Computing All-Pairs Shortest Paths by Leveraging Low Treewidth

Léon Planken[1] Mathijs de Weerdt[1]
Roman van der Krogt[2]

[1]Delft University of Technology

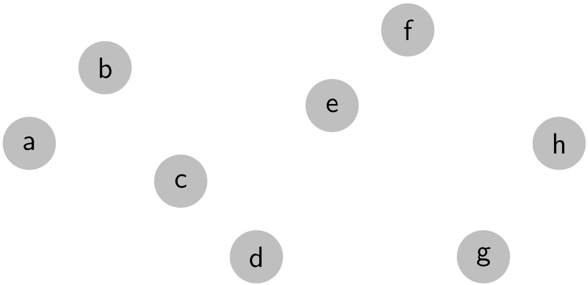
[2]Cork Constraint Computation Centre

15 June 2011

Outline

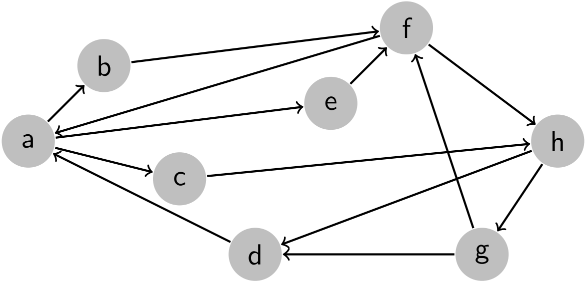
- Computing all-pairs shortest paths
  1. Introduction
  2. Motivation
  3. Existing algorithms
- Leveraging low treewidth

All-pairs shortest paths



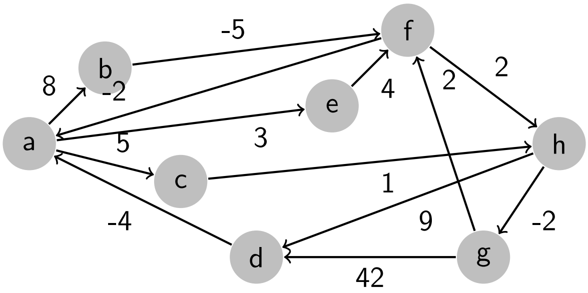
We consider directed graphs graphs on n vertices and m edges.

All-pairs shortest paths



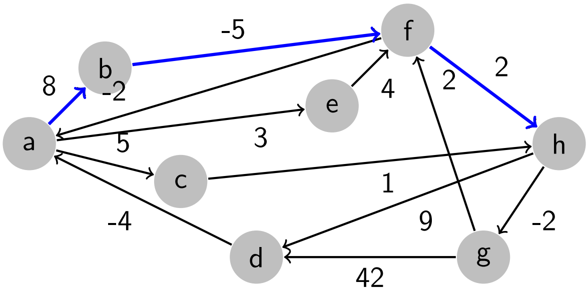
We consider directed graphs graphs on n vertices and m edges.

All-pairs shortest paths



Arcs in the graph are labelled by real-valued weights.

All-pairs shortest paths



We are interested in finding shortest paths. . .

All-pairs shortest paths

D	a	b	c	d	e	f	g	h
a	0	8	5	14	3	3	3	5
b	-7	0	-2	6	-4	-5	-5	-3
c	-1	7	0	10	2	1	-1	1
d	-4	4	1	0	-1	-1	-1	1
e	2	10	7	15	0	4	4	6
f	-2	6	3	11	1	0	0	2
g	0	8	5	13	3	2	0	4
h	-2	6	3	9	1	0	-2	0

... between all pairs of vertices: distance matrix D.

Motivation

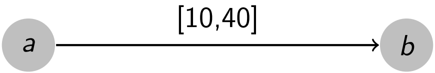
Why is this problem of interest to the ICAPS crowd?

- Shortest paths can clearly be used for spational reasoning
- But for temporal reasoning as well: Simple Temporal Networks

Simple Temporal Networks
Introduction

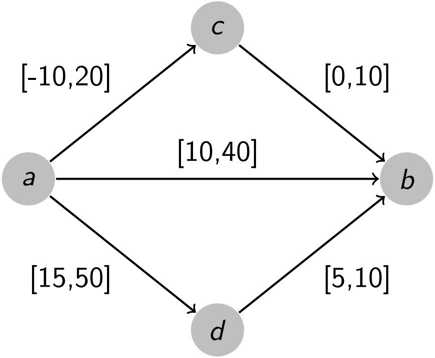
- Proposed in 1991 by Dechter, Meiri and Pearl
- Represent and reason about temporal information
- Nodes represent events
- Arcs represent temporal constraints

Simple Temporal Networks
Example



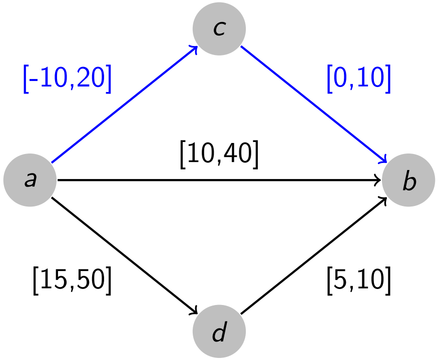
b happens between 10 and 40 time units after a.

Simple Temporal Networks
Example



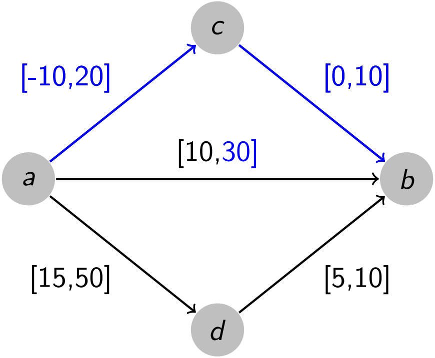
Let's add some constraints.

Simple Temporal Networks
Example



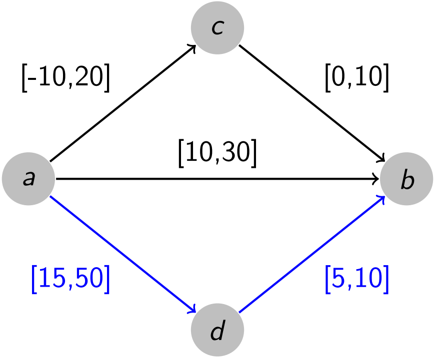
These can be used to infer tighter bounds.

Simple Temporal Networks
Example



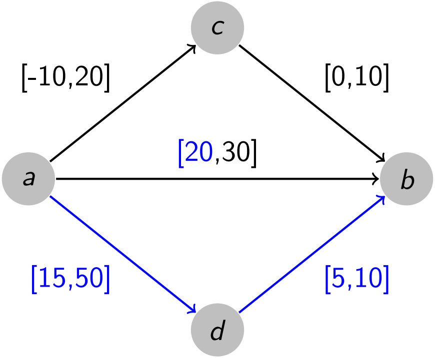
These can be used to infer tighter bounds.

Simple Temporal Networks
Example



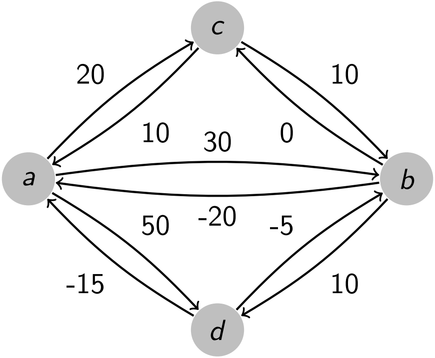
These can be used to infer tighter bounds.

Simple Temporal Networks
Example



These can be used to infer tighter bounds.

Simple Temporal Networks
Example



An equivalent representation uses weighted arcs.

Simple Temporal Networks

Applications

- Scheduling problems (e.g. job shop)
- Temporal planning
- Space missions:
  - NASA's Mars Rover
  - ESA's Mars Express

Algorithms for APSP

- Floyd–Warshall (1959–62): $\mathcal{O}\left(n^3\right)$ time
  - Proposed by Dechter et al. for the STN
  - Very simple to implement

Algorithms for APSP

- Floyd–Warshall (1959–62): $\mathcal{O}\left(n^3\right)$ time
  - Proposed by Dechter et al. for the STN
  - Very simple to implement
- Johnson (1977): $\mathcal{O}\left(nm + n^2 \log n\right)$ time
  - Requires Fibonacci heap (1987)
  - A bit harder to implement
  - Benefits from sparseness

Other algorithms

- Bellman–Ford (1958–62): $\mathcal{O}(nm)$ time
  - Find inconsistency (negative cycles)
  - Find single schedule for events
  - Infer constraints involving a single time point

Other algorithms

- Bellman–Ford (1958–62): $\mathcal{O}(nm)$ time
  - Find inconsistency (negative cycles)
  - Find single schedule for events
  - Infer constraints involving a single time point
- $\mathrm{P^3C}$ (ICAPS'08): $\mathcal{O}\left(nw_d^2\right)$ time
  - Infer constraints for an (interesting) subset of all pairs
  - For this specific problem: state of the art

Outline

- Computing all-pairs shortest paths
- Leveraging low treewidth
    1. Directed path consistency
    2. Snowball
    3. Empirical evaluation
    4. Conclusion

Directed path consistency

Introduction

- Proposed by [Dechter et al., 1991] for determining consistency
- Known from CSP literature
- Given vertex ordering d, runs in $\mathcal{O}\left(nw_d^2\right)$ time
- Prerequisite to $\mathrm{P}^3\mathrm{C}$

Directed path consistency
Algorithm

Given $G = \langle V, E \rangle$ and vertex ordering d:

For $k \leftarrow n$ to 1:

For all $i < j < k$ such that $\{i, k\}, \{j, k\} \in E$:

$w_{i\to j} \leftarrow \min\{w_{i\to j},\ w_{i\to k} + w_{k\to j}\}$

$w_{j\to i} \leftarrow \min\{w_{j\to i},\ w_{j\to k} + w_{k\to i}\}$

$E \leftarrow E \cup \{\{i, j\}\}$

If $w_{i\to j} + w_{j\to i} < 0$ return INCONSISTENT

Return CONSISTENT

Directed path consistency
Algorithm

Given $G = \langle V, E \rangle$ and vertex ordering d:

For $k \leftarrow n$ to 1:

For all $i < j < k$ such that $\{i, k\}, \{j, k\} \in E$:

$w_{i\to j} \leftarrow \min\{w_{i\to j},\ w_{i\to k} + w_{k\to j}\}$

$w_{j\to i} \leftarrow \min\{w_{j\to i},\ w_{j\to k} + w_{k\to i}\}$

$E \leftarrow E \cup \{\{i, j\}\}$

If $w_{i\to j} + w_{j\to i} < 0$ return INCONSISTENT

Return CONSISTENT

Directed path consistency
Algorithm

Given $G = \langle V, E \rangle$ and vertex ordering d:

For $k \leftarrow n$ to 1:

For all $i < j < k$ such that $\{i, k\}, \{j, k\} \in E$:

$w_{i \to j} \leftarrow \min\{w_{i \to j},\ w_{i \to k} + w_{k \to j}\}$

$w_{j \to i} \leftarrow \min\{w_{j \to i},\ w_{j \to k} + w_{k \to i}\}$

$E \leftarrow E \cup \{\{i, j\}\}$

If $w_{i \to j} + w_{j \to i} < 0$ return INCONSISTENT

Return CONSISTENT

Directed path consistency

Algorithm

Given $G = \langle V, E \rangle$ and vertex ordering d:

For $k \leftarrow n$ to 1:

For all $i < j < k$ such that $\{i, k\}, \{j, k\} \in E$:

$w_{i \to j} \leftarrow \min\{w_{i \to j},\ w_{i \to k} + w_{k \to j}\}$

$w_{j \to i} \leftarrow \min\{w_{j \to i},\ w_{j \to k} + w_{k \to i}\}$

$E \leftarrow E \cup \{\{i, j\}\}$

If $w_{i \to j} + w_{j \to i} < 0$ return INCONSISTENT

Return CONSISTENT

Directed path consistency

Algorithm

Given $G = \langle V, E \rangle$ and vertex ordering d:

For $k \leftarrow n$ to 1:

For all $i < j < k$ such that $\{i, k\}, \{j, k\} \in E$:

$w_{i \to j} \leftarrow \min\{w_{i \to j},\ w_{i \to k} + w_{k \to j}\}$

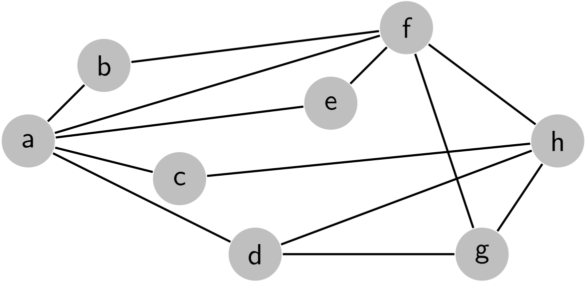
$w_{j \to i} \leftarrow \min\{w_{j \to i},\ w_{j \to k} + w_{k \to i}\}$

$E \leftarrow E \cup \{\{i, j\}\}$

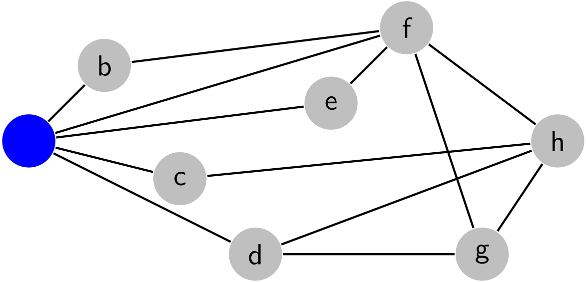
If $w_{i \to j} + w_{j \to i} < 0$ return INCONSISTENT

Return CONSISTENT

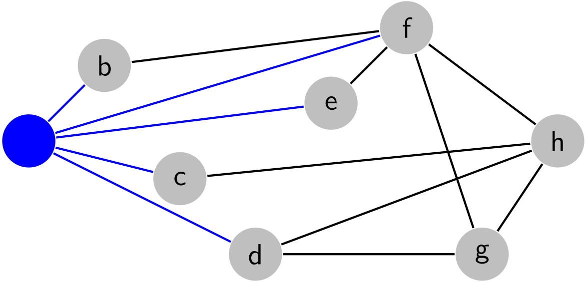
Directed path consistency
Example



Directed path consistency
Example

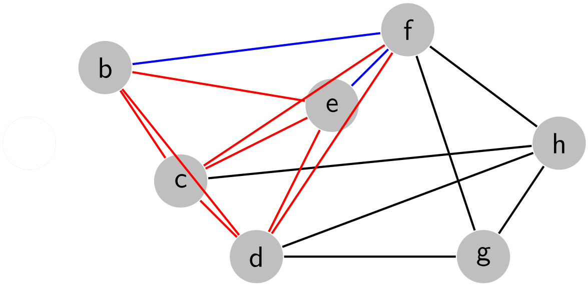


Directed path consistency
Example



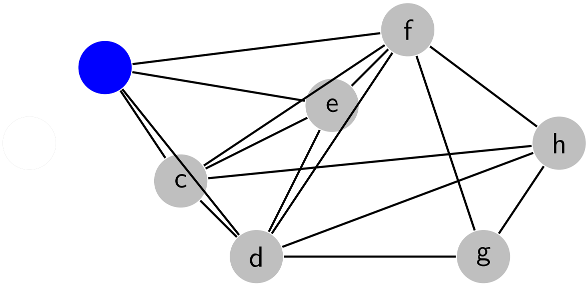
Directed path consistency

Example



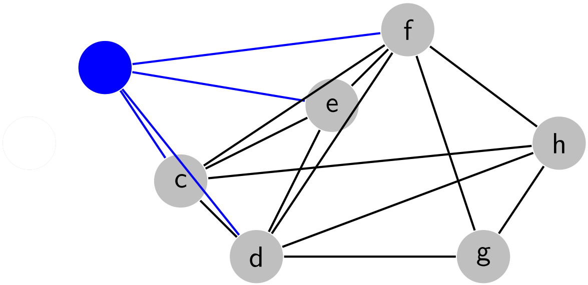
Directed path consistency

Example



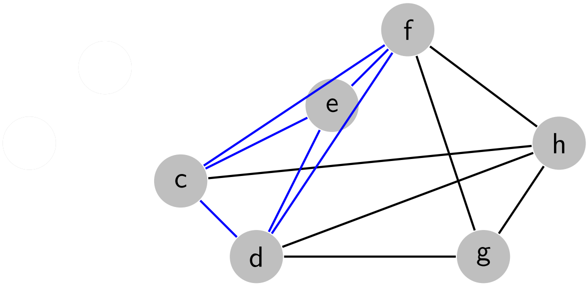
Directed path consistency

Example

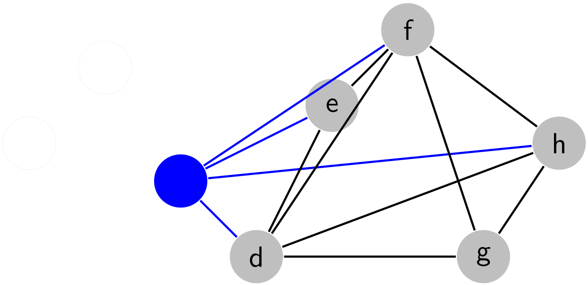


Directed path consistency

Example

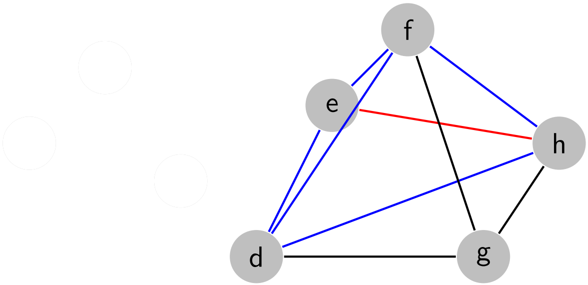


Directed path consistency
Example

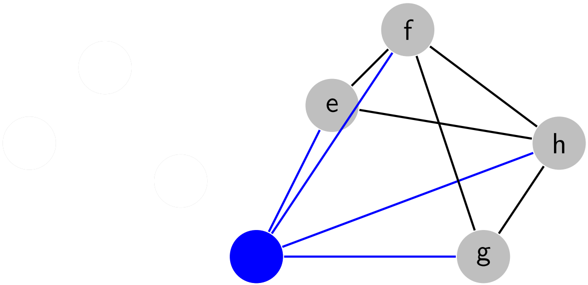


Directed path consistency

Example

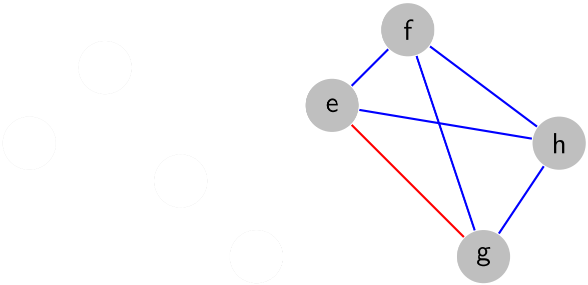


Directed path consistency
Example



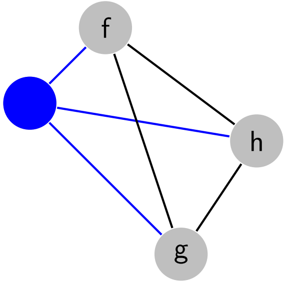
Directed path consistency

Example

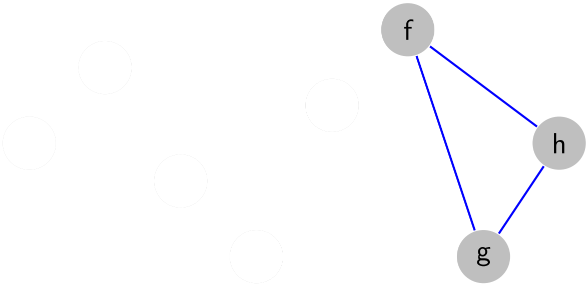


Directed path consistency

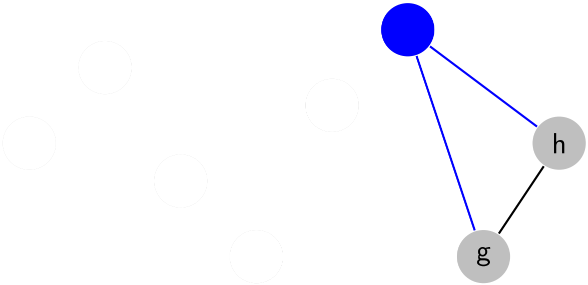
Example



Directed path consistency
Example



Directed path consistency
Example



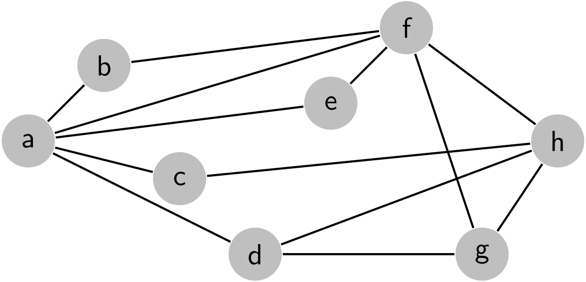
Directed path consistency
Example

Directed path consistency

Example

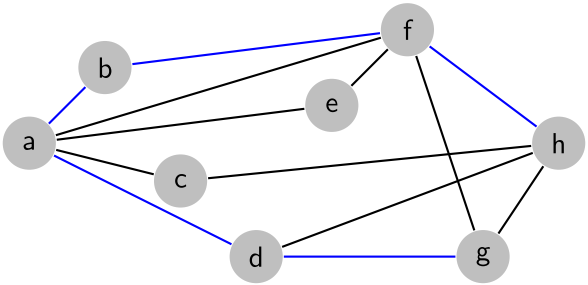
- Edge between last two vertices is minimal
- This example: $w_d = 5$, $\mathrm{fill} = 9$
- Heuristic: minimum degree

Directed path consistency
Example



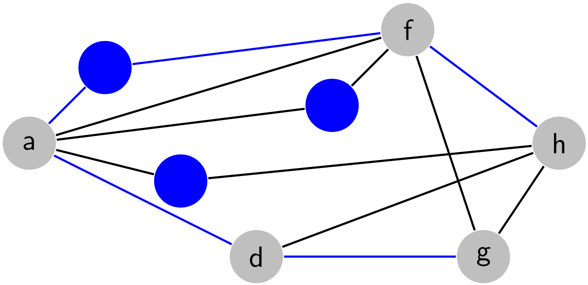
Directed path consistency

Example

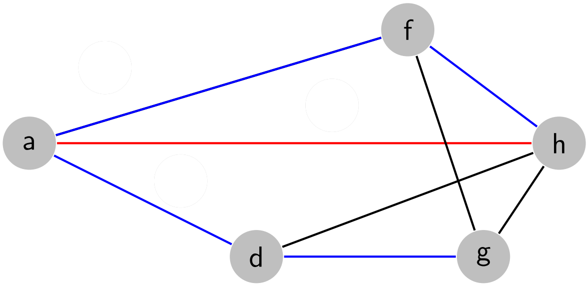


Directed path consistency

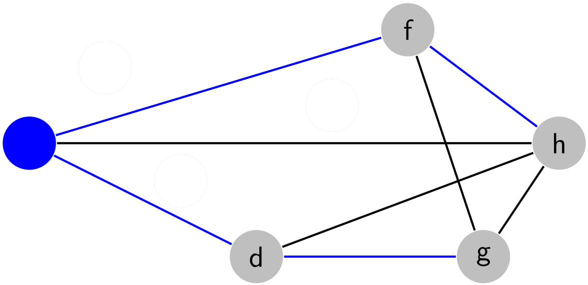
Example



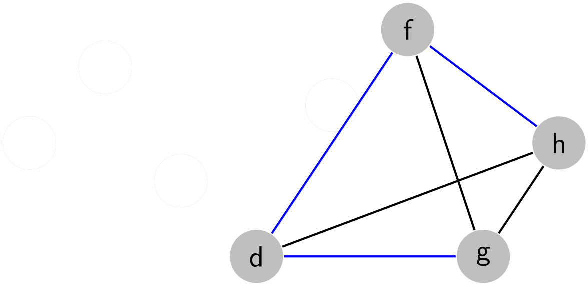
Directed path consistency
Example



Directed path consistency
Example

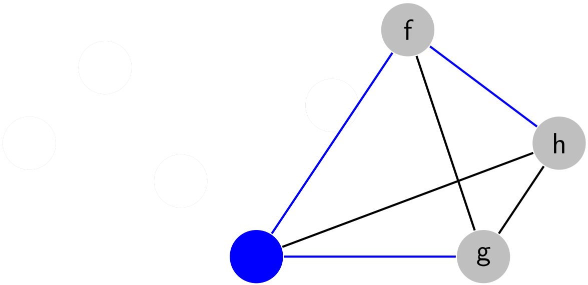


Directed path consistency
Example

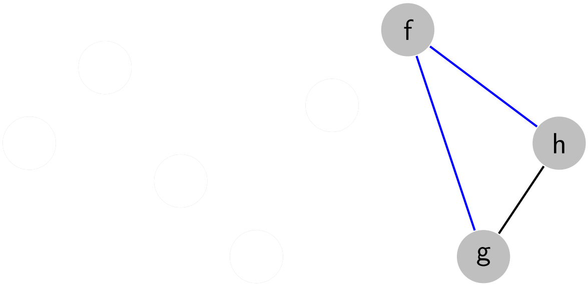


Directed path consistency

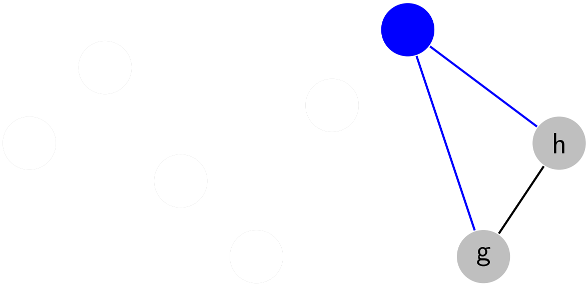
Example



Directed path consistency
Example



Directed path consistency
Example



Directed path consistency
Example

Directed path consistency
Example

- Using minimum degree: $w_d = 3$, $\text{fill} = 2$
- Shortest paths can be found by "looking down"
- We use this in our algorithm

Snowball
Introduction

- Like $\mathrm{P}^3\mathrm{C}$, builds on DPC
- Runs in $\mathcal{O}(nm_c) \subseteq \mathcal{O}(n^2 w_d)$
- Idea: after DPC,
  - $D[1][2]$ and $D[2][1]$ are minimal
  - Shortest path to/from k runs through neighbours $j < k$

Snowball
Algorithm

Given $G = \langle V, E \rangle$ which is DPC along d:

For all $i, j \in V : D[i][j] \leftarrow \infty$

For all $i \in V : D[i][i] \leftarrow 0$

For $k \leftarrow 1$ to n:

For all $j < k$ such that $\{j, k\} \in E$:

For $i \in \{1, \ldots, k-1\}$:

$D[i][k] \leftarrow \min\{D[i][k],\ D[i][j] + w_{j \to k}\}$

$D[k][i] \leftarrow \min\{D[k][i],\ w_{k \to j} + D[j][i]\}$

Return D

Snowball
Algorithm

Given $G = \langle V, E\rangle$ which is DPC along d:

For all $i, j \in V : D[i][j] \leftarrow \infty$

For all $i \in V : D[i][i] \leftarrow 0$

For $k \leftarrow 1$ to n:

For all $j < k$ such that $\{j, k\} \in E$:

For $i \in \{1, \ldots, k-1\}$:

$D[i][k] \leftarrow \min\{D[i][k],\ D[i][j] + w_{j\to k}\}$

$D[k][i] \leftarrow \min\{D[k][i],\ w_{k\to j} + D[j][i]\}$

Return D

Snowball
Algorithm

Given $G = \langle V, E \rangle$ which is DPC along d:

- For all $i, j \in V : D[i][j] \leftarrow \infty$
- For all $i \in V : D[i][i] \leftarrow 0$
- For $k \leftarrow 1$ to n:
  - For all $j < k$ such that $\{j, k\} \in E$:
    - For $i \in \{1, \ldots, k-1\}$:
      - $D[i][k] \leftarrow \min\{D[i][k],\ D[i][j] + w_{j \to k}\}$
      - $D[k][i] \leftarrow \min\{D[k][i],\ w_{k \to j} + D[j][i]\}$
- Return D

Snowball
Algorithm

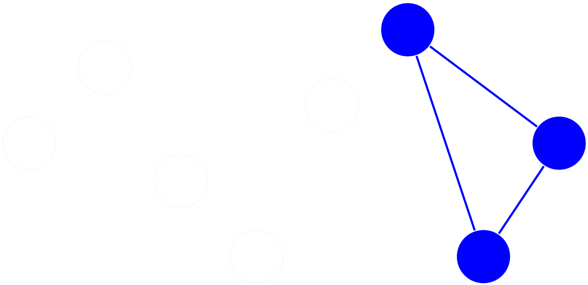
Given $G = \langle V, E \rangle$ which is DPC along d:

- For all $i, j \in V : D[i][j] \leftarrow \infty$
- For all $i \in V : D[i][i] \leftarrow 0$
- For $k \leftarrow 1$ to n:
  - For all $j < k$ such that $\{j, k\} \in E$:
    - For $i \in \{1, \ldots, k-1\}$:
      - $D[i][k] \leftarrow \min\{D[i][k],\ D[i][j] + w_{j \to k}\}$
      - $D[k][i] \leftarrow \min\{D[k][i],\ w_{k \to j} + D[j][i]\}$
- Return D

Snowball
Example

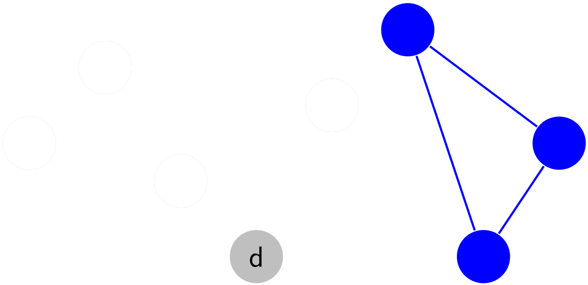
Snowball
Example

Snowball
Example



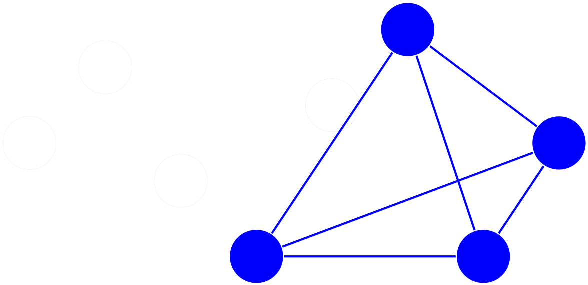
Snowball

Example

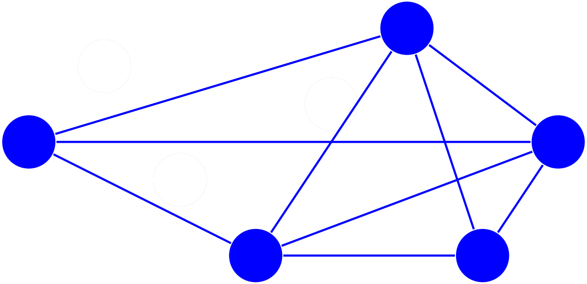


Snowball

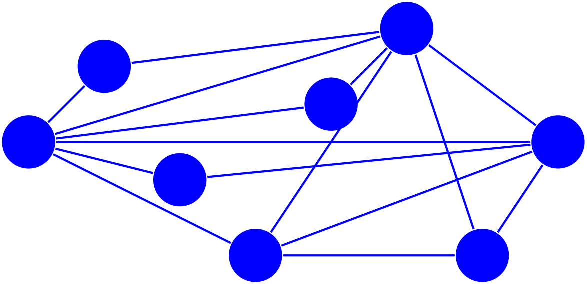
Example



Snowball
Example



Snowball
Example



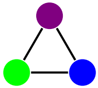
Snowball
Special cases

- If constant w_d exists, it can be found in $\mathcal{O}(n)$ time
  - Then, Snowball runs in $\mathcal{O}(n^2)$ time (optimal).
- Chordal graphs can be identified in $\mathcal{O}(m)$ time
  - Then, Snowball runs in $\mathcal{O}(nm)$ time (also nice).

Snowball
Etymology

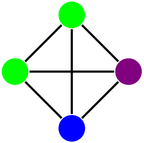
Number of computed shortest paths grows quadratically. . .

Snowball
Etymology



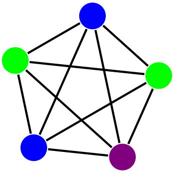
Number of computed shortest paths grows quadratically. . .

Snowball
Etymology



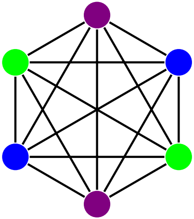
Number of computed shortest paths grows quadratically. . .

Snowball
Etymology



Number of computed shortest paths grows quadratically. . .

Snowball
Etymology



Number of computed shortest paths grows quadratically. . .

Snowball
Etymology

. . . like a snowball.

Empirical evaluation

Introduction

- Compared Floyd–Warshall, Johnson, Snowball
- Java 1.6 (server mode) on Intel Xeon E5430
- Run 10 times, take average CPU time

Empirical evaluation

Benchmark overview

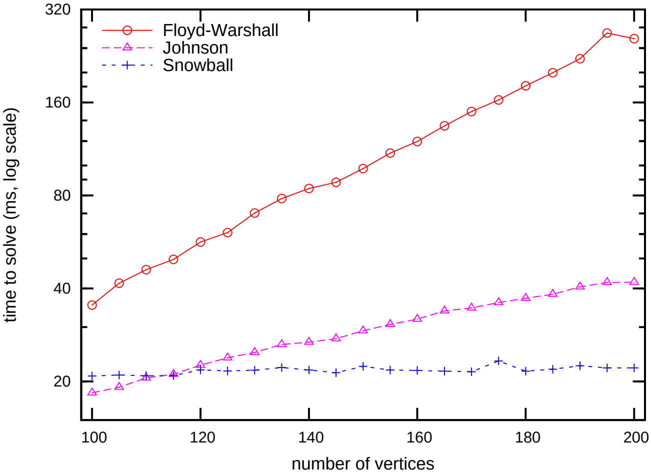
type	#cases	n	m	w_d
Chordal				
- Varying n	130	214–3,125	22,788–637,009	211
- Varying w^*	400	200	985–19,900	5–199
Scale-free				
- Varying n	426	100–200	460–891	38–58
- Varying w_d	190	150	296–2,240	14–103
Diamonds	504	51–379	49–379	2
New York	180	108–4,882	113–8,108	2–51
Job-shop	600	5–241	8–3,840	3–62
HTN	121	500–625	748–1,515	2–144

Empirical evaluation
Benchmark overview

type	#cases	n	m	w_d
Chordal				
- Varying n	130	214–3,125	22,788–637,009	211
- Varying w^*	400	200	985–19,900	5–199
Scale-free				
- Varying n	426	100–200	460–891	38–58
- Varying w_d	190	150	296–2,240	14–103
Diamonds	504	51–379	49–379	2
New York	180	108–4,882	113–8,108	2–51
Job-shop	600	5–241	8–3,840	3–62
HTN	121	500–625	748–1,515	2–144

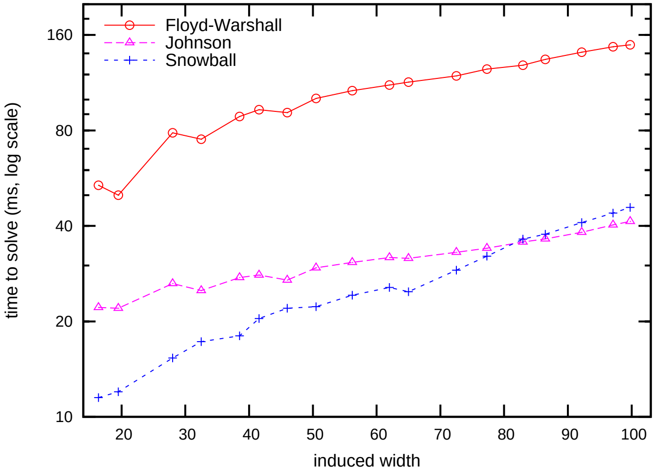
Empirical evaluation

Scale-free graphs, $w_d \in [38, 58]$

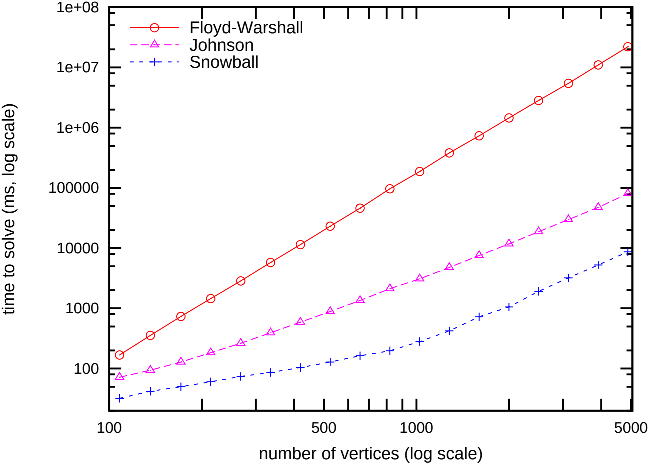


Empirical evaluation

Scale-free graphs, $n = 150$

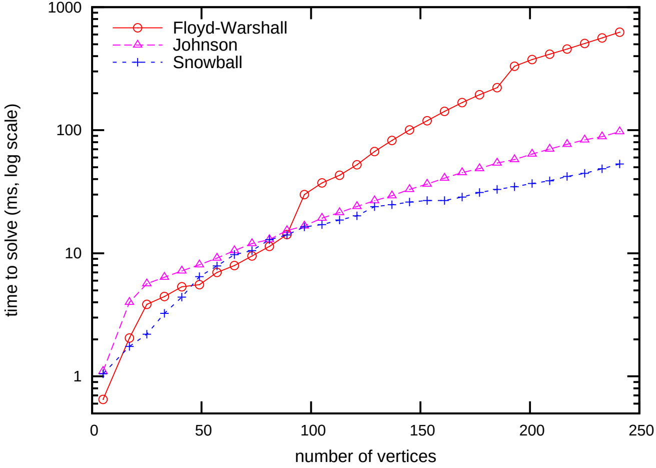


Empirical evaluation
New York City road network, $w_d \in [2, 51]$



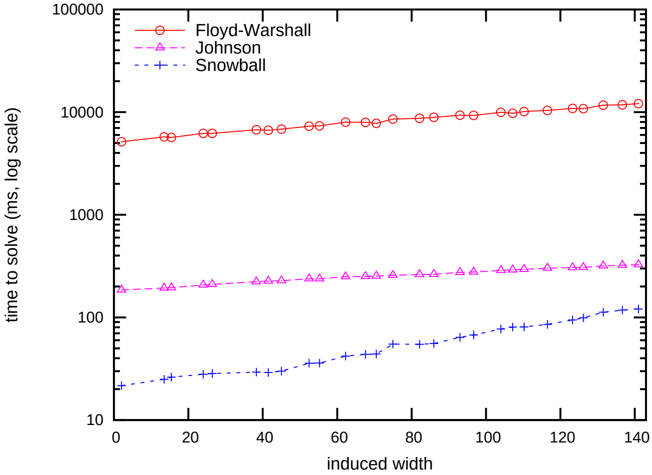
Empirical evaluation

STNs from job-shop benchmarks, $w_d \in [3, 62]$



Empirical evaluation

STNs from HTN benchmarks, $n \in [500, 625]$



Conclusion

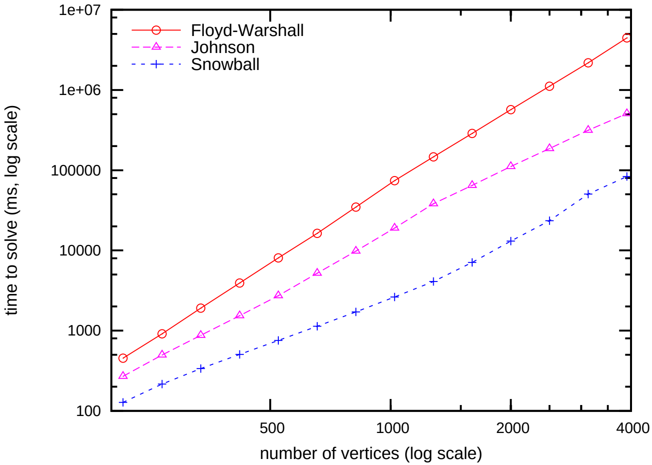
- Proposed a new, simple APSP algorithm:
  - Graphs of constant treewidth: $\mathcal{O}\left(n^2\right)$ time
  - Chordal graphs: $\mathcal{O}\left(nm\right)$ time
  - General graphs: $\mathcal{O}\left(nm_c\right) \subseteq \mathcal{O}\left(n^2 w_d\right)$ time
- Empirically seen to outperform competitors in most cases

Future work

- More efficient Snowball: $\mathcal{O}\left(nw_d^2 + n^2 s\right)$ time, for $s < w_d$
- Compare against [Pettie 2004]: $\mathcal{O}\left(nm + n^2 \log\log n\right)$ time
- Incremental/dynamic version?

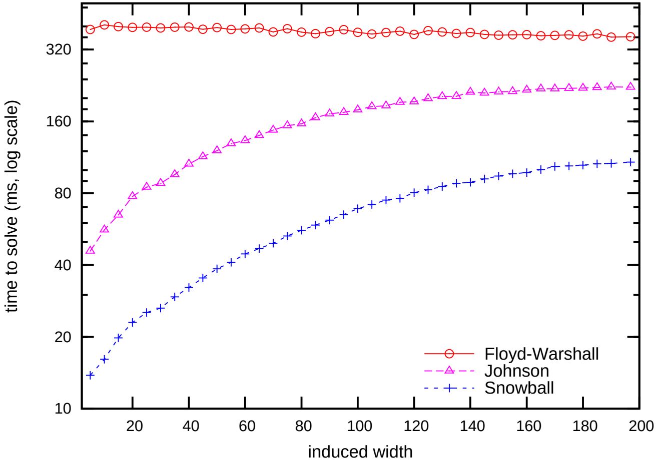
Empirical evaluation

Chordal graphs, $w^* = 211$



Empirical evaluation

Chordal graphs, $n = 200$



Empirical evaluation

"Diamonds" benchmark, $w_d = 2$