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.

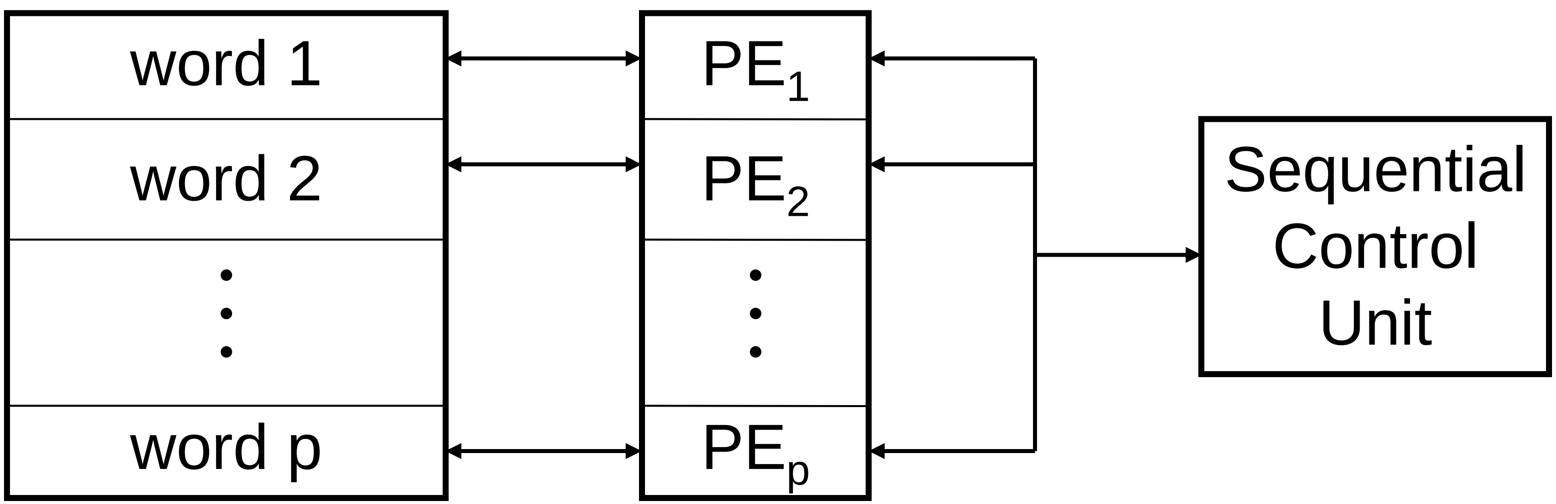
The Ramalingam algorithm for the dynamic update of the APSPs after inserting an edge.

Representing the Ramalingam algorithm on the STAR-machine.

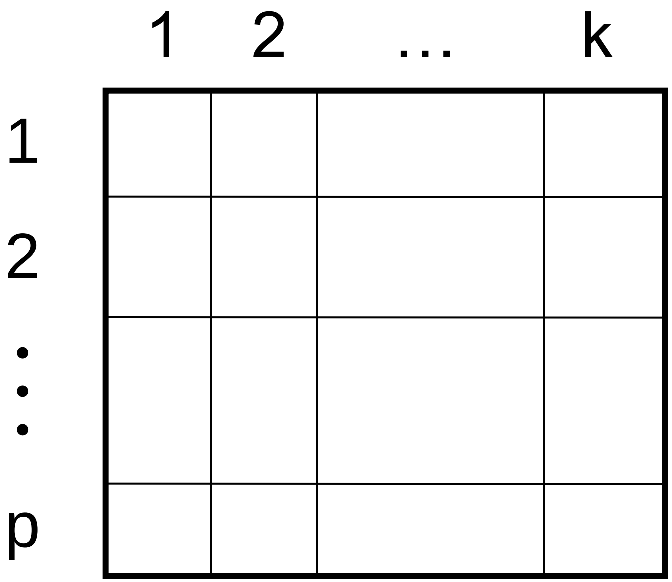
Main advantagies of representing the Ramalingam algorithm on the STAR-machine.

Conclusions.

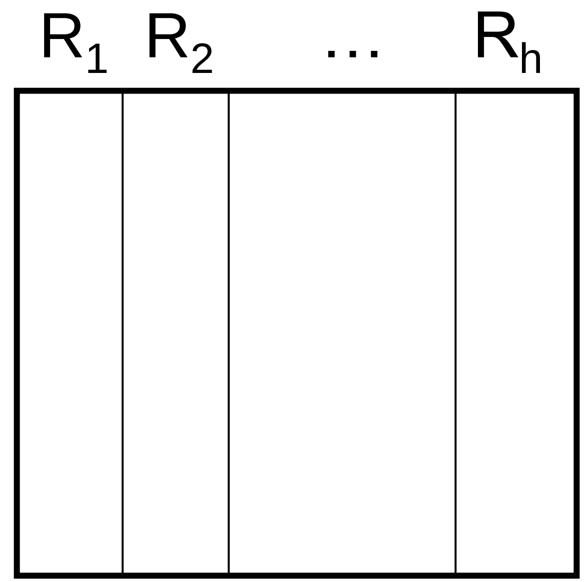
The STAR-machine



Data array



Associative processing unit



Operations for vertical processing

var X,Y: **slice**; i, j: **integer**;

v,w: **word**; T: **table**;

SET(Y) sets all comp. of the slice Y to '1';

CLR(Y) sets all comp. of the slice Y to '0';

Y(i) returns the i-th comp. of the slice Y;

NUMB(Y) returns the number of bits '1' in the slice Y;

FND(Y) returns the ordinal number of the first bit '1' in the slice Y;

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).

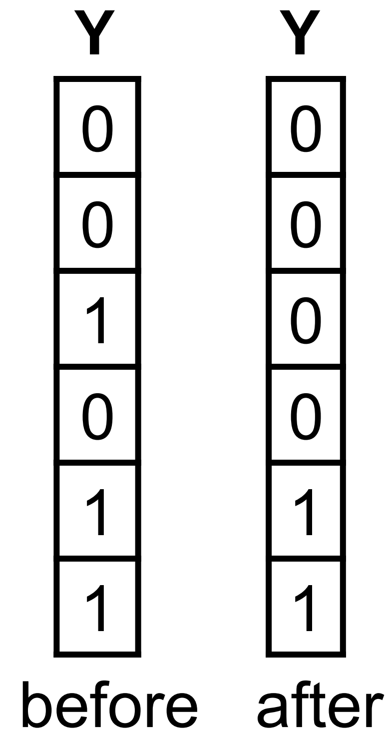
Operations for rows:

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.



STEP(Y)

Preliminaries

- 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

- Pred(u) = { y / y $\rightarrow$ u $\in$ E}.
- Let (i,j) be inserted in G. Vertex u is *affected* in G if dist(u,z) is changed.
- AffectedVert = { y / y is affected in G}.
- *SP(a,b,c)* $\leftrightarrow$ (dist(a,c) = wt(a,b) + dist(b,c) & dist(a,c) $\neq$ infinity).

  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 AffectedSinks.
- Then for every $v \in$ 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 AffectedVert.

It uses the sets WorkSet, AffectedVert, and VisitedVert.

The simplified form of the increm. algorithm for updating the shortest paths subgraph

function InsertUpdate (G, i→ j, z);

Begin WorkSet := {(i,j)}; AffectedVert := {Ø};

VisitedVert := {i};

While WorkSet ≠ Ø **do**

Select and Remove x → u from WorkSet;

if wt(x,u) + dist(u,z) < dist(x,z) **then**

Insert x in AffectedVert;

dist(x,z) **:=** wt(x,u) + dist(u,z);

```
for every y ∈ Pred(x)  do
      if  SP(y,x,z)  and  y not in VisitedVert  then
          Insert  (y,x)  in WorkSet;
          Insert  y  in VisitedVert;
      fi;
    od;
  fi;
 od;
End.
```

The increm. alg. for the dynamic update of the all-pairs shortest paths is given as procedure InsertEdge that uses AffectedSinks.

procedure InsertEdge(G, $i \rightarrow j$, c);

Begin Insert edge into E(G);

wt(i,j) **:**= c;

AffectedSinks **:**= InsertUpdate(G, $i \rightarrow j$, j);

for every $x \in$ AffectedSinks **do**

InsertUpdate(G, $i \rightarrow j$, x);

End.

Associative version ot the Ramalingam increm. 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 funchion 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 increm. algorithm for updating the shortest paths subgraph on the STAR-machine

var AffectedV, 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 w3 $\geq$ dist(u,s) **then**
go to cycle while SOME(X) do
else begin AffectedV(u) **:**= '1';
dist(u,s) **:**= w3;
Perform the proc. ***ComputePred2.***
Let ComputePred2 return the slice Z.
Z1 **:**= Z and (not VisitedV);
VisitedV **:**= VisitedV or Z1;
For every vertex p $\in$ Z1, include the arc (u,p) in WS.
end;
End.

On the STAR-machine, the *simplified form* of the increm. algorithm for updating the shortest paths subgraph is given as procedure **InsertUpdate.**

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 Weight, Cost, Dist, Dist1 and Adj be given. Then **InsertUpdate** returns a slice AffectedV.

Let the slice AffectedV consist of q vertices. Then **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 AffectedV. */
 AffectedSinks := AffectedV;
  while SOME(AffectedSinks) do
   begin z1 := STEP(AffectedSinks);
      InsertUpdate( i,j,h,n,z1,Weight, Cost, Adj,
 Dist, Dist1, AffectedV);
   end;
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 I**nsertEdge** 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.