Distributed Multi-source Regular Path Queries

Maryam Shoaran, Alex Thomo

Regular Path Queries

Useful for expressing desired paths to follow in graph DB's.

E.g. I want to go from Victoria to Munich taking Lufthansa.

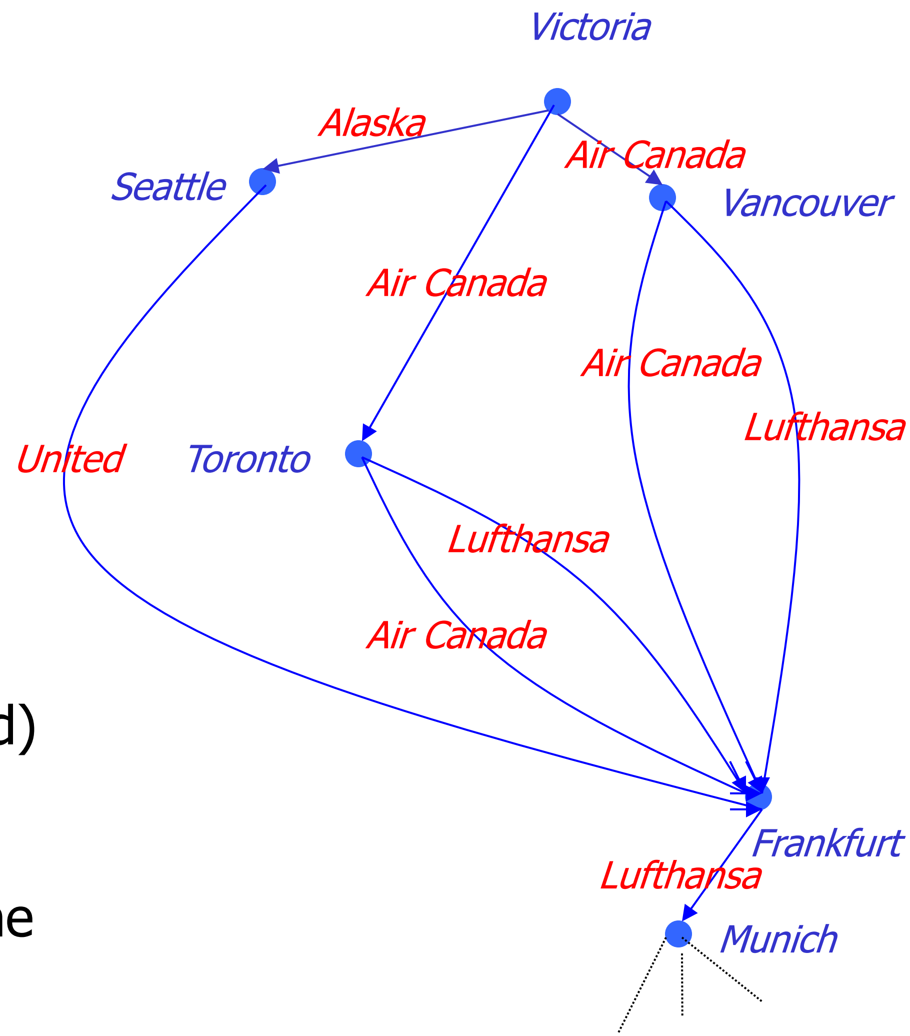
Query: (Lufthansa)*
Answer: Empty

User might repeat querying with:

(Lufthansa+AirCanada)*

…but, this returns too many (unranked) answers.

- Simply, the system doesn't know the user preferences.

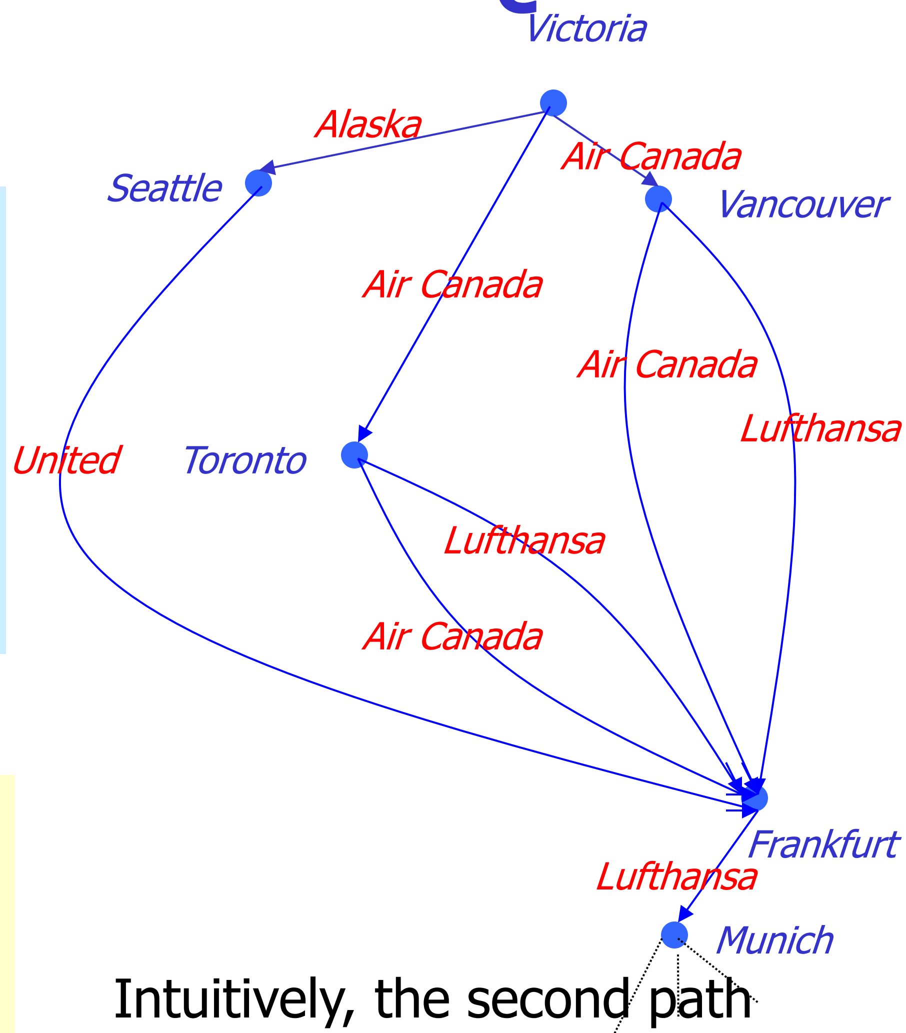


Enhanced Regular Path Queries

Add preference weights:

(Lufthansa:1+AirCanada:2)*

Victoria AirCanada Vancouver Lufthansa Frankfurt Lufthansa Munich	Better than	Victoria AirCanada Vancouver AirCanada Frankfurt Lufthansa Munich



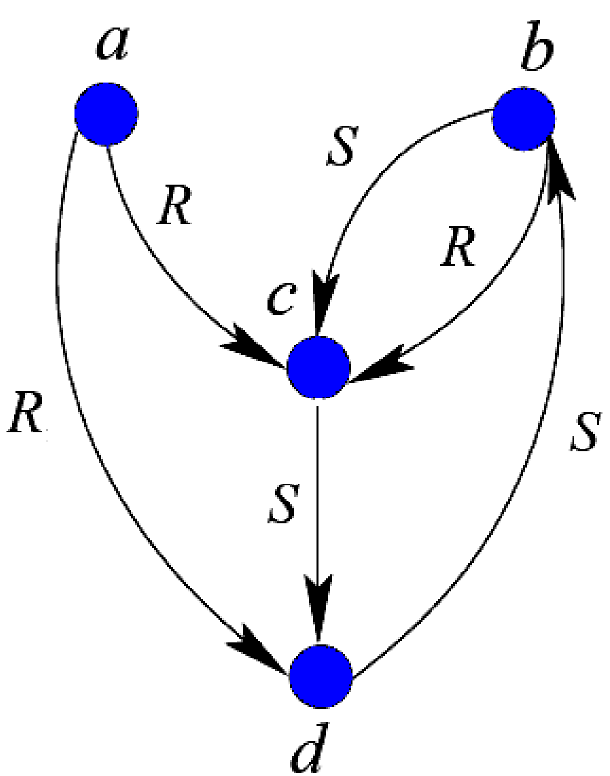
However, which one is better?

Victoria AirCanada Vancouver Lufthansa Frankfurt Lufthansa Munich	or	Victoria AirCanada Toronto Lufthansa Frankfurt Lufthansa Munich

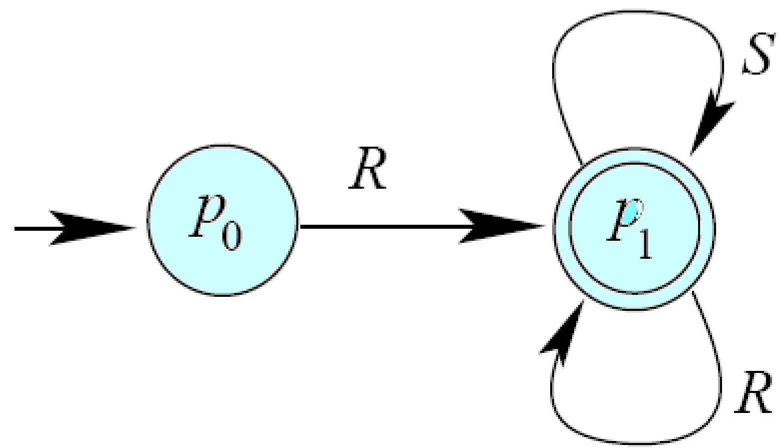
Intuitively, the second path makes me be longer on AirCanada, which I prefer less than Lufthansa.

Databases and Queries

- *DB* is a graph labeled with symbols from Δ



- *Query* is a regular language
- **E.g.** $Q = R \cdot (R+S)^*$



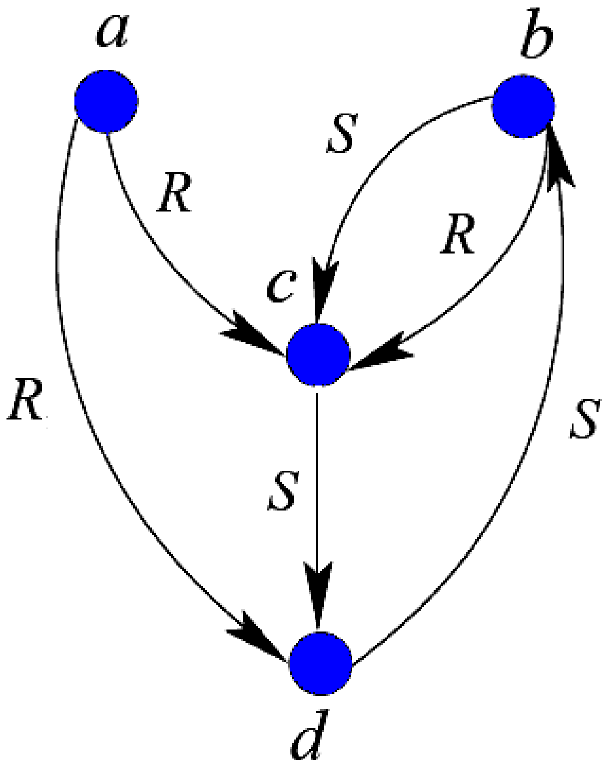
$\text{ans}(Q, DB) =$

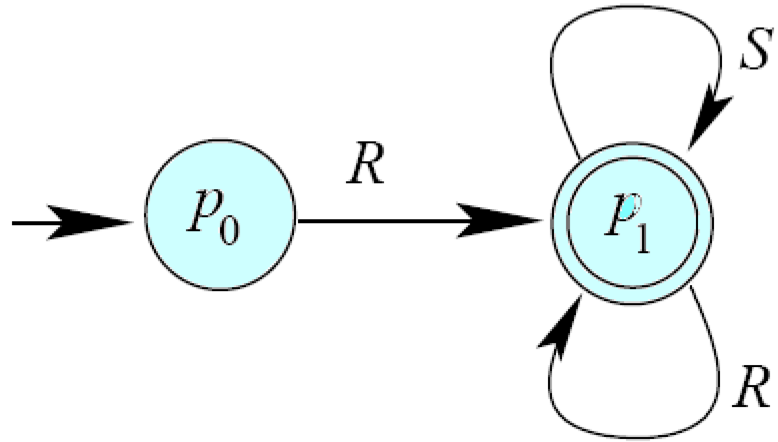
$\{(x, y) \;:\; x \overset{w}{\rightsquigarrow} y$

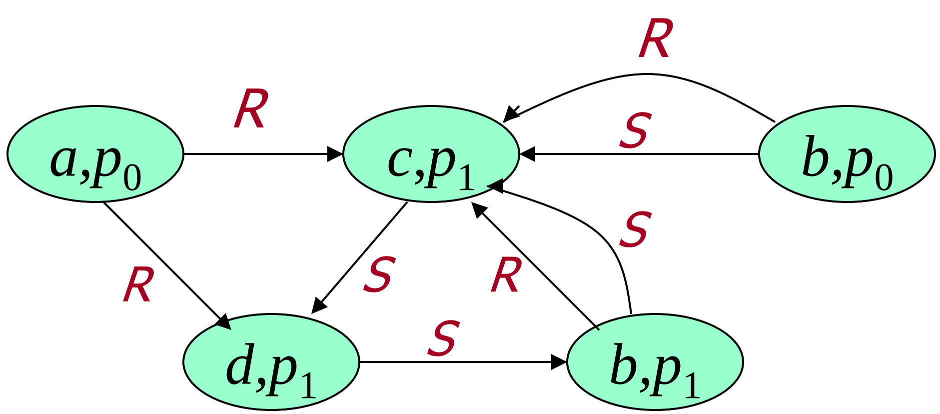
$x, y \in DB, \;\; w \in Q\}$

{(a,d), (a,b)…} for the above query.

Evaluation of queries



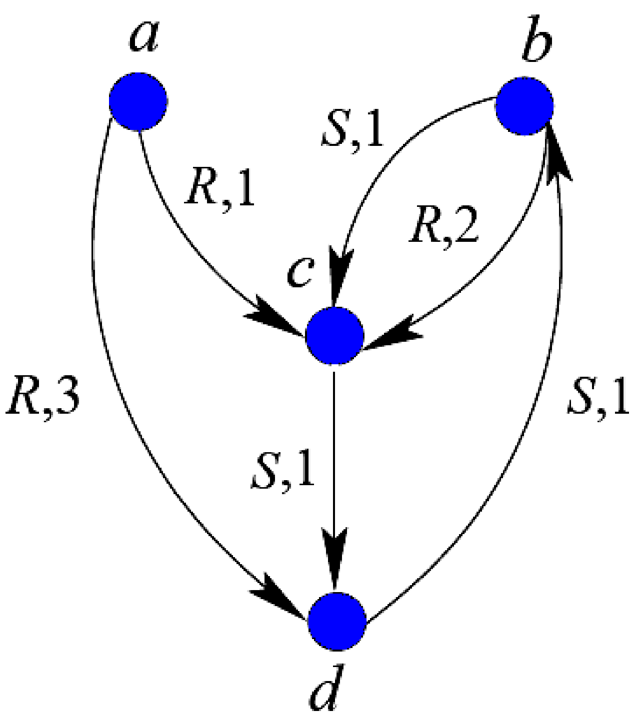




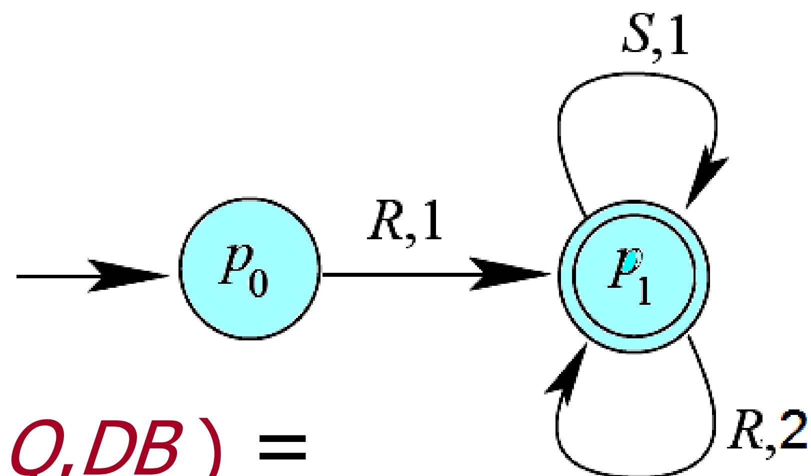
Then, do reachability in the green graph.

Weight Enhanced DB's and Queries

- *DB* is a graph labeled with symbols from $\Delta \times \mathbf{R}^+$



- *Query* is weighted now
- **E.g.** $Q = (R{:}1)\cdot(R{:}2+S{:}1)^*$



ans(Q,DB) =

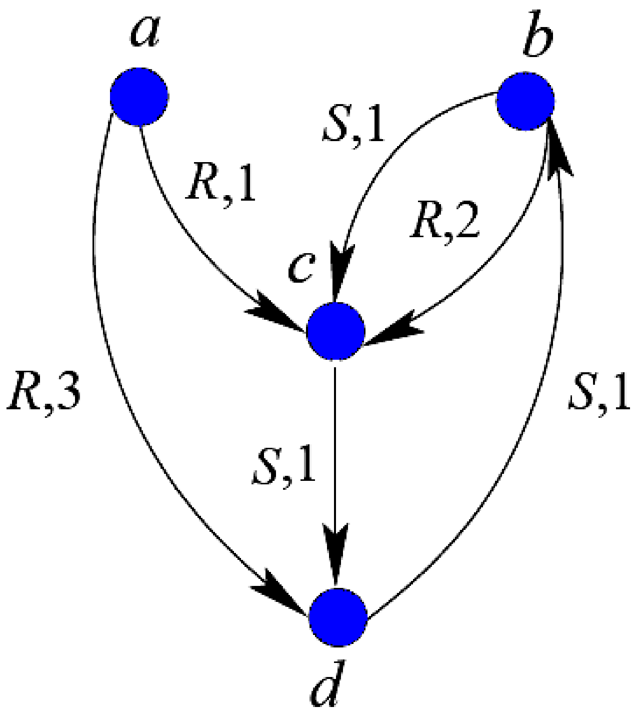
$\{(x,y,n) \;:\; x \overset{w}{\rightsquigarrow} y$

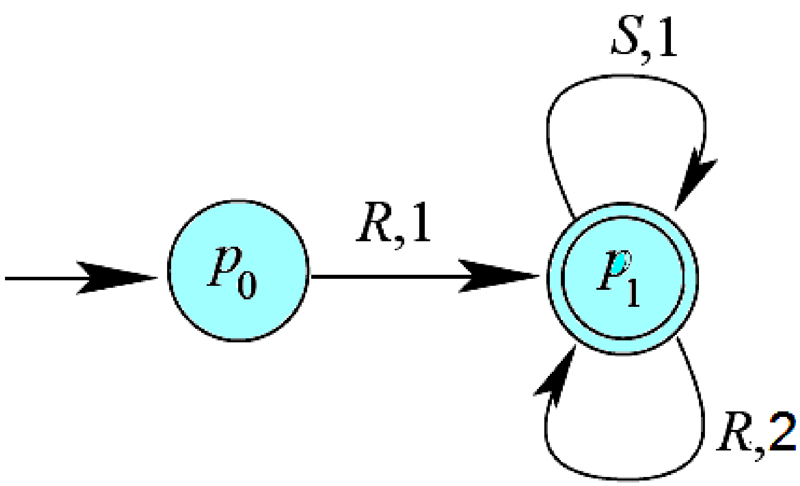
$x,y \in DB,\;\; w \in Q\,,$

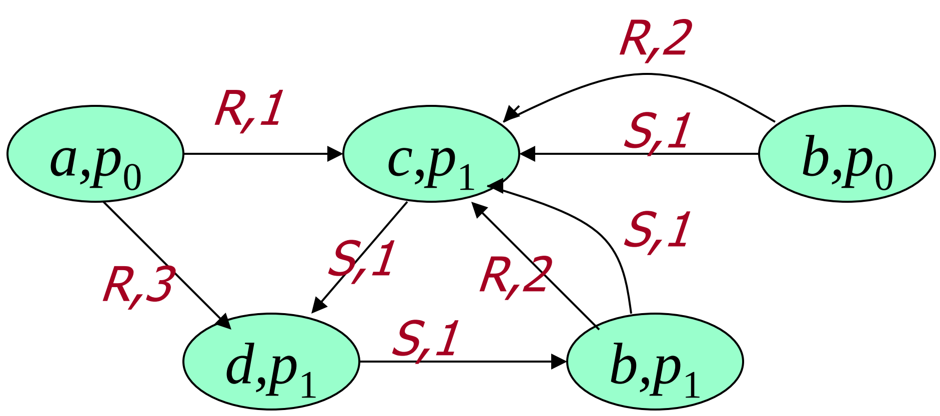
$n = \mathbf{min}\{\mathrm{d}(w) \bullet \mathrm{scale}(\pi)\}\}$

{(a,d,2), (a,b,3)…} for the above query.

Evaluation of weighted queries







Then, compute shortest paths in the green graph.

Variants

- Weighted queries, un-weighted DB’s.
- Un-weighted queries, weighted DB’s.

- Single source.
- Multi source.

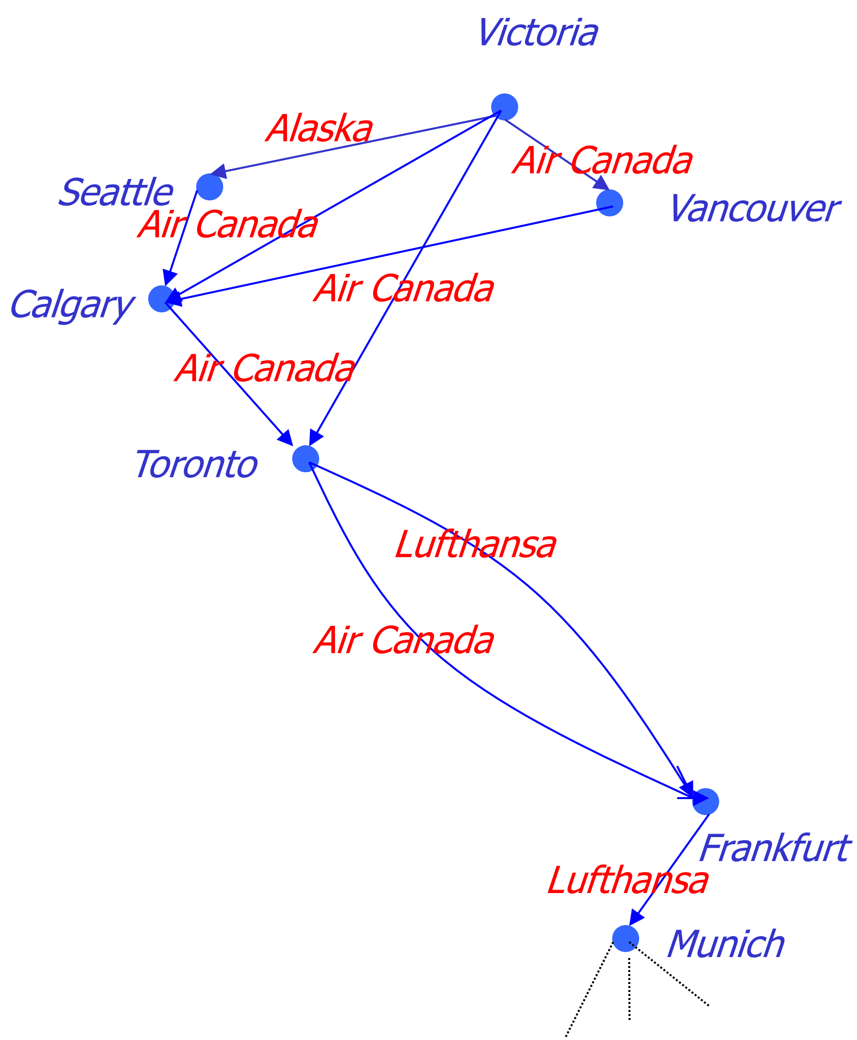
Challenges

- Product graph too big.
  - "On the fly" strategy needed.

- Data might be distributed among a set of peers.
  - A distributed strategy needed.
  - For single source variant see our paper in SAC' 05.

- What about multisource variant?
  - Flloyd-Warshall algorithm can't be used because it needs knowledge of the whole product graph, and we cannot afford to compute it.

Idea



Overlap after Calgary the traversing of paths starting from Seattle, Victoria, and Vancouver.

Distributed Algorithm

- Each DB object is being serviced by a process.

- Query automaton is send first to all the processes.
  - Query automaton is small, (no data transfer here)

- Processes compute the “next” product nodes and send tasks to corresponding neighbor processors.

Distributed Algorithm

- Each process starts by creating an initial task for itself.
  - Tasks are "keyed" by automaton states, with the initial tasks being keyed by the initial state

    $\langle p_0, \{\}, \text{unexpanded}\rangle$

- Each $\langle p, \{\}, \text{unexpanded}\rangle$ at some process P_a is eventually chosen for "expansion."
  - Expansion is the *creation* and *sending* of new tasks to neighbor processes whenever:

    there is an automaton transition originating at state p that matches a database edge originating at object a.

Distributed Algorithm Expansion

- Let $\langle p, \{\}, \text{unexpanded}\rangle$ be chosen for expansion at some process P_a.
- Let (p, R, q, k) be a transition matching a database edge (a, R, b, t).

- Then P_a will send the task $\langle q,\ldots\rangle$ to P_b.

- P_b upon receival of task $\langle q,\ldots\rangle$, will establish a virtual communication channel with P_a for the originating p-task.
  - This channel is weighted by $k{\cdot}t$
  - Completion of the p-task in P_a **depends** on the completion of the q-task in P_b.

Distributed Algorithm Overlapping

- Overlapping of computations happens when:

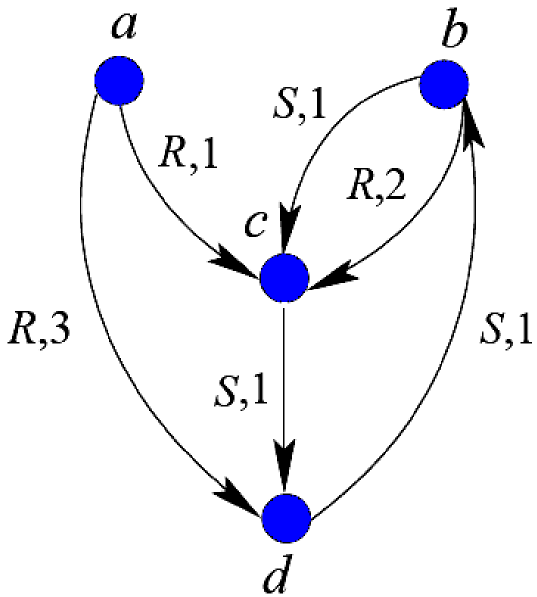
  a process receives the same task multiple times from different neighboring processes.

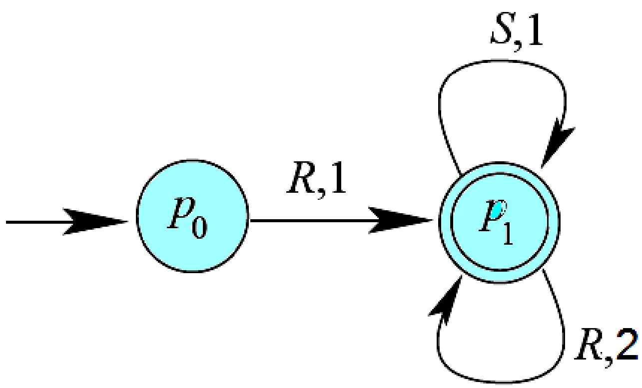
- In such a case:

  the receiving process

  - *does not accept the "new" task, but instead*
  - *creates only a virtual communication channel with the sending process for the originating task.*

Distributed Algorithm Trace 1

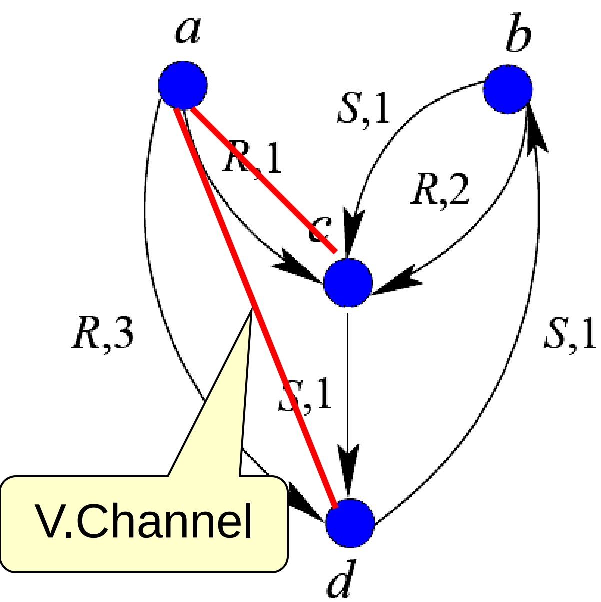


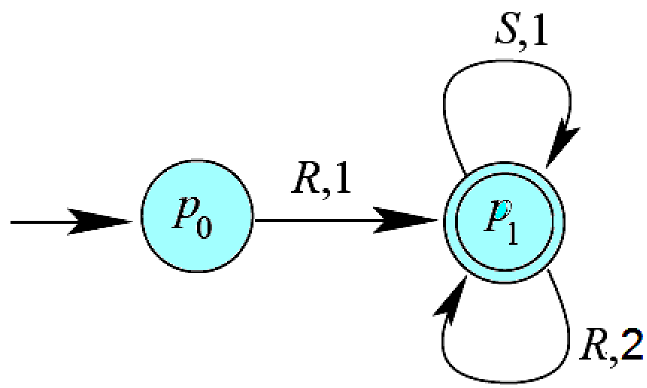


P_a	P_b	P_c	P_d
$\langle p_0, \{\}, \mathrm{u}\rangle$	$\langle p_0, \{\}, \mathrm{u}\rangle$	$\langle p_0, \{\}, \mathrm{u}\rangle$	$\langle p_0, \{\}, \mathrm{u}\rangle$

All processes create a task $\langle p_0, \{\}, \mathrm{u}\rangle$ for themselves.

Distributed Algorithm Trace 2

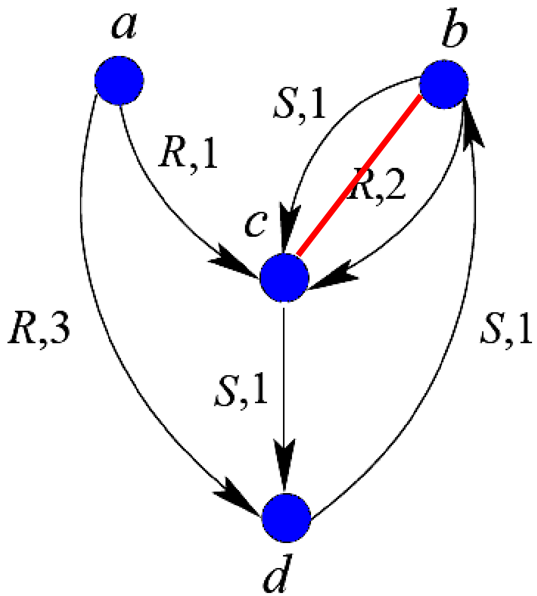


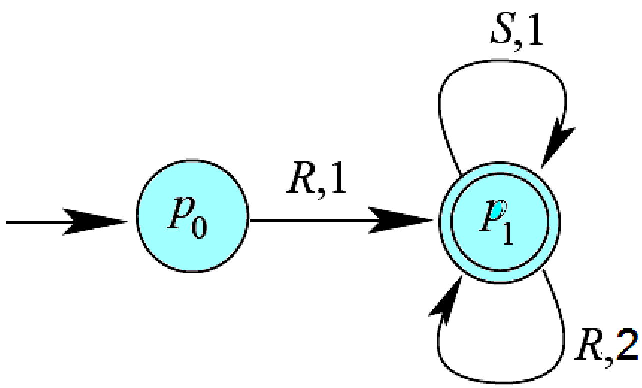


P_a	P_b	P_c	P_d
$\langle p_0, \{(c, 1), (d, 3)\}, e\rangle$	$\langle p_0, \{\}, u\rangle$	$\langle p_0, \{\}, u\rangle$ $\langle p_1, \{(c, 0)\}, u\rangle$	$\langle p_0, \{\}, u\rangle$ $\langle p_1, \{(d, 0)\}, u\rangle$

- P_a expands the tasks $\langle p_0, \{\}, u\rangle$ and sends the task $\langle p_1, \{\}, u\rangle$ to both P_c and P_d.
- P_c and P_d observe that p_1 is a final state and insert (c, 0) and (d, 0) in their p_1-task pair-set.
- P_c and P_d send $\langle c, 1\rangle$ and $\langle d, 3\rangle$ respectively to P_a through the appropriate virtual channels.

Distributed Algorithm Trace 3



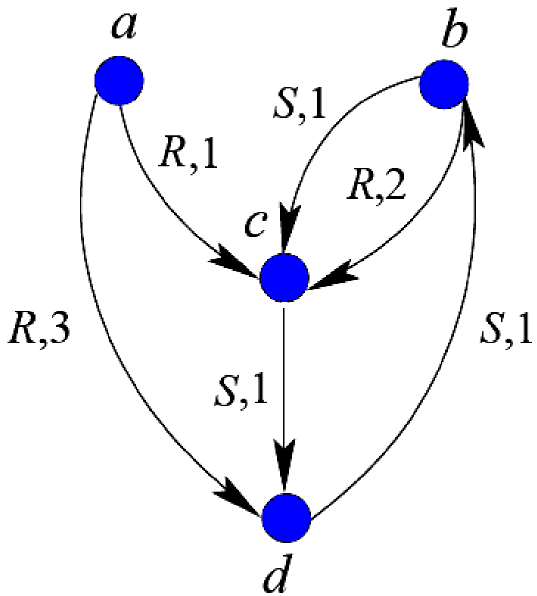


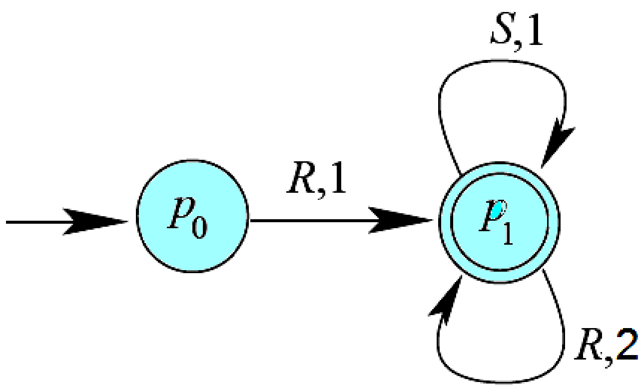
P_a	P_b	P_c	P_d
$\langle p_0, \{(c, 1), (d, 3)\}, e\rangle$	$\langle p_0, \{(c,2)\}, \mathbf{e}\rangle$	$\langle p_0, \{\}, u\rangle$ $\langle p_1, \{(c, 0)\}, u\rangle$	$\langle p_0, \{\}, u\rangle$ $\langle p_1, \{(d, 0)\}, u\rangle$

- P_b expands the p_0-task and sends a p_1-task to P_c.
- P_c has already such a task, so,

  it doesn't create a new task, but only establishes a virtual channel with P_b for the originating p_0 -task.
- Also, P_c sends $\langle c, 2\rangle$ to P_b.

Distributed Algorithm Trace 4

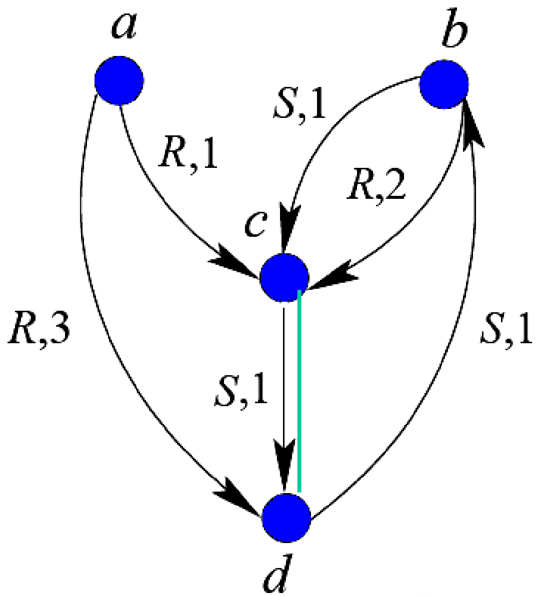


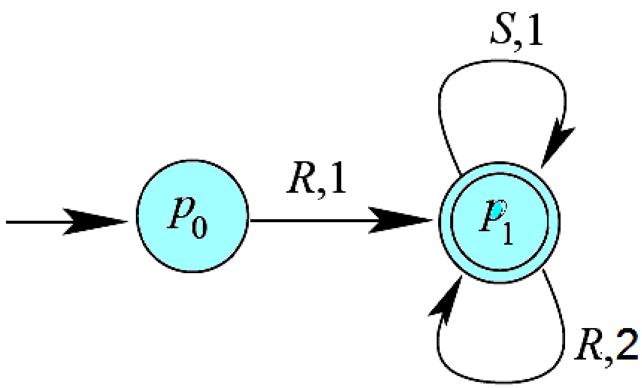


P_a	P_b	P_c	P_d
$\langle p_0$, {(c, 1), (d, 3)}, e$\rangle$	$\langle p_0$, {(c,2)}, e$\rangle$	$\langle p_0$, { }, **e**$\rangle$ $\langle p_1$, {(c, 0)}, u$\rangle$	$\langle p_0$, { }, u$\rangle$ $\langle p_1$, {(d, 0)}, u$\rangle$

- P_c expands the p_0-task and gets stuck.

Distributed Algorithm Trace 5



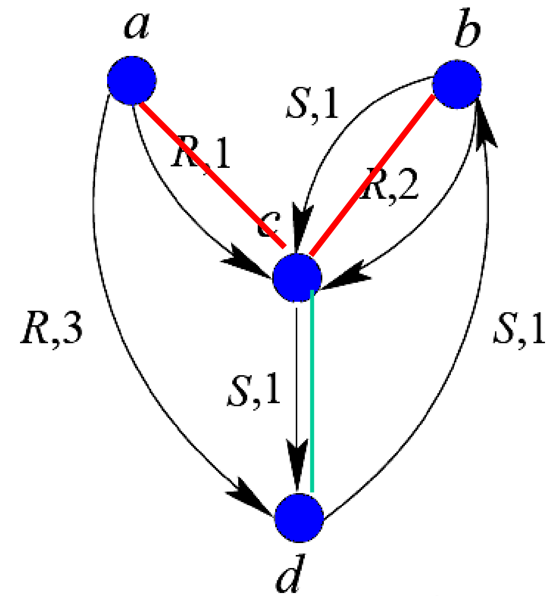


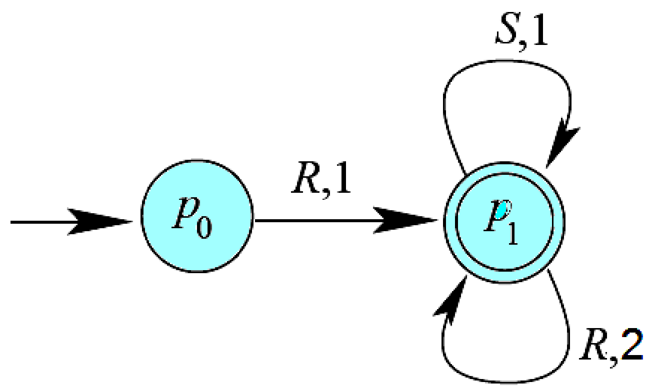
P_a	P_b	P_c	P_d
⟨p_0, {(c, 1), (d, 3)}, e⟩	⟨p_0, {(c,2)}, e⟩	⟨p_0, {}, e⟩ ⟨p_1, {(c, 0)}, **e**⟩	⟨p_0, {}, u⟩ ⟨p_1, {(d, 0)}, u⟩

- P_c expands the p_1 -task and sends a p_1 -task to P_d.
- P_d has already received a p_1 -task before, so,

  it doesn't create a new task, but only establishes a virtual channel with P_c for the originating p_1 -task.

Distributed Algorithm Trace 5





P_a	P_b	P_c	P_d
$\langle p_0$, {(c, 1), (d, **2**)}, e$\rangle$	$\langle p_0$, {(c,2), **(d,3)**}, e$\rangle$	$\langle p_0$, { }, **e**$\rangle$ $\langle p_1$, {(c,0), (d,1)}, **e**$\rangle$	$\langle p_0$, { }, u$\rangle$ $\langle p_1$, {(d, 0)}, u$\rangle$

- P_d sends $\langle$d, 1$\rangle$ to P_c.
- P_c in turn sends:
  - $\langle$d, 2$\rangle$ to P_a
  - $\langle$d, 3$\rangle$ to P_b.
- P_a will update (relax) the weight for d from 3 to 2.

Complexity Discussion

- Upper bound for number of messages: $(E * |\tau|)^2$
- However, E is the number of inter-processor edges.

- If sets of DB nodes are serviced by processors,

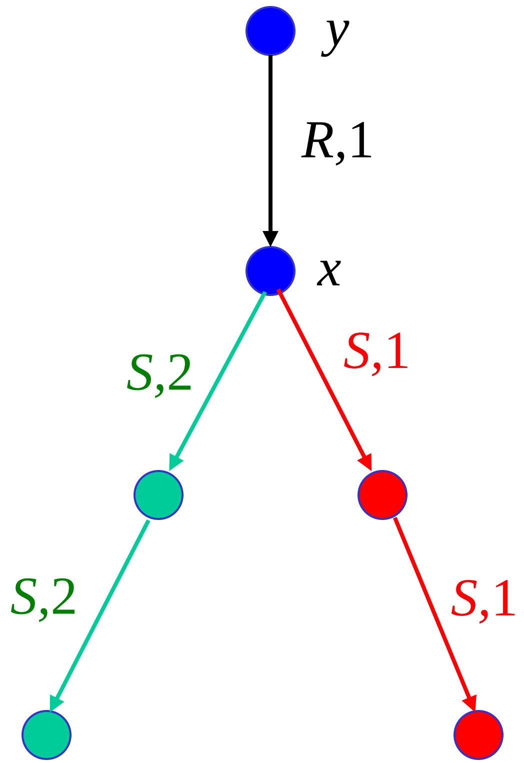
  as opposed to a node per processor,

  then

  the number of messages will be quadratic in the number of processors, not DB edges.

Complexity Discussion

- Delaying back-propagation of query answers, might save a lot of messages.



If x delays the back-propagation of green answers to y, then the (better) red answers will eventually arrive, and be sent to y.

Conclusions

- Introduced enhanced path queries, and concept of scaling query paths.

- Presented a multi-source distributed query evaluation algorithm.
  - Progressive evaluation: i.e. the user sees partial answers very quickly, while waiting for new answers to arrive, and lowering of weights.
  - Even load distribution among processors.

Future Work

- Evaluate the effect of back-propagation delay.
- Investigate the overlapping of multiple queries.
  - Needs query containment.
    - Decidable for un-weighted queries, and weighted DB.
    - Undecidable for weighted queries, and un-weighted DB.
      - (Reduction from equivalence problem for **finance automata** Hashiguchi et. al. 2004)
    - **Open:** What about when both queries and DB's are weighted?

References

- Maryam Shoaran, Alex Thomo. Distributed Multi-source Regular Path Queries. ISPA Workshops 2007: 365-374
- Dan C. Stefanescu, Alex Thomo. Enhanced Regular Path Queries on Semistructured Databases. EDBT Workshops 2006: 700-711
- Dan C. Stefanescu, Alex Thomo, Lida Thomo: Distributed evaluation of generalized path queries. SAC 2005: 610-616
- Gösta Grahne, Alex Thomo. Regular path queries under approximate semantics. Ann. Math. Artif. Intell. 46(1-2): 165-190 (2006)
- Gösta Grahne, Alex Thomo: Query Answering and Containment for Regular Path Queries under Distortions. FoIKS 2004: 98-115
- Gösta Grahne, Alex Thomo. Approximate Reasoning in Semistructured Data. KRDB 2001