Route Planning for Bicycles – Exact Constrained Shortest Paths made Practical via Contraction Hierarchy

Sabine Storandt

University of Stuttgart
Germany

ICAPS 2012

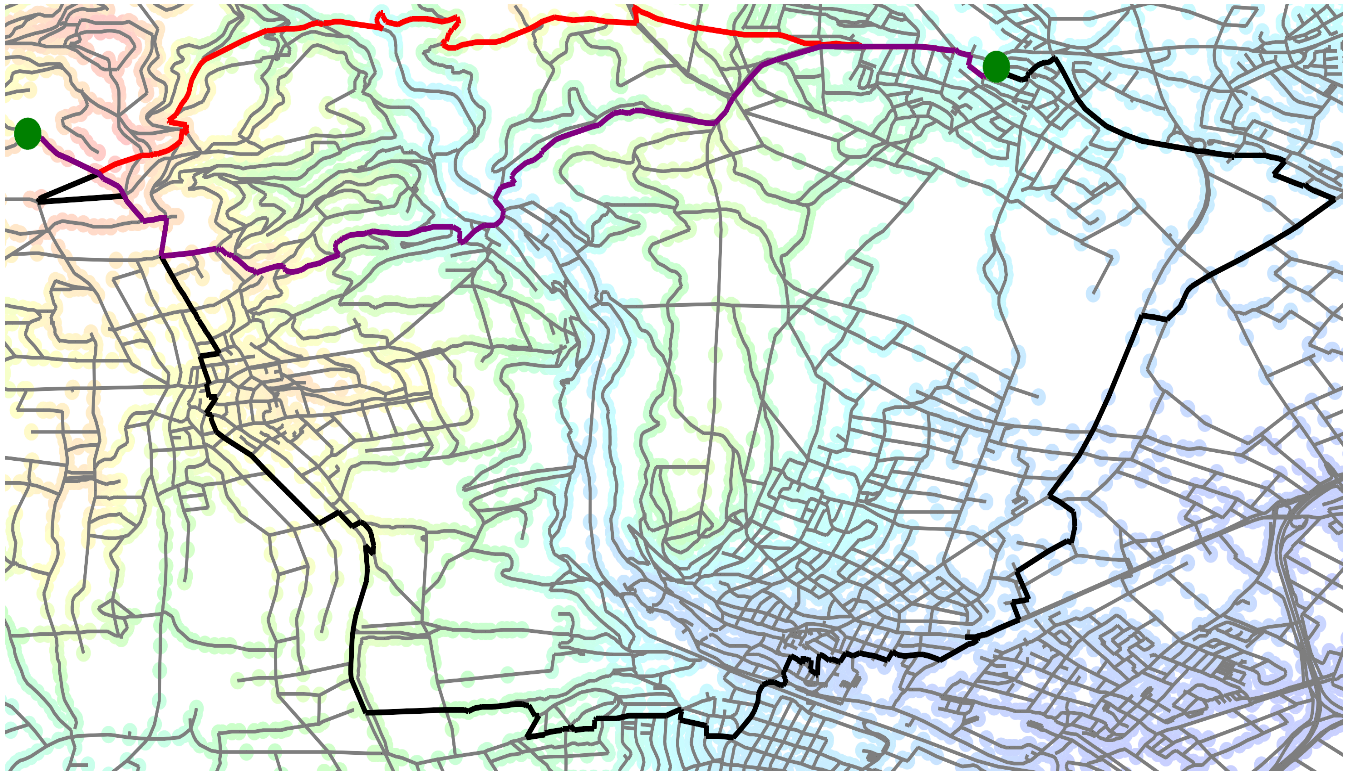
MOTIVATION

Bicycle route from A to B

- should be short
- but bear not too much hard climbs

Optimization Problem

Find the shortest path from A to B with a (positive) height difference smaller than H.

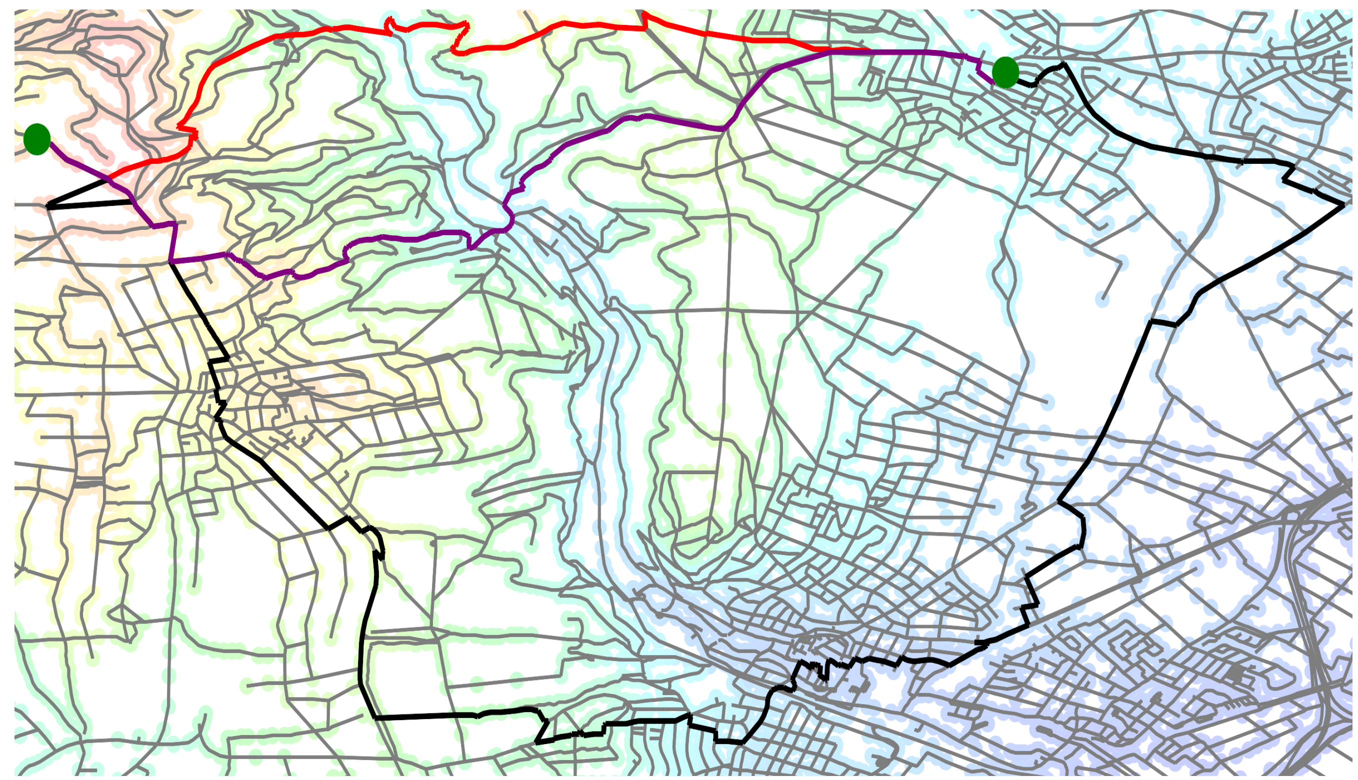


	length	height difference
RED:	7.5km	517m
BLACK:	19.1km	324m
PURPLE:	7.7km	410m

MOTIVATION

Bicycle route from A to B

- should be short
- but bear not too much hard climbs



	length	height difference
RED:	7.5km	517m
BLACK:	19.1km	324m
PURPLE:	7.7km	410m

Optimization Problem

Find the shortest path from A to B with a (positive) height difference smaller than H.

Constrained Shortest Path(CSP)
NP-hard

FORMAL PROBLEM DEFINITION

Given

$G(V, E)$ (street) graph

$c : E \to \mathbb{R}_0^+$ cost

$r : E \to \mathbb{R}_0^+$ resource consumption

Goal

for $s, t \in V$, $R \in \mathbb{R}_0^+$ compute minimal cost path p from s to t whose resource consumption does not exceed R

$\min c(p) = \sum_{e \in p} c(e)$

s.t. $r(p) = \sum_{e \in p} r(e) \leq R$

FORMAL PROBLEM DEFINITION

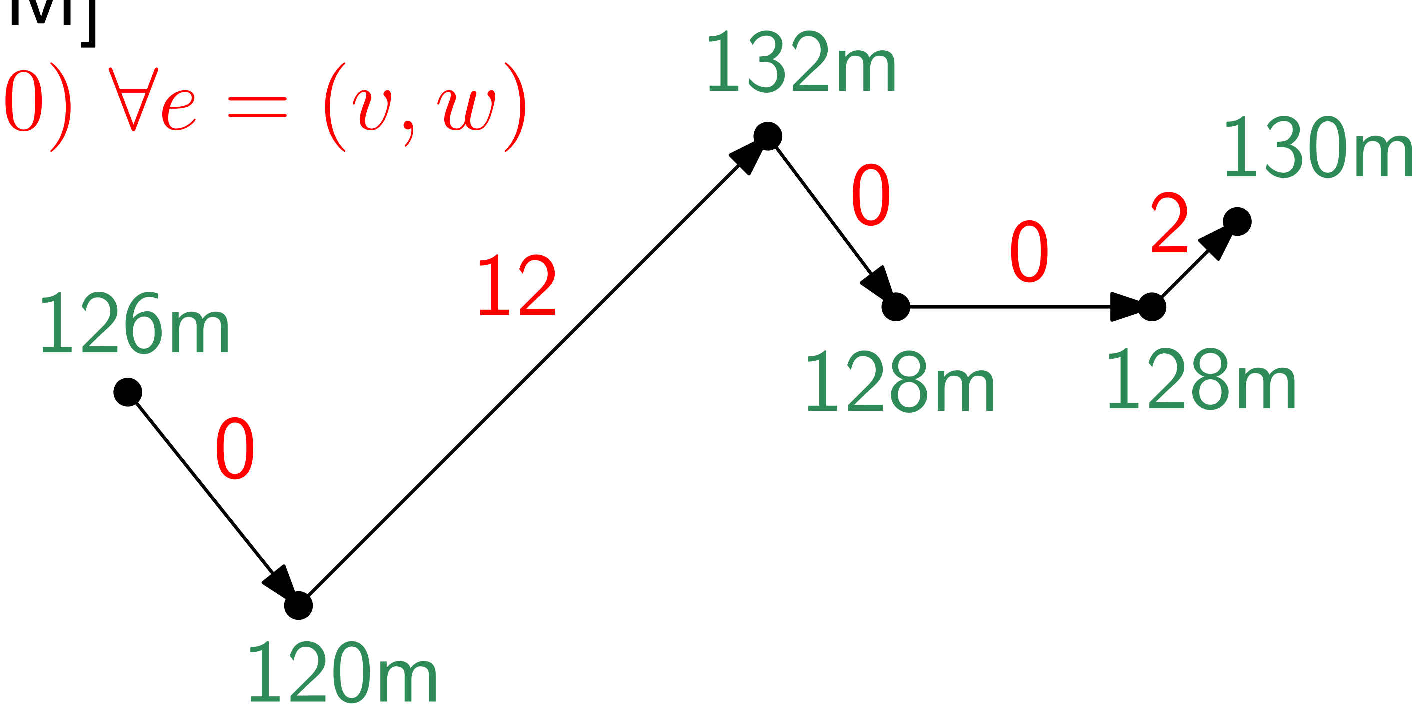
Bicycle Route Planning

costs: euclidean distance [OSM]

resource: positive height difference

$h : V \to \mathbb{Z}$ elevation [SRTM]

$r(e) = \max(h(w) - h(v), 0) \ \forall e = (v, w)$



CONTRIBUTION

Adaption of speed-up techniques for the shortest path problem to reduce

- query time
- space consumption

for exact CSP computation in large street networks.

Focus Contraction Hierarchy

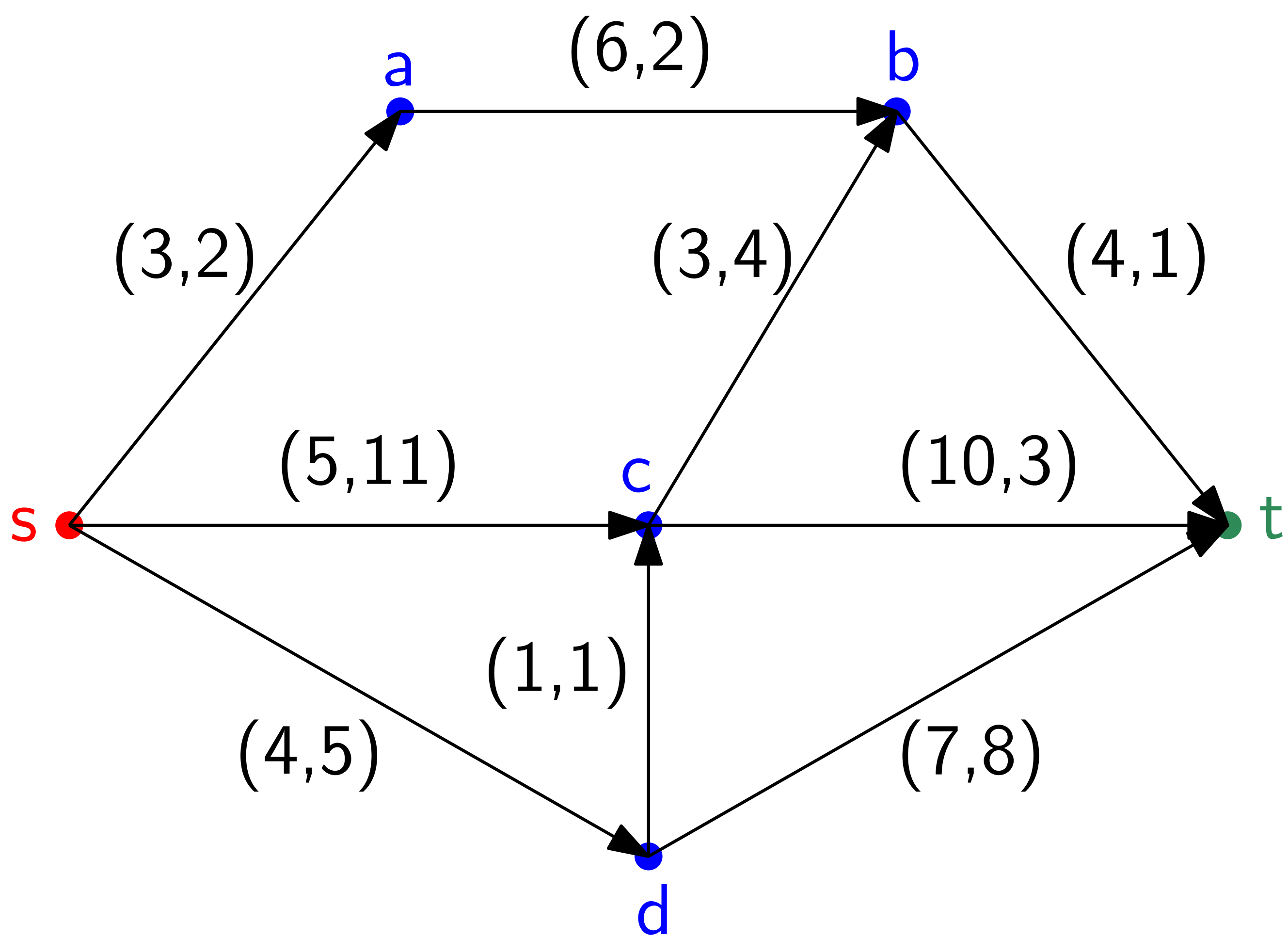
LABEL SETTING ALGORITHM

[Aggarwal, Aneja, and Nair 1982]

Approach
Assign to each node the list of pareto-optimal tuples.

Pareto-optimal $\widehat{=}$ no dominating path exists

p' **dominates** p if
$c(p') \leq c(p)$ and
$r(p') \leq r(p)$



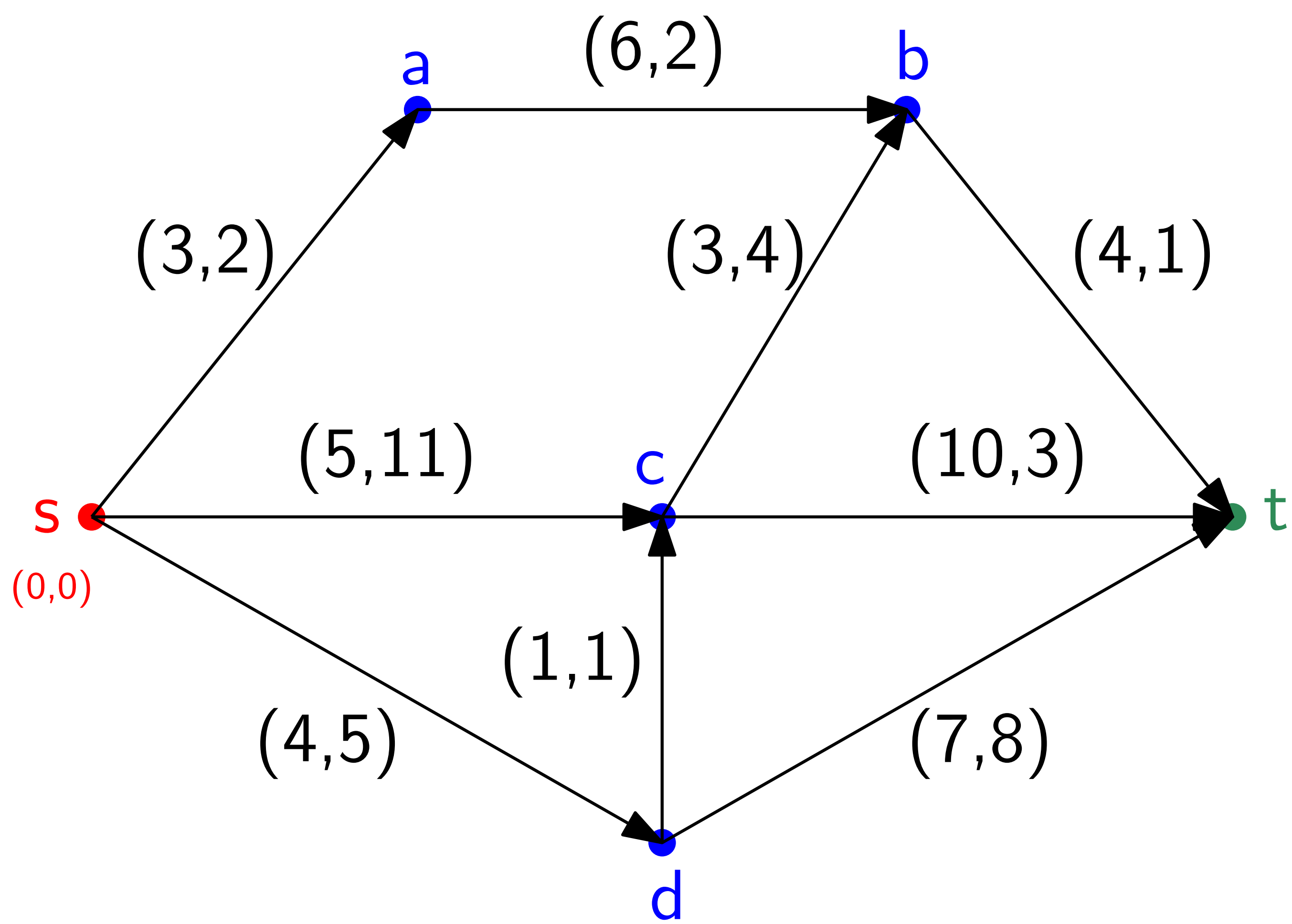
LABEL SETTING ALGORITHM

[Aggarwal, Aneja, and Nair 1982]

Approach
Assign to each node the list of pareto-optimal tuples.

Pareto-optimal $\widehat{=}$ no dominating path exists

p' **dominates** p if
$c(p') \leq c(p)$ and
$r(p') \leq r(p)$



PQ = (0,0,s)

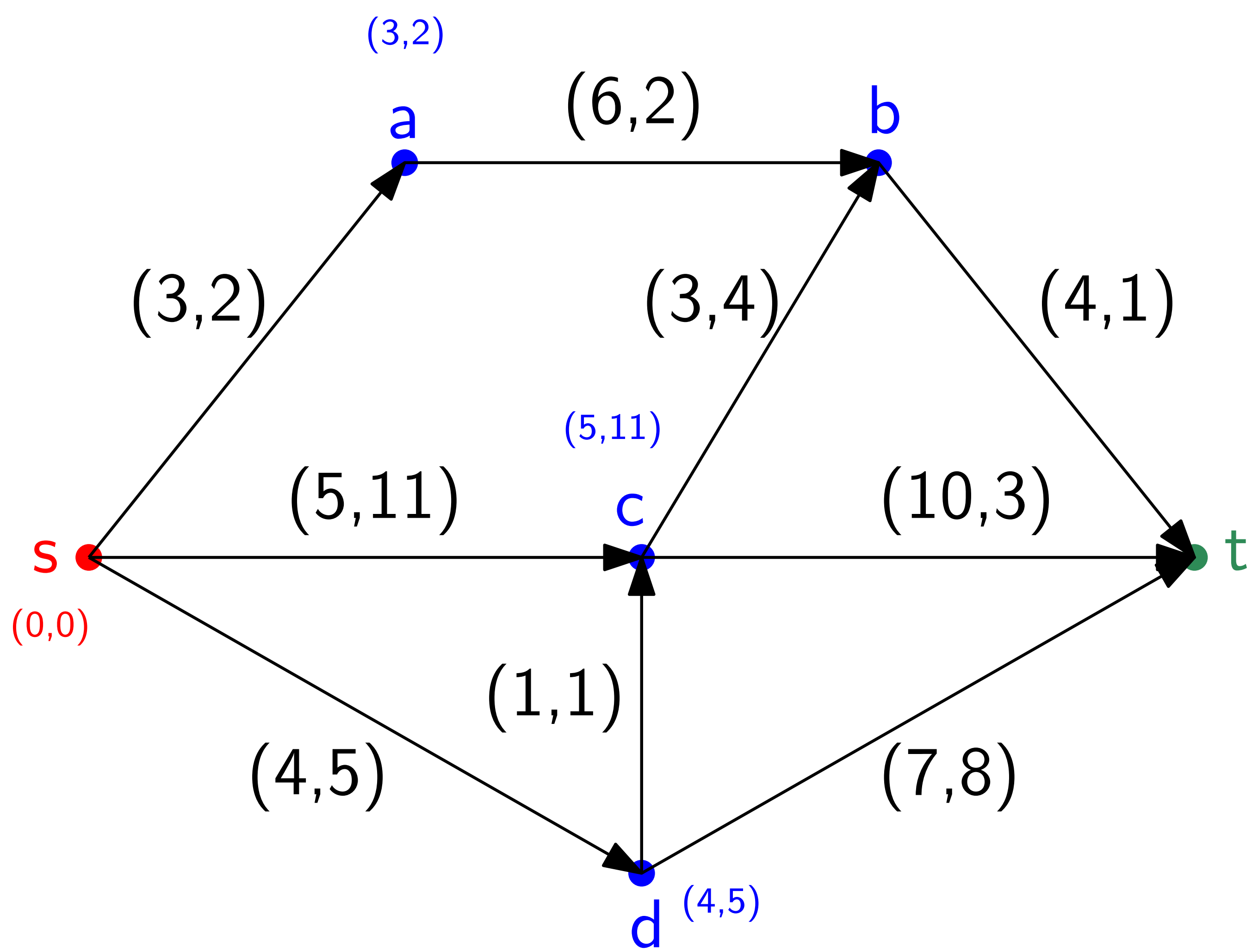
LABEL SETTING ALGORITHM

[Aggarwal, Aneja, and Nair 1982]

Approach
Assign to each node the list of pareto-optimal tuples.

Pareto-optimal $\widehat{=}$ no dominating path exists

p' **dominates** p if
$c(p') \leq c(p)$ and
$r(p') \leq r(p)$



PQ = (3,2,a), (4,5,d), (5,11,c)

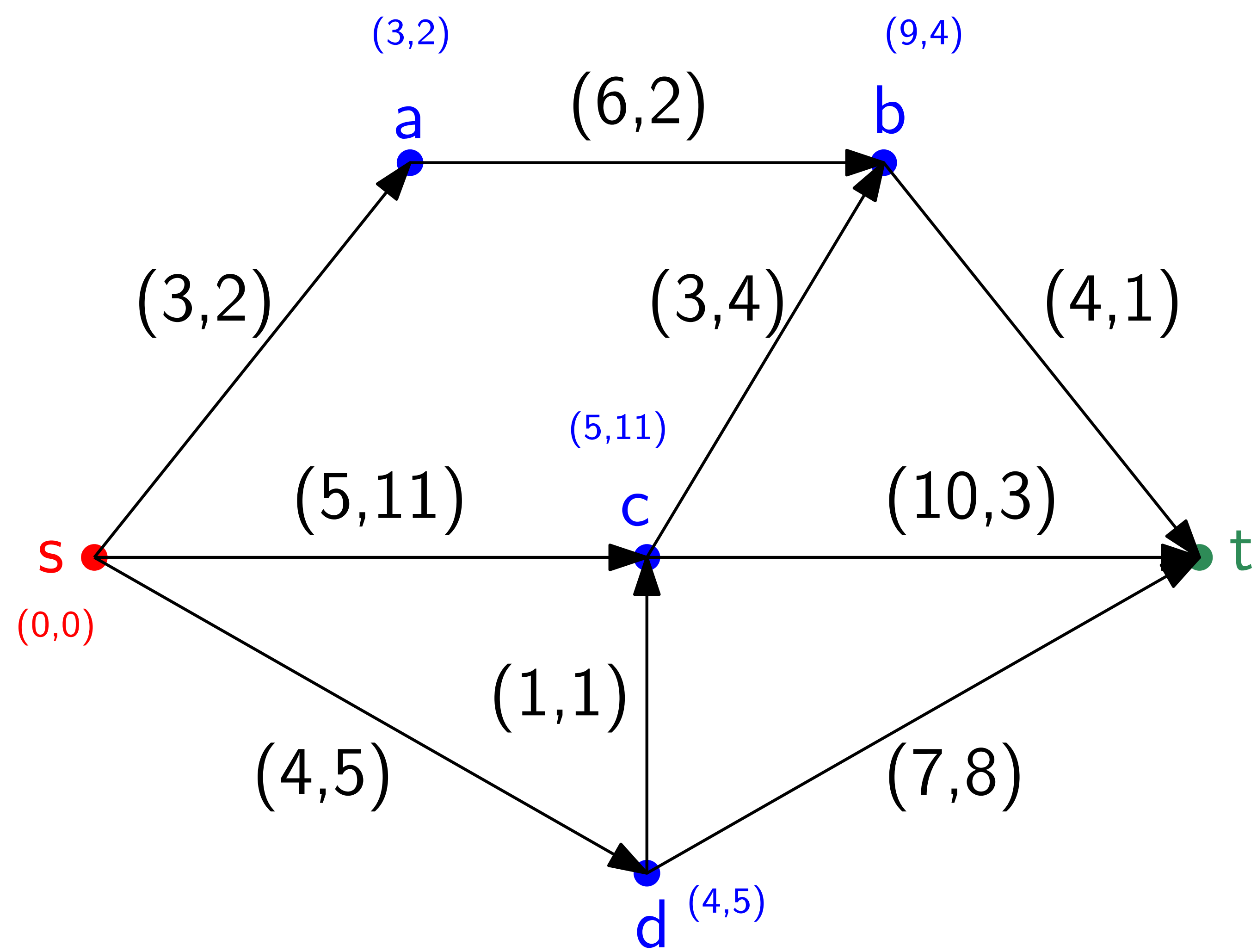
LABEL SETTING ALGORITHM

[Aggarwal, Aneja, and Nair 1982]

Approach
Assign to each node the list of pareto-optimal tuples.

Pareto-optimal $\widehat{=}$ no dominating path exists

p' **dominates** p if
$c(p') \leq c(p)$ and
$r(p') \leq r(p)$



PQ = (4,5,d), (5,11,c), (9,4,b)

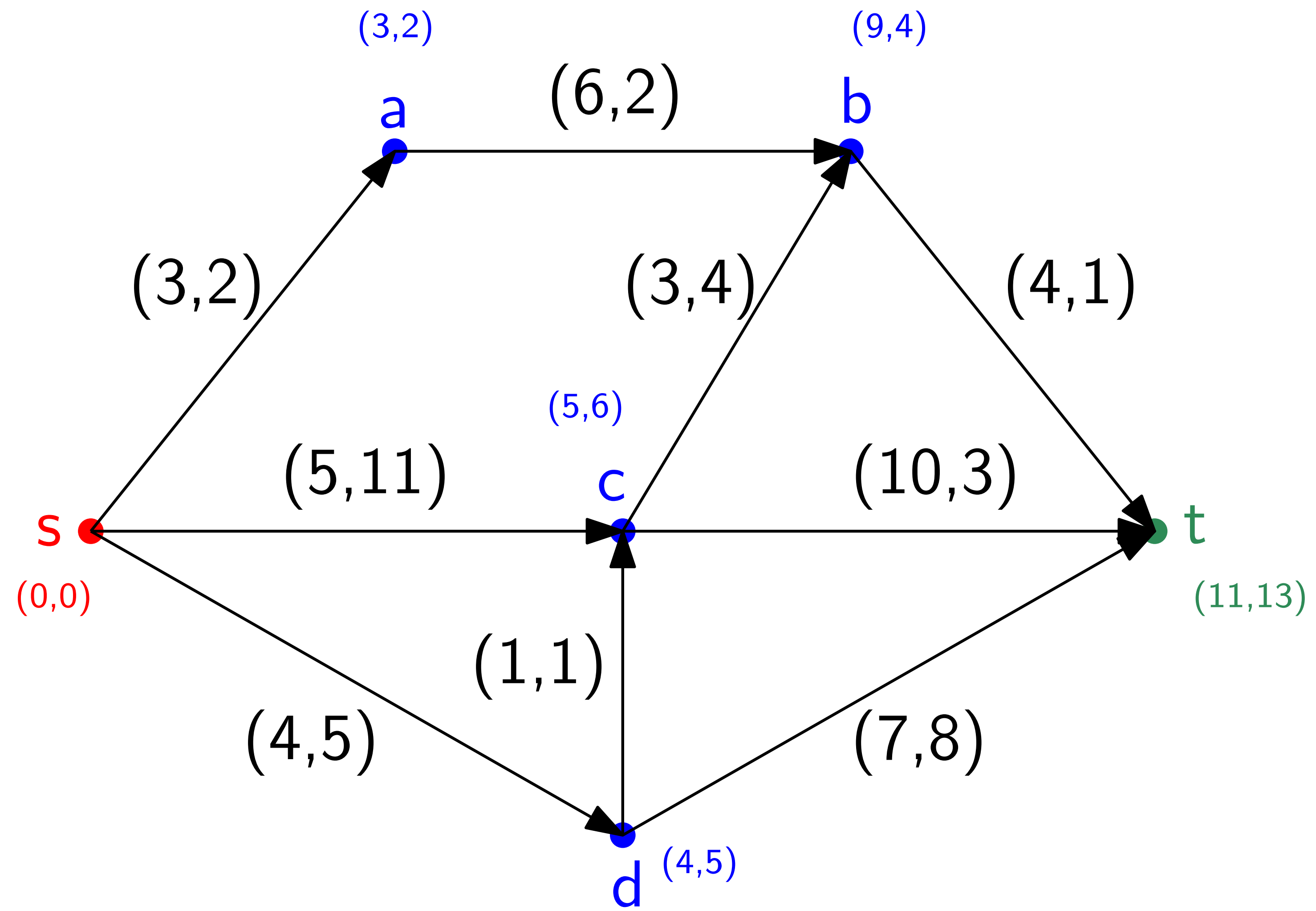
LABEL SETTING ALGORITHM

[Aggarwal, Aneja, and Nair 1982]

Approach
Assign to each node the list of pareto-optimal tuples.

Pareto-optimal $\widehat{=}$ no dominating path exists

p' **dominates** p if
$c(p') \leq c(p)$ and
$r(p') \leq r(p)$



PQ = (5,6,c), (9,4,b)

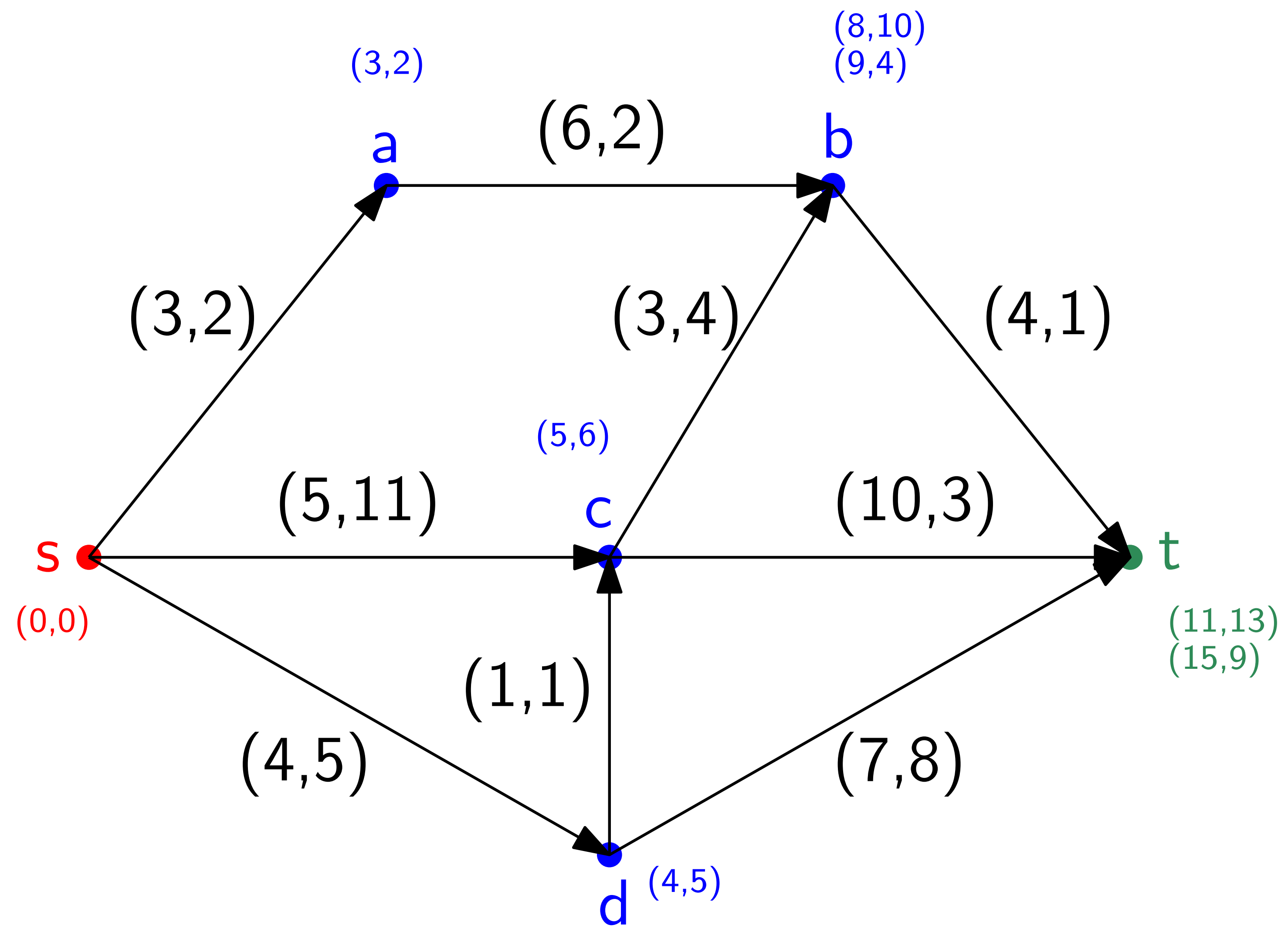
LABEL SETTING ALGORITHM

[Aggarwal, Aneja, and Nair 1982]

Approach
Assign to each node the list of pareto-optimal tuples.

Pareto-optimal $\widehat{=}$ no dominating path exists

p' **dominates** p if
$c(p') \leq c(p)$ and
$r(p') \leq r(p)$



PQ $=$ (8,10,b),(9,4,b)

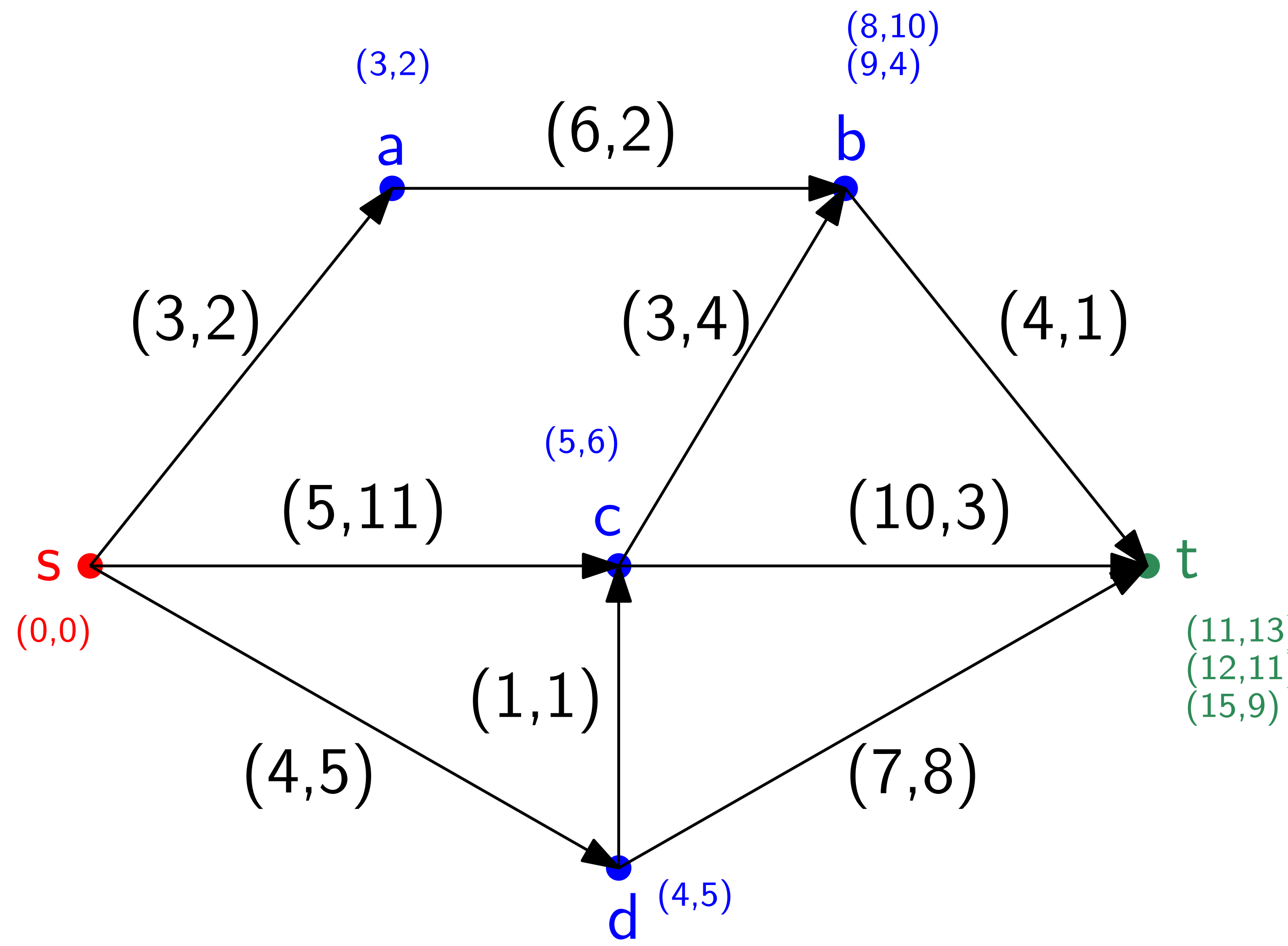
LABEL SETTING ALGORITHM

[Aggarwal, Aneja, and Nair 1982]

Approach
Assign to each node the list of pareto-optimal tuples.

Pareto-optimal $\widehat{=}$ no dominating path exists

p' **dominates** p if
$c(p') \leq c(p)$ and
$r(p') \leq r(p)$



PQ = (9,4,b)

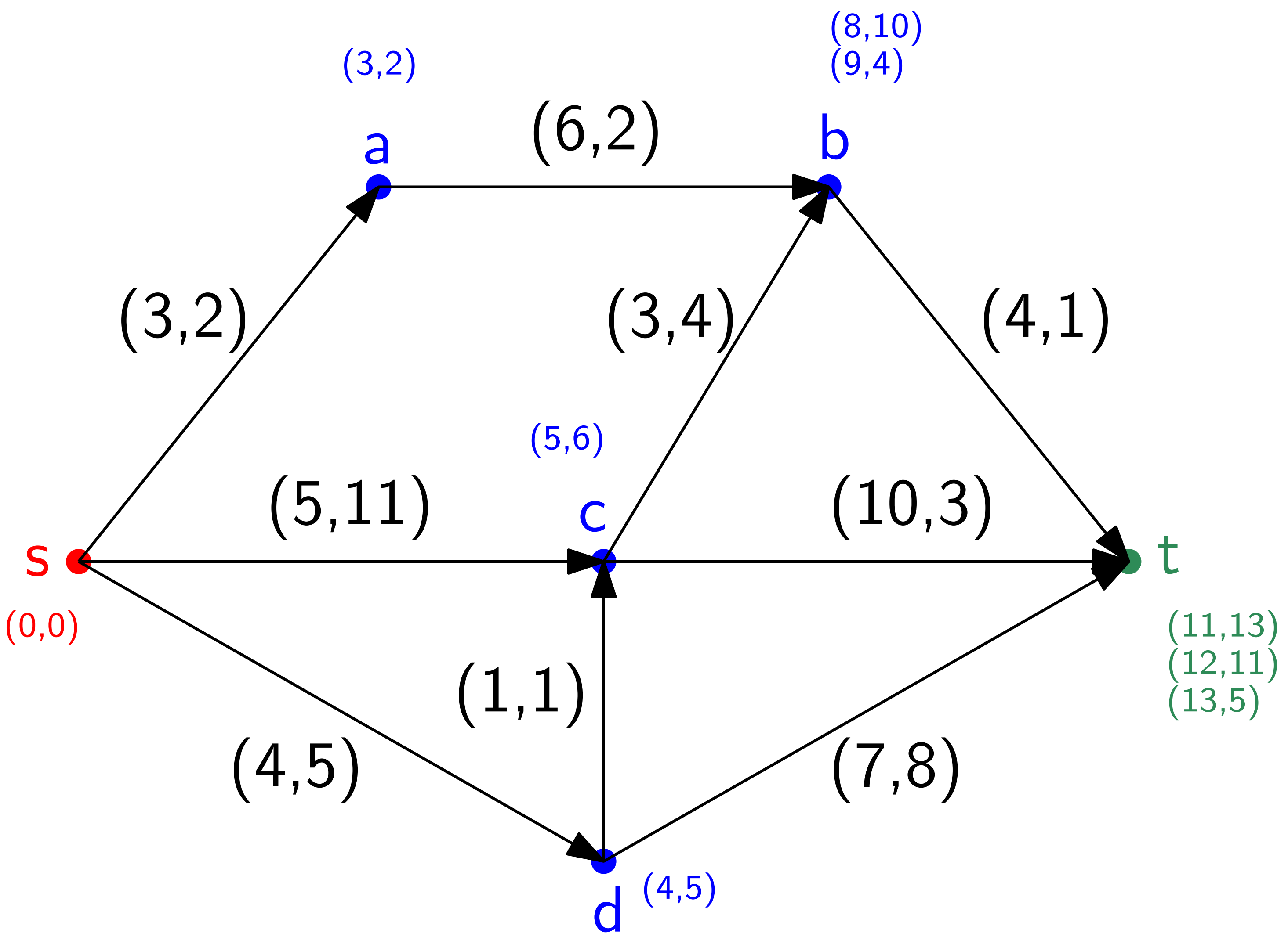
LABEL SETTING ALGORITHM

[Aggarwal, Aneja, and Nair 1982]

Approach
Assign to each node the list of pareto-optimal tuples.

Pareto-optimal $\widehat{=}$ no dominating path exists

p' **dominates** p if
$c(p') \leq c(p)$ and
$r(p') \leq r(p)$



PQ $= \emptyset$

LABEL SETTING ALGORITHM

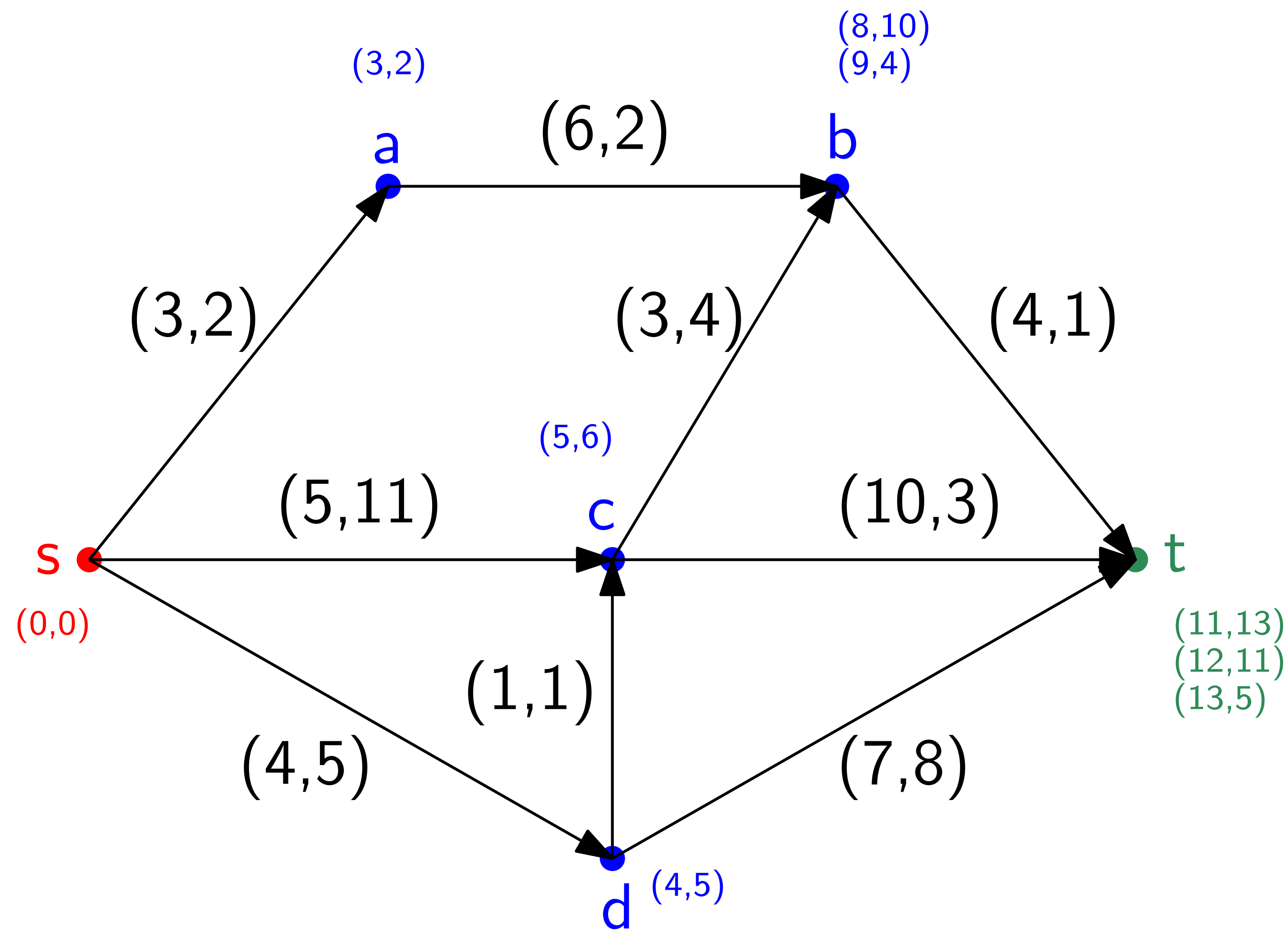
[Aggarwal, Aneja, and Nair 1982]

Approach

Assign to each node the list of pareto-optimal tuples.

Similarities to Dijkstra

- operates directly on the graph
- PQ and edge relaxation
- bidirectional version exists



PQ $= \emptyset$

SIMPLE PRUNING

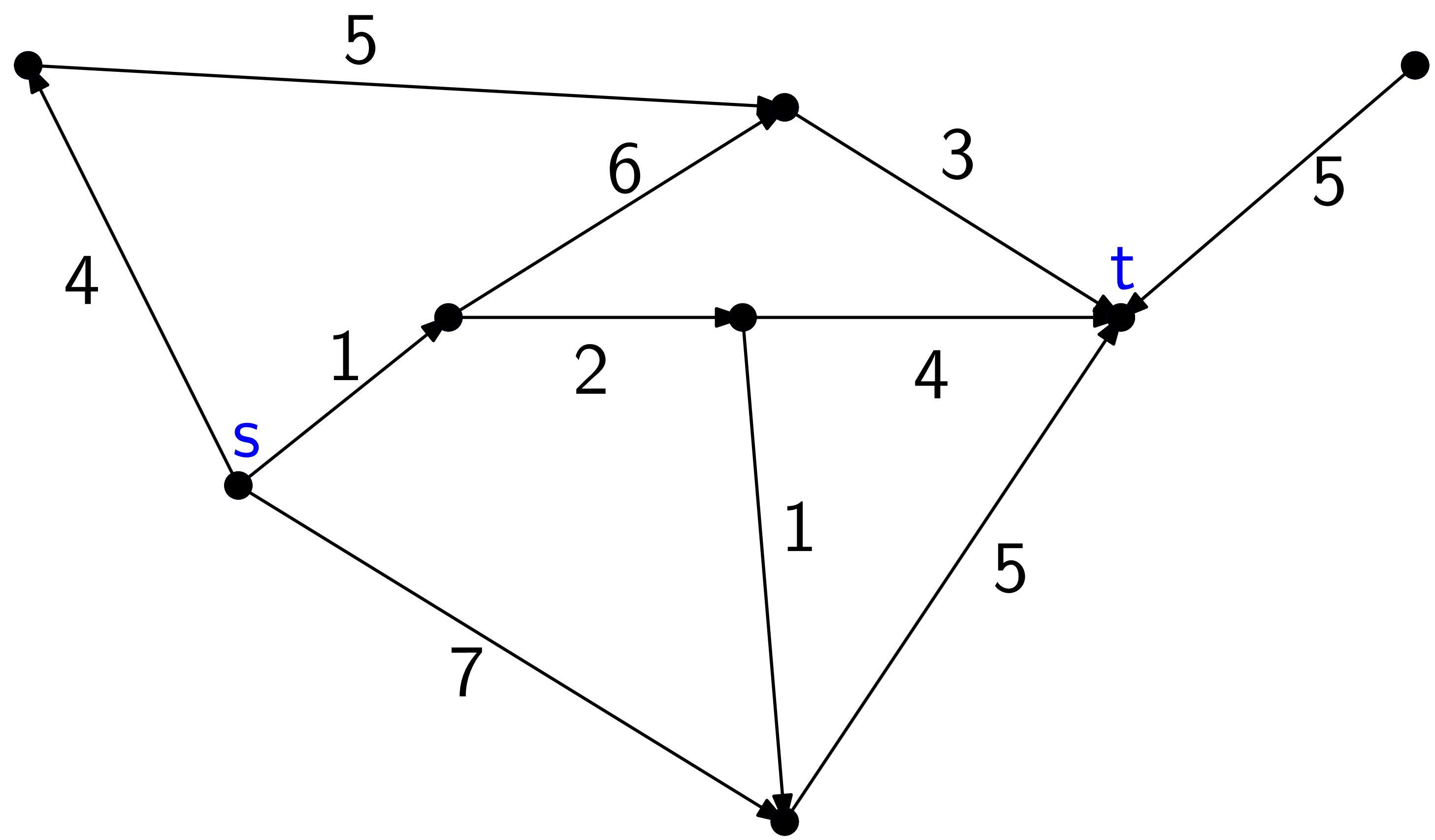
[Aneja, Aggarwal, and Nair 1983]

Idea

Consider only resource consumption

$\forall v \in V$ compute minimal resource consumption r_{min} for a path $s, \cdots, v, \cdots, t$ (via two Dijkstra runs)

Prune all nodes with $r_{min}(v) > R$



SIMPLE PRUNING

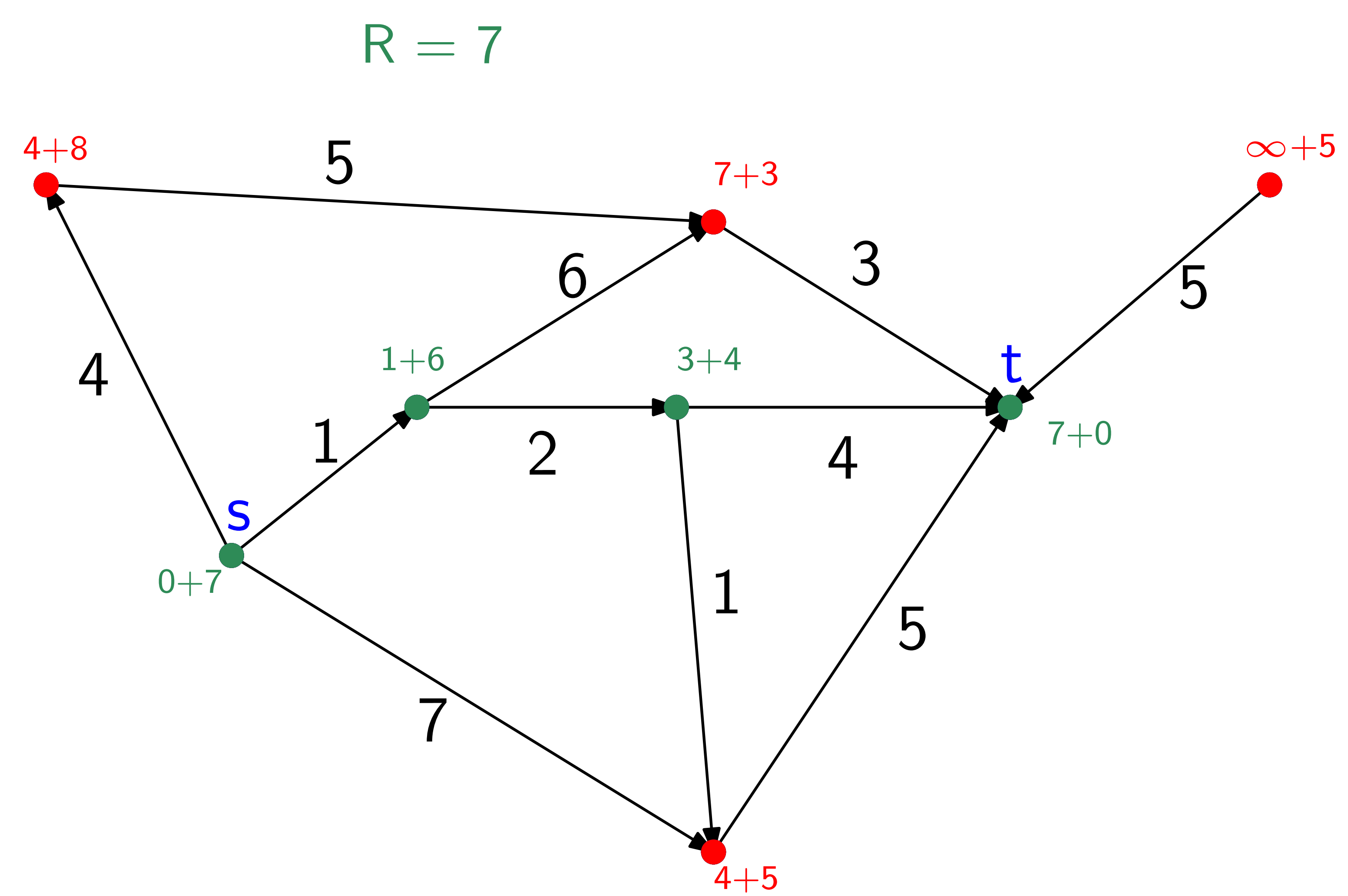
[Aneja, Aggarwal, and Nair 1983]

Idea

Consider only resource consumption

$\forall v \in V$ compute minimal resource consumption r_{min} for a path $s, \cdots, v, \cdots, t$ (via two Dijkstra runs)

Prune all nodes with $r_{min}(v) > R$



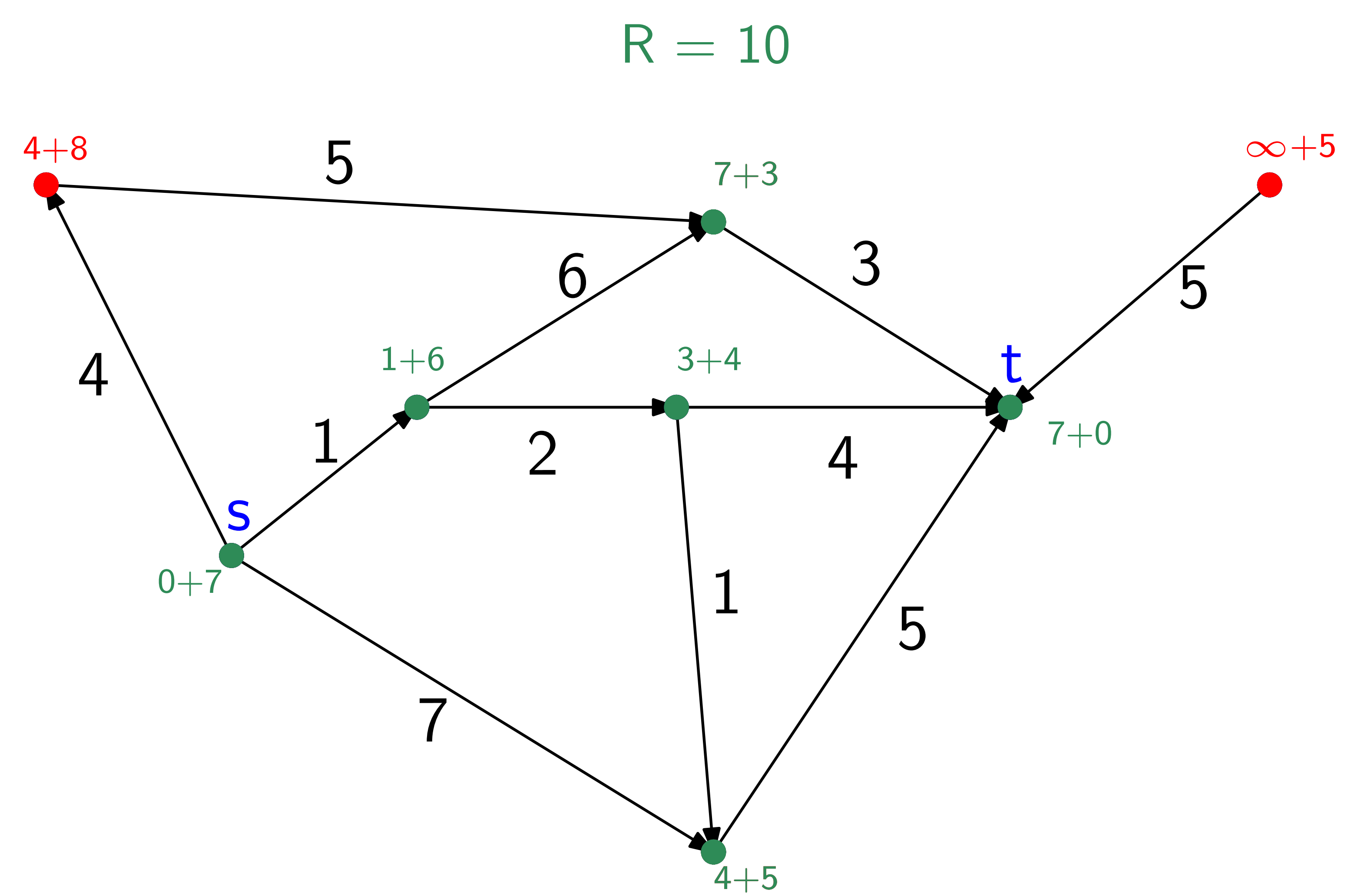
SIMPLE PRUNING

[Aneja, Aggarwal, and Nair 1983]

Idea
Consider only resource consumption

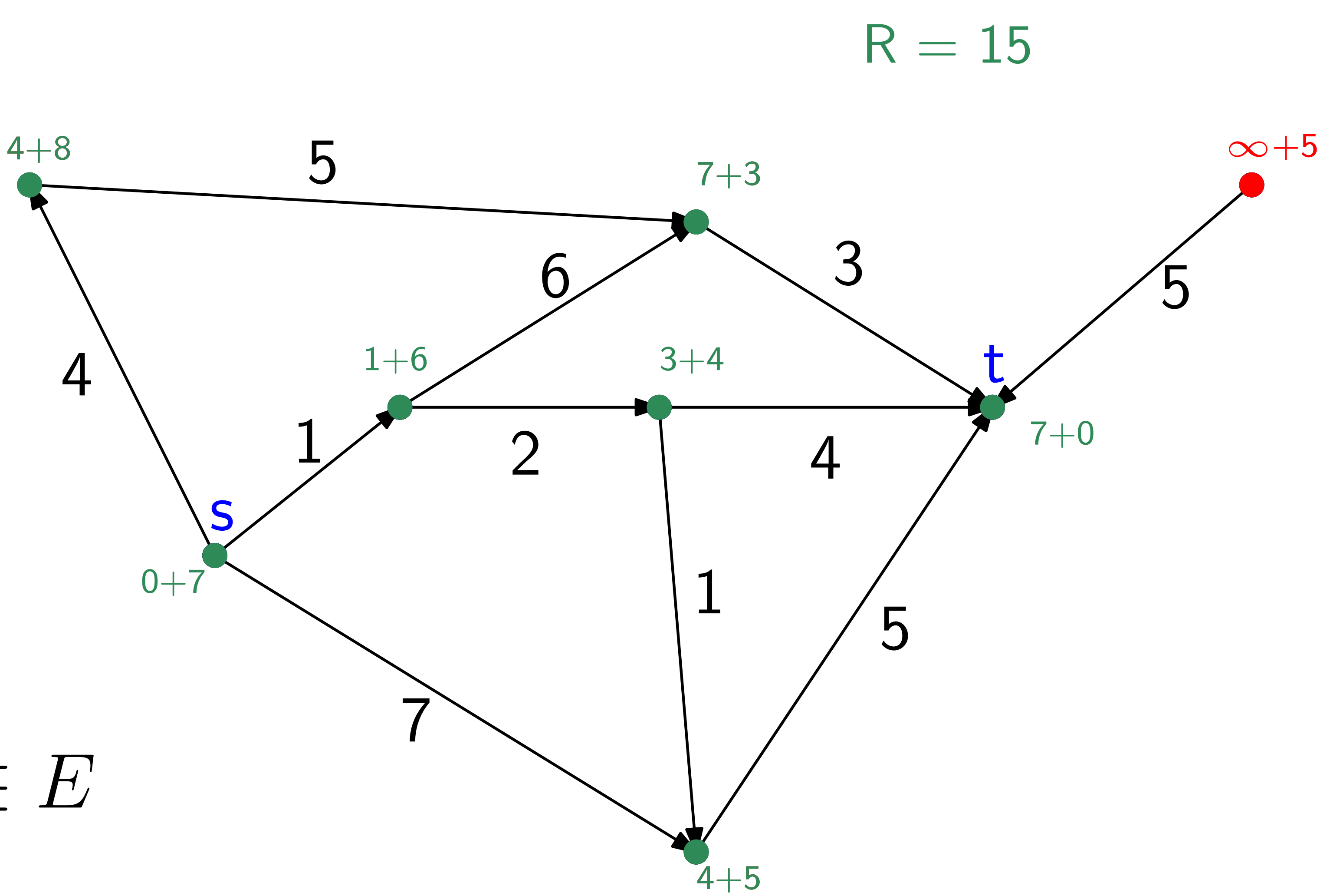
$\forall v \in V$ compute minimal resource consumption r_{min} for a path $s, \cdots, v, \cdots, t$ (via two Dijkstra runs)

Prune all nodes with $r_{min}(v) > R$



SIMPLE PRUNING

[Aneja, Aggarwal, and Nair 1983]



Problem

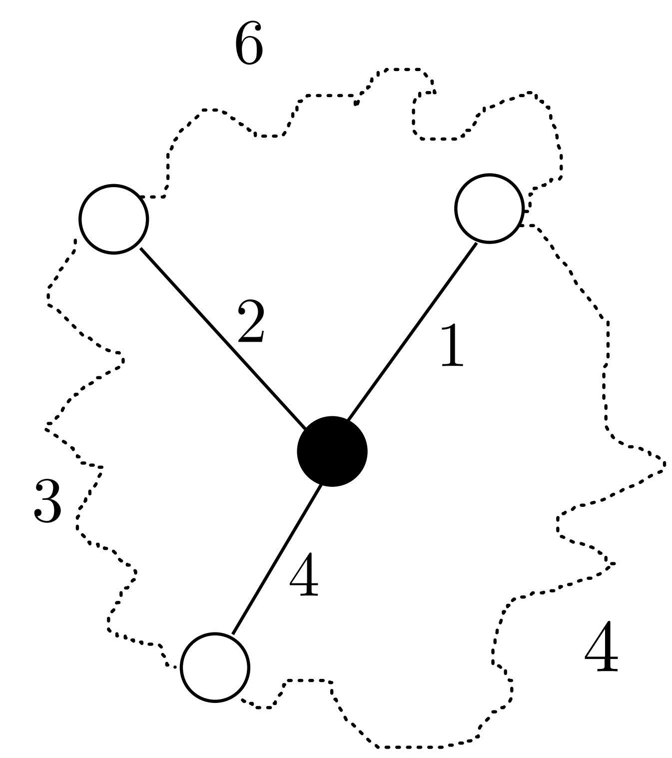
Impact low if

- R is large
- $r(e)$ small for many $e \in E$

CONTRACTION HIERARCHY

[Geisberger et al. 2008]

Graph preprocessing method

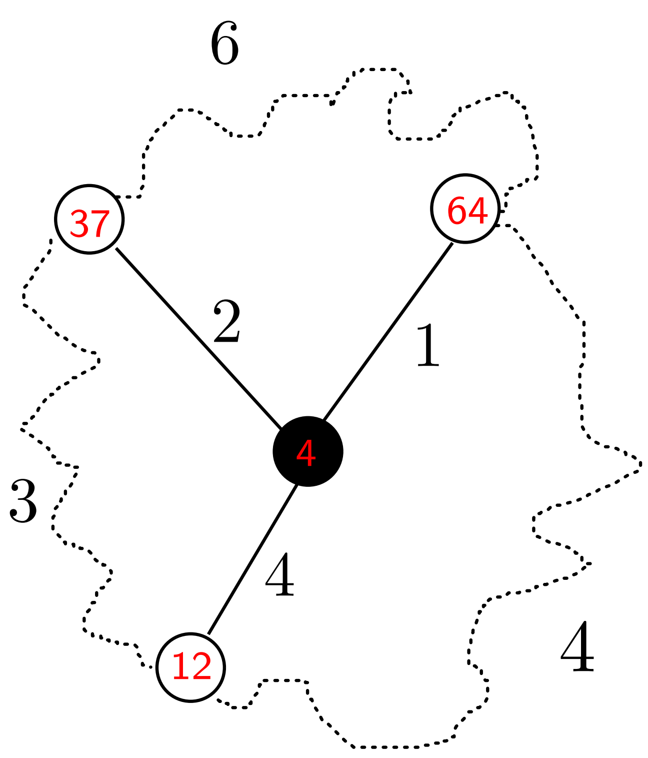


CONTRACTION HIERARCHY

[Geisberger et al. 2008]

Graph preprocessing method

1. Assign distinct importance values to the nodes



CONTRACTION HIERARCHY

[Geisberger et al. 2008]

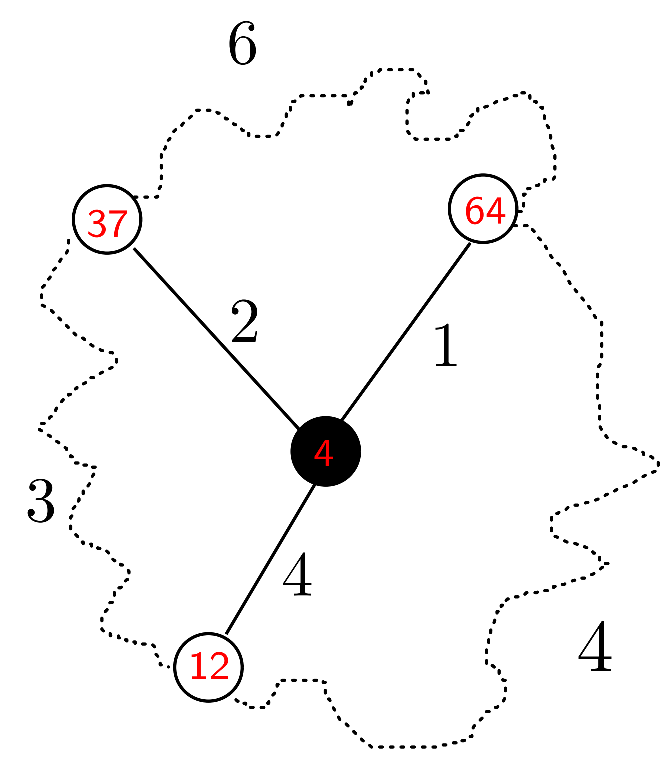
Graph preprocessing method

1. Assign distinct importance values to the nodes
2. Remove nodes one by one in order of importance ('contraction')

    Task: maintain all shortest path distances in remaining graph

    Add shortcut if no witness found

    Witness: path shorter than reference path



CONTRACTION HIERARCHY

[Geisberger et al. 2008]

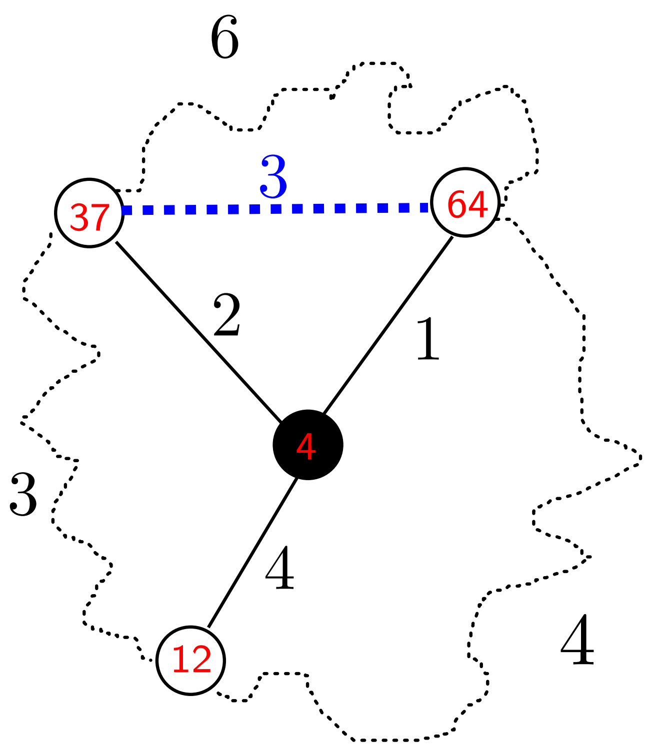
Graph preprocessing method

1. Assign distinct importance values to the nodes
2. Remove nodes one by one in order of importance ('contraction')

    Task: maintain all shortest path distances in remaining graph

    Add shortcut if no witness found

    Witness: path shorter than reference path



CONTRACTION HIERARCHY

[Geisberger et al. 2008]

Graph preprocessing method

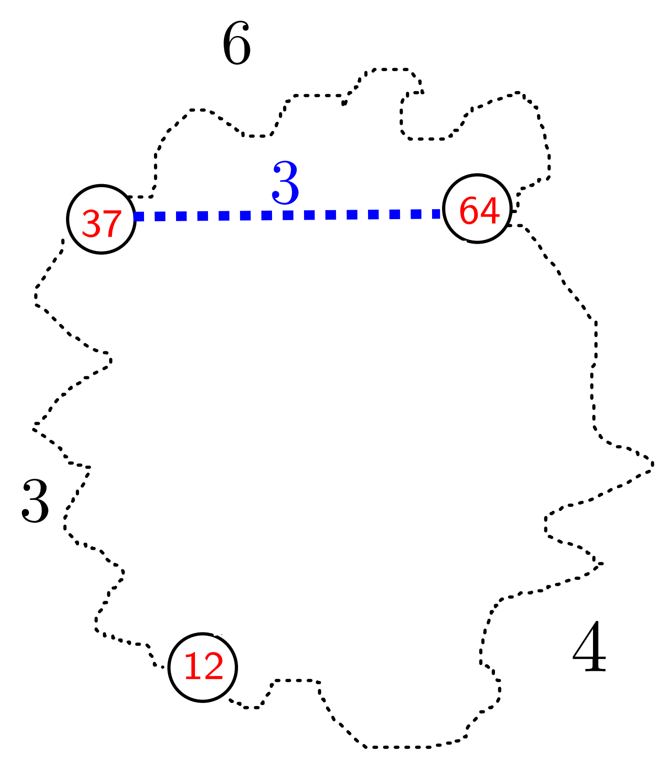
1. Assign distinct importance values to the nodes
2. Remove nodes one by one in order of importance ('contraction')

    Task: maintain all shortest path distances in remaining graph

    Add shortcut if no witness found

    Witness: path shorter than reference path
3. Add all shortcuts to original graph

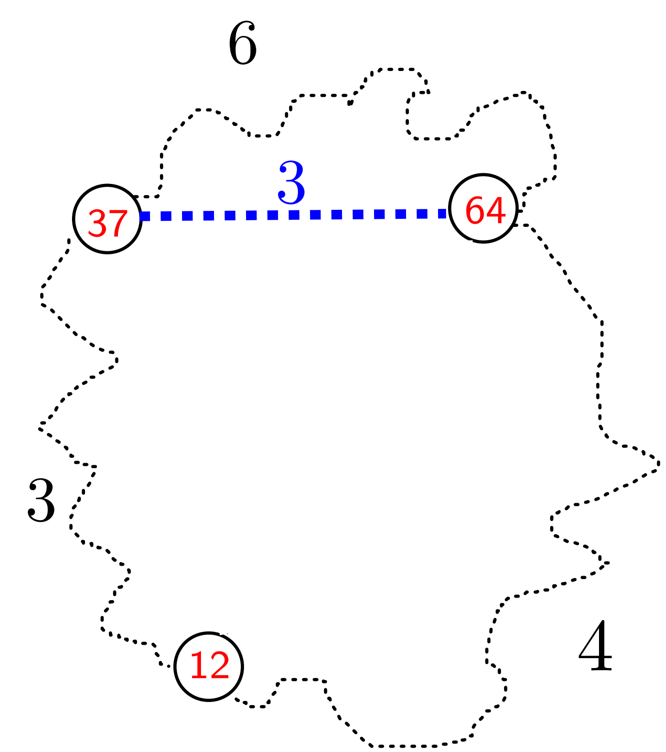
Every SP can be divided into $s \uparrow$ and $\downarrow t$



CONTRACTION HIERARCHY

[Geisberger et al. 2008]

Graph preprocessing method



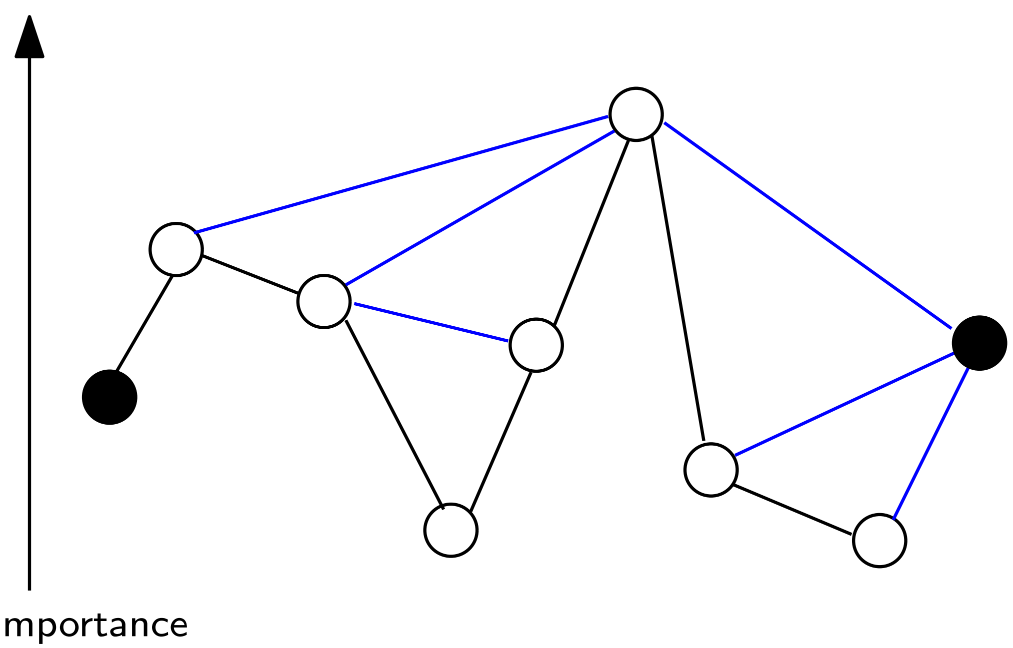
1. Assign distinct importance values to the nodes
2. Remove nodes one by one in order of importance ('contraction')

   Task: maintain all shortest path distances in remaining graph
   Add shortcut if no witness found
   Witness: path shorter than reference path
3. Add all shortcuts to original graph

Every SP can be divided into $s \uparrow$ and $\downarrow t$

Query Answering

bidirectional: only relax edges to nodes with higher importance



WITNESS SEARCH FOR CSP

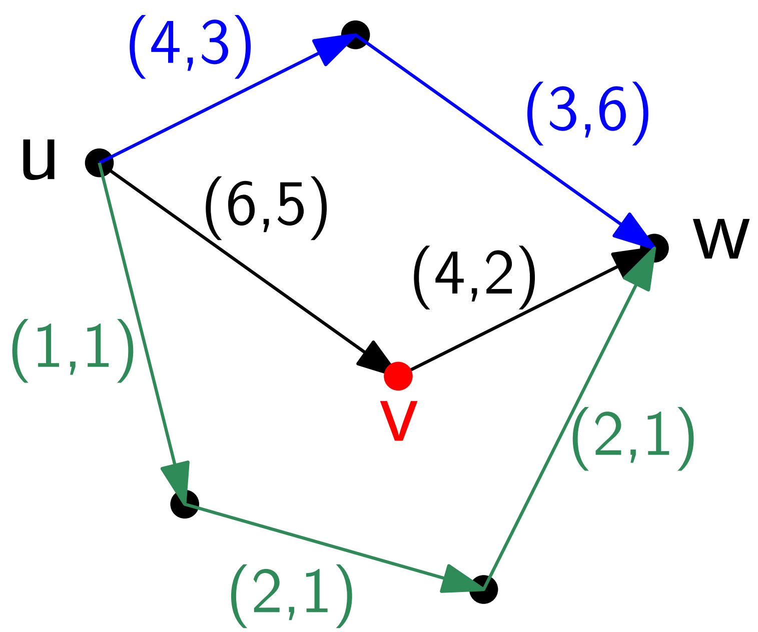
Task maintain all pareto-optimal paths

Witness must dominate reference path

WITNESS SEARCH FOR CSP

Task maintain all pareto-optimal paths

Witness must dominate reference path

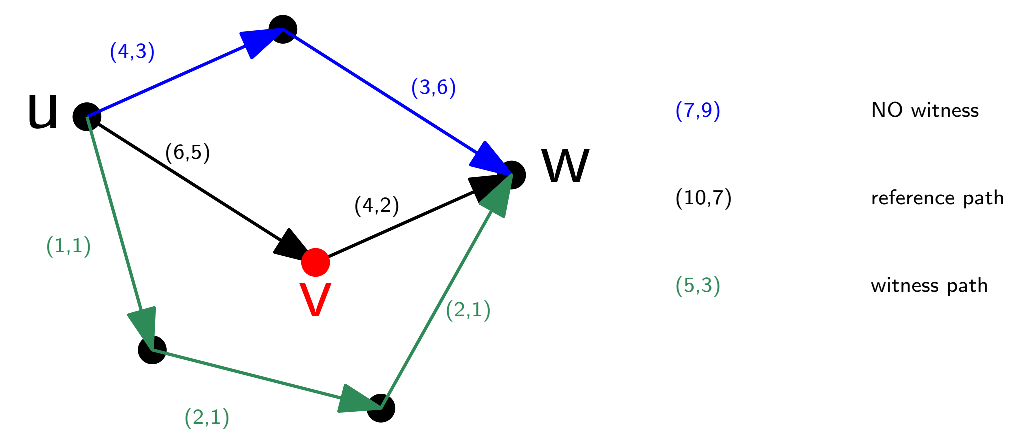


(7,9)	NO witness
(10,7)	reference path
(5,3)	witness path

WITNESS SEARCH FOR CSP

Task maintain all pareto-optimal paths

Witness must dominate reference path



Naive Witness Search

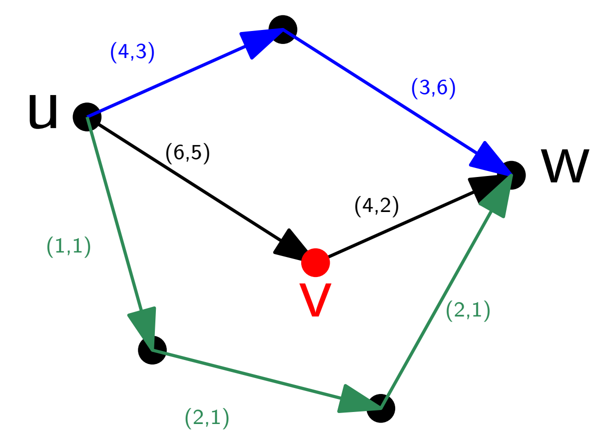
reference path $p = uvw$

- start label setting computation(LSC) in u with $R = r(p)$
- if w receives label with $c \leq c(p), r \leq r(p)$, break
  $\rightarrow$ witness path found
- insert shortcut if no witness was found

WITNESS SEARCH FOR CSP

Task maintain all pareto-optimal paths

Witness must dominate reference path



Naive Witness Search

reference path $p = uvw$

- start label setting computation(LSC) in u with $R = r(p)$
- if w receives label with $c \leq c(p), r \leq r(p)$, break
  $\rightarrow$ witness path found
- insert shortcut if no witness was found

Problem
LSC might be very time and space consuming

WITNESS SEARCH FOR CSP

Basic Idea

Restrict witness search first to paths on the lower convex hull.

Lower Convex Hull(LCH)

for every v-w-path p:
represent $(c(p), r(p))$ as line segment $\lambda c(p) + (1 - \lambda) r(p)$, $\lambda \in [0, 1]$

$p \in LCH(v, w) \Leftrightarrow \exists \lambda \in [0, 1]$ for which line segment of p is minimal

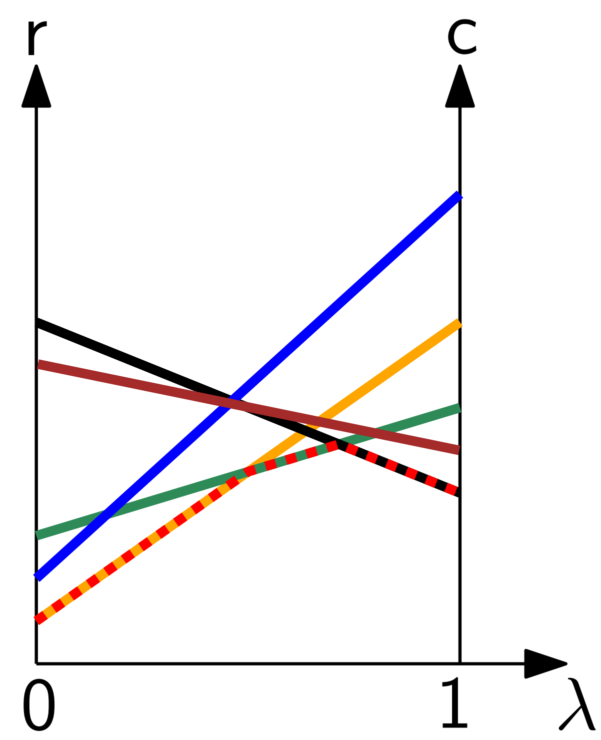
$p_1 : (11, 2)$

$p_2 : (6, 3)$

$p_3 : (1, 8)$

$p_4 : (5, 7)$

$p_5 : (4, 8)$



WITNESS SEARCH FOR CSP

Basic Idea

Restrict witness search first to paths on the lower convex hull.

Advantage

paths on the LCH can be found by a Dijkstra run in G^λ

G^λ: edges have single weight $w(e) = \lambda c(e) + (1 - \lambda) r(e)$

WITNESS SEARCH FOR CSP

Basic Idea

Restrict witness search first to paths on the lower convex hull.

Advantage

paths on the LCH can be found by a Dijkstra run in G^λ

G^λ: edges have single weight $w(e) = \lambda c(e) + (1-\lambda)r(e)$

In which cases does exploring the LCH help?

What if LCH check procedure is inconclusive?

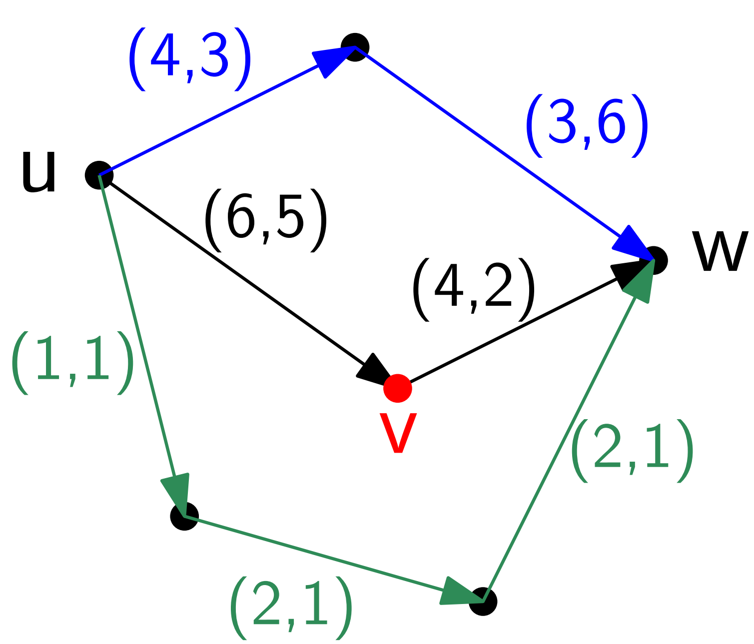
WITNESS SEARCH FOR CSP

In which cases does exploring the LCH help?

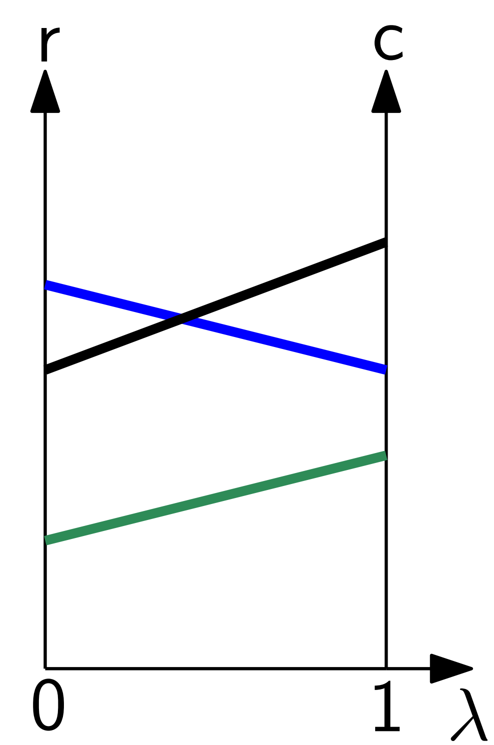
WITNESS SEARCH FOR CSP

In which cases does exploring the LCH help?

1. If dominating path is part of the LCH.
   witness path found, shortcut can be omitted



(7,9)	NO witness
(10,7)	reference path
(5,3)	witness path

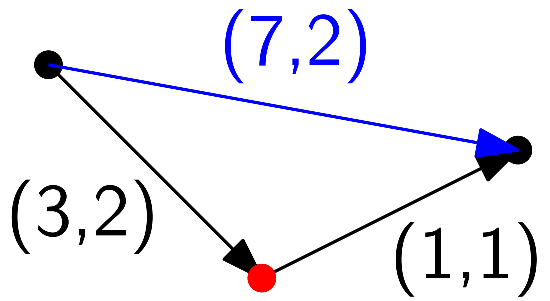


WITNESS SEARCH FOR CSP

In which cases does exploring the LCH help?

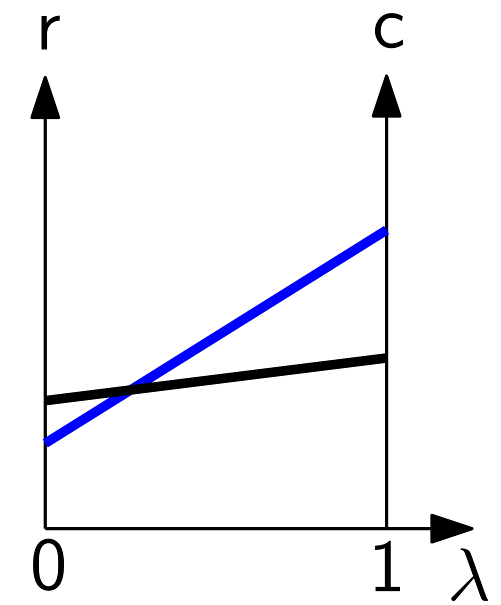
1. If dominating path is part of the LCH.
   witness path found, shortcut can be omitted

2. If reference path is part of the LCH.
   no dominating path exists, shortcut must be inserted



(7,2)

(4,3) reference path



WITNESS SEARCH FOR CSP

What if LCH check procedure is inconclusive?

Reasons

1. Neither p nor a possible witness are part of the LCH.
2. Number of λ support points too small.

Possibilities

- Apply LSC on top.

  or
- Add shortcut without further care.

EXPERIMENTAL RESULTS

Test Graphs 10k - 5.5m nodes

Preprocessing

- $t = 3$ support points led to a conclusive result of the LCH-checker in 62% of the cases
- number of edges in CH-graph about twice the number of original edges (comparable to the conventional case)

Query Answering

- speed-up about two orders of magnitude
- remarkably less space consumption (8GB laptop sufficient, before some queries failed even on a 96GB server)

CONCLUSIONS

Can answer exact CSP queries in graphs with up to 500k nodes in time less than one second!

Also in the paper...

- speed-up via CH for dynamic programming CSP solution
- CSP-variant of arc-flags

Future Work

- combination with other techniques/heuristics (e.g. A^*)
- consider other metric combinations and more complicated scenarios, e.g. edge cost functions

THANK YOU...

... for your attention!

Questions?