4C−6　先行者リストを用いた改訂動的Blocked Warshall法による推移的閉包アルゴリズム†

舘下 直純　大内 東

北海道大学工学部

1 はじめに

二次記憶環境下での推移的閉包を求めるアルゴリズムに動的Blocked Warshall法，先行者リスト付きBlocked Warshall法などがある[1]．我々は，すでに動的Blocked Warshall法に対し，新たな処理順序を持つ改訂動的Blocked Warshall法を提案した[2]．しかし，後続者リストしか保持しない場合には推移的閉包演算の上で必要のない後続者リストを読み込んでくる場合がある．先行者リストを持つことによってその無駄な読み込みを防ぐことができる．

そこで本稿では，改訂動的Blocked Warshall法に先行者リストを用いることを提案し，提案する方法と改訂動的Blocked Warshall法や従来の先行者リスト付きBlocked Warshall法を実験により比較し，その有効性について調べる．

2 データ構造および戦略

ここでは，頂点iから到達可能である頂点の集合を頂点iの後続者リストとして持っている．さらに先行者リストについても同様に，頂点iに到達可能である頂点の集合を頂点iの先行者リストとして持っている．

従来のWarshall法およびWarren法の基本戦略として，
"ある頂点番号k以下の頂点からなる後続者リストが計算済みである．"ということが挙げられる．
これは以下の様に表わすことができる．

1. (i,k) precedes (i,j) for all $k<j$, for all i.
2. (j,k) precedes (i,j) for all $k<j$, for all i.

ここで(i,j)は，頂点iから頂点jへ辺があるかどうかを調べ，それがあるときは書換えをすること(jの後続者リストをiの後続者リストに重複しないように加えること)を意味する．そのような検査と書換えを合わせて要素(i,j)の処理と呼ぶ．

さらに，ここで用いられるデータは，その推移的閉包が主記憶の容量に対して大きすぎて全てのデータを主記憶に入れて計算することが不可能なものであるとする．そうすると主記憶の大きさにあうようにいくつかに分割して計算を行なわなければならない．これらを後続者リスト(先行者リストを持つときはこれも)として持っている場合には，推移的閉包を計算するにつれてそのリストは大きくなり，最初の分割では主記憶の容量に合わなくなってしまう．もし分割が静的であれば分割をより小さいものにして再計算しなければならない結果となる．あるいは，それを恐れて分割をあまりにも小さいものにして計算すると一度に処理できるものをわざわざ分けて計算するのだから当然効率の悪いものとなる．そこで処理中に主記憶が溢れた場合に柔軟に対応して処理するための動的分割を考える．

3 改訂動的Blocked Warshall法

我々がすでに提案している方法で，動的Blocked Warshall法を改訂したものである[2]．

列分割に属する列にあたる頂点の後続者リストは，最初の一つはその列分割の処理が始まる前に主記憶に読み込んでくる．次の頂点の後続者リストからは，その頂点の処理が始まる前に主記憶に読み込んでくる．そして対角ブロックを一つずつ大きくしてゆく方法で，そうすることによって主記憶の容量を無駄なく使えるようになるというものである．
非対角ブロックの処理は従来の動的Blocked Warshall法と同じである[1]．

4 先行者リスト付き動的Blocked Warshall法

従来の動的Blocked Warshall法に先行者リストを用いた方法である[1]．
非対角行の処理を行なう際に，実際にその非対角行の後続者リストが必要かどうかを先行者リストを用いて判定し，本当に必要な場合にのみその後続者リストを二次記憶から主記憶上に読み込んでくるというものである．

このように先行者リストを用いることにより，必要の無い後続者リストの二次記憶からの読み込みを防ぐことができる．しかし，主記憶上に対角ブロックの先行者リストも保持しなければならないので，その分だけ主記憶上に読み込める後続者リストの量が減る，要するに一度に処理できる量が減るという欠点もある．

5 先行者リスト付き改訂動的Blocked Warshall法

今回我々が提案する方法で，改訂動的Blocked Warshall法と先行者リスト付き動的Blocked Warshall法を組み合わせたアルゴリズムである．

5.1 アルゴリズムの概略

アルゴリズムの概略をFig.1に示し，例として7×7行列での処理順序をFig.2に示す．Fig.2の中で，d_iはPhase Iでの処理順序，s_iはPhase II，p_iはPhase IIIでの処理順序である．

† Revised Dynamic Blocked Warshall Algorithm with Predecessors for Computing the Transitive Closure.
Naosumi TATESHITA, Azuma OHUCHI
Fuculty of Engineering, Hokkaido University

```
For each column partition (columns Jb ~ )
{Phase I : Processing of diagonal block}
k := Jb − 1
repeat
   k := k + 1
   for j := Jb to k − 1 do
      i := k
      if tuple(i, j) exists then
         add succ.list of j to succ.list of i.
      if memory overflow then
         process (A), End of Phase I.
   for i := Jb to k do
      j := k
      if tuple(i, j) exists then
         add succ.list of j to succ.list of i.
      if memory overflow then
         process (B), End of Phase I.
until memory overflow

(columns Jb to Je inclusive)

{Phase II : Processing of off-diagonal rows}
for i := 1 to n do (∉ Jb to Je)
   for j := Jb to Je do
      if tuple(i, j) exists (by predecessor) then
         add succ.list of j to succ.list of i.

{Phase III : Predecessor Updates of off-diagonal nodes}
for j := 1 to n do (∉ Jb to Je)
   for i := Jb to Je do
      if tuple(i, j) exists then
         add pred.list of i to pred.list of j.
```

Fig.1 先行者リスト付き改訂動的 Blocked Warshall 法

5.2 アルゴリズムの説明

列分割に属する列にあたる頂点の後続者リストと先行者リストは，最初の一つはその列分割の処理が始まる前に主記憶に読み込んでくる．次の頂点の後続者及び先行者リストからは，その頂点の処理が始まる前に主記憶に読み込んでくる．そしてその列分割の Phase III まですべて終わってから，その列分割に属するれつにあたる頂点の後続者リストおよび先行者リストを全て二次記憶に書き戻す．

列分割に属さない列にあたる頂点の後続者リストは，必要ならばその頂点の非対角行の処理が始まる前に主記憶に読み込んで，処理が終わったらすぐに二次記憶に書き戻す．

(A) i 行の行方向の処理中に主記憶の容量が溢れた場合

いま確定している列分割の最後の列に対応する $i-1$ の後続者リスト及び先行者リストを二次記憶に書き戻し，列分割は $i-2$ までと考え対角ブロックを形成する．それから i 行の残った部分の行方向の処理を行ない，i の後続者リストおよび先行者リストを二次記憶に書き戻す．それによって，非対角行の処理の際には $i-1$ 行と i 行の処理を省略することができる．

		s1	s2	s3		
		s4	s5	s6		
p1	p4	d1	d3	d7	p7	p10
p2	p5	d2	d4	d8	p8	p11
p3	p6	d5	d6	d9	p9	p12
		s7	s8	s9		
		s10	s11	s12		

Fig.2 7 × 7 行列における中央の列分割中の処理順序

(B) j 列の列方向の処理中に主記憶の容量が溢れた場合

現在処理中の列の前の $j-1$ までを列分割と考え，対角ブロックを形成する．処理を終えた j 行については非対角での処理が終わったものと考え後続者リストと先行者リストを二次記憶に書き戻す．それによって，非対角行の処理の際には j 行の処理を省略することができる．

もし，全ての処理が終わったときに各頂点の完全な先行者リスト (逆閉包) が必要ないならば Phase III で対角ブロックの左側 (Jb より小さい番号) の先行者リストは書き換える必要はない．(p1 ～ p6 は省略することができる．) というのは，それらの先行者リストは後の処理で参照されることはないからである．

5.3 先行条件の充足性

もし，(i,j) が対角ブロックにあれば，すべての (i,k) はいつもその行の (i,j) よりも左側に現れる．それらはすでに処理済みとなっているので 1 番目の先行条件は満たす．2 番目の先行条件については，すべての (j,k) はすでに確定した対角ブロックかそれらの行の左側にあるので，これも同様に処理済みである．よってこれも満たしている．次に (i,j) が非対角ブロックにあるときは，現在の列分割の前の列分割まではすでに処理済みであり，現在の列分割の非対角行を左から処理しているので，1 番目の先行条件については満たしている．2 番目の先行条件についても，(j,k) は先程と同じく満足している．よって対角ブロック，非対角ブロックいずれにおいても 2 個の先行条件を満たしているので，この順序で処理することに問題はない．

6 おわりに

非対角ブロックにおいて関係があることが多い場合については，先行者を使うメリットが少なくなる．つまりもともとの関係の密度が小さい場合にのみ先行者を使ったほうが効率が良くなる．
今後の課題として，Blocked Warren 法における動的分割のより効率的な方法を考案することなどが挙げられる．
実験結果等の詳細については，当日発表する．

参考文献

[1] Rakesh Agrawal and H.V.Jagadish : Direct Algorithms for Computing the Transitive Closure of Database Relations : Proc.13th Conference VLDB Brighton,pp.255-266 (1987).

[2] 舘下, 大柳, 大内 : 動的 Blocked Warshall 法による推移的閉包アルゴリズム: 情報処理学会アルゴリズム研究会研究報告,89-AL-11-4 (1989).