计算具有不可靠结点分布式网络可靠度的一个因子分解算法*)

A Factoring Algorithm for Reliability Evaluating of Distributed Networks with Imperfect Nodes

孙艳蕊　崔立彦　张祥德

(东北大学理学院　沈阳110004)

Abstract This paper presents several reliability preserving reductions. By using these reductions and factoring theorem, an efficient algorithm was proposed to evaluate the distributed program reliability(DPR)of distributed networks with imperfect nodes. The time complexity of the algorithm is $O(N\cdot(|V|+|E|))$, where N is the total number of the nodes in the generated tree, $|V|$ and $|E|$ are the node and link number of the network, respectively. In order to show the efficiency of the algorithm, the DPRs of different networks were evaluated by computer experiments on Pentium 120 PC. The number of leaves in generated tree and overall computing time are much less than that of other algorithm.

Keywords Distributed network, Imperfect node, Distributed program reliability, Factoring theorem, Algorithm complexity

1 引言

随着计算机技术的迅速发展,计算机已在各个领域得到广泛的应用。越来越多的部门,象通讯、金融、国防、工业控制等领域,对计算机产生了很强的依赖性。这些系统的计算机一旦发生故障,将带来不可估量的损失。分布式网络以其可靠、坚固、快速响应、易于修改和扩充、资源共享等优点,而被广泛应用,其可靠度的计算成为人们关注的重要课题。目前,国内外学者对一般网络可靠度进行了较多的研究[1~4,8,9],但对于具有不可靠结点的分布式网络的分布程序可靠度(Distributed Program Reliability ,DPR)的研究还不多见[6,7]。所谓分布程序可靠度是指分布计算网络中一个给定的程序可以被成功实现的概率,一个程序被成功实现是指执行该程序所需要的所有数据文件已从网络的各结点处得到。文[6]给出一个可靠度算法,认为是目前计算分布程序可靠度比较有效的算法。其基本思想是用与源点关联的所有边将网络分成一系列较小的彼此不相交的网络,然后反复重复这个过程,直到找到满足条件的子网络或者失效的网络。将原始网络作为根点,产生的每个满足条件的子网络作为结点生成一事件树。算法的时间复杂性是$O(N\cdot|V|\cdot|E|)$,其中$|V|$和$|E|$分别表示网络的结点数与边数,N 表示事件树的叶点数。对于比较稠密的网络,与源点关联的边较多,将会产生较大的 N。本文利用因子分解定理[4],结合保持网络可靠度不变的缩简给出了一个计算 DPR 的因子分解算法。最后用 Pascal 语言编程对文[6]中的网络进行了计算,结果表明,本文算法产生的 N 比文[6]中的小得多,计算速度也快得多,尤其是对较为复杂的网络,更能显示算法的这一特点,因而本文算法更为有效。

在本文中假设网络的结点和边以一定的概率工作;整个网络及每个元件(结点和边)只有两个状态:正常工作或失效;整个网络及每个元件的工作状态在统计上是相互独立的。

2 记号

这部分给出一些记号:

s: program 所在的结点,称为网络的源点。

$G(V,E,FM)$: 结点集合为 V,边集合为 E,完成 program 所需的所有数据文件(data-file)集合为 FM 的无向分布计算网络,简记为 G(在不引起混淆的情况下 G 也记作一般网络)。

$|V|,|E|$: 分别表示网络的结点数与边数。

$1,2,\cdots,|V|$:表示网络的结点。

$p_i,(q_i)$:网络的元件 i(边或结点)工作(失效)的概率,$p_i+q_i=1$。

(u,v):结点为 u 和 v 的边。

FM:在 G 中完成 program 所必需的所有数据文件构成的集合。

$G*e$:在 G 中融合结点 v 进入结点 u,这里 $e=(u,v)$。

$G-e$:从 G 中删除边 e。

$deg(u)$:表示结点 u 的度数,即网络中与 u 关联的边数。

$N[v]$:与结点 v 关联的所有结点构成的集合,称为结点 v 的邻点集合。

$F[v]$:结点 v 处分布的数据文件构成的集合。

$DPR(G)$: 网络 G 的分布程序可靠度。

$R(G)$:网络 G 的一般可靠度。

3 因子分解与网络的缩减

3.1 因子分解公式(因子定理)[4]

计算具有可靠结点的网络 G 的可靠度的因子分解算法就是反复应用下述因子分解公式:

$$R(G)=p_eR(G*e)+q_eR(G-e) \quad (1)$$

在每次因子分解后,对网络 $G*e$ 和 $G-e$ 进行可靠度保护缩减,这样循环下去,直到缩减的网络可靠度为1或0为止(即条件已得到满足或条件已不能被满足,例如,求 s-t 两终端可靠度时,当缩减的网络中的汇点 t 已融合到源点 s 或 s 与 t 已不连通时,对应缩减网络的可靠度分别为1或0),对无向网络的每条边因子定理都成立。对于具有可靠结点的一般的无向网络,因子分解算法是计算其可靠度的一类非常有效的方法。对

*)国家自然科学基金资助项目(编号:19701006)。

于具有不可靠结点的无向网络，由于边 $e=(u,v)$ 失效有几种可能：一是边失效，一是结点失效，或者边与结点同时失效，因此因子分解公式改变如下：

$$R(G)=p_A R(G|A\text{ 发生})+(1-p_A)R(G|A\text{ 不发生}) \quad (2)$$

其中 A 表示边 e 和结点 u,v 同时运行的事件，p_A 表示事件 A 发生的概率。

具有不可靠结点的分布式网络，求其分布程序可靠度时，若程序所在的点 s 失效，则 $DPR(G)=0$，不妨设源点 s 永远正常工作，应用公式(2)，我们有：

$$DPR(G)=p_s[p_e p_v DPR(G*e)+(1-p_e p_v)DPR(G-e)] \quad (3)$$

其中 e 是端点为 s 和 v 的边，在网络 $G-e$ 中的结点 v 的可靠度 p_v 用 $p'_v=p_v q_e/(q_v+p_v q_e)$ 代替。

3.2 网络缩减

一般地，因子分解算法都结合网络的缩减，下面我们给出分布式网络中1-度点、2-度点、平行边的保持网络可靠度不变的缩减。

·1-度点缩减　设 $deg(u)=1$，若 v 与 u 关联，即 $v\in N(u)$：

1)如果在结点 u 处分布的数据文件不是完成程序所需要的，即 $F(u)\cap FM=\varnothing$，则删除结点 u 及与之关联的边，网络的分布程序可靠度不变；

2)若 $F(u)\subseteq F(v)$，则删除结点 u 及与之关联的边，网络的分布程序可靠度不变；

3)若 $F(u)\cap FM\neq\varnothing$，且在 $F(u)\cap FM$ 中的数据文件至少有一个在其它结点处分布的数据文件集合中不存在，则去掉结点 u 及与之关联的边，同时将 u 点处分布的数据文件加入到 v 点处分布的数据文件集合中，则 DPR(G)=$p_v p_e$*DPR($G-e$)，其中 $e=(u,v)$。

·2-度点缩减　设 $deg(u)=2$，v,w 是 u 的两个邻点，即 $v,w\in N(u)$，e_1,e_2 为与 u 关联的两条边，则当分布在 u 点处的数据文件同时包含在分布在 v,w 点处的数据文件集合中，即 $F(u)\subseteq F(v)\cap F(w)$ 时，有：DPR(G)=DPR(G')，其中 G' 是 G 删除结点 u 及与之关联的边 e_1 和 e_2 后，在 v,w 之间增加一条可靠度为 $p_{e1}p_{e2}p_v$ 的边 e 所得到的网络。

·平行边的缩简　设 e_1,e_2 是和 u,v 关联的两条边，用可靠度为 $1-q_{e1}q_{e2}$ 的边 e 代替 e_1,e_2，网络的可靠度不变。

4 算法及其复杂性

这部分利用公式(3)，结合保持可靠度不变的缩简，给出一个算法并讨论其复杂性。算法的基本思想是选一个与源点关联的边 $e=(s,v)$，应用公式(3)分解成两个较小网络：$G-e$ 和 $G*e$。对于网络 $G-e$，若 s 与其它结点不连通或其它各结点处分布的数据文件已不能满足完成程序所需时，则停止对 $G-e$ 的分解，此时 DPR($G-e$)=0，否则对网络 $G-e$ 继续应用公式(3)；对于网络 $G*e$，完成程序所需的数据文件集合是 $FM-F[v]$，仍记为 FM，当 $FM=\varnothing$ 时，说明程序已成功实现，停止对 $G*e$ 的分解，此时 DPR($G*e$)=1，否则对网络 $G*e$ 继续应用公式(3)。如此下去，直到由公式(3)可以计算出网络的 DPR 为止。算法如下：

```
开始
输入：无向分布计算网络 G，各个结点及边正常工作的概率，program
      所在的点 s，完成 program 所需的数据文件集合 FM，及各个结
      点处分布的数据文件集合。
输出：分布程序可靠度 DPR(G)*p。
结束
```

```
Function DPR(G)
begin{DPR(G)}
  {修正完成 program 所需的数据文件集合 FM，及各个结点处分布
的数据文件集合}
  FM←FM-F[s]; F[s]←∅;
  For i=2 to|V|
  F[i]←F[i]∩FM;
  if FM=∅ then return(1)
      else
    begin
      查找1-度点;
      if 1-度点找到，记为 v，then degree1(v);
        return(Ω * DPR(G));
        查找2-度点;
        if 2-度点找到，记为 v，then degree2(v);
        if FM=∅ then return(1)
        else{应用因子分解定理}
          begin
            从 s 的邻点集合 N[s]中选取满足|F[v]|= max_{v_i∈N[s]} (|F
            [v_i]|)的结点 v；若|F[u_1]|=|F[u_2]|= max_{v_i∈N[s]} (|F[v_i]
            |)，则 u_1,u_2 中任取一点；
            从 G 中去掉边 e=(s,v);
            if 去掉该边后使 s 与网络其它部分不连通 then
            P_1←0
            else
            begin
              if deg(v)=1 then degree1(v);
              if deg(v)=2 then degree2(v);
              P_1←(1-p_e p_v) * DPR(G-e); p_v←p_v q_e/(q_v+p_v q_e)
            end;
            将 v 融合到结点 s; FM∅FM-F[v]
            If FM=∅ then
              P_2←p_e p_v; return(P_1+P_2)
            else
            begin
              对于 G 的每个结点 i, F[i]←F[i]∩FM;
              if G 存在平行边，then 进行平行边的缩简;
              P_2←p_e p_v * DPR(G * e); return(P_1+P_2)
            end
        end
end.
Procedure degree1(v)
begin
  求与 v 关联的结点 u;
  if F(v)∩FM=∅，then 删除结点 v 及与之关联的边;Ω←1;
  if F(v)⊆F(u)，then 删除结点 v 及与之关联的边;Ω←1;
  if F(v)∩FM≠∅ and F(v)∩FM 中的数据文件至少有一个在其
    它结点处分布的数据文件集合中不存在 then
    begin
      去掉结点 v 及与之关联的边
      F[u]←F[u]∪F[v];G←G-e 其中 e=(u,v);Ω←p_v p_e
    end
end.
Procedure degree2(u)
begin
  求与 u 关联的结点 v,w,e_1,e_2 为与 u 关联的两条边;
  if F(u)⊆F(v)∩F(w)then
    删除 u 点及与之关联的边 e_1 和 e_2，增加可靠度为 p_u p_e1 p_e2 的边 wv
    得到网络 G'
    DPR(G)←DPR(G')
end.
```

算法的实现实际上是一个二叉树(生成树)的生成过程。树的根点为 G，对 G 进行因子分解后得到的两个子网络 $G*e$ 和 $G-e$，这里 $e=(s,v)$。缩简后分别作为 G 的左孩子和右孩子(即二叉树的左右两个分枝或结点)，设每次因子分解收缩边得到的子网络总作为左孩子，并且给其父与左、右孩子之间的边赋权为 $p_v p_e$ 和 $1-p_v p_e$；若子网络的可靠度是1或0，则对应的分枝成为叶点。将每个左叶点到根点的路径中各边的权作乘积，然后再将它们加到一起乘上源点的可靠度即得到网络 G 的分布程序可靠度。

算法每次更新数据文件所需的时间至多是 $|V|$；每次选进行因子分解的边所需时间至多是 $O(|V|)$；求因子分解得到的子网络所需时间最多需 $O(|V|+|E|)$(算法中网络是用各点的邻接表表示的)；1-度结点、2-度结点及平行边的缩简在常数时间内完成；因此计算二叉树中每个叶点对应的 DPR 中的每一项所需时间不超过 $O(|V|+|E|)$。设生成的二叉树的叶点数为 N，那么算法的时间复杂度是 $O(N\cdot(|V|+|E$

|))。

利用因子分解定理计算网络可靠度时，算法的复杂性与二叉树中产生的叶点数目 N 有关，但叶点数 N 和计算时间取决于因子分解过程中边的选取。对于具有可靠结点网络可靠度的计算，文[5,8]提出了选边的优化策略，这些选边策略取决于算法所采用的网络化简类型，以及网络本身的结构和性质。一般选边的优化策略都比较复杂，虽然优化选边会使二叉树的叶点数目减少，但要花费很多时间，从理论上考虑，采用选边的优化策略可能使复杂性降低，但计算效率并不一定提高[3]。由于本文计算的是分布式网络可靠度，我们给出的选边原则并没有从网络结构本身考虑而是从各结点处分布的数据文件考虑的。本文采用的选边方法是在 s 的邻点集合中选取具有下述性质的结点 v：在该点处分布的数据文件集合的模最大，若这样的结点不止一个，则任取其中一点作为 v，这样 $e=(s,v)$ 就是用于因子分解的边。按照对 $G-e$ 和 $G*e$ 停止分解的原则可知，利用我们的选边方法生成的二叉树的叶点数不多于任意选边法生成的二叉树中的叶点数。

上述算法可以推广到计算具有不可靠结点的分布式网络的其它可靠度。

5 计算结果与讨论

利用本文算法，采用 Pascal 语言编程，在 Pentium 120计算机上对文[6]中的8个基准网络 $G_{i,j}$ 和 $G_{10,10}$ 进行了计算。网络 $G_{i,j}(j\leqslant i)$ 就是有 i 个结点，且结点1,2,…,j 构成一个完全图的网络。作为例子图1给出了网络 $G_{8,6}$。

5.1 计算结果

设 program 放置在结点1处，实现 program 所需的数据文件集合为 $\{F_1,F_3,F_5\}$，各个结点处分布的数据文件如表1，运行的结果如表2，其中 E_t 和 P_t 分别表示利用 *ENR/KW* 算法[6]和本文算法计算 DPR 所需的全部计算时间；E_t，P_t 分别表示利用 *ENR/KW* 算法和本文算法计算 DPR 所产生的事件树和生成树的叶点数目。

表1 各结点的数据文件分布表

结点	数据文件集合	结点	数据文件集合
1	F_1, F_2, F_3	6	F_6, F_7, F_8
2	F_2, F_3, F_4	7	F_1, F_7, F_8
3	F_3, F_4, F_5	8	F_1, F_2, F_8
4	F_4, F_5, F_6	9	F_3, F_7, F_8
5	F_5, F_6, F_7	10	F_1, F_4, F_7

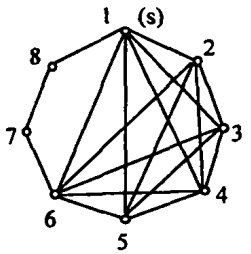

图1 网络 $G_{8,6}$

表2 利用本文算法和文[6]中的算法计算 DPR 的运行结果

网络	E_t	P_t	本文循环次数	E_t(微秒)	P_t(微秒)	P_t/E_t	*DPR*
$G_{8,4}$	16	5	6	5	—		0.89155135
$G_{10,4}$	20	5	6	14	—		0.88935531
$G_{8,6}$	72	15	19	15	5	0.33	0.89889609
$G_{8,7}$	289	48	63	55	20	0.36	0.89906110
$G_{10,7}$	462	48	63	148	20	0.14	0.89905679
$G_{8,8}$	1196	195	259	222	90	0.41	0.89908997
$G_{10,8}$	2556	195	259	817	91	0.11	0.89909191
$G_{10,9}$	17832	900	1233	5640	442	0.08	0.89909852
$G_{10,10}$		5871	7827		2830		0.89909963

注：'—'程序运行结果显示'0'微秒；循环次数是指算法应用因子定理的次数。

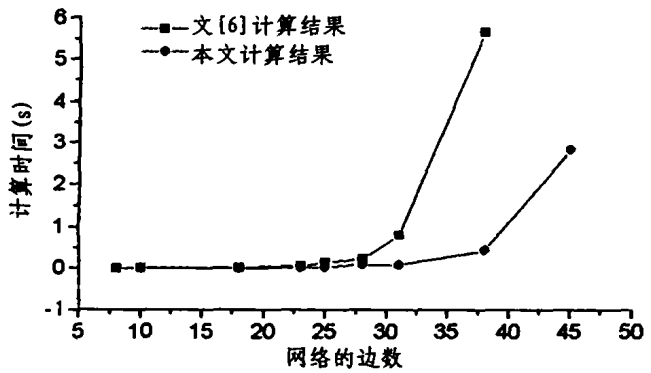

图2 本文与文[6]算法的计算结果比较

5.2 讨论

表2为利用本文算法与文[6]中的算法 ENR/KW 计算 DPR 的结果，通过比较可知本文算法具有如下优点：

1)利用本文算法计算 DPR 时所产生的叶点数少。由于两个算法的时间复杂性都与叶点数成正比，因此本文算法的时间复杂性也要小得多。

2)运算时间比文[6]小得多。图2是两种算法所需计算时间的对比图，可以直观地反映出利用本文算法和算法 ENR/KW 计算 $G_{i,j}$ 网络的 DPR 所需要的时间随网络边数变化的趋势。由图2及表2中的 P_t/E_t 可以看出，随着网络边数及复杂性的增加，本文算法所用计算时间增长的速率明显低于 ENR/KW 算法，尤其是边数大于30以后，这一变化趋势更为明显。

参考文献

1 Rai S, Veeraraghavan M, Trivedi K S. A survey of efficient reliability computation using disjoint products approach. Networks, 1995, 25: 146～163

2 Luo T, Trivedi K S. An improved algorithm for coherent-system reliability. IEEE Trans Reliability, 1998, 47(1): 73～78

3 Page L B, Perry J E. A practical implementation of the factoring theorem for network reliability. IEEE Trans Reliability, 1988, 37(3): 259～267

(下转第71页)

5)适应度函数:由(9)式约束下的(8)式确定。

实验中,取约束条件值 m 为10;问题规模 n 从10到100以增量10变化;w_{ij}取0-500之间的均匀分布随机数。实验结果如表1所示:

表1 快速 BMDA 求解多约束背包问题性能

问题规模(n)	实际最优值	算法求解值	BMDA 运行时间(秒)	快速 BMDA 运行时间(秒)
10	2040	2040	0.49	0.28
20	2851	2851	7.69	5.17
30	3040	3040	13.08	3.79
40	4465	4465	22.96	6.37
50	6729	6729	53.72	15.16
60	8206	8206	53.06	53.17
70	9437	9437	111.23	96.95
80	10513	10513	190.65	110.22
90	12062	12062	317.57	126.33
100	13179	12775	362.18	165.22

从表1可知,快速 BMDA 求解此问题具有很好的性能,除问题规模为100外,其它情况下,算法都得到了最优解,与文[10]利用改进的遗传算法求解多约束问题结果比较,本文得到了更好的结果。说明了利用 BMDA 概率模型算法求解此类问题的精确性。

另一种评价进化算法求解问题性能的指标是算法执行过程所经历的函数评价次数[1],这种评价比简单地给出算法经历的进化代数更有说服力,因为它综合考虑了群体规模和进化代数两种因素。图1给出了快速 BMDA 求解问题规模和函数评价次数之间的关系,可以看出,随着问题规模的增加,函数评价次数基本上呈一种线性关系,而不是一种指数关系。

表1给出了快速 BMDA 和 BMDA 在相同硬件环境下求解问题的速度,采用快速 BMDA 显著减少了求解问题的计算复杂性,从而提高了算法的运行速度。应该指出这种时间的统计只是针对两种算法同等条件进行比较,因此只具有相对比较意义,由于不同的硬件和软件条件,难以和其它算法比较。

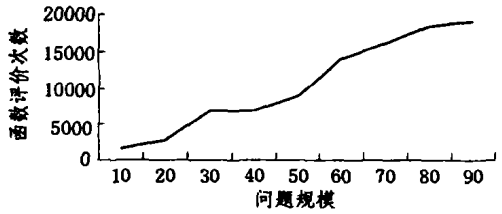

图1 快速 BMDA 求解多约束背包问题的函数评价次数

结束语 针对基本遗传算法存在的连锁问题,本文讨论了基于概率分析模型的遗传算法,这种算法的特点是把自然进化算法和构造性数学分析方法结合,以指导对问题空间的有效搜索。本文结合 BMDA 算法讨论了二阶概率分析进化算法的快速计算问题,提出了一种快速 BMDA 算法,文中分析了快速计算的原理,并利用这种快速算法对多约束背包问题进行了求解,结果表明,该算法具有胜任进化算法所要求的精确性和快速性。本文认为在这一方面还有许多值得进一步探讨的问题,包括研究高阶概率进化算法的快速计算问题;研究如何将现有的一些降阶算法引入概率分析进化算法,使得可以利用低阶概率分析进化算法求解高阶问题,达到快速、精确和可靠地求解问题的目的。

参考文献

1 Muehlenbein H. From recombination of genes to the estimation of distributions I. Binary parameters. Parallel Problem Solving from Nature, 1996, 4: 178～187

2 Baluia S. Fast probabilistic models for combinatorial optimization. In: Proc. of the 15th National Conf. on Artificial Intelligence (AAA-98) AAA Press, 1998. 469～476

3 Pelikan M, Muehlenbein H. Marginal distributions in evolutionary algorithms. http://www-illigal.ge.uiuc.edu/~Pelikan/

4 林亚平. 概率进化算法及其研究进展. 计算机研究与发展, 2001, 38(1): 43～49

5 Pelikan M. The bivariate marginal distribution algorithm. Advanced in soft computing-Engineering Design and Manufacturing, London: Springer-Verlag, 1998. 521～535

6 Ackley D H. A Connectionist Machine for Genetic Hill climbing. Boston: Kluwer Academic, 1987

7 Bonet J D, et al. MIMIC: Finding Optima by Estimating Probability Densities, Advances in Neural Information Processing System. The MIT Press, Cambridge, 1997

8 Baluia S, Davies S. Using optimal dependency-trees for combinatorial optimization learning the structure of the search space. In: Proc of the 14th Intl. Conf. on Machine Learning. Morgan Kaufmann, 1997. 30～38

9 Pelikan M, Goldberg D E. BOA: The Bayesian Optimization Algorithm: [IlliGAL Report No. 98013]. UIUC, IlliGAL Genetic Algorithm Aboratory, 1998

10 Lin F T, et al. Applying the Genetic Approach to Simulated Annealing in Solving Some NP-Hard Programs. IEEE Trans. on Systems, MAN and Cybernetics, 1993, 23(6): 1752～1767

11 林亚平, 杨小林. 快速概率分析进化算法及其性能研究. 电子学报, 2001, 29(2): 178～181

(上接第113页)

4 Page L B, Perry J E. Reliability of directed networks using the factoring theorem. IEEE Trans Reliability, 1989, 39(5): 556～561

5 Satyanarayana A, Chang M K. Network reliability and factoring theorem. Networks, 1983, 13: 107～120

6 Ke W J, Wang S D. Reliability evaluation for Distributed Computing Networks with imperfect nodes. IEEE Trans on Reliability, 1997, 46(3): 342～349

7 Kumer A, Agrawal D P. A generalized algorithm for evaluating distributed program reliability. IEEE Trans Reliability, 1993, 42(3): 416～426

8 Wood R K, Factoring algorithm for computing K-terminal network reliability. IEEE Trans Reliability, 1986, 35(3): 269～278

9 Factoring and reductions for networks with imperfect vertices. IEEE Trans Reliability. 1991, 40(2): 210～217