维普资讯 http://www.cqvip.com

网络最大流问题求解的符号 ADD 增广路径算法*)

徐周波　古天龙　赵岭忠
(桂林电子工业学院计算机系　桂林 541004)

摘　要　本文通过对网络及网络最大流问题的符号代数判定图(ADD)描述,将网络中的结点和边用 ADD 隐式表示,并利用 Gabow 的容量变尺度算法的主要思想,将一般网络最大流问题化为一系列的单位容量网络最大流问题,结合 Hachtel 等的单位容量网络最大流问题的求解算法,给出了网络最大流问题求解的符号 ADD 增广路径算法,简称为符号 ADD 算法。与 Dinic 算法、Karzanov 算法相比,本文算法的空间复杂度得到了改善。实验结果表明,本文算法是切实有效的,且可处理更大规模的问题。

关键词　符号算法,最大流,代数判定图(ADD),剩余网络

An Augmenting-Path-Based Symbolic ADD Algorithm for Maximum Flow in Networks

XU Zhou-Bo　GU Tian-Long　ZHAO Ling-Zhong
(School of Computer Science, Guilin University of Electronic Technology, Guilin 541004)

Abstract　In this paper, the augmenting-path-based symbolic ADD (Algebraic Decision Diagram) algorithm for maximum flow in networks is proposed. In the algorithm, the network and the maximum flow problem are formulated via ADD (Algebraic Decision Diagram), and Hachtels' symbolic algorithm for maximum flow in unit capacity networks is integrated with Gabow's scaling algorithm to transfer the general problem into a sequence of maximum flow problem in unit capacity network. The simulation results show that the novel symbolic algorithm can improve the space complexity, compared with Dinic's algorithm and Karzanov's algorithm, and can be used to handle larger-scale general network flow problems.

Keywords　Symbolic algorithms, Maximum flow, Algebraic decision diagram (ADD), Residual network

1　引言

最大流问题是典型的组合优化问题,已广泛应用于电力、交通、通信、计算机网络等工程领域和物理、化学等科学领域。随着问题规模的逐步增大,传统的最大流问题的求解算法,如 Dinic 算法[1]、Karzanov 算法[2]等,难免要涉及状态空间或者变量组合的显式枚举,使最大流问题的求解受到了组合爆炸复杂性的制约。符号算法是通过引入额外的布尔变量(也称为符号)来表示各种对象之间的一种关系,其泛指基于各种判定图—二叉判定图(BDD－Binary Decision Diagram)[3]及其扩展形式的问题求解算法,它是一种有效的图形、数学描述技术,其优点是能够以较小的空间存储较大规模的有向图。基于 BDD 的符号算法已广泛应用于逻辑综合、形式化硬件验证、电路测试等领域。Hachtel 等在 Dinic 算法的基础上,利用 OBDD 来表示单位容量网络和描述网络最大流问题,给出了单位容量网络最大流问题的符号 OBDD 求解算法[4]。该算法能够比常规最大流算法处理更大规模的问题,在一定程度上缓减了状态爆炸问题。但算法仅局限于求解单位容量网络的最大流。代数判定图(ADD－Algebraic Decision Diagram)是 BDD 的一种扩展形式[5],其是表示和操作伪布尔函数的有力工具。基于 ADD 的符号算法已成功地应用于矩阵乘[5]、最短路径计算[5,6]和组合电路的时间分析[7]等领域。

基于此,本文通过对单位容量网络最大流问题的符号 OBDD 算法及 ADD 做了进一步的研究,给出了一种基于增广路径的网络最大流问题的符号 ADD 求解算法。与 Dinic 算法和 Karzanov 算法相比,本文算法的空间复杂度得到了改善。实验结果表明,本文算法是切实有效的,且可比 Dinic 算法、Karzanov 算法处理更大规模的问题。

2　预备知识

2.1　最大流问题

定义 1　流网络(下面简称网络)N 是一个四元组(G,s,t,C),其中:G 是一个有向加权图,$G=(V,A)$,$|V|$表示 G 的结点数,$|A|$表示 G 的弧数;$s\in V$ 为源点;$t\in V$ 为终点,除了 s 和 t 外,V 中的其它顶点称为中间点;$C:A\to R^+$(非负实数)为弧上的权函数,弧$(i,j)\in A$ 上的权 $C(i,j)$称为容量,记为 c_{ij},并记最大容量为 $U(=\max\{c_{ij}\mid(i,j)\in A\})$。

定义 2　网络 $N=(G,s,t,C)$上的一个流 f 是指从 N 的弧集 A 到 R^+的一个函数,即对任意弧$(i,j)\in A$,赋予一个非负实数 $f(i,j)$,称 $f(i,j)$为弧(i,j)上的流量,记为 f_{ij}。若流 f 满足如下两式:

$$\sum_{\{j\mid(i,j)\in A\}} f_{ij}=\sum_{\{j\mid(j,i)\in A\}} f_{ji},\ \forall i\in V-\{s,t\} \tag{1}$$

$$0\leqslant f_{ij}\leqslant c_{ij},\ \forall(i,j)\in A \tag{2}$$

则称 f 为可行流。其中等式(1)表示在网络的中间点处保持

*)本文工作得到国家自然科学基金(60243002)、教育部留学归国人员基金及广西自然科学基金(0448072)的资助。徐周波　硕士研究生,主要研究领域为符号调度技术、符号模型检验、软件测试。古天龙　教授、博士生导师、博士,主要研究领域为形式化方法、符号调度技术、符号模型检验、Petri 网,离散事件/混杂系统理论及应用等。赵岭忠　博士研究生,主要研究领域为形式化技术、符号算法。

维普资讯 http://www.cqvip.com

流守恒，称为流量守恒条件，不等式(2)称为容量约束。流出源点 s 的净流即为流 f 的值 $|f|$，即：

$$|f|=\sum_{\{i|(s,i)\in A\}}f(s,i)-\sum_{\{i|(i,s)\in A\}}f(i,s)$$

最大流即流值(流量)最大的可行流。

对于网络 $N=(G,s,t,C)$，网络最大流问题就是找一个网络 N 上流值最大的可行流，即找一个流 $\{f_{ij}\}$，使得流 f 的流值达到最大，并满足上述的式(1)和式(2)条件。

定义 3 设 f 为网络 $N=(G,s,t,C)$ 上的一个可行流。网络 N 关于流 f 的剩余网络 $N(f)=(G(f)=(V,A(f)),s,t,C(f))$，其中：

$$A(f)=\{(i,j)|(i,j)\in A,f_{ij}<c_{ij}\}\cup\{(i,j)|(j,i)\in A,f_{ji}>0\},$$

$$c_{ij}(f)=\begin{cases}c_{ij}-f_{ij}, & (i,j)\in A,f_{ij}<c_{ij},\\ f_{ji}, & (j,i)\in A,f_{ji}>0\end{cases}$$

其中 $c_{ij}(f)$ 称为边 (i,j) 的剩余容量；Δ—剩余网络 $G(f)$ 是指由剩余容量不小于 Δ 的边组成的网络。

定义 4 层次网 $N_L(V_L,E_L,s,t,c)$[1]，若它满足：①层次网上的每一个结点都在从源点 s 到终点 t 的路径上；②对每个结点 v，射入结点 v 的入度不大于结点 v 的出度；那么称该层次网为正层次网。

2.2 数判定图(ADDs)

定义 5 一个 ADD 就是表示基于变量 $x_1,x_2,\cdots,x_n$ 的一簇伪布尔函数 $f_i:\{0,1\}^n\to S$ 的有一个直接根结点的有向无环图 $G(V\cup T,E)$，它满足：

(1) S 为 ADD 代数结构的有限值域，且 $S\subseteq Z$(整数集合)；

(2)没有射出边的结点称为终结点，终结点集合记为 T。对 $\forall t\in T$，均被标识为值域 S 中的一个元素 $s(t)$。

(3)除终结点外的其它结点称作内结点，记内结点的集合为 V。对于 $\forall v\in V$，均由变量名 $var(v)$ 标识，且有两条射出边。v 取值 0 后的射出边称为 0-边，v 取值 1 后的射出边称为 1-边。所有边的集合记为 E。0-边和 1-边所指向的结点分别记为 $low(v)$ 和 $high(v)$。

(4)图中的每一个结点对应唯一一个函数 f_i。

(5)在 ADD 的有向路径上，每个变量至少出现一次。

(6)在给定变量序 $\pi:x_1<x_2<\cdots<x_n$ 下，在每条有向路径上都必须以该变量次序出现，也就是说若 x_i 在 x_j 前，则不存在一条路径，使得 x_j 排在 x_i 前。

在图形中一般用圆圈表示内结点，用方框表示终结点。通常，假设边的方向向下，0-边用虚线表示，1-边用实线表示。

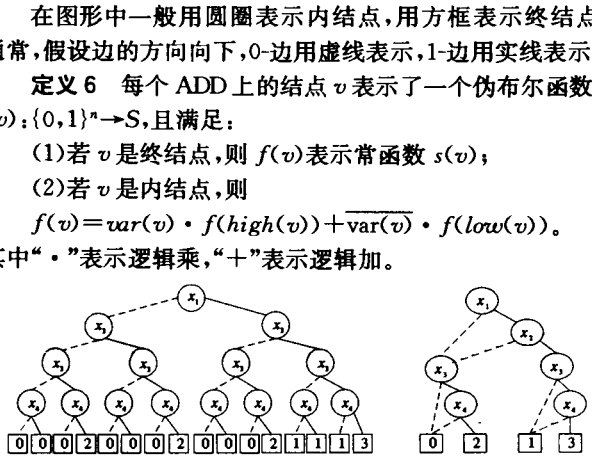
定义 6 每个 ADD 上的结点 v 表示了一个伪布尔函数 $f(v):\{0,1\}^n\to S$，且满足：

(1)若 v 是终结点，则 $f(v)$ 表示常函数 $s(v)$；

(2)若 v 是内结点，则

$f(v)=var(v)\cdot f(high(v))+\overline{var(v)}\cdot f(low(v))$。

其中“·”表示逻辑乘，“+”表示逻辑加。



(a) 完全二叉树　　(b) ADD

图 1　伪布尔函数 $f=x_1x_2+2x_3x_4$ 的表示

由此可见，对于变量的一组赋值，所得到的函数值由根结点到一个终端点的一条路径决定。这条路径所对应的分支由变量的这组赋值来决定，该分支的终结点所标识的值就是变量在这组赋值下所对应的函数值。不同的变量序的选择对于 ADD 的规模有着极大的影响，因此在求伪布尔函数的 ADD 时，必须说明变量的顺序。

例：$f=x_1x_2+2x_3x_4$，其中 $x_1<x_2<x_3<x_4$，它的完全二叉树及 ADD 表示如图 1 所示。

由图 1 可见，用 ADD 表示伪布尔函数 $f=x_1x_2+2x_3x_4$ 只需要 10 个结点，占用空间较小，这就便于在函数上进行其它操作。

3 网络最大流问题的符号 ADD 算法

下面通过对网络及网络最大流问题的伪布尔函数描述，将状态空间转化为 ADD 表示，并利用 Gabow 的容量变尺度技术[8]，将一般网络最大流问题化为一系列的单位容量网络最大流问题，在此基础上，利用 Hachtel 等提出的单位容量网络最大流问题的求解算法来求解最大流。

3.1 网络最大流问题的符号 ADD 描述

在进行网络最大流问题的符号 ADD 算法求解时，网络 $N=(G,s,t,C)$ 和最大流问题都是用伪布尔公式来表示的，而伪布尔公式在计算机内用 ADD 表示，于是网络及最大流问题被隐式的表示。

将网络 $N=(G,s,t,C)$ 的顶点进行 $n(=\lceil\log_2|V|\rceil)$ 位长的二进制编码，设编码后，$x=(x_0,\cdots,x_{n-1})$，$y=(y_0,\cdots,y_{n-1})$，其中 $x_i,y_i\in\{0,1\}(i=0,\cdots,n-1)$，那么网络 N 的 $C:A\to Z$ 即可由伪布尔函数 $E(x,y):\{0,1\}^n\times\{0,1\}^n\to Z$ 表示，且有：

$$E(x,y)=\begin{cases}c_{xy}xy,若(x,y)\in A\\ 0,否则\end{cases}$$

其中 c_{xy} 为网络 N 中弧 (x,y) 的容量。由此可见，网络 N 的符号 ADD 表示为 $N(E(x,y),s(x),t(y))$。

如图 2 所示，初始流为零的网络 $N=(G,s,t,C)$，其的符号 ADD 表示如图 2(c)所示。

类似地，网络 N=(G,s,t,C)上的流 f 可由伪布尔函数 F(x, y):$\{0,1\}^n\times\{0,1\}^n\to Z$ 表示，且有：

$$F(x,y)=\begin{cases}f_{xy}xy,若(x,y)\in A\\ 0,否则\end{cases}$$

其中 f_{xy} 为图 G 中弧 (x,y) 的流量。

基于上述对网络 $N=(G,s,t,C)$ 的符号 ADD 描述，2.1 节中所描述的网络最大流问题的符号 ADD 描述为：

$$Max\{\backslash_{xy}^{+}Ite(s(x),F(x,y),0)\}\qquad(3)$$

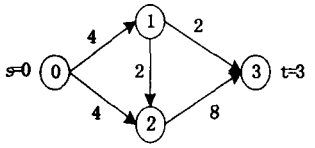
并满足如下两式：

$$\backslash_{x}^{+}f(x,y)=\backslash_{z}^{+}F(y,z),\forall y\in V(y)-\{s(y),t(y)\}\qquad(4)$$

$$F(x,y)\leqslant E(x,y)\qquad(5)$$

其中 Ite，Abstraction(在上文由\表示)为 ADD 的一些操作，此外还有 Apply 操作，详细请参见文[5]。公式(3)指求解从源点 $s(x)$ 流出的流值最大的可行流。公式(4)为流量守恒条件，其中 $F(y,z)$ 表示起点以 y 表示，终点以 $z=(z_0,\cdots,z_{n-1})$ $(z_i\in\{0,1\},i=0,\cdots,n-1)$ 为表示的弧上的流。公式(5)为容量约束条件。

维普资讯 http://www.cqvip.com

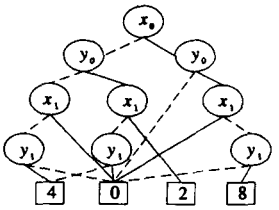


结点编码：
0：00
1：01
2：10
3：11

(a) 初始流为零的网络 N

	00	01	10	11
00	0	4	4	0
01	0	0	2	2
10	0	0	0	8
11	0	0	0	0

(b) 网络 N 的邻接矩阵



(c) 图 N 的 ADD 表示

图 2　网络及其 ADD 表示

3.2　递归构造 Δ_i-剩余网络

根据 3.1 节中网络最大流问题的符号 ADD 描述，求得整个网络所对应的 ADD 表示 $N(E(x,y),s(x),t(y))$。

设 $F_0(x,y)$为网络 $N(E(x,y),s(x),t(y))$上的初始流。在 N 中递归构造 Δ_i-剩余网络 $N_i=(E_i(x,y),s(x),t(y))(i=1,\cdots,\lfloor\log_2 U\rfloor+1)$，其中 $E_i(x,y)$为：

$$P(x,y)=E(x,y)-\sum_{j=0}^{i-1}F_j(x,y)+\sum_{j=0}^{i-1}F_j(y,x) \tag{6}$$

$$E_i(x,y)=\begin{cases}P(x,y),P(x,y)\geqslant\Delta_i=2^{\lfloor\log_2 U\rfloor-i+1}\\0,\text{否则}\end{cases} \tag{7}$$

其中 $F_i(x,y)((i=1,\cdots,\lfloor\log_2 U\rfloor+1))$为网络 N_i 上的最大流。根据 Gabow 的容量变尺度技术[8]，在 N_i 中的每条增广路径最多可增广 Δ_i 的流。由 Δ_i-剩余网络的特性，其上每一条边的容量均大于等于 Δ_i。现对 Δ_i-剩余网络 N_i 进行如下操作，将边容量大于等于 Δ_i 的边容量规划为 1。由此即得到 Δ_i-剩余网络 N_i 的单位容量网络。所以只要求得 N_i 的单位容量网络的最大流 $f_i(x,y)$，就可求得 Δ_i-剩余网络 N_i 的最大流 $F_i(x,y)=f_i(x,y)\cdot\Delta_i$。而整个网络 N 上的最大流就等于这些 N_i 上的流的叠加。

设 $F'(x,y)$为由 Δ_m- 剩余网络 $N_m(m=0,\cdots,i;i<\lfloor\log_2 U\rfloor+1)$上的最大流 $F_m(x,y)$叠加而得的流。由 Δ-剩余网络的定义易知，Δ_{i+1}-剩余网络 N_{i+1} 中的边由以下两部分组成：一部分为 $F'(y,x)$，另一部分为在构造 N_i 后 $E(x,y)$中尚未使用的容量大于 Δ_{i+1}的边 $A_{i+1}(x,y)$。由此可见可充分利用现有的计算结果 $F'(x,y)$来提高算法的效率，故 Δ_{i+1}-剩余网络 N_{i+1}可表示为 $N_{i+1}=(A_{i+1}(x,y)+F'(y,x),s(x),t(y))$。

由公式(6)(7)可见，N_i 上的最大流 $F_i(x,y)\leqslant E_i(x,y)$。设 $E_{i-1}'(x,y)$为在 N_{i-1}中求得最大流 $F_{i-1}(x,y)$后 $E(x,y)$中尚未使用的容量大于 0 的边，可见在原网络 N 中尚未使用的容量大于 0 的边 $E'_0(x,y)=E(x,y)$，则在构造 N_i 后 $E(x,y)$中尚未使用的容量大于 0 的边 $R_i(x,y)=E'_{i-1}(x,y)-E_i(x,y)$，而在 $E_i(x,y)$中推入最大流 $F_i(x,y)$后，在 $E_i(x,y)$中尚未使用的容量大于 0 的边 $R'_i(x,y)=E_i(x,y)-(F_i(x,y)-F_i(x,y)\cdot F'(y,x))$。可见在 N_i 上求得最大流 $F_i(x,y)$后，$E(x,y)$中尚未使用的容量大于 0 的边 $R(x,y)=R_i(x,y)+R'_i(x,y)=E'_{i-1}(x,y)-F_i(x,y)+F_i(x,y)\cdot F'(y,x)$。故在 N 中递归构造 Δ_{i+1}-剩余网络 $N_{i+1}(A_{i+1}(x,y)+F'(y,x),s(x),t(y))$可由下式计算得到：

$$A_{i+1}(x,y)=\begin{cases}R(x,y),R(x,y)\geqslant\Delta_{i+1}=2^{\lfloor\log_2 U\rfloor-i}\\0,\text{否则}\end{cases}$$

3.3　单位容量网络最大流问题的符号 OBDD 算法[4]

单位容量网络最大流问题的符号算法的主要思想是通过构造层次网，对层次网的每一层求边的极大匹配来寻求极大边不相交路径，并通过极大边不相交的增广路径来计算极大的增广流。而这些极大的边不相交集中的路径是平行的，在算法中是隐式表示的。

由于篇幅有限，下面我们仅给出该算法的实现步骤，详细请参见文[4]。

给定单位容量网络 $N_i(E_i(x,y),s(x),t(y))$，及 N_i 上的初始流 $f_i(x,y)=0$，求网络 N_i 的最大流的符号算法如下。

Step1　通过宽度优先搜索方法来构造层次网，第 i 层搜索到的结点存储在 $Layer[i](x)$中，其中 $Layer[i](x)$是 BDD 变量。若终点 $t(y)$不在 $Layer[i](x)$，则求得源网络 N_i 的最大流 $f_i(x,y)$，算法结束。

Step2　构造正层次网，即从终点 $t(y)$起，进行反向搜索，并对每相邻的两层网进行极大边不相交路径的求解。

Step3　在所建立的正层次网的基础上，从源点 s 起，对正层次网上的各层边进行选择来进行流推进，使得第 l 层与第 $l+1$ 层上的边所构成的所有路径均不相交，且至少存在一条从源点 s 到终点 t 的增广路径。由单位容量网络的特性可见，在每条边不相交的路径上增广的流均为 1。因此，该相的最大流增广路径即为所求的边不相交路径 $P(x,y)$。

Step4　构造 N_i 的剩余网络，有 $E_i(x,y)=E_i(x,y)\cdot\overline{P(x,y)}+P(y,x)$，并将当前流叠加到流 $f_i(x,y)$中，有

$$f_i(x,y)=(f_i(x,y)+P(x,y))\cdot\overline{f_i(x,y)\cdot P(y,x)}\cdot\overline{f_i(y,x)\cdot P(x,y)};$$

转 Step1。

3.4　网络最大流问题的符号 ADD 算法实现步骤

给定网络 $N=(G,s,t,C)$，及 N 上的初始流 $F=0$，求网络 N 的最大流的符号 ADD 算法如下。

Step1　网络 $N=(G,s,t,C)$的符号 ADD 表示 $N(E(x,y),s(x),t(y))$，设 N 上的初始流 $F(x,y)=$zero，初始时 $E(x,y)$中尚未使用的容量大于 0 的边 $R(x,y)=E(x,y)$，$\Delta=2^{\lfloor\log_2 U\rfloor}$，$i=0$，$A_0(x,y)=$zero。

Step2　若 $\Delta<1$，算法结束，求得最大流 $F(x,y)$；否则进行下一步。

Step3　$E(x,y)=R(x,y)+A_i(x,y)$，$i=i+1$；构造网络 $N(E(x,y),s(x),t(y))$的 Δ-剩余网络 $N_i(A_i(x,y)+F(y,x),s(x),t(y))$，有

$$A_i(x,y)=\begin{cases}E(x,y),E(x,y)\geqslant\Delta\\0,\text{否则}\end{cases}$$

$R(x,y)=E(x,y)-A_i(x,y)$。

Step4　求 Δ-剩余网络 $N_i(A_i(x,y)+F(y,x),s(x),t(y))$的单位容量网络 $N_{01}(E_{01}(x,y),s(x),t(y))$，其中 $E_{01}(x,y)=01\,add(A_i(x,y)+F(y,x))$，函数 $01add(f)$是指将函数 f 转化为只有 0 和 1 终结点的 ADD。

Step5　利用单位容量网络最大流问题的符号算法来求 $N_{01}(E_{01}(x,y),s(x),t(y))$网络的最大流 $f(x,y)$；若在构造 N_{01}网络的层次网时，终点 $t(y)$不在层次网络上，那么在 Δ-剩

(下转第 54 页)

维普资讯 http://www.cqvip.com

数量达到百万时，系统完成概率在99.688%左右。

总结 本文在分析基于iSCSI的IP存储广域网可用性中，从用户的角度出发，提出了基于任务完成概率的新的可用性评价标准，并在理论上对IP-SWAN系统进行了可用性分析，分析结果表明IP-SWAN系统具有良好的可用性。

未来应该使用任务完成概率对更多的网络存储系统进行可用性研究，以验证这种分析方法的有效性。另外，可以考虑是否还有其它的可用性度量标准，能更准确、更全面地评价存储系统的可用性。

参考文献

1 Chang F, Ji M, Leung S-T, et al. Myriad: cost-effective disaster tolerance. In: Conf. on File and Storage Technologies, Monterey, CA, Jan. 2002. 103～106

2 Cancio G, Fisher S M, Folkes T, et al. The DataGrid Architecture: [Data Grid TechReport DataGrid-ATF-01]. 2001

3 Oggerino C. High Availability Network Fundamentals, Cisco Press, May 2001

4 熊伟. 基于网络的虚拟存储服务及其相关问题的研究: [南开大学信息技术科学学院博士学位论文]. 2004

5 Siewiorek D, Swarz R. The Theory and Practice of Reliable System Design, Digital Press, 1982

（上接第40页）

余网络 N_i 上求得最大流，转Step7。

Step6 ①计算 Δ-剩余网络 N_i 上的流 $F_\Delta(x,y)=f(x,y)\cdot\Delta$，并将 $F_\Delta(x,y)$ 叠加到 $F(x,y)$ 上，则有 $F(x,y)=F(x,y)+F_\Delta(x,y)-F_\Delta(x,y)\cdot 01add(F(y,x))-F_\Delta(y,x)\cdot 01add(F(x,y))$；

②构造 N_i 的剩余网络 $N'_i(R_i(x,y)+F(y,x),s(x),t(y))$，其中 $R_i(x,y)=A_i(x,y)-(F_\Delta(x,y)-F_\Delta(x,y)\cdot 01add(F(y,x)))$；令 $A_i(x,y)=R_i(x,y)$；

③求 N'_i 的 Δ-剩余网络 $N''_i(A'_i(x,y)+F(y,x),s(x),t(y))$，有

$$A'_i(x,y)=\begin{cases}R_i(x,y), R_i(x,y)\geqslant\Delta;\\ 0, \text{否则}\end{cases}$$

④求 Δ-剩余网络 N''_i 的单位容量网络 $N_{01}(E_{01}(x,y),s(x),t(y))$，其中 $E_{01}(x,y)=01add(A'_i(x,y)+F(y,x))$，转Step5。

Step7 $\Delta=\Delta/2$，转Step2。

4 实验结果及分析

为了检验基于增广路径的符号ADD算法的性能，本文利用Colorado大学的CUDD软件包[9]，在运行平台Windows2000，P4 1500MHz CPU，128MB RAM上，通过随机产生的一些随机图，将本文算法与Dinic算法及Karzanov算法进行了大量的实验对比。

在实验中，我们随机产生不同大小、密度及边容量的有向加权图。一般来说，对于随机图 $N(E(x,y),s(x),t(y))$，$E(x,y)$ 的ADD中的共享结点是非常少的，此可近似认为是符号算法的最差情况[4]。基于这些随机图，本文算法与Dinic算法及Karzanov算法的比较结果如表1所示。当有向加权图较大时，在Dinic算法和Karzanov算法产生溢出错时，而本文算法仍能够在较短的时间内计算出最大流。此外，对具有相同大小、密度的有向加权图，本文算法的执行时间随边容量的变小而减少，这主要是因为在相同大小及密度的条件下，随着有向加权图的边容量的减小，那么 $E(x,y)$ 的ADD的终结点数会随之减少，共享结点就会越多，相应的ADD总结点数就会随之减少，为此算法的性能就越好。

以上实验结果表明，本文算法的空间复杂度较低，可处理更大规模的问题。

表1 符号ADD算法与Dinic算法的执行时间比较表

名称	节点数	边数	容量	最大流	符号ADD算法		Dinic算法		Karzanov算法	
					相数	时间	相数	时间	相数	时间
r1. max	200	5.0×10^2	$[0,10^4]$	4.042×10^3	14	1.031	4	0.01	7	0.015
r2. max	500	2.0×10^5	$[0,10^3]$	1.88288×10^5	10	18.703	4	17.344	4	9.391
r3. max	448	2.0×10^5	$[0,10^3]$	2.19194×10^5	10	17.468	5	18.266	4	7.547
r4. max	800	6.392×10^5	$[0,10^5]$	1.246914×10^7	15	82.672	4	219.844	4	235.813
r5. max	10^3	4.995×10^5	[0,1]	9.99×10^2	1	0.015	溢出错		溢出错	
r6. max	10^3	4.995×10^5	$[0,10^3]$	2.46855×10^5	10	69.594	溢出错		4	5171.594
r7. max	10^3	9.99×10^5	$[0,10^3]$	4.82324×10^6	10	89.031	溢出错		4	4227.078
r8. max	10^3	4.995×10^5	$[0,10^5]$	7.88754×10^6	15	180.55	溢出错		溢出错	

参考文献

1 Dinic E A. Algorithm for solution of a problem of maximum flow in networks with power estimation. Soviet Math Dokl, 1970, 11 (8): 1277～1280

2 Karzanov A V. Determining the maximum flow in a network by the method of preflows. Soviet Math Dokl, 1974, 15(3): 434～437

3 Akers S B. Binary decision diagrams. IEEE Transaction on Computer, 1978, 27(6): 509～516

4 Hachtel G D, Somenzi F. A symbolic algorithm for maximum flow in 0-1 networks. Formal Methods in System Design, 1997, 10 (2-3): 207～219

5 Bachar R I, Frohm E A, et al. Algebraic decision diagrams and their applications. Formal Methods in Systems Design, 1997, 10 (2-3): 171～206

6 Bachar R I, Hachtel G D, et al. An ADD-based algorithm for shortest path back-tracing of large graphs. In: Proc. Great Lakes Symposium on VLSI, Notre Dame, 1994. 248～251

7 Bachar R I, Cho H, et al. Timing analysis of combinational circuits using ADD's. In: Proc. of the European Conf. on Design Automation, Paris, France, 1994. 625～629

8 Gabow H N. Scaling algorithms for network problems. Journal of Computer and System Sciences, 1985, 31(2): 148～168

9 Somenzi F. CUDD: CU decision diagram package release 2. 3. 1. http://vlsi. Colorado. edu/～fabio/CUDD/cuddIntro. html, 2001