On the dynamization of shortest path overlay graphs

Work partially supported by the Future and Emerging Technologies Unit of EC (IST priority - 6th FP), under contract no. FP6-021235-2 (project ARRIVAL)

Francesco Bruera Serafino Cicerone
Gianlorenzo D'Angelo Gabriele Di Stefano
Daniele Frigioni

Deptartment of Electrical and Information Engineering
University of L'Aquila - Italy
{cicerone,gdangelo,gabriele,frigioni}@ing.univaq.it

Important issue on railways

Answer to timetable queries as fast as possible

Known solution

- A weighted directed graph represents a timetable
- Shortest paths algorithms answer to timetable queries
- Use speed-up techniques for shortest paths and preprocessed information

Purpose of this work

- Compute preprocessed information used by speed-up techniques
- Update preprocessed information used by speed-up techniques

Outline

Outline

Outline

Speed-up techniques for shortest paths

- Pruning of Dijkstra's algorithm (SWW00, MSSWW05, GKW06)
- Geometric Information (WW03, WWZ05)
- Landmark (GH05)
- Arc-labelling (KMS05)
- Hierarchical decomposition (SWZ02, HSW06, SS06, DHMSW06)
- Combinations (HSWW06, GKW06)

Dynamization for speed-up techniques

- Geometric Information (WWZ03)
- Landmark (DW07)
- Hierarchical decomposition (SS07)

Outline

Multi-level overlay graphs for shortest paths

Hierarchical speed-up technique for shortest paths (HSW06)
Experimentally fast when applied to timetable information

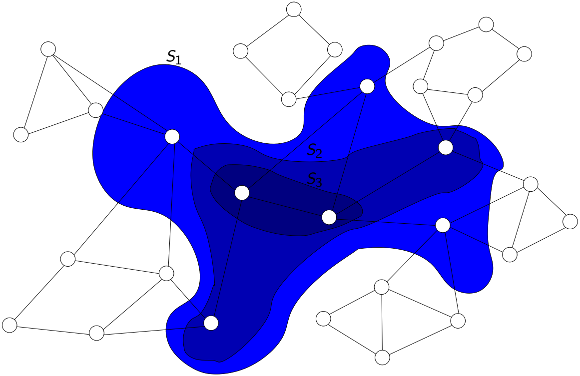
Results

- **Compute** and store efficiently a multi-level overlay graph
- Answer to distance queries theoretically faster than Dijkstra's algorithm by using a multi-level overlay graph
- **Update** a multi-level overlay graph when an edge weight of the graph changes

Outline

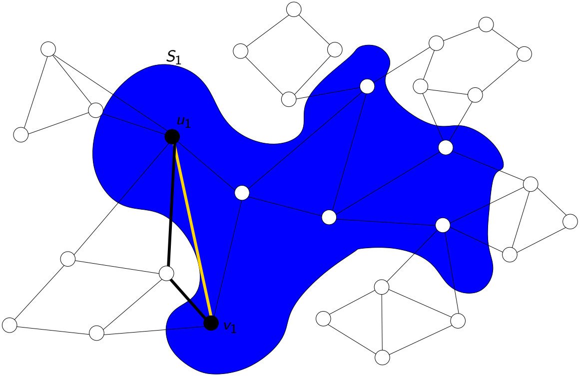
Given a graph $G = (V, E)$ and a a sequence $V \equiv S_0 \supset S_1 \supset \ldots \supset S_l$ of subsets of V

$\mathcal{M} = (V, E \cup E_1 \cup \ldots \cup E_l)$ where
$\forall i = 1, 2, \ldots, l$, edge $(u, v) \in S_i \times S_i$ belongs to $E_i \Leftrightarrow$ each shortest path from u to v does not contain a node in S_i $(w(u, v) = d(u, v))$



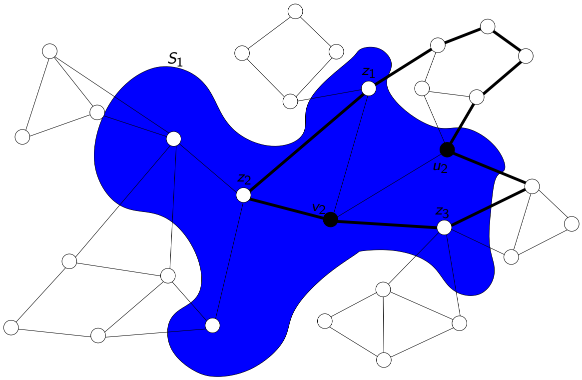
Given a graph $G = (V, E)$ and a a sequence $V \equiv S_0 \supset S_1 \supset \ldots \supset S_l$ of subsets of V

$\mathcal{M} = (V, E \cup E_1 \cup \ldots \cup E_l)$ where
$\forall i = 1, 2, \ldots, l$, edge $(u, v) \in S_i \times S_i$ belongs to $E_i \Leftrightarrow$ each shortest path from u to v does not contain a node in S_i $(w(u, v) = d(u, v))$



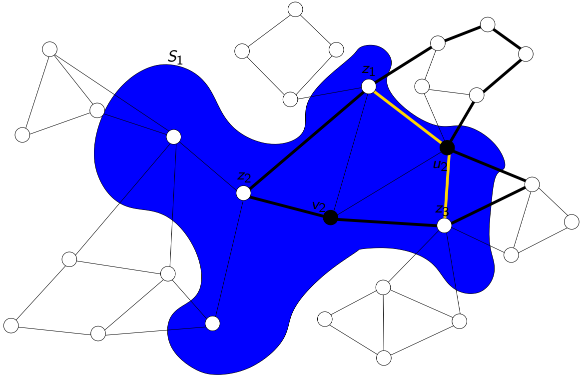
Given a graph $G = (V, E)$ and a a sequence $V \equiv S_0 \supset S_1 \supset \ldots \supset S_l$ of subsets of V

$\mathcal{M} = (V, E \cup E_1 \cup \ldots \cup E_l)$ where
$\forall i = 1, 2, \ldots, l$, edge $(u, v) \in S_i \times S_i$ belongs to $E_i \Leftrightarrow$ each shortest path from u to v does not contain a node in S_i $(w(u, v) = d(u, v))$



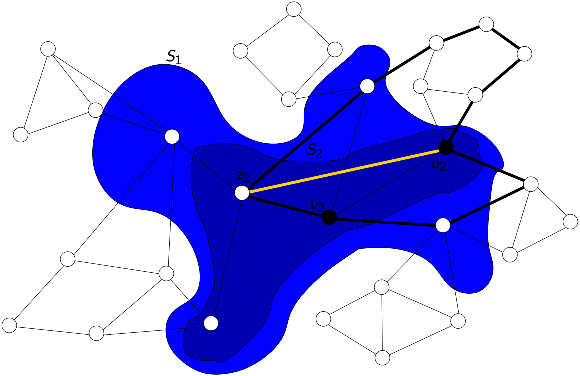
Given a graph $G = (V, E)$ and a a sequence $V \equiv S_0 \supset S_1 \supset \ldots \supset S_l$ of subsets of V

$\mathcal{M} = (V, E \cup E_1 \cup \ldots \cup E_l)$ where
$\forall i = 1, 2, \ldots, l$, edge $(u, v) \in S_i \times S_i$ belongs to $E_i \Leftrightarrow$ each shortest path from u to v does not contain a node in S_i $(w(u, v) = d(u, v))$



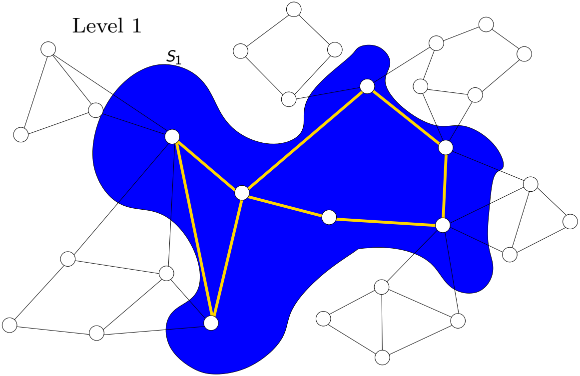
Given a graph $G = (V, E)$ and a a sequence $V \equiv S_0 \supset S_1 \supset \ldots \supset S_l$ of subsets of V

$\mathcal{M} = (V, E \cup E_1 \cup \ldots \cup E_l)$ where
$\forall i = 1, 2, \ldots, l$, edge $(u, v) \in S_i \times S_i$ belongs to $E_i \Leftrightarrow$ each shortest path from u to v does not contain a node in S_i $(w(u, v) = d(u, v))$



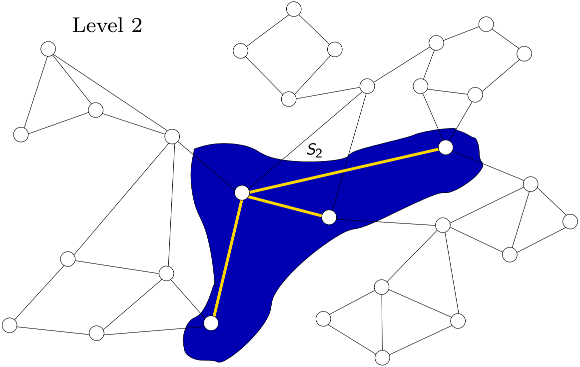
Given a graph $G = (V, E)$ and a a sequence $V \equiv S_0 \supset S_1 \supset \ldots \supset S_l$ of subsets of V

$\mathcal{M} = (V, E \cup E_1 \cup \ldots \cup E_l)$ where
$\forall i = 1, 2, \ldots, l$, edge $(u, v) \in S_i \times S_i$ belongs to $E_i \Leftrightarrow$ each shortest path from u to v does not contain a node in S_i $(w(u, v) = d(u, v))$



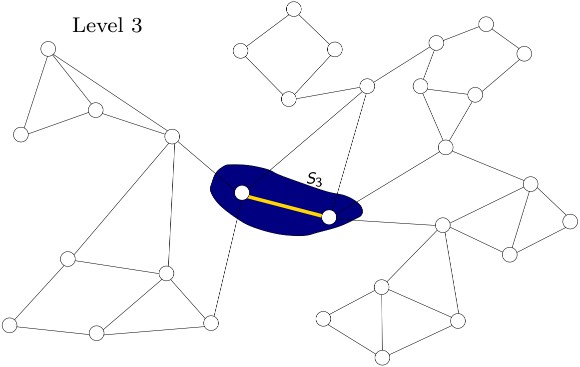
Given a graph $G = (V, E)$ and a a sequence $V \equiv S_0 \supset S_1 \supset \ldots \supset S_l$ of subsets of V

$\mathcal{M} = (V, E \cup E_1 \cup \ldots \cup E_l)$ where
$\forall i = 1, 2, \ldots, l$, edge $(u, v) \in S_i \times S_i$ belongs to $E_i \Leftrightarrow$ each shortest path from u to v does not contain a node in S_i $(w(u, v) = d(u, v))$



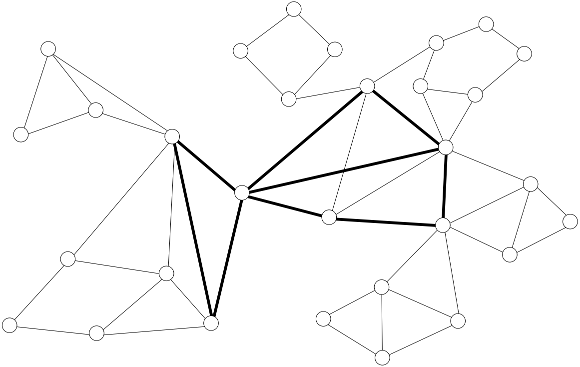
Given a graph $G = (V, E)$ and a a sequence $V \equiv S_0 \supset S_1 \supset \ldots \supset S_l$ of subsets of V

$\mathcal{M} = (V, E \cup E_1 \cup \ldots \cup E_l)$ where
$\forall i = 1, 2, \ldots, l$, edge $(u, v) \in S_i \times S_i$ belongs to $E_i \Leftrightarrow$ each shortest path from u to v does not contain a node in S_i $(w(u, v) = d(u, v))$

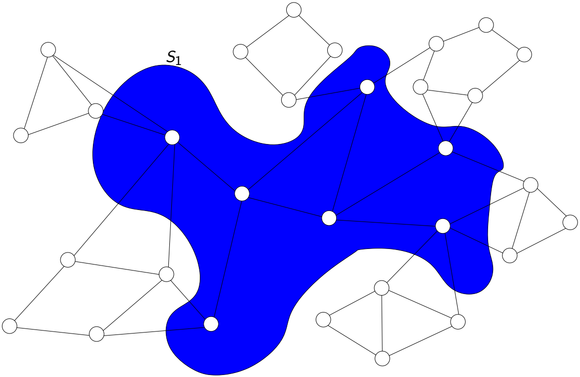


Given a graph $G = (V, E)$ and a a sequence $V \equiv S_0 \supset S_1 \supset \ldots \supset S_l$ of subsets of V

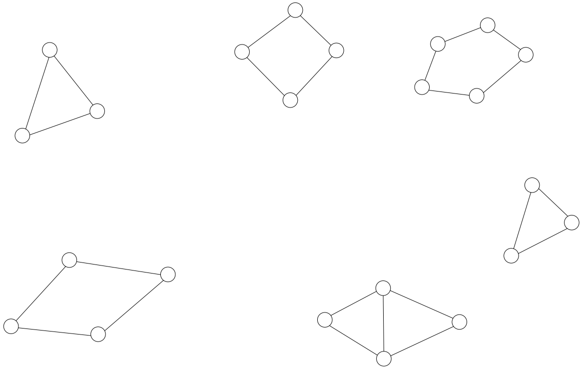
$\mathcal{M} = (V, E \cup E_1 \cup \ldots \cup E_l)$ where
$\forall i = 1, 2, \ldots, l$, edge $(u, v) \in S_i \times S_i$ belongs to $E_i \Leftrightarrow$ each shortest path from u to v does not contain a node in S_i $(w(u, v) = d(u, v))$



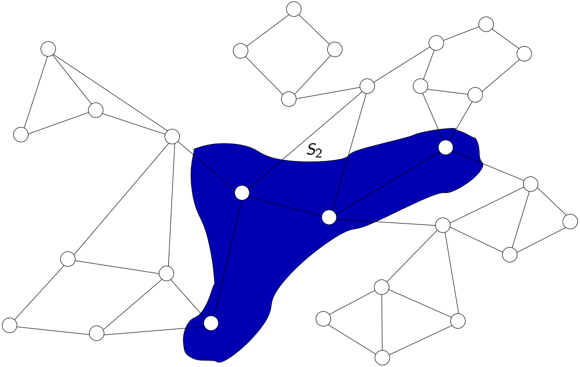
Components tree $T_{\mathcal{M}}$: tree of connected components induced by $V \setminus S_i$



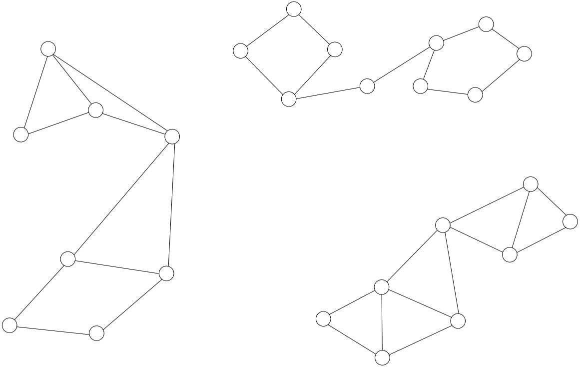
Components tree $T_{\mathcal{M}}$: tree of connected components induced by $V \setminus S_i$



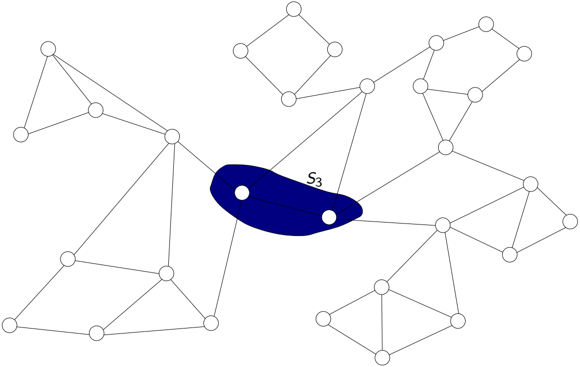
Components tree $T_{\mathcal{M}}$: tree of connected components induced by $V \setminus S_i$



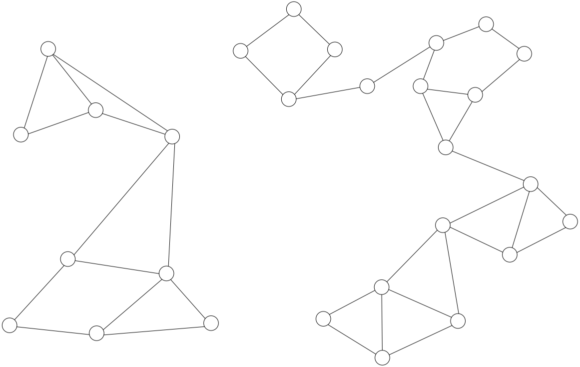
Components tree $T_{\mathcal{M}}$: tree of connected components induced by $V \setminus S_i$



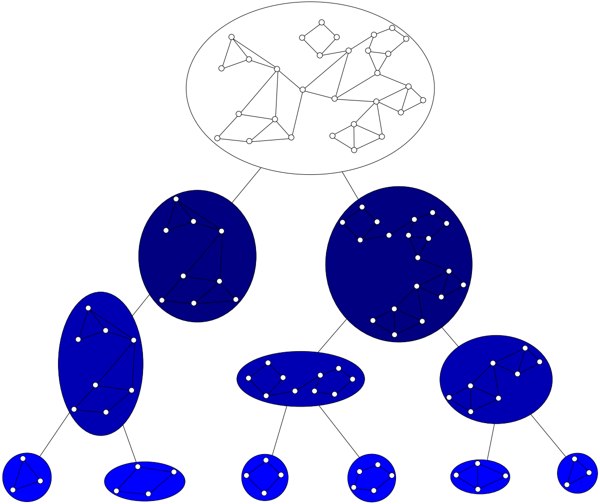
Components tree $T_{\mathcal{M}}$: tree of connected components induced by $V \setminus S_i$



Components tree $T_{\mathcal{M}}$: tree of connected components induced by $V \setminus S_i$



Components tree $T_{\mathcal{M}}$: tree of connected components induced by $V \setminus S_i$

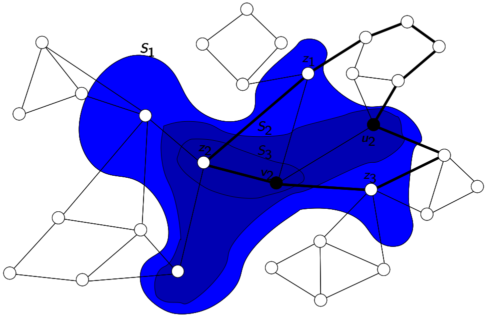


Outline

Outline

Definition (Barrier levels)

- $P_u(v)$ set of nodes x such that x is different from u and v, and x belongs to at least one shortest path from u to v in G.
- $s_u(v) = \begin{cases} \max\{maxlevel(x) \mid x \in P_u(v)\} & \text{if } P_u(v) \not\equiv \emptyset \\ 0 & \text{if } P_u(v) \equiv \emptyset \end{cases}$



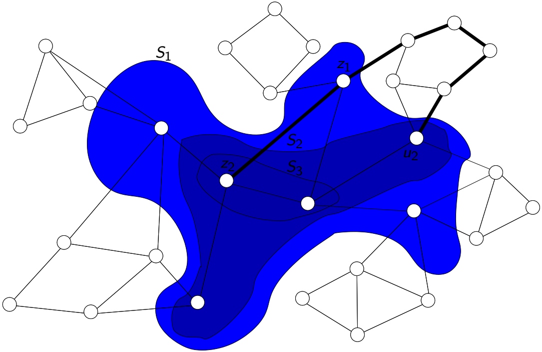
$$P_{u_2}(v_2) = \{\ldots, z_1, z_2, \ldots, z_3 \ldots\}$$

$$s_{u_2}(v_2) = 3$$

Property 1

Lemma

$(u, v) \in S_j \times S_j$ is a level edge of level j if and only if there exists a path from u to v in G and $s_u(v) < j$.



$(u_2, z_2) \in S_1 \times S_1$
$(u_2, z_2) \in S_2 \times S_2$
$s_{u_2}(z_2) = 1$ (given by z_1)

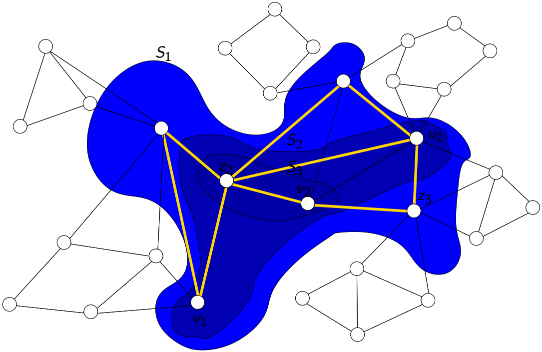
$j = 1 \quad s_{u_2}(z_2) \geq j$
$j = 2 \quad s_{u_2}(z_2) < j$

(u_2, z_2) is a 2-level edge

Property 2

Lemma

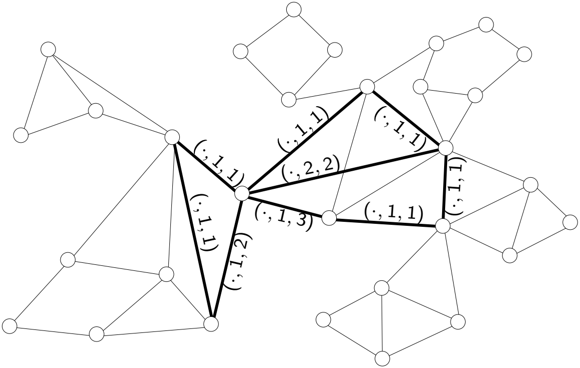
If $e = (u, v) \in \bigcup_{i=1}^{l} E_i$, then there exist $j, k \in \mathbb{N}$, $1 \leq j \leq k \leq l$, such that $e \in E_i$, $\forall i \in \{j, \ldots, k\}$, and $e \notin E_i$, $\forall i \notin \{j, \ldots, k\}$.



$(v_1, z_2) \in E_1, E_2$
$(z_2, v_2) \in E_1, E_2, E_3$
$(u_2, z_3) \in E_1$
$(z_2, u_2) \in E_2$

Definition

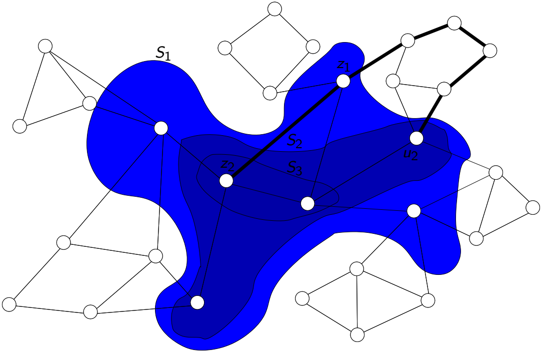
$$w_{\mathcal{M}}(u,v) = (\bar{d}(u,v), f(u,v), \ell(u,v)) = \begin{cases} (d(u,v), s_u(v)+1, \min\{maxlevel(u), maxlevel(v)\}) & \text{If } (u,v) \text{ is a level edge} \\ (w(u,v), 0, 0) & \text{otherwise} \end{cases}$$



Property 3

Lemma

$(u, v) \in S_1 \times S_1$ is a level edge if and only if there exists a path from u to v in G and $s_u(v) < \min\{maxlevel(u), maxlevel(v)\}$.



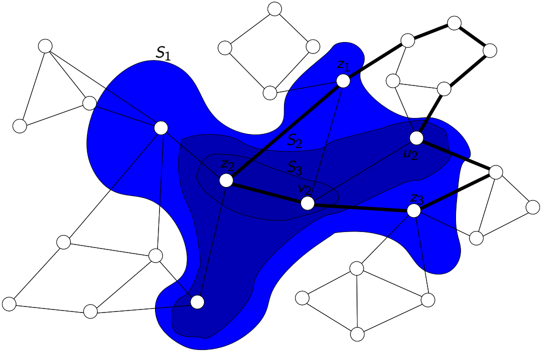
$s_{u_2}(z_2) = 1$ (given by z_1)
$\min\{maxlevel(u_2), maxlevel(z_2)\} = 2$
$s_{u_2}(z_2) < \min\{maxlevel(u_2), maxlevel(z_2)\}$
(u_2, z_2) is a level edge

$s_{u_2}(v_2) = 3$ (given by z_2)
$\min\{maxlevel(u_2), maxlevel(v_2)\} = 2$
$s_{u_2}(v_2) > \min\{maxlevel(u_2), maxlevel(v_2)\}$
(u_2, v_2) is not a level edge

Property 3

Lemma

$(u, v) \in S_1 \times S_1$ is a level edge if and only if there exists a path from u to v in G and $s_u(v) < \min\{maxlevel(u), maxlevel(v)\}$.



$s_{u_2}(z_2) = 1$ (given by z_1)
$\min\{maxlevel(u_2), maxlevel(z_2)\} = 2$
$s_{u_2}(z_2) < \min\{maxlevel(u_2), maxlevel(z_2)\}$
(u_2, z_2) is a level edge

$s_{u_2}(v_2) = 3$ (given by z_2)
$\min\{maxlevel(u_2), maxlevel(v_2)\} = 2$
$s_{u_2}(v_2) > \min\{maxlevel(u_2), maxlevel(v_2)\}$
(u_2, v_2) is not a level edge

Outline

In order to compute $s_u(v)$

We define

- a weighted graph $G_u = (V, E, w_u)$ for each $u \in S_1$
- an algebraic structure $(\mathcal{K}, \min_{\mathcal{K}}, \oplus_{\mathcal{K}})$

such that, if $w_u : E \rightarrow \mathcal{K}$, then

- for each $v \in V$ the distance from u to v in G_u is $d_u(v) = (d(u, v), s_u(v))$
- $d_u(v)$ can be computed by Dijkstra's shortest paths algorithm

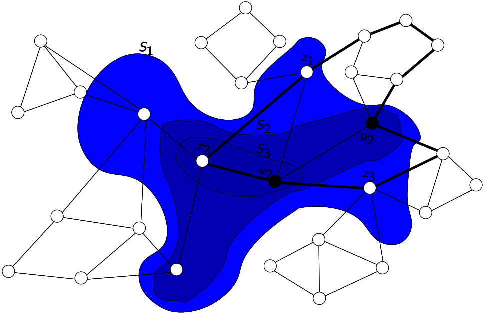
Theorem

$(\mathcal{K}, \min_{\mathcal{K}}, \oplus_{\mathcal{K}}, (\infty, 0), (0, 0))$ *is a closed semiring.*

Theorem

$d_u(v) = (d(u, v), s_u(v))$, *for each* $v \in V$.

We can use Dijkstra's shortest paths algorithm to compute $d(u, v)$ and $s_u(v)$



Outline

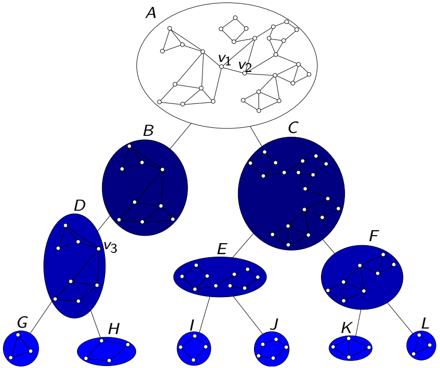
For each node $v \in V$, we store $\mathtt{S}[v] = maxlevel(v)$

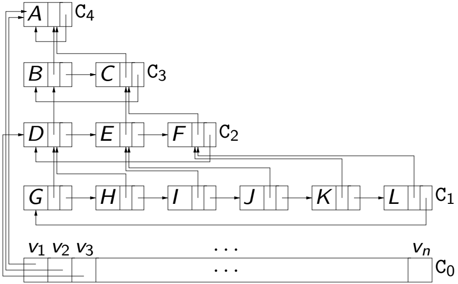
OVERLAY

For each $u \in S_1$	$\|S_1\|$ times
1. Compute G_u	$O(n+m)$
2. Run Dijkstra on G_u from u and get $d(u,v)$ and $s_u(v)$	$O(m + n\log n)$
3. For each $(u,v), v \in S_1$	
4. If $(s_u(v) < \min\{\mathtt{S}[u], \mathtt{S}[v]\}$ and $d(u,v) \neq \infty)$ then	$O(n)$
5. $w_{\mathcal{M}}(u,v) := (d(u,v), s_u(v)+1, \min\{\mathtt{S}[u], \mathtt{S}[v]\})$	

Lemma

OVERLAY requires $O(|S_1|(m + n\log n))$ time



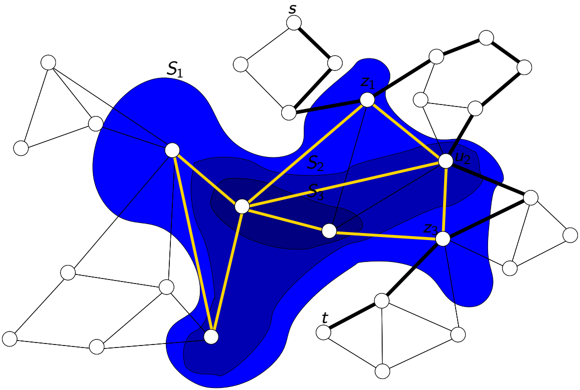


Components tree

Is computed in linear time and space

Outline

1. Introduction
   - Previous works
   - Results of the paper
   - Multi level overlay graphs
2. Computation of Multi level overlay graphs
   - Characterization of level edges
   - Computation of barrier levels
   - Computation of $\mathcal{M}$ and $T_{\mathcal{M}}$
   - Shortest paths queries
3. Maintenance of Multi-level overlay graphs
4. Conclusions



The distance from s to t in $\mathcal{M}_{st}$ is the same that in G (HSW06)

Build $\mathcal{M}_{st}$ and compute distance in $\mathcal{M}_{st}$ costs less then compute distance in G

Outline

1. Introduction
   - Previous works
   - Results of the paper
   - Multi level overlay graphs
2. Computation of Multi level overlay graphs
   - Characterization of level edges
   - Computation of barrier levels
   - Computation of $\mathcal{M}$ and $T_{\mathcal{M}}$
   - Shortest paths queries
3. Maintenance of Multi-level overlay graphs
4. Conclusions

Purpose

Update a multi-level overlay graph $\mathcal{M}$ of a graph G and the components tree when G changes as a consequence of a weight increase or a weight decrease operation on an edge of G

Note

- Distance queries can be answered as in the static case
- The components tree does not change

- To built $\mathcal{M}$, we use $|S_1|$ times Dijkstra's algorithm
- To update $\mathcal{M}$, we use the dynamic algorithm of FMN00
- It updates a shortest paths tree while weight increase or weight decrease modifications occur
- Procedure OVERLAY does not store the shortest paths tree rooted in the nodes in S_1
- Hence, we need a procedure that first computes the shortest paths trees and then computes $\mathcal{M}$

OVERLAY-2

COMPUTE T_u, $u \in S_1$

1. For each $u \in S_1$
2. Compute G_u
3. Run Dijkstra on G_u from u and get T_u containing $d(u, v)$ and $s_u(v)$

COMPUTE $\mathcal{M}$

4. For each $(u, v) \in S_1$
5. If $(s_u(v) < \min\{\mathtt{S}[u], \mathtt{S}[v]\}$ and $d(u, v) \neq \infty)$ then
6. $w_{\mathcal{M}}(u, v) := (d(u, v), s_u(v) + 1, \min\{\mathtt{S}[u], \mathtt{S}[v]\})$

Lemma

OVERLAY-2 requires $O(|S_1|(m + n) \log n))$ *time and* $O(|S_1|(n + m))$ *space*

UPDATE-$\mathcal{M}$

1. Update graphs $G_u, u \in S_1$ — $O(|S_1|n)$
2. Update trees $T_u, u \in S_1$ — $O(\Delta\sqrt{m}\log(n))$
3. COMPUTE $\mathcal{M}$ — $O(|S_1|n + m)$

Lemma

UPDATE-$\mathcal{M}$ requires $O(|S_1|n + m + \Delta\sqrt{m}\log(n))$ time

Here Δ is the number of pairs in $S_1 \times V$ that changes the distance as a consequence of a modification of G

Outline

Results

- Compute a multi-level overlay graph in $O(|S_1|(m + n \log n))$
- Answer to distance queries theoretically faster than Dijkstra's algorithm
- Update a multi-level overlay graph in $O(|S_1|n + m + \Delta\sqrt{m}\log(n))$

Future works

Experimental evaluation