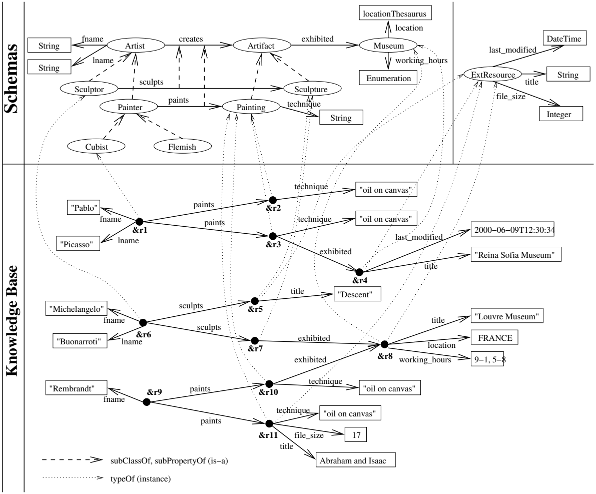
Designing a Path-oriented Indexing Structure for a Graph Structured Data

Stanislav Bartoň

Masaryk university, Brno

June 1, 2005

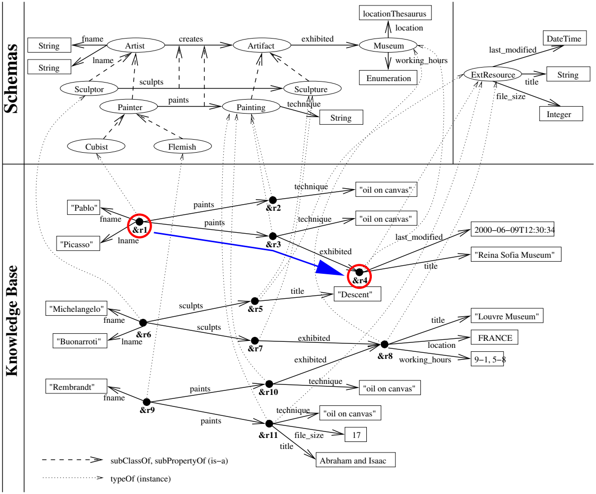
An example of an RDF Graph



ρ operators

- Firstly introduced in the context of Semantic Web
- Designed to study complex relationships between entities defined as Complex Associations
- Can be generalized into terms of graphs and a problem of searching paths in them

ρ-path operator

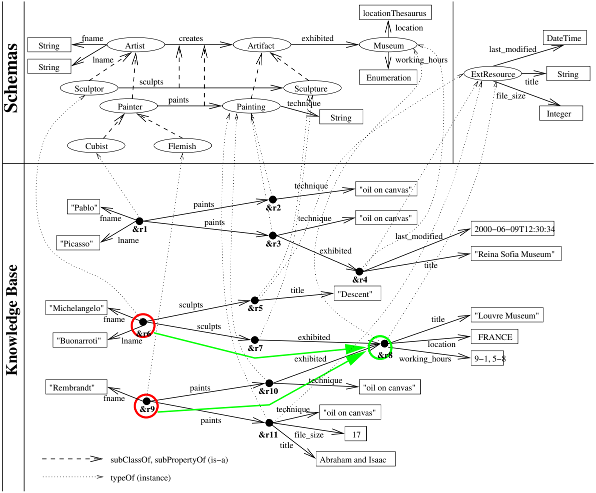


ρ-path operator

ρ-*path* operator definition in graph theory terms:

$$\rho\text{-}\mathbf{path}(\mathbf{x}, \mathbf{y}) = \{p = (v_1 e_1 v_2 e_2 \dots e_n v_{n+1}) | v_1 = x \wedge v_{n+1} = y \wedge p \text{ } \textit{is acyclic}\}$$

ρ-connection operator



ρ-connection operator

ρ-*connection* operator definition in graph theory terms:

ρ-**connection**$(\mathbf{x}, \mathbf{y}) = \{(p_1, p_2) | p_1 = (v_1 e_1 v_2 e_2 \ldots e_n v_{n+1}), p_2 = w_1 h_1 w_2 h_2 \ldots h_n w_{m+1}) \wedge v_1 = x \wedge w_1 = y \wedge v_{n+1} = w_{m+1} \wedge p_1, p_2 \text{ are acyclic}\}$

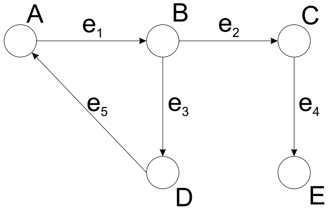
Designing the indexing structure

- Adjacency matrix
  - Great graph description, simple transitive closure computation
  - Can be easily modified to store paths themselves rather then just amounts of them
  - The use of matrix algebra is limited to relatively small graphs due to space and time complexity
- With graph transformations towards graph simplification $\Rightarrow$ transformed graph must have similar properties as the original graph had
  - Graph segmentation (*vertex clustering*, *graph to forest of trees*)
  - The transitive closure of segment graph models all paths from the original graph

Designing the indexing structure

- Path type matrix
  - Instead of storing the amounts of paths, keeps the paths themselves
  - $+$ and $*$ replaced by path concatenation and set union
  - enables cycle detection during computation
- Vertex clustering
  - two pass algorithm that divides the graph into several subgraphs of predefined size
  - the vertices that are *close* to each other are put to same subgraphs
  - very general, non-restrictive technique that can be applied to arbitrary directed graph

Path type matrix transitive closure



$$M = \begin{pmatrix} & A & B & C & D & E \\ A & & \{(e_1)\} & & & \\ B & & & \{(e_2)\} & \{(e_3)\} & \\ C & & & & & \{(e_4)\} \\ D & \{(e_5)\} & & & & \\ E & & & & & \end{pmatrix}$$

Path type matrix transitive closure

$$M = \begin{array}{c c} & \begin{array}{ccccc} A & B & C & D & E \end{array} \\ \begin{array}{c} A \\ B \\ C \\ D \\ E \end{array} & \left(\begin{array}{ccccc} & \{(e_1)\} & & & \\ & & \{(e_2)\} & \{(e_3)\} & \\ & & & & \{(e_4)\} \\ \{(e_5)\} & & & & \\ & & & & \end{array}\right) \end{array}$$

$$M^2 = \begin{array}{c c} & \begin{array}{ccccc} A & B & C & D & E \end{array} \\ \begin{array}{c} A \\ B \\ C \\ D \\ E \end{array} & \left(\begin{array}{ccccc} & & \{(e_1 e_2)\} & \{(e_1 e_3)\} & \\ \{(e_3 e_5)\} & & & & \{(e_2 e_4)\} \\ & & & & \\ & & & & \\ & & & & \end{array}\right) \end{array}$$

$$M + M^2 = \begin{array}{c c} & \begin{array}{ccccc} A & B & C & D & E \end{array} \\ \begin{array}{c} A \\ B \\ C \\ D \\ E \end{array} & \left(\begin{array}{ccccc} & \{(e_1)\} & \{(e_1 e_2)\} & \{(e_1 e_3)\} & \\ & \{(e_3 e_5)\} & \{(e_2)\} & \{(e_3)\} & \{(e_2 e_4)\} \\ & & & & \{(e_4)\} \\ \{(e_5)\} & & & & \\ & & & & \end{array}\right) \end{array}$$

Path type matrix transitive closure

$$M = \begin{array}{cc} & \begin{matrix} A & B & C & D & E \end{matrix} \\ \begin{matrix} A \\ B \\ C \\ D \\ E \end{matrix} & \begin{pmatrix} & \{(e_1)\} & & & \\ \boxed{} & \boxed{} & \{(e_2)\} & \{(e_3)\} & \\ & & & & \{(e_4)\} \\ \{(e_5)\} & & & & \\ & & & & \end{pmatrix} \end{array}$$

$$M^2 = \begin{array}{cc} & \begin{matrix} A & B & C & D & E \end{matrix} \\ \begin{matrix} A \\ B \\ C \\ D \\ E \end{matrix} & \begin{pmatrix} & & \{(e_1 e_2)\} & \{(e_1 e_3)\} & \\ \boxed{\{(e_3 e_5)\}} & & & & \{(e_2 e_4)\} \\ & & & & \\ & & & & \\ & & & & \end{pmatrix} \end{array}$$

$$M + M^2 = \begin{array}{cc} & \begin{matrix} A & B & C & D & E \end{matrix} \\ \begin{matrix} A \\ B \\ C \\ D \\ E \end{matrix} & \begin{pmatrix} & \{(e_1)\} & \{(e_1 e_2)\} & \{(e_1 e_3)\} & \\ & \{(e_3 e_5)\} & \{(e_2)\} & \{(e_3)\} & \{(e_2 e_4)\} \\ & & & & \{(e_4)\} \\ \{(e_5)\} & & & & \\ & & & & \end{pmatrix} \end{array}$$

Path type matrix transitive closure

$$M = \begin{array}{rl} & \begin{array}{ccccc} A & B & C & D & E \end{array} \\ \begin{array}{c} A \\ B \\ C \\ D \\ E \end{array} & \begin{pmatrix} & \{(e_1)\} & & & \\ & & \{(e_2)\} & \{(e_3)\} & \\ & & & & \{(e_4)\} \\ \{(e_5)\} & & & & \\ & & & & \end{pmatrix} \end{array}$$

$$M^2 = \begin{array}{rl} & \begin{array}{ccccc} A & B & C & D & E \end{array} \\ \begin{array}{c} A \\ B \\ C \\ D \\ E \end{array} & \begin{pmatrix} & & \{(e_1 e_2)\} & \{(e_1 e_3)\} & \\ \{(e_3 e_5)\} & & & & \{(e_2 e_4)\} \\ & & & & \\ & & & & \\ & & & & \end{pmatrix} \end{array}$$

$$M + M^2 = \begin{array}{rl} & \begin{array}{ccccc} A & B & C & D & E \end{array} \\ \begin{array}{c} A \\ B \\ C \\ D \\ E \end{array} & \begin{pmatrix} & \{(e_1)\} & \{(e_1 e_2)\} & \{(e_1 e_3)\} & \\ & \{(e_3 e_5)\} & \{(e_2)\} & \{(e_3)\} & \{(e_2 e_4)\} \\ & & & & \{(e_4)\} \\ \{(e_5)\} & & & & \\ & & & & \end{pmatrix} \end{array}$$

Path type matrix transitive closure

$$M = \begin{array}{cc} & \begin{array}{ccccc} A & B & C & D & E \end{array} \\ \begin{array}{c} A \\ B \\ C \\ D \\ E \end{array} & \begin{pmatrix} & \{(e_1)\} & & & \\ & & \{(e_2)\} & \{(e_3)\} & \\ & & & & \{(e_4)\} \\ \{(e_5)\} & & & & \\ & & & & \end{pmatrix} \end{array}$$

$$M^2 = \begin{array}{cc} & \begin{array}{ccccc} A & B & C & D & E \end{array} \\ \begin{array}{c} A \\ B \\ C \\ D \\ E \end{array} & \begin{pmatrix} & & \{(e_1 e_2)\} & \{(e_1 e_3)\} & \\ \{(e_3 e_5)\} & & & & \{(e_2 e_4)\} \\ & & & & \\ & & & & \\ & & & & \end{pmatrix} \end{array}$$

$$M + M^2 = \begin{array}{cc} & \begin{array}{ccccc} A & B & C & D & E \end{array} \\ \begin{array}{c} A \\ B \\ C \\ D \\ E \end{array} & \begin{pmatrix} & \{(e_1)\} & \{(e_1 e_2)\} & \{(e_1 e_3)\} & \\ & \{(e_3 e_5)\} & \{(e_2)\} & \{(e_3)\} & \{(e_2 e_4)\} \\ & & & & \{(e_4)\} \\ \{(e_5)\} & & & & \\ & & & & \end{pmatrix} \end{array}$$

Path type matrix transitive closure

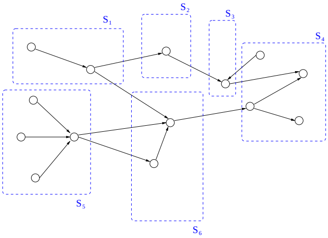
$$M = \begin{array}{cc} & \begin{array}{ccccc} A & B & C & D & E \end{array} \\ \begin{array}{c} A \\ B \\ C \\ D \\ E \end{array} & \begin{pmatrix} & \{(e_1)\} & & & \\ & & \{(e_2)\} & \{(e_3)\} & \\ & & & & \{(e_4)\} \\ \{(e_5)\} & & & & \\ & & & & \end{pmatrix} \end{array}$$

$$M^2 = \begin{array}{cc} & \begin{array}{ccccc} A & B & C & D & E \end{array} \\ \begin{array}{c} A \\ B \\ C \\ D \\ E \end{array} & \begin{pmatrix} & & \{(e_1 e_2)\} & \{(e_1 e_3)\} & \\ \{(e_3 e_5)\} & & & & \{(e_2 e_4)\} \\ & & & & \\ & & & & \\ & & & & \end{pmatrix} \end{array}$$

$$M + M^2 = \begin{array}{cc} & \begin{array}{ccccc} A & B & C & D & E \end{array} \\ \begin{array}{c} A \\ B \\ C \\ D \\ E \end{array} & \begin{pmatrix} & \{(e_1)\} & \{(e_1 e_2)\} & \{(e_1 e_3)\} & \\ & \{(e_3 e_5)\} & \{(e_2)\} & \{(e_3)\} & \{(e_2 e_4)\} \\ & & & & \{(e_4)\} \\ \{(e_5)\} & & & & \\ & & & & \end{pmatrix} \end{array}$$

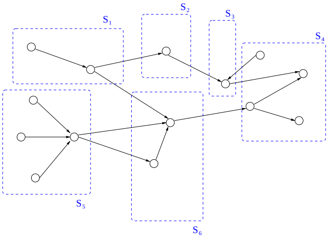
Graph segmentation

- **Segment S** in a graph G : $S = (V_S, E_S) : V_S \subseteq V \ \wedge \ E_S = \{e \in E \mid RIGHT\_VERTEX(e) \in V_S \ \vee LEFT\_VERTEX(e) \in V_S\}$



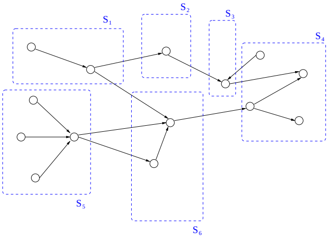
Graph segmentation

- **Segment S** in a graph G : $S = (V_S, E_S) : V_S \subseteq V \ \wedge \ E_S = \{e \in E \mid RIGHT\_VERTEX(e) \in V_S \ \vee LEFT\_VERTEX(e) \in V_S\}$
- EDGES_OUT(S) $= \{e|e \in E_S \wedge LEFT\_VERTEX(e) \in V_S \wedge RIGHT\_VERTEX(e) \notin V_S\}$
- EDGES_IN(S) $= \{e|e \in E_S \wedge RIGHT\_VERTEX(e) \in V_S \wedge LEFT\_VERTEX(e) \notin V_S\}$



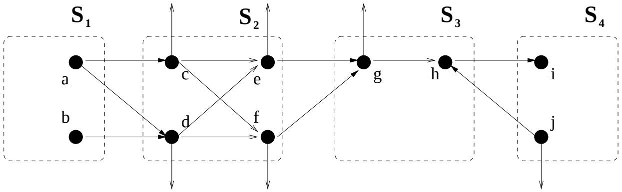
Graph segmentation

- **Segment S** in a graph G : $S = (V_S, E_S) : V_S \subseteq V \;\wedge\; E_S = \{e \in E \mid RIGHT\_VERTEX(e) \in V_S \;\vee LEFT\_VERTEX(e) \in V_S\}$
- EDGES_OUT(S) $= \{e|e \in E_S \wedge LEFT\_VERTEX(e) \in V_S \wedge RIGHT\_VERTEX(e) \notin V_S\}$
- EDGES_IN(S) $= \{e|e \in E_S \wedge RIGHT\_VERTEX(e) \in V_S \wedge LEFT\_VERTEX(e) \notin V_S\}$
- **Segmentation S(G)**$= \{S|S \text{ is a segment of } G\} \;\wedge \forall S, S' \in S(G), S \neq S' : V_S \cap V_{S'} = \emptyset \;\wedge \bigcup_{S \in S(G)} V_S = V$



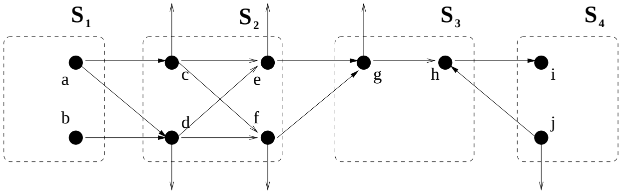
Sequence of segments

- **Sequence of segments** $(S_1 \ldots S_m) = S_1, \ldots S_l \in S(G),\ 1 \leq i \leq m-1 : EDGES\_OUT(S_i) \cap EDGES\_IN(S_{i+1}) \neq \emptyset$



Sequence of segments

- **Sequence of segments** $(S_1 \ldots S_m) = S_1, \ldots S_l \in S(G),\ 1 \leq i \leq m-1 : EDGES\_OUT(S_i) \cap EDGES\_IN(S_{i+1}) \neq \emptyset$
- **Connecting path** $p = (e_1 e_2 \ldots e_n)$ in a segment sequence $(S_1 \ldots S_m)$:
  $p \in (S_1 \ldots S_m) : e_1 \in EDGES\_OUT(S_1) \cap EDGES\_IN(S_2) \ \wedge$
  $e_n \in EDGES\_OUT(S_{m-1}) \cap EDGES\_IN(S_l) \ \wedge \ \exists i_2, i_3, \ldots i_{m-1} : 1 <$
  $i_2 < i_3 < \ldots < i_{m-1} < n : \{e_2, \ldots e_{i_2}\} \subseteq E_{S_2} \ \wedge \ \{e_{i_2}, \ldots e_{i_3}\} \subseteq$
  $E_{S_3} \ \wedge \ \ldots \ \wedge \ \{e_{i_{l-2}}, \ldots e_{i_{m-1}}\} \subseteq E_{S_{m-1}}$



Paths

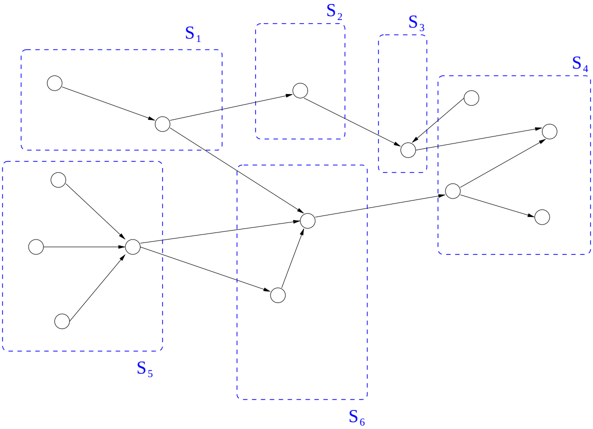
- Acyclic path $p = (v_1 e_1 v_2 e_2 \dots e_n v_{n+1})$ in G :
$1 \leq i \leq n, 1 \leq j \leq n+1, i \neq j : e_i \in E \ \wedge \ v_i, v_j \in V \ \wedge \ v_i = LEFT\_VERTEX(e_i) \ \wedge \ v_{i+1} = RIGHT\_VERTEX(e_i) \ \wedge \ v_i \neq v_j$

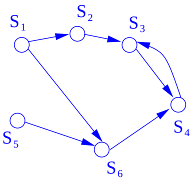
Paths

- Acyclic path $p = (v_1 e_1 v_2 e_2 \ldots e_n v_{n+1})$ in G : $1 \leq i \leq n, 1 \leq j \leq n+1, i \neq j : e_i \in E \ \wedge \ v_i, v_j \in V \ \wedge \ v_i = LEFT\_VERTEX(e_i) \ \wedge \ v_{i+1} = RIGHT\_VERTEX(e_i) \ \wedge \ v_i \neq v_j$
- **Proper segment sequence** for a path $p = (v_1 e_1 v_2 e_2 \ldots e_n v_{n+1})$: $S(p) = (S_1 \ldots S_m) : S(p)$ *is a segment sequence* $\wedge 1 \leq i_1 < i_2 < \ldots < i_l \leq n+1 : \{v_1, \ldots v_{i_1}\} \subseteq V_{S_1} \wedge \{v_{i_1}, \ldots v_{i_2}\} \subseteq V_{S_2} \wedge \ldots \wedge \{v_{i_l}, \ldots v_{n+1}\} \subseteq V_{S_l}$

Segment graph

- Segment graph of G: $SG(G) = (S(G),\ X),\ X = \{h | h = (S_i, S_j) \Leftrightarrow 1 \leq i, j \leq k \wedge EDGES\_OUT(S_i) \cap EDGES\_IN(S_j) \neq \emptyset\}$





Representing paths in G by segment sequences in S(G)

Lemma

If a graph $G = (V, E)$ has a segmentation $S(G)$ that forms a graph $SG(G)$, any path $p = (v_1 e_1 v_2 e_2 \ldots e_n v_{n+1})$ in G can be represented by its proper segment sequence in $S(G)$ and this representation is unique.

Lemma

If a graph $G = (V, E)$ has a segmentation $S(G)$ that forms a graph $SG(G)$, a segment sequence in $S(G)$ represents either some path in G or an empty path.

Preliminary evaluation

- If we generate all possible segment sequences in $S(G)$, we get all possible paths in G

Preliminary evaluation

- If we generate all possible segment sequences in $S(G)$, we get all possible paths in G
  + Fast generation of dense path representations

Preliminary evaluation

- If we generate all possible segment sequences in $S(G)$, we get all possible paths in G
  + Fast generation of dense path representations
  - Problem with disconnected segment sequences

Preliminary evaluation

- If we generate all possible segment sequences in $S(G)$, we get all possible paths in G
  - \+ Fast generation of dense path representations
  - \- Problem with disconnected segment sequences
  - \- A huge amount of paths can be found in dense graphs between almost any two vertices, the number of such paths grows exponentially with the maximal length of a path allowed

Preliminary evaluation

- If we generate all possible segment sequences in $S(G)$, we get all possible paths in G
  - $+$ Fast generation of dense path representations
  - \- Problem with disconnected segment sequences
  - \- A huge amount of paths can be found in dense graphs between almost any two vertices, the number of such paths grows exponentially with the maximal length of a path allowed

$\Rightarrow$ A need for l-ρ-index which is a variation of rho-index, where only those paths between two vertices with length $\leq l$ and some paths having length $> l$ are indexed

Preliminary evaluation

- If we generate all possible segment sequences in $S(G)$, we get all possible paths in G
  - $+$ Fast generation of dense path representations
  - \- Problem with disconnected segment sequences
  - \- A huge amount of paths can be found in dense graphs between almost any two vertices, the number of such paths grows exponentially with the maximal length of a path allowed

$\Rightarrow$ A need for l-ρ-index which is a variation of rho-index, where only those paths between two vertices with length $\leq l$ and some paths having length $> l$ are indexed

- Computational overhead bound with a weight computation of each segment sequence stored $\Rightarrow$ An upper bound on a maximal number of connecting paths to be computed to find the one with a lowest weight

Weights in G and S(G) - Definitions

- Weight of vertex v: $w(v) \in < 1, \infty >$
- Weight of path $p = (v_1 e_1 v_2 e_2 \dots e_n v_{n+1}) : w(p) = \sum_{i=1}^{n+1} w(v_i)$
- Set of weights of segment sequence: $|(S_1 \dots S_m)| = \{w(p) | p \in (S_1 \dots S_m)\}$
- Weight of segment sequence $\|(S_1 \dots S_m)\| = min(|(S_1 \dots S_m)|)$

Facts about weights in G and S(G)

- The relation between $(v_1 e_1 v_2 e_2 \ldots e_n v_{n+1})$ and $(S_1 \ldots S_m)$
  - !! $m \leq n + 1 \implies$ the segment sequence representation is always shorter or of the same length as the path it represents

Facts about weights in G and S(G)

- The relation between $(v_1 e_1 v_2 e_2 \dots e_n v_{n+1})$ and $(S_1 \dots S_m)$
  - !! $m \leq n+1 \Longrightarrow$ the segment sequence representation is always shorter or of the same length as the path it represents
  - !! $||(S_1 \dots S_m)|| \leq w(p) \Longrightarrow$ the proper segment sequence for a path p has always lower or the same weight as the path it represents

Proposing the limit l

Lemma

If a graph $G = (V, E)$ has a segmentation $S(G)$ that forms a graph $SG(G)$ then for a limit l, segment sequences in $S(G)$ having weight $\leq l$ represent all paths in G that have weight $\leq l$.

Proof.

Lets assume that there is a path p with $w(p) \leq l$ and that it is not present in the result represented by segment sequences with $\|(S_1 \dots S_m)\| \leq l$. This would imply that the $S(p) > w(p)$ but this is contradictory to the previous facts. $\square$

Upper bound on the minimal number of connecting paths

Lemma (An upper bound on a maximal number of connecting paths to be computed to find the one with a lowest weight)

A connecting path for a segment sequence $(S_1 \dots S_m)$ with the lowest weight is a path in CPs with the lowest weight.

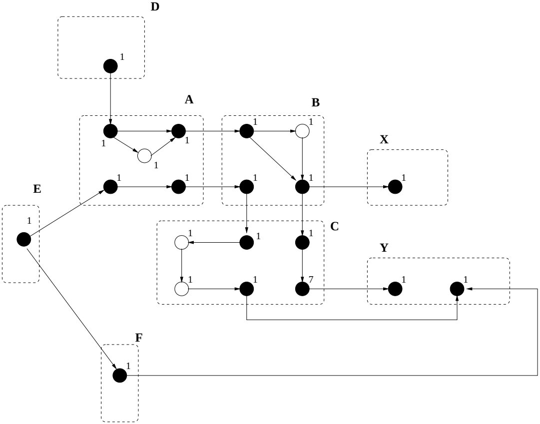
- CPs is a set of connecting paths that have for each combination of common edges for each two neighboring segments in $(S_1 \dots S_m)$ minimal weight.
- The upper bound is represented by the number of combinations of common edges picked from $m - 1$ sets of common edges .

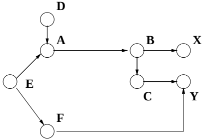
Recursively applying the graph segmentation

- What if the segment graph SG(G) of the indexed graph is not small enough to be described by a path type matrix?
- Intuitively, the graph segmentation can be applied again to the segment graph SG(G) forming the SG(SG(G)).
- But what happens to the vertices' weights? Segments do not have weights assigned, since the segment's shortest traversal is context dependent.
- How to propose the vertices' weights to upper levels of the indexing structure?

Assigning weight to a segment

- !!! Segment sequence $(EABX)$ is disconnected
- !!! $||(ABC)|| = 3$, $||(ABX)|| = 4 \Longrightarrow w(B) = 1$ or 2 ?





Altering the weight definitions for an iteration step

- G = (V, E), G' = SG(G) = (S(G), E'), G" = (SG(SG(G)) = (S(S(G)), E")
- Weight of vertex $v \in V$: $w(v) \in < 1, \infty >$
- Weight of path $p = (v_1 e_1 v_2 e_2 \dots e_n v_{n+1}) : w(p) = \sum_{i=1}^{n+1} w(v_i), p \in G$
- **Connecting segment sequence** $(A_1 \dots A_m) \in G'$ for $(S_1 \dots S_m)$ $\in G''$ denotes a path $(A_1 e_1 A_2 \dots e_{k-1} A_k)$ in G' where $e_1 \in (EDGES\_OUT(S_1) \cap EDGES\_IN(S_2)), e_{k-1} \in (EDGES\_OUT(S_{m-1}) \cap EDGES\_IN(S_m))$ and $(S_1 \dots S_m)$ is a proper segment sequence for $(A_1 e_1 A_2 \dots e_{k-1} A_k)$.

Altering the weight definitions

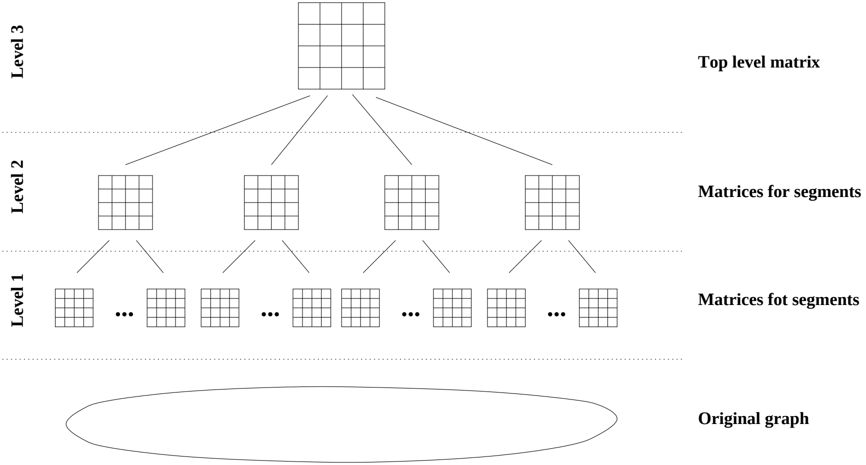
- Set of weights of segment sequence:
$$|(S_1 \dots S_m)| = \begin{cases} \{w(p) | p \in (S_1 \dots S_m)\}, S \in S(G) \\ \{|(A_1 \dots A_k)| \; |(A_1 \dots A_k) \in (S_1 \dots S_m)\}, S \in S(S(G)) \end{cases}$$
- Weight of segment sequence $\|(S_1 \dots S_m)\| = min(|(S_1 \dots S_m)|)$

ρ-index's structure

ρ-index comprises of:

- Each segment is represented by its path type matrix
- EDGES_IN and EDGES_OUT are also stored for each segment
- Path type matrix of a segment graph at the topmost level

Outline of a ρ-index's structure

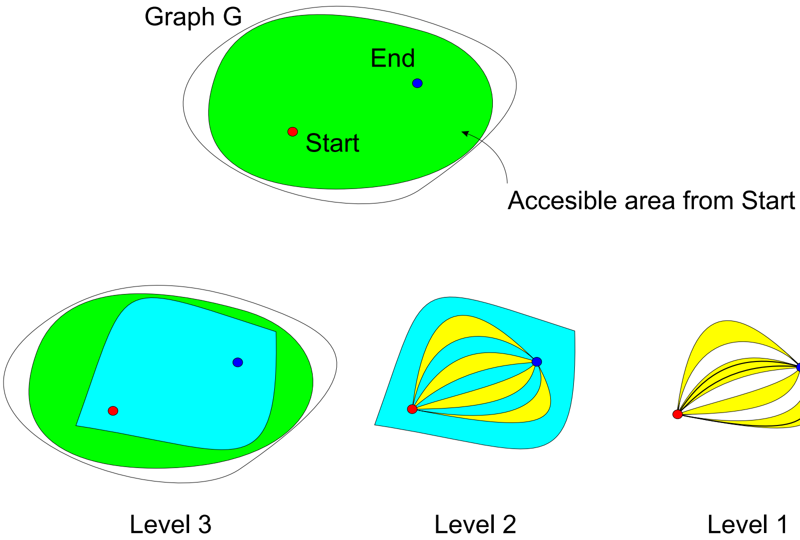


Creating $\rho - index$

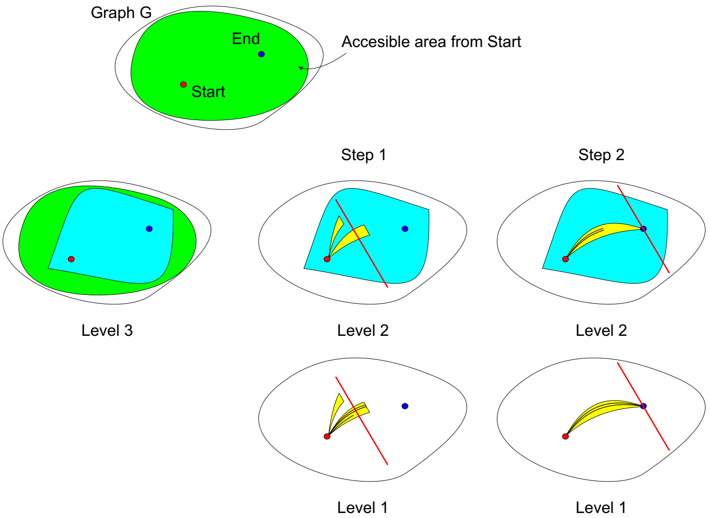
Creating algorithm:

1. Segmentation of the indexed graph G using the *vertex clustering* transformation
2. Creation of path type matrix for each segment, subsequent transitive closure computation
3. Creation of a segment graph $SG(G)$
4. If the segment graph is not small enough $\longrightarrow$ repeat previous steps

Path search algorithm - Breadth First



Path search algorithm - Depth First



Practical experience with the approximative ρ-index

- The approximative (k, l)-ρ-index implementation:
  - Limiting parameters pseudo k and l

  $\rightarrow$ k - limits the number of segment sequences stored in one matrix field

  $\rightarrow$ l - limits the degree of computation of the transitive closure of the matrices representing segments and the top matrix

$\Rightarrow$ Insufficiency of the implemented $(k,\ l)$ parameters lead to the design of the correct l - variant of the ρ-index by proposing weights of vertices and segment sequences to the design of the indexing structure

Practical experience with the approximative ρ-index

- Index efficiency is very dependent on a size of the cluster used to segment the graph
  - using the same (k, l) parameters lead into different number of paths indexed
  - the lower the size of the cluster the more precise results gained
- The unlimited variant of the ρ-index can be achieved using a small size of a cluster

$\Rightarrow$ small number of stored segment sequences in each matrix field

$\Rightarrow$ enables complete transitive closure computation for each segment

Future research

- Implementation and full evaluation of the l-ρ-index variation including optimized depth first search algorithm
- Explore the impact of a graph segmentation strategy to the indexing structure
  - Optimization of the vertex clustering technique
  - Further research of other segmentation techniques
- $(k, l) - \rho - index$ - another variation of ρ-index where only the first k paths of length $\leq l$ are indexed
- Explore the possibilities of distributing the ρ-index

Thank you for your attention.