A]. Proceedings of CSI Sixth International Conference on Software Engineering [C]. Washington DC, USA: IEE, 2012. 1 – 7.

[4] HUANG Han, YANG Zhongming, HAO Zhifeng. Automated test case generation based on differential evolution with relationship matrix for iFogSim toolkit[J]. IEEE Transactions on Industrial Informatics, 2018, 14(11): 5005 – 5016.

[5] GOPI P, RAMALINGAM M, MARUTHAPERUMAL A K, et al. Weighted particle swarm optimization algorithm for test data generation[A]. International Conference on Computing, Analytics and Security Trends[C]. Pune, India: IEEE, 2016. 35 – 39.

[6] 傅博. 基于蚁群算法的软件测试数据自动生成[J]. 计算机工程与应用, 2007, 43(12): 97 – 99.
FU Bo. Automated software test data generation based on colony algorithm[J]. Computer Engineering and Application, 2007, 43(12): 97 – 99. (in Chinese)

[7] BIDGOLI A M, HAGHIGHI H, NASAB H Z, SABOURI H. Using swarm intelligence to generate test data for covering prime paths[A]. International Federation for Information Processing [C]. Rome, Italy: Springer-Verlag, 2017. 132 – 147.

[8] LI K, ZHANG Z, LIU W. Automatic test data generation based on ant colony optimization[A]. Proceedings of the Fifth International Conference on Natural Computation [C]. Washington DC, USA: IEE, 2009. 216 – 220.

[9] SRIVASTAVA P R, RAI V K. An ant colony optimization approach to test sequence generation for control flow based software testing[J]. Communications in Computer & Information Science, 2009, 31(12): 345 – 346.

[10] SHARMA B, GIRDHAR I, et al. Software coverage: a testing approach through ant colony optimization[A]. Proceedings of the Second International Conference on Swarm [C]. Andhra Pradesh, India: Springer-Verlag, 2011. 618 – 625.

[11] AYARI K, BOUKTIF S, ANTONIOL G. Automatic mutation test input data via ant colony[A]. Proceedings of the Ninth Annual Conference on Genetic and Evolutionary Computation[C]. London, UK: ACM, 2007. 1074 – 1081.

[12] LI Huizhong, LAM C Peng. Software test data generation using ant colony algorithm [J]. International Journal of Computer, Control, Quantum and Information Engineering, 2007, 1(1): 126 – 129.

[13] SRIVASTAVA P R, BABY Km. Automated software testing using metahurestic technique based on an ant colony optimization[A]. International Symposium on Electronic System Design[C]. Bhubaneswar, India: IEEE, 2010. 235 – 240.

[14] BOUCHACHIA A, MITTERMEIR R, et al. Nature-inspire techniques for conformance testing of object-oriented software[J]. Applied Soft Computing, 2010, 10(3): 730 – 745.

[15] BAUERSFELD S, WAPPLER S. An approach to automatic input sequence generation for GUI testing ant colony optimization[A]. Proceedings of the 13 Annual Conference on Genetic and Evolutionary Computation[C]. Dublin, Ireland: ACM, 2011. 251 – 252.

[16] MAO Chengying, XIAO Lichuan. Adapting ant colony op-

timization to generate test data for software structural testing[J]. Swarm and Evolutionary Computation, 2015, 20(2):23 – 26.

[17] SHARIFIPOUR H, SHAKERI M, HAGHIGHI H. Structural test data generation using a memetic ant colony optimization based on evolution strategies[J]. Swarm and Evolution Computation, 2018, 40:76 – 91.

[18] BUENO P M S, JINO M. Automatic test data generation for program paths using genetic algorithms[J]. Software Engineering and Knowledge, 2002, 12(6):691 – 709.

[19] WATKINS A, HUFNAGEL E M. Evolutionary test data generation: A comparison of fitness functions[J]. Software Particle and Experience, 2006, 36(1):95 – 116.

[20] YERESIME S, SANTANU K R. A genetic algorithm based approach for test data generation in basis path testing[J]. The International Journal of Soft Computing and Software Engineering, 2013, 3(3):326 – 332.

[21] 谢晓园,徐宝文,史亮,聂长海. 面向路径覆盖的演化测试用例生成技术[J]. 软件学报, 2009, 20(12):3117 – 3136.
XIE Xiao-yuan, XU Bao-wen, SHI Liang, NIE Chang-hai. Genetic test case generation for path-oriented testing[J]. Journal of Software, 2009, 20(12):3117 – 3136. (in Chinese)

[22] MEI Jia, WANG Sheng-yuan. An improved genetic algorithm for test cases generation oriented paths[J]. Chinese Journal of Electronics, 2014, 23(3):494 – 498.

[23] 张岩,巩敦卫. 基于搜索空间自动缩减的路径覆盖测试数据进化生成[J]. 电子学报, 2012, 40(5):1011 – 1016.
ZHANG Yan, GONG Dun-wei. Evolutionary generation of test data for path coverage based on automatic reduction of search space[J]. Acta Electronica Sinica, 2012, 40(5):1011 – 1016. (in Chinese)

[24] 张岩,巩敦卫. 基于稀有数据扑捉的路径覆盖测试数据进化生成方法[J]. 计算机学报, 2013, 36(12):2212 – 2223.
ZHANG Yan, GONG Dun-wei. Evolutionary generation of test data for paths coverage based on scarce data capturing[J]. Chinese Journal of Computers, 2013, 36(12):2212 – 2223. (in Chinese)

[25] 田甜,巩敦卫. 消息传递并行程序路径覆盖测试数据生成问题的模型及其进化求解方法[J]. 计算机学报, 2013, 36(11):2212 – 2223.
TIAN Tian, GONG Dun-wei. Model of test data generation for path coverage of message-passing parallel programs and its evolution-based solution[J]. Chinese Journal of Computers, 2013, 36(11):2212 – 2223. (in Chinses)

[26] 巩敦卫,任丽娜. 回归测试数据进化生成[J]. 计算机学报, 2014, 37(3):489 – 499.
GONG Dun-wei, REN Li-na. Evolutionary generation of regression test data[J]. Chinese Journal of Computers, 2014, 37(3):489 – 499. (in Chinese)

[27] 夏春艳,张岩,宋丽. 基于节点概率的路径覆盖测试数据进化生成[J]. 软件学报, 2016, 27(4):802 – 815.
XIA Chun-yan, ZHANG Yan, SONG Li. Evolutionary generation of test data for paths coverage based on node probability[J]. Journal of Software, 2016, 27(4):802 – 815. (in Chinese)

[28] 李爱国,张艳丽. 基于 PSO 的软件结构测试数据自动生成方法[J]. 计算机工程, 2008, 34(6):93 – 97.
LI Aai-guo, ZHANG Yan-li. Automatic generation method of test data for software structure based on PSO[J]. Computer Engineering, 2008, 34(6):93 – 97. (in Chinese)

[29] NARMADA N, DURGA P M. Automatic test data generation for data flow testing using particle swarm optimization[A]. Proceedings of the Third International Conference on Contemporary Computing[C]. Noida, India: IEEE, 2010. 1 – 12.

[30] 史娇娇,姜淑娟,韩寒,王令赛. 自适应粒子群优化算法及其在测试数据生成中的应用[J]. 电子学报, 2013, 41(8):1555 – 1559.
SHI Jiao-jiao, JIANG Shu-juan, HAN Han, WANG Ling-sai. Adaptive particle swarm optimization algorithm and its application in test data generation[J]. Acta Electronica Sinica, 2013, 41(8):1555 – 1559. (in Chinese)

[31] 廖伟志. 基于路径自动分割的测试数据生成方法[J]. 电子学报, 2016, 44(9):2254 – 2261.
LIAO Wei-zhi. Test data generation based on automatic division of path[J]. Acta Electronica Sinica, 2016, 44(9):2254 – 2261. (in Chinese)

[32] AHMED M A, HERMADI I. GA-based multiple paths test data generator[J]. Computer Operation and Research, 2008, 35(10):3107 – 3124.

[33] 巩敦卫,张婉秋. 基于自适应分组的大规模路径覆盖测试数据进化生成[J]. 控制与决策, 2011, 26(7):979 – 983.
GONG Dun-wei, ZHANG Wan-qiu. Evolutionary generation of test data for many paths coverage based on adaptive grouping[J]. Control and Decision, 2011, 26(7):979 – 983. (in Chinese)

[34] GONG D W, ZHANG W Q, YAO X J. Evolutionary generation of test data for many paths coverage based on grouping[J]. Journal of System and Software, 2011, 84(12):2222 – 2233.

[35] MARAGATHAVALLI P, KANMANI S, KIRUBAKAR J S, et al. Automatic program instrumentation in generation of test data using genetic algorithm for multiple paths cov-

erage[A]. Proceedings of 2012 International Conference on Advances in Engineering, Science and Management [C]. Tamil Nadu, India: IEEE, 2012. 349 – 353.

[36] LI A G, ZHANG Y L. Automatic generating all-path test data of a program based on PSO[A]. Proceedings of World Congress on Software Engineering[C]. Xiamen, China: IEEE, 2009. 189 – 193.

[37] 聂鹏,耿技,秦志光. 多路径粒子群优化自动测试用例生成算法[J]. 计算机集成制造系统,2012,18(1):216 – 223.
NIE P, GEN J, QIN Z-G. Multi-path oriented swarm optimization automatic test case generation algorithm[J]. Computer Integr Manuf System, 2012, 18(1): 216 – 223. (in Chinese)

[38] WANG S T, WU H. A novel algorithm for multi-path test data generation[A]. Proceedings of the Fourth International Conference on Digital Manufacturing and Automation[C]. Qingdao, China: IEEE, 2013. 58 – 60.

[39] HAN X, LEI H, WANG Y S. Multiple paths test data generation based on particle swarm optimization[J]. IET Software, 2017, 11(2): 41 – 47.

[40] YAO X, GONG D. Genetic algorithm-based test data generation for multiple paths via individual sharing[J]. Computational Intelligence and Neuroscience, 2014, 2014(3): 59 – 70.

[41] ZHU Ziming, XU Xiong. Improved evolutionary generation of test data for multiple paths in search-based software testing[J]. 2017 IEEE Congress on Evolutionary Computation[C]. Washington DC, USA: IEEE, 2017. 612 – 620.

[42] 巩敦卫,张岩. 一种新的多路径覆盖测试数据进化生成方法[J]. 电子学报,2010,38(6):1299 – 1304.
GONG Dun-wei, ZHANG Yan. Novel evolutionary generation approach to test data for multiple paths coverage[J]. Acta Electronica Sinica, 2010, 38(6): 1299 – 1304. (in Chinese)

[43] ZHANG Y, GONG D, YAO X, LU Q. Generating test data covering multiple paths using genetic algorithm incorporating with reducing input domain[A]. Proceedings of Chinese Intelligent Systems Conference[C]. Mudanjiang, China: Springer-Verlag, 2017. 739 – 747.

作者简介

廖伟志 男. 1974 年 1 月出生,广西凤山人. 博士、教授、硕士生导师. 主要从事软件测试、计算智能方面的研究工作.
E-mail: 1507715916@qq.com

夏小云 男. 1982 年 10 月出生,江西南昌人. 博士、副教授、硕士生导师. 主要从事计算智能、机器学习方面的研究工作.
E-mail: scutxxy@gmail.com