第 33 卷 第 7 期 计 算 机 工 程 2007 年 4 月
Vol.33 No.7 Computer Engineering April 2007
·网络与通信· 文章编号：1000—3428(2007)07—0120—03 文献标识码：A 中图分类号：TP393

分布式网络性能监测的探针部署方法研究

钱 进，贺贵明
(武汉大学计算机学院，武汉 430079)

摘 要：对于分布式网络性能的监测，监测探针的部署必须在代价和覆盖范围之间进行权衡。该文以最小化监测探针数目为目标，在链路覆盖和消息覆盖约束条件下，讨论了一种基于遗传算法的网络性能监测探针部署方法，并且在传统遗传算法的基础上提出了使用基于边编码的遗传算法解决监测探针的部署问题。实验表明基于遗传算法及其改进算法的监测探针部署方法比贪婪算法具有更好的性能。

关键词：网络性能监测；集合覆盖；贪婪算法；遗传算法

Research on Probe Deployment of Distributed Network Performance Monitor

QIAN Jin, HE Guiming
(School of Computer Science, Wuhan University, Wuhan 430079)

【Abstract】 The probe deployment of distributed network monitor must be tradeoff between cost and coverage. A new approach to network monitor's deployment based on genetic algorithm is presented with the goal of minimizing the number of network monitors subjected to link coverage and message coverage. The genetic algorithm based on edge coding is preferred to traditional genetic algorithm. The simulation indicates that the performance of the algorithm is better than the greedy algorithm.

【Key words】 Network performance monitor; Set coverage; Greedy algorithm; Genetic algorithm

网络监测是网络管理的基础，监测探针的部署必须在代价和覆盖范围之间进行权衡。监测探针过多则费用高，监测探针过少则可能不能覆盖整个被管网络并且响应时间过长。监测探针的部署一直是网络监测研究的主要问题之一。

现有的研究一般将监测探针的部署问题映射为一个最小集合覆盖(Set Cover)问题。SC问题被证明是NP-Hard问题，其典型的求解方法是启发式贪婪算法及其改进算法[1]，这类算法能找到一个能够覆盖所有链路的最小节点集合，但算法没有考虑监测流量和响应时间。文献[2]以最小化监测总流量和最小化最大监测响应时间为目标，结合移动代理的特性提出了一种启发式监测代理部署算法，但这类方法必须基于移动代理实现。文献[3]以网络流为监测单位，讨论了以最大化覆盖监测流的效益和最小化部署监测探针代价为目标的监测探针部署问题，不过效益函数的选取比较困难。

1 性能监测探针部署问题建模

我们把被管理网络建模成一个有向图 $G(V,E)$，图的顶点V表示网络中顶点，边E表示连接路由器的通信链路。节点和边的数量分别表示为 $|V|$ 和 $|E|$。用$P_{s,t}$表示一个IP数据包从源节点S到目的节点t所经历的路径。对每个节点 $x \in P_{s,t}$，$P_{x,t}$包含在$P_{s,t}$中。对每个节点 $s \in V$，通过合并所有的路径$P_{s,t}$，$\forall t \in V$必然得到一个树型拓扑。这颗树称作节点S的路由树RT(routing tree)，用T_s表示。T_s定义了从节点S到V中所有其它节点的路由路径。

假设给沿着任意一对节点 $s,t \in V$ 之间的路径$P_{s,t}$发送一个消息关联成本$C_{s,t}$。为了监测链路 $e \in E$，必须选择一个节点 $s \in V$ 作为监测探针，使e被包含在S的路由树中，即 $e \in T_s$。节点S需要向链路e的两个端节点分别发送探针包(通常链路的监测都需要同时采集两个端点上的信息，例如单向时延的测量)，除了链路e之外，这两个探针包经历了相同的路径。

一个监测探针只能测量其路由树中链路的性能，因此一个可以监测网络中所有链路的监测系统需要确定监测站的集合 $S \subseteq V$ 和管理域划分，它们必须满足两个约束条件：

(1)链路覆盖约束。保证所有链路都被 S 中节点的路由树覆盖，即 $\bigcup_{s \in S} T_s = E$；

(2)消息覆盖约束。定义消息集合 $A \subset \{m(s,u) \mid s \in S, u \in V\}$，其中每一个消息 $m(s,u)$代表从监测站 S 到节点 u 的探针消息。消息覆盖约束保证对每一条边 $e=(u,v) \in E$，都存在节点 $s \in S$ 使得 $e \in T_s$，且 A 包含消息 $m(s,u)$和 $m(s,v)$。

上述两个约束条件本质上确保了每一条链路都被某个监测探针监测。满足上述约束的一个(S,A)对被称作一个可行解。本文目前仅考虑未加权的链路监测问题。事实上，本文的结论很容易扩展到加权的情况：给每个节点关联一个成本，寻找 $S \subseteq V$ 使得 S 中监测站的总成本最小。

2 求解方法

从上一节的分析可以看出链路监测问题与最小集合覆盖问题类似。集合覆盖问题已被证明是 NP-Hard 问题，贪婪式搜索算法是其中一种比较著名的简单易行的方法。本文根据顶点覆盖问题的解决思路，提出了使用混合遗传算法来解决监测探针的部署问题。

2.1 贪婪搜索算法

首先把给定的链路监测问题映射成一个集合覆盖问题，然后使用启发式贪婪算法解决这个集合覆盖问题。在映射中，

作者简介：钱 进(1973－)，男，博士生，主研方向：计算机网络；贺贵明，博士、教授、博导

收稿日期：2006-05-24 **E-mail：**qianjin88@hotmail.com

边的集合E映射为元素集合Z，集合Ω包含每个节点$v \in V$ 的生成树集合 $Q_v = \{e \mid e \in T_v\}$。启发式贪婪算法的的主要思想是在每一次循环中，选择包含最多未覆盖元素的集合Q_{Ri}。

贪婪算法求解监测探针部署问题实际上是在每次迭代过程中，选取其最短路径树可以覆盖未被监测链路数目最多的节点作为部署节点。该算法简单易行，但是由于每次都是搜索局部最优解，因此得到的解往往不是最优解，部署的节点数目并不是最少的。

2.2 遗传算法解决监测探针部署问题

遗传算法不需要求解问题的任何信息而仅需要目标函数的信息，并且不受搜索空间是否连续或可微的限制。在遗传算法中，通过随机方式产生若干个所求解问题的数字编码，即染色体，形成初始群体；通过适应度函数给每个个体一个数值评价，淘汰低适应度的个体，选择高适应度的个体参加遗传操作，经过遗传操作后的个体集合形成下一代新的种群。对这个新种群进行下一轮进化。

文献[2]提出了一种基于点编码的启发式演化算法来解决顶点覆盖问题。我们借助该算法的思想来解决监测探针部署问题。

(1)编码机制

我们将被选择的部署点集合 $\vec{V}'$ 表示成一个二进制向量 $\vec{x} = x_1 x_2 ... x_n$，如果 $x_i = 1$，那么第 ith 个顶点属于 $\vec{V}'$，并且以该顶点为根的最短路径树上的链路也都被覆盖；否则第 ith 个顶点不属于 $\vec{V}'$。初始化群体中所有可行解都按照这种编码方式形成串。

(2)适应度函数

为了避免无效解，即避免链路没有被全部覆盖的情况，将被覆盖的链路数作为适应度函数，即被覆盖的链路数越大，该串的适应度越高。适应度函数 f($\vec{x}$)的求解方法如下：

```
public int Fitness( x )
{ V = φ ;
  while( x ≠ φ )
  {从 x 中取一基因位 x_i ;
   求取以 ith 节点为根的最短路径树 T;
   求取 T 中包含的链路集合 LT;
   V = V ∪ LT ;
   x = x - x_i ;}
  return |V| }
```

由于最短路径树的计算复杂度为 O(n2) (Dijkstra 算法，n 为节点数目)，因此，计算一个适应度函数的算法复杂度就为 O(n3)。

(3)控制参数(control parameters)

在实际操作中，需要适当地确定某些参数的值以便提高选优的效果。

串长，即字符串所含字符的个数，记为 L。在本算法中，根据编码机制，串长为节点个数，为定值，不能改变；群体的容量，即每一代群体所包含字符串的个数，记为 n。根据选择的方式，每一代群体的个数都会较上一代少；施行交换算子的概率即交换率(crossover rate)，记为 Pc；施行突变算子的概率，即突变率(mutation rate)，记为 Pm，在本算法中，若群体容量大于 300，取 Pc=0.6，Pm=0.001，若小于 300，取 Pc=0.9，Pm=0.01；对迭代次数，本算法选择 1 000 次。

基于点编码的遗传算法的最大缺点在于无效解的产生，它们可能出现在最初的种群(population)，也可能产生于正常的交叉和编译操作。无效解的传播将最终影响遗传算法的收敛速度和结果的正确性。为解决该问题而引入的评价函数一方面无法从根本上杜绝无效解的传播，另一方面增加了算法的计算复杂度。

2.3 改进遗传算法解决监测探针部署问题

为了消除无效解，本文提出基于边编码的遗传算法，通过边集合与点集合的对应关系，可确保所有解都是有效的。算法的基本流程和上述遗传算法一致，但在以下几个方面有区别。

(1)编码机制

该算法的个体是一个复合二进制向量 $\vec{e} = \vec{e_1}\vec{e_2}...\vec{e_m}$，$\vec{e_i}$ 对应于第 i条边，表示该条边位于以某个顶点为根最短路径树下。假设第i条边位于以k个顶点为根的最短路径树下，并且这k个顶点的集合为V_i，为了缩短编码的位数，$\vec{e_i}$ 采用 $\log_2 |V_i|$ 位(向上取整)来表示。如第 5 条边位于以顶点{2,5,7,8}为根的最短路径树下，则 $\vec{e_i}$ =00 表示第i条边被节点 2 所监视；$\vec{e_i}$ =01 表示第i条边被节点 5 所监视；$\vec{e_i}$ =10 表示第i条边被节点 7 所监视；$\vec{e_i}$ =11 表示第i条边被节点 8 所监视。

(2)适应度函数

该算法的评价函数为所选节点的个数，所选节点个数越少表明代价越小，适应度也越好。评价函数求解的方法如下：

```
Fitness( e )
1 V ← φ
2 while  e ≠ φ
3   for   i → 1 to m
4     do if B( e_i )>|V_i|  then
5          e_i 的最高位变异为 0;
6   vi ← e_i 所对应的节点
7   V ← V ∪ {vi}
8   e = e - e_i
9 return  |V|
```

在第 4 行中，B($\vec{e_i}$)表示 $\vec{e_i}$ 按照二进制计算以后得到的值，在经过一系列演化计算后，它可能大于 $|V_i|$，即出现不可行解，根据编码的方法可知这种情况下 $\vec{e_i}$ 的最高位必定为 1。第 5 行是通过变异以保证为可行解。

(3)选择算子

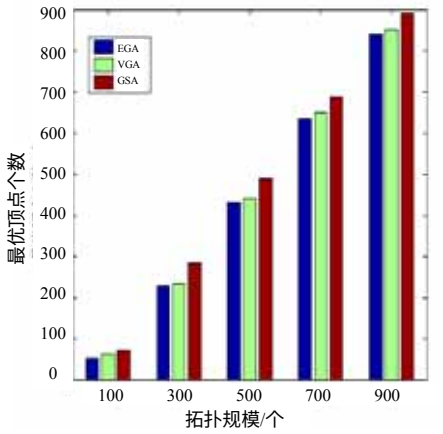
本算法采取的则是按比例选择的模式，即评价函数为 fi 的个体以 $\frac{1/f_i}{\sum 1/f_k}$ 的概率继续存在，其中分母为父代中所有个体适应度的倒数之和。因为在我们的算法中，评价函数越小，适应度越高。

通过以上的分析可以看出，基于边的编码方式只包括可行解。由编码方式可知，任一条边至少被一个最短路径树覆盖，这保证了所有的边都被覆盖，并且以后的评价函数计算也消除了无效解，所以解码得到的顶点集合是可行解。但是，基于边的编码方式使初始时的编码和最后的解码变得复杂了，不过这个代价并不能掩盖其优势。

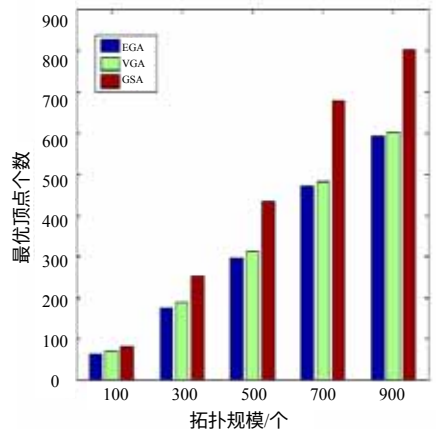
3 仿真实验

我们使用GT-ITM[4]工具产生的拓扑来进行实验，比较了贪婪算法(GSA)、传统基于点编码的遗传算法(VGA)和改进后基于边编码的遗传算法(EGA)的性能。

我们用 Waxman 模型和 Transit-Sit 模型各生成 5 组数据，分别包括 100、300、500、700 和 900 个点。图 1(a)是选用 Waxman 模型时，VGA、GSA 与 EGA 最优监测探针个数的比较(扩大 10 倍以后的相对值)。图 1(b)为选用 transit-stub 模型时，VGA、GSA 与 EGA 最优监测探针个数的比较(扩大 10 倍以后的相对值)。为了比较 VGA 和 EGA，采用了相同的交叉和变异参数。从图中可以看出，EGA 得到的监测探针数都是最小的，其次是 VGA 的计算结果，最次是 GSA。另外，EGA 和 VGA 所得到的最优解比较集中，而 GSA 由于随机地选择每一条边，导致最优解的随机性很大。



(a)wax 类型拓扑



(b)TS 类型拓扑

图 1 VGA 与 EGA、GSA 在不同拓扑规模下的性能比较

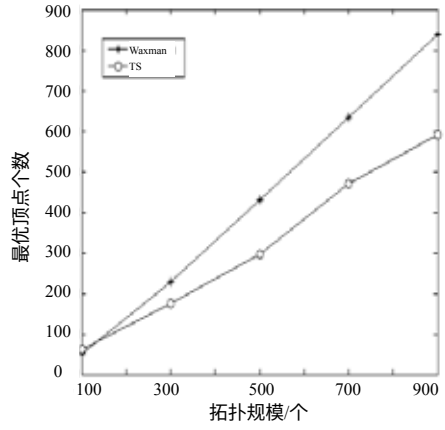


图 2 EGA 不同规模不同类型监测顶点个数比较

我们将使用改进遗传算法得到的监测探针部署数目以图 2 表示。可以看出监测探针数目基本上随拓扑节点数目呈线性增长，监测代价不会在拓扑节点数目增加时增长过快。图 3 是 EGA 的适应度函数在 wax-300 拓扑下的进化过程，可以看出 EGA 算法能够迅速收敛。

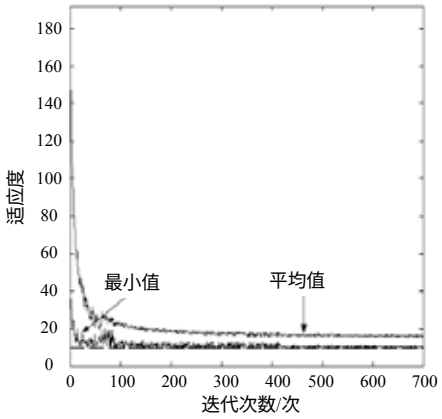


图 3 EGA 在 wax-300 中的进化过程

4 结束语

本文以最小化监测探针数目为目标，在链路覆盖和消息覆盖约束条件下，讨论了分布式网络性能监测的探针部署方法。该方法借助于遗传算法解决顶点覆盖问题的思想，将基于点编码的遗传算法应用到网络监测探针的部署问题中，并针对其引入无效解的缺点，在传统遗传算法的基础上提出了使用基于边编码的遗传算法解决监测探针部署问题。

实验表明基于遗传算法及其改进算法的网络性能监测探针部署方法比贪婪算法具有更好的性能，使用改进后的遗传算法能够得到更少的监测探针数目，并且收敛速度快。

参考文献

1 Khuri S, Bäck T. An Evolutionary Heuristic for the Minimum Vertex Cover Problem[M]//Hopf J. Saarbrcken: Genetic Algorithms Within the Framework of Evolutionary Computation. Max Planck Institut fr Informatik, 1994: 86-90.

2 朱培红. 基于移动多 Agent 的分布式网络性能监测的研究[D]. 武汉: 武汉大学, 2004-05.

3 Kyoungwon S, Yang Guo, Kurose J, et al. Locating Network Monitors: Complexity, Heuristics and Coverage[C]//Proc. of Infocom'05. 2005.

4 Calvert K, Doar M, Zegura E W. Modeling Internet Topology[J]. IEEE Communications Magazine, 1997, 35(6).

（上接第 93 页）

因此 TCP 吞吐量的测量对网络性能评估有重要意义。本文通过实验方式得出了背景流量类型（TCP、UDP 或二者共存）、竞争带宽资源的 TCP 连接数目对主动测量 TCP 吞吐量的影响，并通过实验区分了 TCP 吞吐量和网络可用带宽。

TCP 吞吐量测量工具 NetThruPut 产生的负载一般为测量时间与所得吞吐量值之积，对带宽较小的专用网络来说，测量工具本身产生的负载对网络运行有很大影响。因此 TCP 吞吐量不适合在线长期测量，工具的测量时间可以选择，一般为 10s 或 60s。

参考文献

1 Netperf: A Benchmark for Measuring Network Performance[Z]. 2005. http://www.netperf.org/netperf/ DownloadNetperf.html.

2 Blum R. 网络性能开源工具包[M]. 梁金昆, 译. 北京: 清华大学出版社, 2005.

3 潘亚军. TCP 吞吐量测量方法研究[D]. 北京: 首都师范大学, 2005.