Associative Version of the Ramalingam Incremental Algorithm for the Dynamic All-Pairs Shortest-Path Problem

A.S. Nepomniaschaya

Institute of CM&MG  SD RAS
Novosibirsk, 2017
Plan

Introduction.
The STAR-machine.
Preliminaries.
   The Ramalingam algorithm for the dynamic update of the APSPs after inserting an edge.
Representing the Ramalingam algorithm on the STAR-machine.
Main advantages of representing the Ramalingam algorithm on the STAR-machine.
Conclusions.
The STAR-machine

Sequential Control Unit
Data array

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>...</th>
<th>k</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>p</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Associative processing unit

<table>
<thead>
<tr>
<th>R_1</th>
<th>R_2</th>
<th>...</th>
<th>R_h</th>
</tr>
</thead>
</table>
Operations for vertical processing

\textbf{var} \ X, Y: \ 	exttt{slice}; \ i, j: \ 	exttt{integer};

\texttt{v, w: \ word; \ T: \ table;}

\texttt{SET(Y)}  sets all comp. of the slice \ Y to '1';
\texttt{CLR(Y)}  sets all comp. of the slice \ Y to '0';
\texttt{Y(i)}  returns the \ i-th comp. of the slice \ Y;
\texttt{NUMB(Y)} returns the number of bits '1' in the slice \ Y;
\texttt{FND(Y)}  returns the ordinal number of the first bit '1' in the slice \ Y;
\texttt{STEP(Y)} returns the same result as \ FND(Y) and then resets the first bit '1' to '0';
Bitwise Boolean operations:
X and Y, X or Y, not Y, X xor Y.

Predicate: SOME(Y).

Operations for rows:
TRIM(i,j,w), REP(i,j,v,w), ADD(v,w).

Operations for matrices:
ROW(i,T) returns the i-th row of T;
COL(i,T) returns the i-th column of T.
Let $G = (V,E)$ be a digraph, $V = \{1,2,...,n\}$ and $E$ is the set of $m$ arcs. Let $wt(e)$ be a weight function, where $wt(e) \geq 0$.

In any arc $e = (u,v)$, $u \rightarrow v$, $u$ is the tail of $e$ and $v$ is its head.

The shortest path between two vertices in $G$ is a path with the minimal sum of weights of its arcs.

We consider graphs with a distinguished vertex $z$ called 'sink'.

Let $dist(u,z)$ denote the length of the shortest path from $u$ to the sink $z$. 
• $\text{Pred}(u) = \{ y / y \rightarrow u \in E \}$.

• Let $(i,j)$ be inserted in $G$. Vertex $u$ is affected in $G$ if $\text{dist}(u,z)$ is changed.

• $\text{AffectedVert} = \{ y / y \text{ is affected in } G \}$.

• $\text{SP}(a,b,c) \leftrightarrow (\text{dist}(a,c) = \text{wt}(a,b) + \text{dist}(b,c) \& \text{dist}(a,c) \neq \text{infinity})$.
  $\text{SP}(a,b,c)$ verifies whether $(a,b)$ belongs to the shortest path from $a$ to $c$. 
The Ramalingam incremental algorithm for updating the all-pairs shortest paths

- Let \((i,j)\) be inserted in \(G\). The algorithm runs as follows.

- At first, it computes the set \(\text{AffectedSinks}\).

- Then for every \(v \in \text{AffectedSinks}\), it applies the simplified form of the incremental alg. for updating the shortest paths subgraph with a sink.

The simplified form computes the set \(\text{AffectedVert}\). It uses the sets \(\text{WorkSet}, \text{AffectedVert}, \) and \(\text{VisitedVert}\).
The simplified form of the incremental algorithm for updating the shortest paths subgraph

**function** InsertUpdate \((G, i \rightarrow j, z)\);

**Begin**

- \(\text{WorkSet} := \{(i,j)\}\);
- \(\text{AffectedVert} := \{\emptyset\}\);
- \(\text{VisitedVert} := \{i\}\);

**While** \(\text{WorkSet} \neq \emptyset\) do
  - Select and Remove \(x \rightarrow u\) from \(\text{WorkSet}\);
    - if \(\text{wt}(x,u) + \text{dist}(u,z) < \text{dist}(x,z)\) then
      - Insert \(x\) in \(\text{AffectedVert}\);
      - \(\text{dist}(x,z) := \text{wt}(x,u) + \text{dist}(u,z)\);
for every $y \in \text{Pred}(x)$ do 
    if $\text{SP}(y,x,z)$ and $y$ not in $\text{VisitedVert}$ then 
        Insert $(y,x)$ in $\text{WorkSet}$; 
        Insert $y$ in $\text{VisitedVert}$; 
    fi; 
od; 
fi; 
od; 
End.
The incremental algorithm for the dynamic update of the all-pairs shortest paths is given as procedure `InsertEdge` that uses `AffectedSinks`.

```plaintext
procedure InsertEdge(G, i → j, c);
Begin
Insert edge into E(G);
wt(i,j) := c;
AffectedSinks := InsertUpdate(G, i → j, j);
for every x ∈ AffectedSinks do
   InsertUpdate(G, i → j, x);
End.
```
Associative version of the Ramalingam incremental algorithm for updating the all-pairs shortest paths

**The data structure:**

An *adjacency* matrix Adj;

a matrix *Weight* that consists of n fields having h bits each;

a matrix *Cost* that consists of n fields having h bits each;

a matrix *Dist* that consists of n fields having h bits each;

a matrix *Dist1* that consists of n fields having h bits each;

a slice *AffectedV* that saves positions of affected vertices.
On the STAR-machine, we first present the function InsertUpdate. It uses the auxiliary proc. **ComputePred2** that defines *in parallel* the tails of arcs \((y,u)\) for which the predicates \(SP(y,u,s}\) are true.

We have obtained that **ComputePred2** takes \(O(h)\) time.
The simplified form of the increment. algorithm for updating the shortest paths subgraph on the STAR-machine

```pascal
varAffectedV, VisitedV, Z, Z1: slice(Adj);
WS: array [1..2,1..m] of integer;
X: slice(WS);

Begin CLR(AffectedV); CLR(VisitedV); CLR(X);
Write (i,j) in the first row of WS.
X(1) := ‘1’; VisitedV(i) := ‘1’;
while SOME(X) do
begin k := STEP(X);
 Remove the arc (u,p) from the k-th row of WS;
 Compute w3 := wt(u,p) + dist(p,s);
```
if $w_3 \geq \text{dist}(u,s)$ then
go to cycle while SOME($X$) do
else begin
    AffectedV($u$) := ‘1’;
dist($u$,s) := $w_3$;
Perform the proc. \textit{ComputePred2}.
Let ComputePred2 return the slice $Z$.
$Z_1 := Z \text{ and } \neg \text{VisitedV}$;
VisitedV := VisitedV or $Z_1$;
For every vertex $p \in Z_1$, include the arc $(u,p)$ in WS.
end;
End.
On the STAR-machine, the simplified form of the increment algorithm for updating the shortest paths subgraph is given as procedure \texttt{InsertUpdate}.

\textbf{Claim 1.} Let a graph $G$ have $n$ vertices and a sink $s$. Let an arc $(i,j)$ be inserted in $G$. Let the matrices \texttt{Weight}, \texttt{Cost}, \texttt{Dist}, \texttt{Dist1} and \texttt{Adj} be given. Then \texttt{InsertUpdate} returns a slice \texttt{AffectedV}.

Let the slice \texttt{AffectedV} consist of $q$ vertices. Then \texttt{InsertUpdate} takes $O(qh)$ time.
procedure InsertEdge(i,j,h,n: integer;
    v0: word(Trim); var Adj: table; var Weight,
    Cost, Dist, Dist1: table);
/* Here wt(i,j) = v0 . */
var AffectedV, AffectedSinks: slice(Adj);
    z1: integer;
Begin Insert v0 in the matrix Weight;
    Insert v0 in the matrix Cost;
    Include the arc (i,j) in the matrix Adj;
Perform InsertUpdate for the sink \( s = j \);

/* Recall that it returns the slice \( \text{AffectedV} \). */

\( \text{AffectedSinks} := \text{AffectedV}; \)

\( \text{while} \ \text{SOME}(\text{AffectedSinks}) \ \text{do} \)

\( \text{begin} \ z1 := \text{STEP}(\text{AffectedSinks}); \)

\( \quad \text{InsertUpdate}(i,j,h,n,z1,\text{Weight}, \text{Cost}, \text{Adj}, \)
\( \quad \quad \text{Dist}, \text{Dist1}, \text{AffectedV}); \)

\( \text{end}; \)

\( \text{End}; \)
Conclusions

- We have proposed the associative version of the Ramalingam incremental alg. for the dynamic update of the all-pairs shortest paths.
- It has been given as procedure `InsertEdge` those correctness has been proved.
- We have obtained that `InsertEdge` takes $O(hkr)$ time per an insertion, where $k$ is the number of affected sink vertices, $r$ is the total sum of affected vertices for different sink vertices.
- We have shown the main advantages of representing the associative version of the incremental algorithm on the STAR-machine.