基于蚁群算法的猜测符号执行的路径搜索

李 航 臧 洌 甘 露
(南京航空航天大学计算机科学与技术学院 南京 211106)

摘 要 符号执行作为一种基本的程序分析技术,已被广泛应用于软件测试领域。研究表明,即使在现有的查询优化技术的支持下,约束求解也仍然是符号执行中最耗时的部分。猜测符号执行的思想是将多次约束求解合并成一次求解,从而减少约束求解消耗的时间。但是,猜测的成功率受猜测深度和路径搜索方向的影响,尤其是路径搜索的方向在较大程度上决定了整体猜测的成功率。因此,引导路径搜索向成功率高的方向进行,对提高猜测符号执行的整体效率至关重要。在猜测符号执行的路径搜索过程中引入蚁群算法,根据节点条件信息初次确定分支路径的权重,在多次迭代中根据分支路径的覆盖情况更新权重,通过权重决定路径搜索的方向。实验表明,该方法有效提升了猜测符号执行的效率。

关键词 蚁群算法,猜测符号执行,约束求解,路径搜索
中图法分类号 TP311 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2018.06.025

Search of Speculative Symbolic Execution Path Based on Ant Colony Algorithm

LI Hang ZANG Lie GAN Lu
(School of Computer Science and Technology, Nanjing University of Aeronautics & Astronautics, Nanjing 211106, China)

Abstract Symbolic execution has been widely used in the field of software testing. The research shows that constraint solving is the most time-consuming part in symbolic execution, even though some optimization techniques are used. The speculative symbolic execution reduces the consuming time of constraint solved by making several continuous constraint solving merge into once. The success rate of every time guess is affected by the depth of conjecture and the direction of search, especially the direction of search. Therefore, how to guide the path search to conduct in the direction of success is very important to improve the efficiency of speculative symbolic execution. In this paper, ant colony algorithm was used to search the path. Firstly, according to the node condition, the weight of branch path was determined. Then, the weight of a branch was updated according to whether this branch is covered in every time guess. This paper chose the direction of search based on the weight of branch. The experimental results show that the proposed method can improve the efficiency of speculative symbol execution effectively.

Keywords Ant colony algorithm, Speculative symbolic execution, Constraint solving, Path search

1 引言

符号执行技术是一种重要的程序分析技术,由King等[1]于1976年提出。符号执行使用符号化的输入来代替实际的输入,程序中的操作被转化为相应符号表达式的操作。在遇到分支指令时,程序的执行也相应地分叉,以探索每个分支,分支条件被加入到当前路径的路径条件中。通过调用约束求解器对路径条件进行求解来判断程序路径的可行性。如果路径条件有解,则说明路径可行;如果路径条件无解,则终止对该路径的分析[2]。

目前,符号执行面临着两方面的困难和挑战:1)路径爆炸问题,针对该问题的相关研究较多,例如通过消除冗余路径缓解路径爆炸问题[3]、通过摘要方法解决路径爆炸问题、采用分段式分析的分段符号执行[4];2)约束求解问题,其是符号执行过程中最为耗时的部分,有研究表明其大约占到执行时间的40%~90%[2]。文献[5]通过消除冗余约束条件来缩短约束求解的时间,文献[6]使用拓展的并行KLEE约束求解器来缩短约束求解的时间,文献[7]使用机器学习的方法来简化复杂约束。

针对约束求解耗时的问题,目前主要采用"查询优化"的技术。典型的查询优化有反例缓存、约束独立性分析和实例化。虽然查询优化一定程度地降低了约束求解的时间开销,但约束求解的用时占比依旧很高。为此,有学者提出了猜测符号执行(Speculative Symbolic Execution, SSE)[8],其主要思路是将多次约束求解合并成一次约束求解,通过减少约束求解器的调用次数来减少约束求解的耗时。SSE方法提升效率的关键在于提高每次猜测的成功率,而提高猜测成功率的关键又在于制定优秀的路径搜索策略。文献[2]进行了"真分支

到稿日期:2017-03-05 返修日期:2017-06-09

李 航(1991-),男,硕士,CCF会员,主要研究方向为软件测试,E-mail:422813555@qq.com;臧 洌(1965-),女,硕士,副教授,主要研究方向为机器学习、入侵检测,E-mail:zangliwen@nuaa.edu.cn(通信作者);甘 露(1991-),女,硕士,主要研究方向为机器学习、软件缺陷预测。

优先”策略和“假分支优先”策略的实验对比，在不同的测试程序下，两种方法各有优劣。本文利用蚁群算法的思想，将分支节点看作蚂蚁路径搜索中的障碍物，左右分支的可满足性大小作为路径的长度（权重）。在每次路径猜测迭代中，基于分支路径覆盖的次数（信息素）来更新分支的权重，并将其作为下次路径搜索时的信息导向。该方法能够有效利用分支可行性大小这一关键信息，并动态更新分支权重，使得每次猜测的成功率尽可能地高。实验证明，利用分支可行性信息进行路径选择，提高了猜测的成功率，减少了回溯次数，也提高了猜测符号执行的效率。

2 猜测符号执行

传统的符号执行在路径分支处会将分支条件加入到对应的路径条件中，然后调用约束求解器来判断该路径条件是否可满足，如果可满足，则继续向下进行路径搜索，否则将该路径分支剔除。猜测符号执行引入了猜测思想，即遇到分支条件语句时不一定调用约束求解器，而是假设路径条件可满足，从而跳过此次判定，直到当前探索路径中未判定的分支数达到一定的数量 $k(k>0)$或者到达路径末端时再调用约束求解器。如果当前路径条件可满足，则继续向下进行路径探索；否则需要进行系统状态的回退，将当前路径上的不可行分支剪掉。

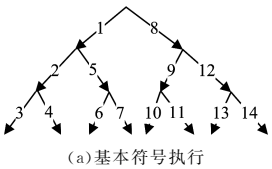
如果使用普通的符号执行，按照深度优先策略，完成图 1 示例程序的路径空间的搜索需要进行 14 次约束求解。而如果使用 SSE 方法，当路径上积累的未判定分支达到最大猜测深度时（路径上积累的未判定分支个数的最大值）才调用约束求解器进行判定，因此共需要进行 8 次约束求解。

```
1. int x,y;
2. if (x<0)
3.    x=-x;
4. if (y<0)
5.    y=-y;
6. x=x+y;
7. if (x>2)
8.    output(x);
```

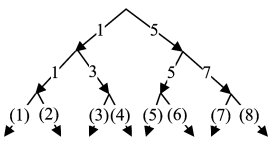
图 1 示例程序

Fig. 1 Sample program

图 2 显示的是图 1 中的程序简化后的路径空间，每条边上的数字表示该分支的可行性是在第 n 次约束求解中判定的，最大猜测深度为 3。图 3 和图 4 是其细化的符号执行树，每个节点包含了当前路径条件以及各变量对应的符号值。



(a)基本符号执行



(b)猜测符号执行

图 2 约束求解的次数

Fig. 2 Number of constraint solving

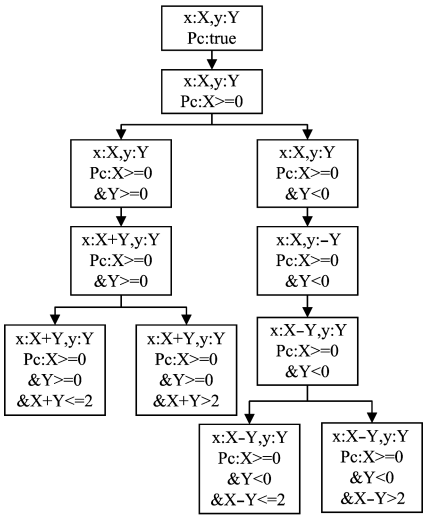


图 3 if(x<0)假分支方向的路径空间

Fig. 3 Path space in false branch direction of if (x<0)

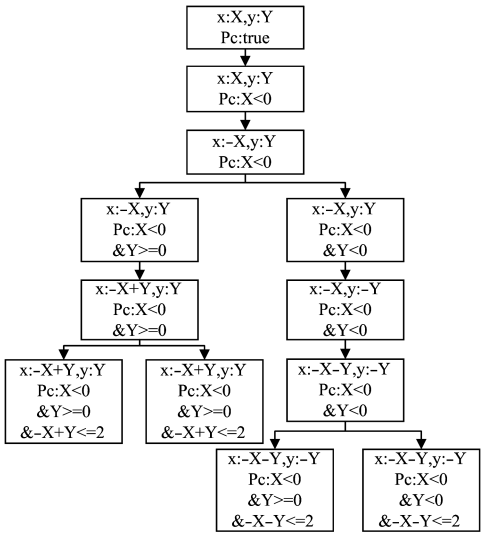


图 4 if(x<0)真分支方向的路径空间

Fig. 4 Path space in the true branch direction of if (x<0)

3 蚁群算法

蚁群算法由意大利学者 Dorigo 等人首先提出，他们充分利用蚁群搜索食物的过程与旅行商问题（TSP）之间的相似性，解决了 TSP 问题[9]。蚁群算法的思想是借助前边走过的蚂蚁所留下的分泌物的多少，来指导后续蚂蚁选择哪条路径，分泌物越多的路径被选择的概率越大[10]。受蚂蚁这种群体行为的启发，使用正反馈手段搜索最短路路径或最优路径。

以 TSP 问题来说明基本蚁群算法的框架。设有 n 个城市，$d_{ij}(i,j=1,2,\cdots,n)$表示城市 i 和城市 j 之间的距离，$\tau_{ij}(t)$表示在 t 时刻城市 i 和城市 j 之间的信息量，以此来模拟蚂蚁的分泌物。设共有 m 只蚂蚁，用 $p_{ij}^{k}(t)$表示在 t 时刻蚂蚁由城市 i 转移到城市 j 的概率，则：

$$p_{ij}^{k}(t)=\begin{cases}\dfrac{\tau_{ij}^{\alpha}(t)\ \eta_{ij}^{\beta}(t)}{\sum\limits_{r\in allowed_k}\tau_{ir}^{\alpha}(t)\ \eta_{ir}^{\beta}(t)}, & j\in allowed_k\\ 0, & \text{otherwise}\end{cases}\tag{1}$$

其中，$allowed_k$ 表示蚂蚁 k 下一步允许走过的城市的集合，

α 表示路径上的信息量对蚂蚁选择路径所起作用的大小，η_{ij} 表示从城市 i 转移到城市 j 的期望程度，例如可以选取 $\eta_{ij}=1/d_{ij}$，β 表示 η_{ij} 的作用。当 $\alpha=0$ 时，蚁群算法即为传统的贪心算法；而当 $\beta=0$ 时，其就成为了纯粹的正反馈的启发式算法。经过 n 个时刻，蚂蚁可以走完所有的城市，完成一次循环，每只蚂蚁走过的路径就是一个解。根据式(2)来更新路径上的信息量：

$$\tau_{ij}(t+1)=(1-p)\cdot\tau_{ij}(t)+\Delta\tau_{ij} \tag{2}$$

其中，$p\in(0,1)$，表示信息量 $\tau_{ij}(t)$ 随时间的推移而衰减的程度。信息量 $\Delta\tau_{ij}$ 为：

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

其中，$\Delta\tau_{ij}^{k}$ 表示蚂蚁 k 在本次循环中在城市 i 和城市 j 之间留下的信息量，其计算方法由模型而定。

4　结合蚁群算法的路径搜索

4.1　分支条件的可满足性大小

文献[2]在研究猜测方向对成功率的影响时发现，对于大多数程序，真分支不可行的概率较高(见表 3)。其原因有两点：1)对于条件为等式的分支语句，真分支边的路径空间十分狭窄；2)编程人员习惯把特殊情况放在 then 块中进行处理，而将非特殊情况放在 else 块中进行处理，或者相反，将特殊情况放在 else 块中进行处理，而将非特殊情况放在 then 块中进行处理。无论是哪种原因，都意味着程序的执行树往往是不对称的，而是向分支语句的某一分支侧倾斜。

针对第一种原因，假设遇到了等式条件 if($A==1$)，那么真分支成立的条件就为当且仅当 A 为 1，而假分支成立的条件则较宽泛，只要 A 不为 1 即可。在解集的大小方面，真分支的解集远小于假分支的解集，如果当前的路径条件中还有其他关于 A 的限制语句，则此节点的真分支的可行性明显小于假分支的可行性。例如，如果路径条件中有其他关于 A 的限制条件 $A<0$，那么此节点的真分支将不可行。

针对第二种原因，特殊情况一般是边界值或是几个离散的点值，也就是说条件语句基本还是等式，解集较小或只有一个元素。总结这两种情况发现，真假分支的解集大小对分支的可行性的影响较大。这是因为路径条件是由若干个分支条件求交集，每个分支条件的解集越大，从而其与其他解集相交的可能性就越大，路径条件有解的可能性也就越大。

对于变量 X，它在 m 和 n 节点某一分支的解集分别为 A 和 B，在当前节点处的解集为 D，D 中只有一个元素。如果 D 没有落在 A 和 B 的交集 C 中，则当前路径不可行，如图 5(a)所示。如果解集 D 很大，则很有可能与 A 和 B 都相交，当前路径可行，如图 5(b)所示。

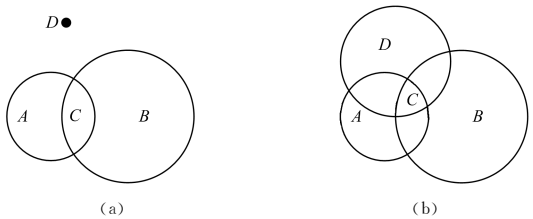


图 5　解集的大小对路径条件有解的影响

Fig. 5　Effect of solution size on solution of path conditions

本文将分支条件的可行性引入到路径选择中，将分支的可行性大小设置为分支边的权重，并将其作为蚁群算法中蚂蚁选择路径的依据，根据蚂蚁每次走过的分支情况来更新分支权重。

4.2　分支权重的初始化与更新

分支条件是布尔表达式，变量取值与其对应的表达式的真假值可以体现在真值表中，因此可以借助真值表来确定真假分支解的占比，并将其作为条件分支的初始权重。

例如，对于条件语句 if(A && $B\,\|\,C$)，其对应的真值表如表 1 所列，其中条件为真和条件为假的占比分别为 5/8 和 3/8，因此将真分支和假分支的权重初始值设为 5/8 和 3/8。

表 1　if (A && $B\,\|\,C$)的真值表

Table 1　Truth table for if (A && $B\,\|\,C$)

A	B	C	A && $B\,\|\,C$
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

对于单个条件的判断语句，例如 if($A==1$)，if($B!=0$)，if($A>B$)，if(!A)，由于无法从真值表中判断真假分支可行性的大小，因此只能人为分析并设定分支权重。例如，对于条件语句 if($A==1$)，当且仅当 A 取 1 时条件成立，而不成立的 A 的取值则非常多，因此为真分支设定一个较小的权重，如 0.1。而对于条件语句 if($B!=0$)，情况则刚好相反，它的假分支被赋予了一个很小的权重。另外，还存在无法判断左右分支可行性大小的 if 语句，如对于 if($A>B$)这种比较大小的判断语句，则认为其左右分支的可行性相同，设定初始权重都为 0.5。

初始化所有的分支权重后进行路径的搜索，当搜索到某个节点时选择权重较大的分支作为搜索方向。这些分支在被搜索后需要更新权重。权重的局部更新策略如下：

$$\tau_j(t+1)=\begin{cases}\tau_j(t)+\dfrac{1}{10n_{ij}}, & \text{猜测成功的分支}\\ \tau_j(t)-\dfrac{1}{10n_{ij}}, & \text{猜测失败的分支}\end{cases} \tag{4}$$

其中，$\tau_j(t)$ 表示 t 时刻节点 j 某一分支的权重，n_{ij} 表示从路径的初始猜测节点 i 到节点 j 的节点个数。一次猜测结束后将形成一个猜测段，如果节点 j 的分支在此猜测段中，则增大其权重，相当于蚂蚁经过后增加了该路径的信息量。若本次猜测失败，则剪除部分不可行的分支；同时，由于下一次猜测遇到这些分支时其不可行的概率依旧较高，因此为了纠正搜索方向，需要将这部分分支的权重调小。

4.3　路径搜索算法 SHS-A

猜测符号执行的路径搜索使用的是猜测式启发搜索(Speculative Heuristic Search，SHS)，该搜索是普通的启发式搜索方法的变体[2]。当从开放列表 openlist 中取出一个节点后，搜索过程会从该节点开始向下探索，从而形成猜测段，最后调用约束求解器进行判定。本文通过结合 SHS 算法与蚁

群算法，提出了一种 SHS-A 算法，该算法确定每次猜测时路径搜索的方向，并及时更新相关分支的权重。SHS-A 算法的具体描述如算法 1 所示。

算法 1 SHS-A

输入：最大猜测深度 K，状态 s，猜测段 ss

```
1. begin
2.  将状态 s 添加至开放列表 openlist
3.  while openlist ≠Ø do
4.    s←pickState(openlist);ss←{s};
5.    segmentDone=false;d=0;
6.    while segmentDone=false do
7.      if(初次访问节点 s)//设置权重
8.        probSymCalculate(s_Tp);
9.        probSymCalculate(s_Fp);
10.     if(s_Tp>s_Fp)//比较真假分支的权重
11.       ss←{s_T};d+=1;
          //探索 s 真分支指向的下一个节点 s_T;
12.     else
13.       ss←{s_F};d+=1;
          //探索 s 真分支指向的下一个节点 s_F;
14.     if (s_T 或 s_F 有未访问过的后继节点)
15.       将 s_T 或 s_F 添加至开放列表 openlist;
16.     if(d==K)//达到最大猜测深度;
17.         调用约束求解器;
18.         segmentDone=true;
19.         if(ss 可满足)
20.           根据式(4)修改分支权重;
21.         else
22.           根据式(4)修改分支权重;
23.           剪除 ss 上的不可行节点，并从开放列表中删除;
24.   endwhile
25. endwhile
```

4.4 SHS-A 算法分析

每次进行路径搜索时，若开放列表 openlist 不为空，则从 openlist 中任选节点，并将标志位 segmentDone 设置为 false（见算法 1 中的第 2－5 行）。接着进行循环，目的是将节点扩展成猜测段，每次循环时，如果节点被初次探索，则初始化分支权重（见算法 1 中的第 7－9 行）。然后比较分支权重，选择权重较大的分支作为搜索方向，如果有未访问的后继节点，则将它加入开放列表 openlist 中（见算法 1 中的第 10－15 行）。当循环次数达到最大猜测深度 K 时，调用约束求解器，如果猜测段可行，则修改分支权重，否则在修改完分支权重后删除猜测段上的不可行分支（见算法 1 中的第 16－23 行）。算法 1 运行至开放列表openlist 为空时结束，整个算法的时间复杂度为 $O(n^2)$，空间复杂度为 O(1)。

SHS-A 算法的优点在于权重的设计和更新。首先，分支权重代表可行性的大小，选择可行性大的分支时，得到可行解的概率大，猜测成功率高。其次是更新机制，由于前后分支条件间存在关联性，一个分支的可行率高并不代表每条路径运行此分支的可行概率都高，相反，有些权重大的分支容易造成猜测失败。更新机制可以在猜测失败时逐步调整分支权重，使失败率高的分支权重小于成功率高的分支权重。

图 6 中给出了已搜索过的路径 s 上的一段路径 $A_TB_TC_TD_T$，再次搜索时从节点 A 的假分支出发至节点 C，此时面临是搜索 C 的真分支还是假分支的问题。在此有两种情况：1）C 在被 s 覆盖之前真分支的权重较大，在更新之后仍然是真分支的权重较大，那么本次搜索至节点 C 时，仍然会选择真分支作为搜索方向，而由于之前路径段 C_TD_T 被成功搜索过，因此此次可行的概率仍然很高；2）C 的真分支初始权重较小，优先搜索 C 的假分支，但多次路径搜索在搜索到 C 的假分支时发现路径不可行，导致 C 的假分支权重逐渐减小，真分支权重逐渐增大。当 s 覆盖 C 的真分支后，C 的真分支权重首次超过了假分支权重。因此，本次猜测至节点 C 时，将搜索真分支 C_TD_T，而避开失败率较高的假分支 C_F。

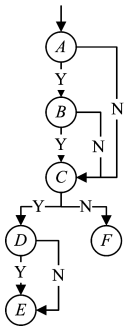


图 6 程序流程图片段

Fig. 6 Fragment of program flowchart

5 实验

5.1 实验设计

Symbolic PathFinder(SPF)是 NASA Ames 实验室研发的基于 Java PathFinder(JPF)的开源符号执行工具。JPF 的两个核心模块 JVM[JPF] 和搜索引擎模块 Search Strategy 实现了路径空间的探索，SPF 在此基础上对 JPF 进行了扩展：为每一个符号化的变量都附加了一个符号值，而且提供了统一的约束求解器调用接口。S_2PF 又对 SPF 进行了扩展：首先，修改了搜索引擎模块，使用猜测式启发搜索引擎来实现 SHS 算法；其次，JVM[JPF] 模块添加了新的选择生成器 choice generator，用于保存探索过的每一个分支的可行性，它位于路径空间的分叉点，以实现某些特殊的搜索策略[2]。本文实验使用的是 S_2PF，搜索引擎利用选择生成器中的分支权重信息（可行性的大小）来实现 SHS-A 的搜索策略。

表 2 列出了实验中所用的程序，其中一部分经常被用于进行相关的研究[11]，其余是 SIR 的测试集程序[12]。代码行数为 53～1200，不可行分支所占的比例为 0～47%。

表 2 实验中所用的程序

Table 2 Programs used in experiment

Program	Description
WBS	刹车系统
TreeMap	红黑树、数据结构
BinTree	二分搜索树、数据结构
BinomialHeap	二项堆、数据结构
FibHeap	Fibonacci 堆
List	有序双向链表、数据结构
ArrayPartition	用一个元素作为基准，将数组分为两部分
BinaryHeap	数组实现的二叉堆
OrdSet	数组实现的有序集

实验使用了优化的 SSE 方法，在“真分支边优先”“假分支边优先”和本文的 SHS-A 路径搜索策略下对搜索时间和约束求解时间进行了对比。根据文献[2]中的实验结论，本实验中的最佳猜测深度设置为 6。

5.2　实验结果与分析

图 7 对比了 3 种搜索策略下的路径搜索时间，图 8 对比了 3 种搜索策略下的约束求解时间。

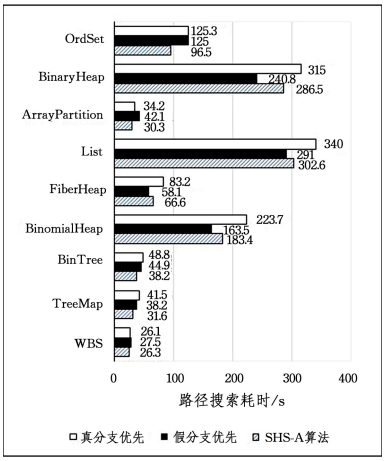


图 7　3 种搜索策略的搜索耗时

Fig. 7　Search time of three search strategies

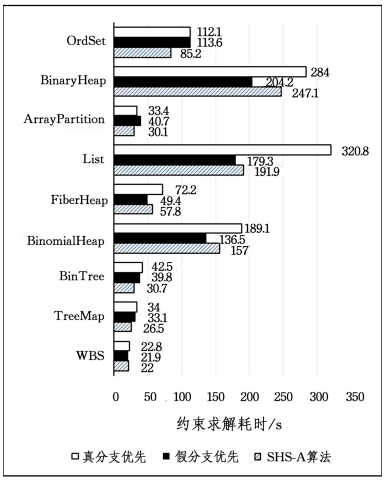


图 8　3 种搜索策略的约束求解耗时

Fig. 8　Constraint solution time of three search strategies

可以看到，本文提出的 SHS-A 算法的路径搜索策略比“真分支优先”和“假分支优先”更具优势，尤其是在真假分支的不可行占比适中的情况下，效率提升极为明显。虽然有些程序使用“假分支优先”路径搜索策略时比本文方法的用时更少，但在现实的测试程序中，事先是不可能知晓分支不可行占比的，因此本文的方法在多数情况下仍然效率占优。

对于 WBS 程序，因为其程序路径空间中的不可行分支占比为 0，猜测总是成功，所以本文的策略与“真分支优先”“假分支优先”相比基本没有优势。而有些程序的不可行分支占比过高（如 List，BinaryHeap），它们的不可行真分支占比达到了 70%以上，猜测失败的概率较高，所以即使是使用了本文的搜索策略，效率提升得也不明显。对于表 2 中的BinomialHeap，FiberHeap 和 BinaryHeap 程序，本文搜索策略的效果反而较差，这是因为在这 3 个程序的路径空间中假分支的不可行占比接近 0（见表 3），也就是说对比本文的搜索策略，“假分支优先”策略的成功率明显更高，约束求解的时间开销小。

表 3　程序真假分支的约束求解结果统计

Table 3　Result of constraint solution of true and false branch of programs

程序	可行的真分支数	不可行的真分支数	可行的假分支数	不可行的假分支数
WBS	13823	0(0%)	13823	0(0%)
TreeMap	11567	10566(48%)	15438	6695(30%)
BinTree	9214	9771(52%)	13167	5818(31%)
BinomialHeap	70270	23576(25%)	93846	0(0%)
FiberHeap	25006	8572(26%)	33008	570(2%)
List	30276	80952(73%)	97800	13428(12%)
ArrayPartition	3610	812(18%)	1651	2771(63%)
BinaryHeap	16489	88250(84%)	101431	3308(3%)
OrdSet	25852	13736(35%)	25373	14215(36%)

结束语　针对如何提高猜测符号执行效率的问题，本文提出了一种基于蚁群算法的路径搜索算法。该算法根据蚂蚁寻食的正反馈特点，设计了根据分支可行性大小来选择分支路径的搜索策略。首先，根据节点分支条件的解集大小，为分支赋予初始权重，将其作为第一只蚂蚁的寻径选择依据；然后，每当一只蚂蚁搜寻过一条路径 S 后，更新路径 S 上每个分支的权重，并将其作为下一只蚂蚁的寻径选择依据。这样既可以提高本次猜测的成功率，又能指导后续猜测向成功率高的方向搜索，从而提高了整体的猜测成功率。实验表明，本文的路径搜索策略比单一方向的搜索策略有明显的效率提升。SHS-A 算法搜索策略的不足之处在于分支权重的初始化。真值表中的变量可能代表一个布尔表达式，当变量代表一个等式或是不等式时，将其看作一个变量来处理，这将会造成初始权重与实际情况不符。因此，未来的研究工作应着眼于细化布尔表达式，使权重的设定更贴近真实情况。

参 考 文 献

[1] KING J C. Symbolic execution and program testing [J]. Communications of the ACM, 1976, 19(7): 385-394.

[2] ZHANG Y F. Improving the Scalability and Feasibility of Symbolic Execution [D]. Changsha: National University of Defense Technology, 2013. (in Chinese)
张羽丰. 符号执行可扩展性及可行性关键技术研究[D]. 长沙：国防科学技术大学，2013.

[3] WANG H, LIU T, GUAN X, et al. Dependence Guided Symbolic Execution [J]. IEEE Transactions on Software Engineering, 2017, 43(3): 252-271.

[4] YAN T. Dynamic Symbolic Execution with Segmented[D]. Shanghai: East China Normal University, 2015. (in Chinese)
颜婷. 分段式分析方法在动态符号执行中的应用[D]. 上海：华东师范大学，2015.

[5] ZOU Q, HUANG W, AN J, et al. Redundant constraints elimination for symbolic execution[C]// IEEE Information Technology, Networking, Electronic and Automation Control Conference. IEEE, 2016: 235-240.

[6] KANG W T. Optimization of Constraint Solver for KLEE, A

Dynamic Symbolic Execution Tool [D]. Chengdu: University of Electronic Science and Technology, 2014. (in Chinese)
康文涛. 符号执行工具 KLEE 约束求解优化设计与实现[D]. 成都:电子科技大学,2014.

[7] LI X, LIANG Y, QIAN H, et al. Symbolic execution of complex program driven by machine learning based constraint solving[C]// Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering. ACM, 2016: 554-559.

[8] ZHANG Y, CHEN Z, WANG J. Speculative Symbolic Execution [C]// IEEE, International Symposium on Software Reliability Engineering. IEEE, 2012: 101-110.

[9] DORIGO M. Ant Colonies for the Traveling Salesman Problem [J]. Biosystems, 1997, 43(2): 73-81.

[10] CHEN L, SHEN J, QIN L, et al. An adaptive ant colony algorithm based on equilibrium of distribution [J]. Journal of Software, 2003, 14(8): 1379-1387.

[11] STAATS M, PĂSĂREANU C. Parallel symbolic execution for structural test generation[C]// Proceedings of the 19th International Symposium on Software Testing and Analysis. ACM, 2010: 183-194.

[12] DO H, ELBAUM S, ROTHERMEL G. Supporting Controlled Experimentation with Testing Techniques: An Infrastructure and its Potential Impact[J]. Empirical Software Engineering, 2005, 10(4): 405-435.

(上接第 116 页)

[2] FAN C L, CHAN Y C, ZHANG Z K. Robust remote authentication scheme with smart cards[J]. Computers & Security, 2005, 24(8): 619-628.

[3] LI C T, HWANG M S. An efficient biometrics-based remote user authentication scheme using smart cards[J]. Journal of Network and Computer Applications, 2010, 33(1): 1-5.

[4] LIAO Y P, HSIAO C M. A novel multi-server remote user authentication scheme using self-certified public keys for mobile clients[J]. Future Generation Computer Systems, 2013, 29(3): 886-900.

[5] TSENG Y M, WU T Y, WU J D. A pairing-based user authentication scheme for wireless clients with smart card[J]. Informatics, 2008, 19(2): 285-302.

[6] CHAUDHRY S A. A secure biometric based multi-server authentication scheme for social multimedia networks[J]. Multimedia Tools & Applications, 2016, 75(20): 1-21.

[7] XIA P Z, CHEN J H. Three-factor authentication scheme for multi-server environments based on elliptic curve cryptography [J]. Application Research of Computers, 2017, 34(10): 3061-3067. (in Chinese)
夏鹏真,陈建华,一个基于椭圆曲线密码的多服务器环境下三因子认证协议[J]. 计算机应用研究,2017,34(10):3061-3067.

[8] WANG D, WANG P. Two Birds with One Stone: Two-Factor Authentication with Security Beyond Conventional Bound[J]. IEEE Transactions on Dependable and Secure Computing, 2016, PP(99): 1.

[9] WANG D. Robust biometric-based user authentication scheme multi-server environment [J]. IEEE Systems Journal, 2015, 9(3): 816-823.

[10] DODIS Y, REYZIN L. Fuzzy extractors: how to generate strong keys from biometrics and other noisy data[M]// Advances in Cryptology — EUROCRYPT 2004. Berlin: Springer, 2004: 523-540.

[11] KOBLITZ N. Elliptic curve Cryptosystem[M]// Mathematics Computing, 1987: 203-209.

[12] MILLER V. Uses of elliptic curves in cryptography[M]// Advances in Cryptology — CRYPTO'85 Proceedings. Berlin: Springer, 1986: 417-426.

[13] DOLEV D, YAO A C. On the security of public Key Protocols [J]. IEEE Transactions on Information Theory, 1981, 29(2): 198-208.

[14] KOCHER P, JAFFE J, JUN B. Differential power analysis[C]// 19th Annual International Cryptology Conference. Berlin: Springer, 1999: 388-397.

[15] MESSER T, DAB E, SLOAN R. Examining smart-card security under the threat of power analysis attacks[J]. IEEE Transactions on Computers, 2002, 51(5): 541-552.

[16] BRIER E, CLAVIER C, OLIVIER F. Correlation power analysis with a leakage model[C]// Proceedings of the 6th International Conference on Cryptographic Hardware and Embedded Systems. Berlin: Springer, 2004: 16-29.

[17] GAO Y, CHENG S L. Building network security channel-Virtual network technology [J]. Traffic Information and Security, 2001, 19(1): 30-32. (in Chinese)
高岩,程胜利. 构筑网络安全信道-虚拟专用网技术[J]. 交通信息与安全,2001,19(1):30-32.

[18] GUO D, WEN Q, LI W. Analysis and Improvement of Chaotic Map Based Mobile Dynamic ID Authentication Key Agreement Scheme[J]. Wireless Personal Communications an International Journal, 2015, 83(1): 35-48.

[19] YANG S P. Formal Analysis of Security Protocol and BAN logic [D]. Guiyang: Guizhou University, 2007. (in Chinese)
杨世平,安全协议及其 BAN 逻辑分析研究[D]. 贵阳:贵州大学,2007.

[20] KILINC H H, YANIK T. A survey of SIP Authentication and Key Agreement Schemes IEEE Communications[J]. Surveys & Tutorials, 2014, 16(2): 1005-1023.

[21] YANG L, MA J F. Trusted Mutual Authentication Scheme with Smart Cards and Passwords[J]. Journal of University of Electronic Science and Technology of China, 2011, 40(1): 128-133. (in Chinese)
杨力,马建峰. 可信的智能卡口令双向认证方案[J]. 电子科技大学学报,2011,40(1):128-133.

[22] WANG Y F. A Smart Card Password Authentication Scheme Study [J]. Computer Applications and Software, 2011, 28(9): 295-297. (in Chinese)
王亚飞. 一种基于智能卡口令认证方案的研究[J]. 计算机应用与软件,2011,28(9):295-297.