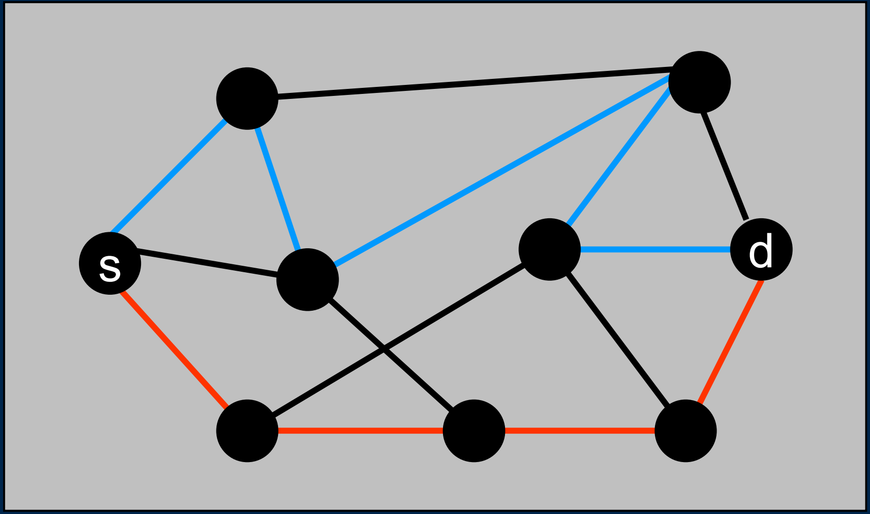
Diverse routing in SRLG networks

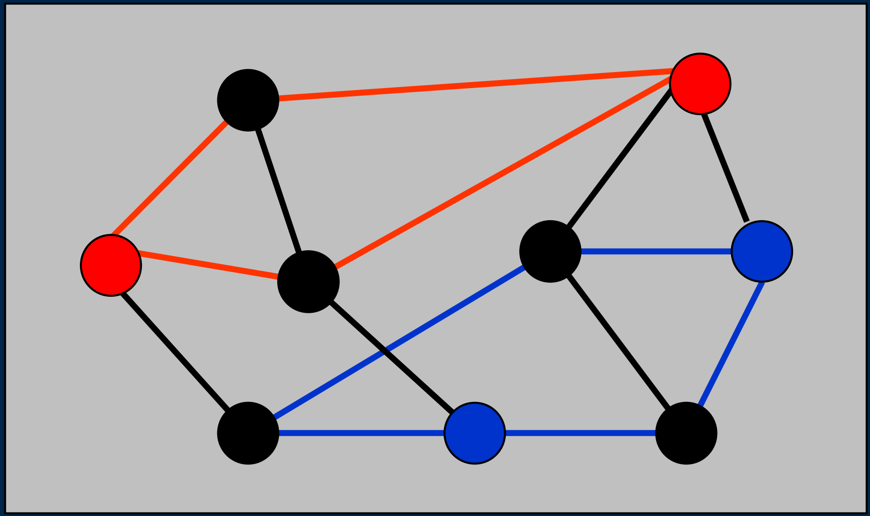
Introduction

- Networks are everywhere in modern society
- Internet, telephone (wired/wireless), ATMs, stock market, etc.
- Connection failure sometimes unacceptable
- Failure caused by router, cable,server, software, power, etc.
- Single / protected path

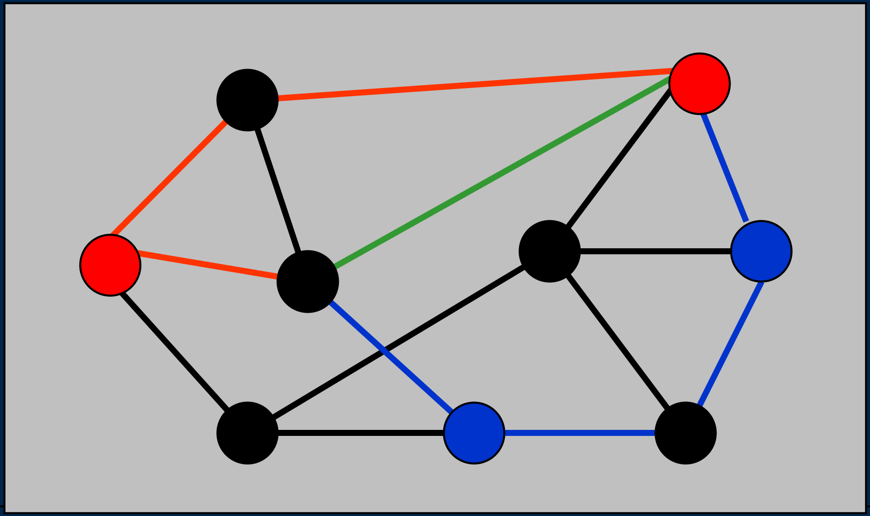


Protected path

- Dedicated / shared protection



Dedicated protection

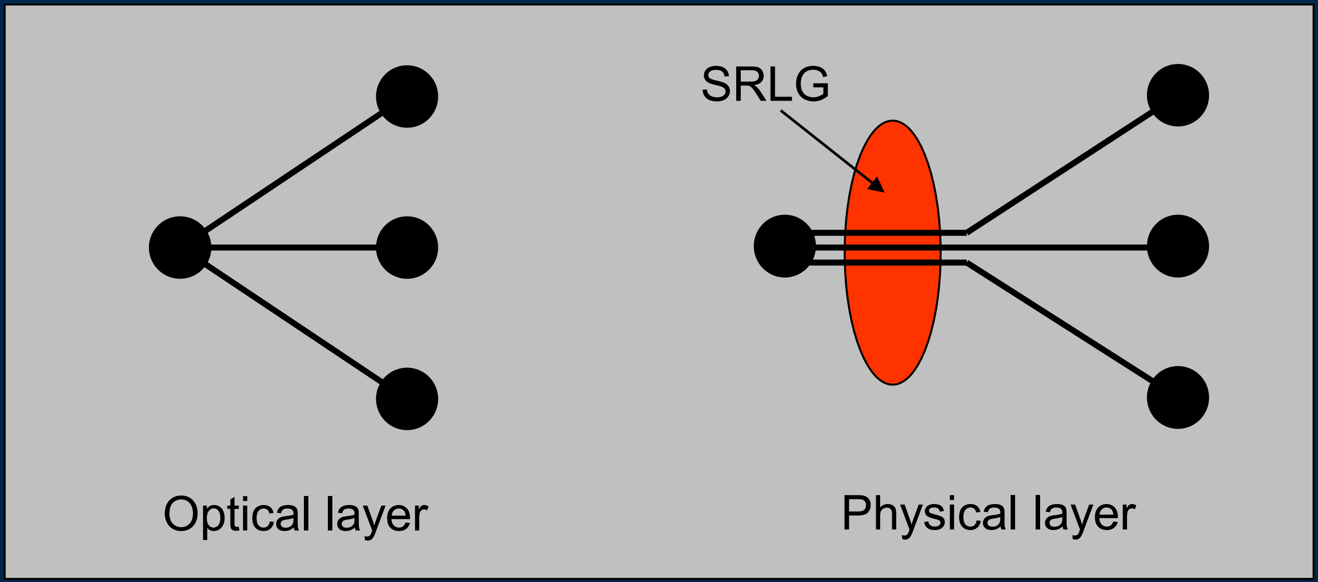


Shared protection

Shared Risk Link Group (SRLG)

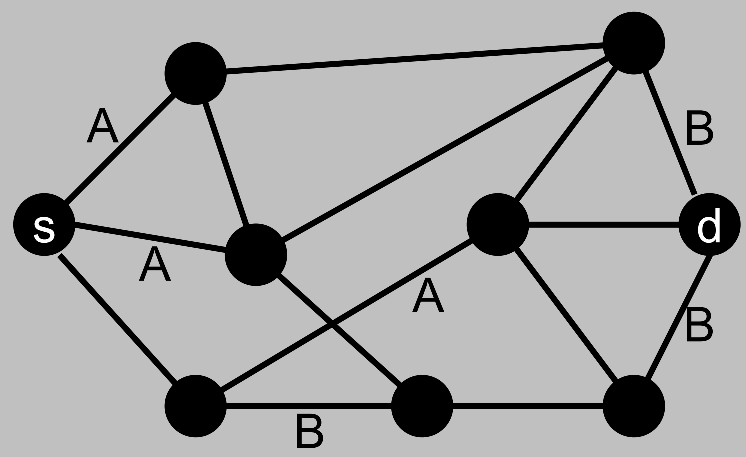
- Optical networks consist of at least two layers

  - Optical layer
  - Physical layer



Shared Risk Link Group (SRLG)

Try yourself:
Find two paths in the network between node s and node d such that they do not share a common SRLG.



example

Problem definition

- Find two SRLG-disjoint paths between two nodes
- With minimal cost or shortest
- Exact algorithm (always return optimal solution, if exists)

Complexity

- Problem is NP-complete
  - given solution easy to verify
  - finding a solution is difficult
- If routed serially, traps. Choice of first path limits second path
- Minimal cost
- Balance between performance and near-optimality

SRLG-tree algorithm

First steps SRLG-tree:

• Bhandari’s algorihm gives shortest protected path (no SRLGs)

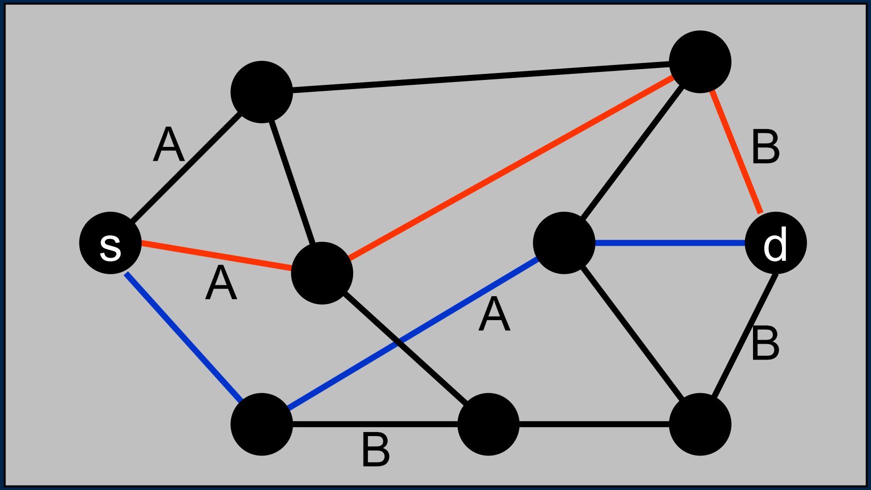
• If found path SRLG-disjoint then optimal solution

Define primary path PP and back path BP

• Else pick shared SRLG, for example A, and recalculate the protected path twice where:

1) A not in PP
2) A not in BP

Example

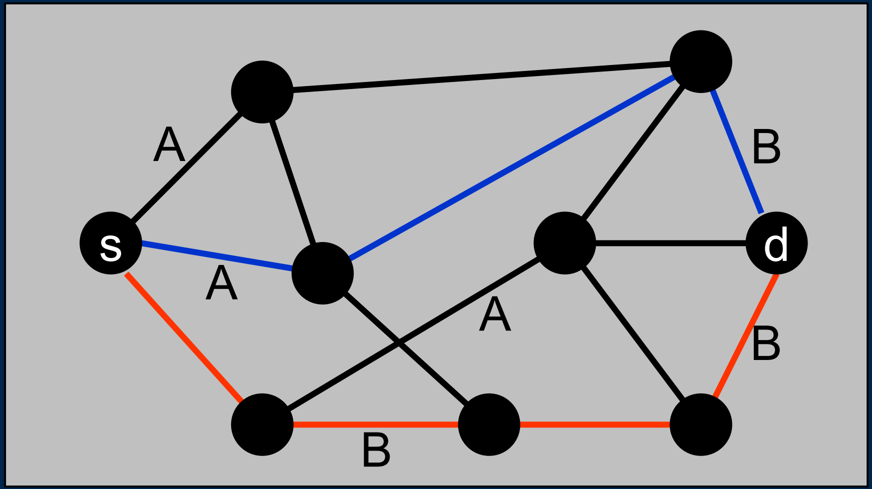


Shortest path

SRLG A shared

Make 2 new searches

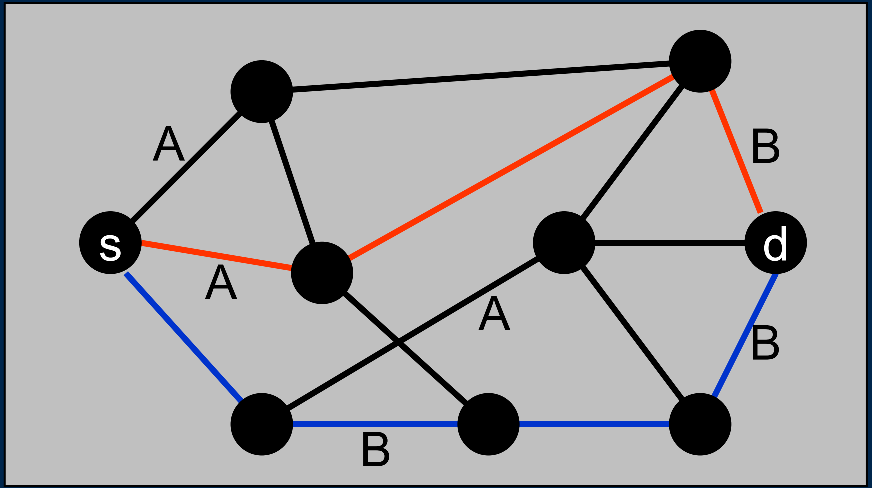
Example: the two new searches



1st new search

A not in PP
BP no limitations
(A,-)

SRLG B shared

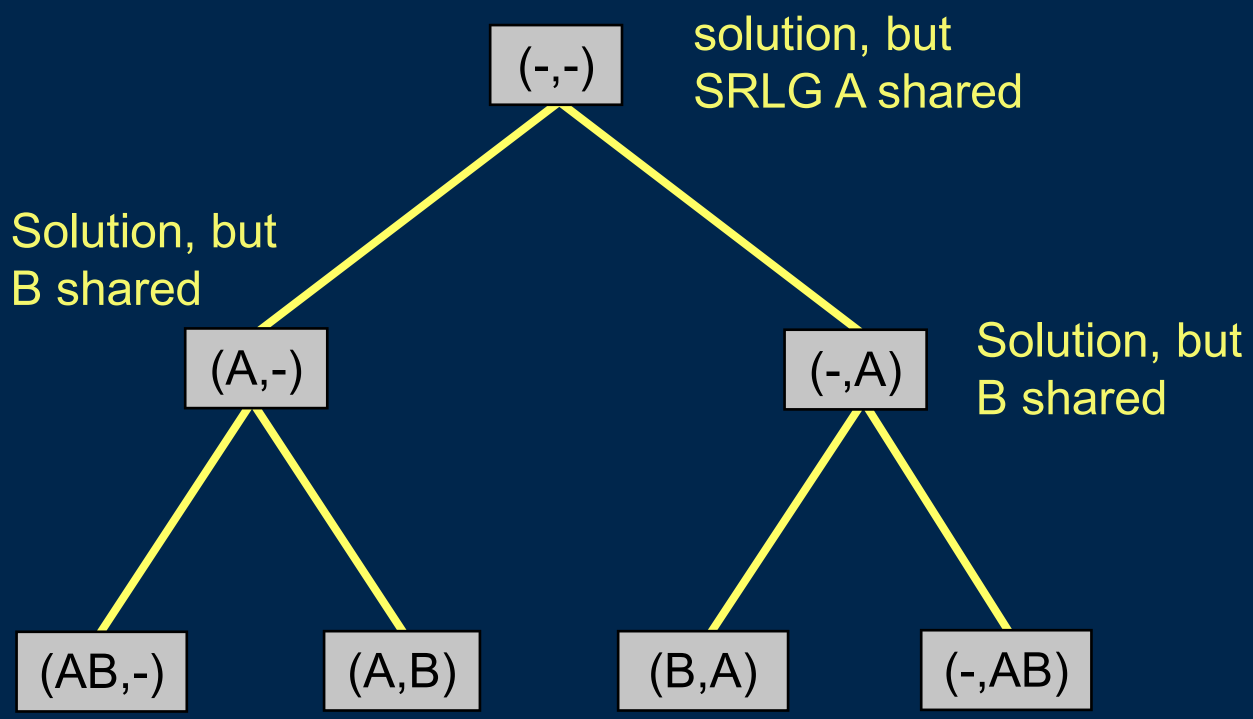


2nd new search

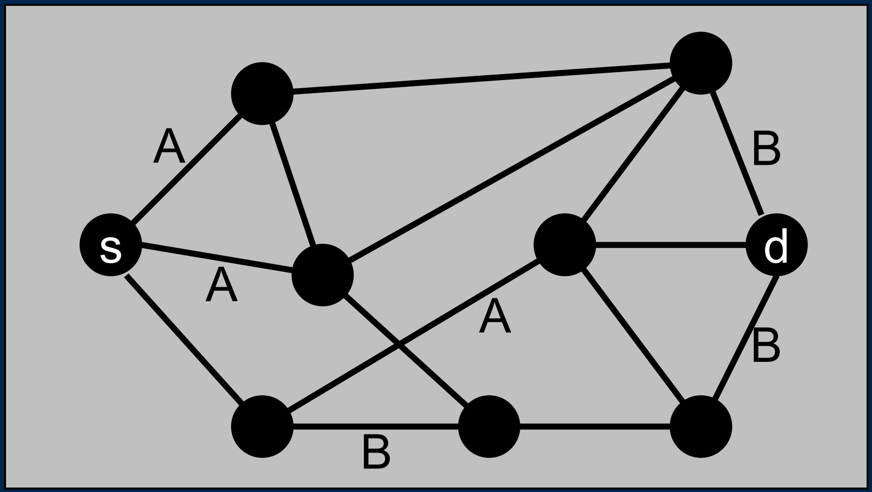
A not in BP
PP no limitations
(-,A)

SRLG B shared

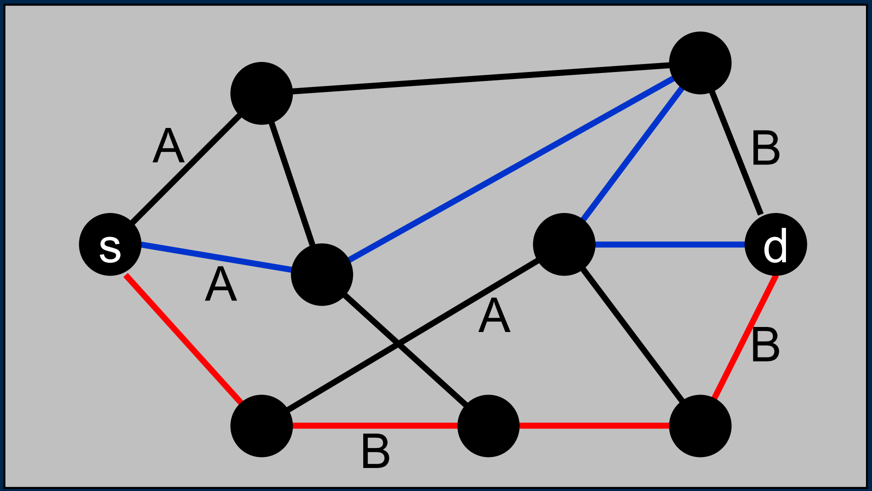
Example: tree structure



Example: (AB,-) and (A,B)



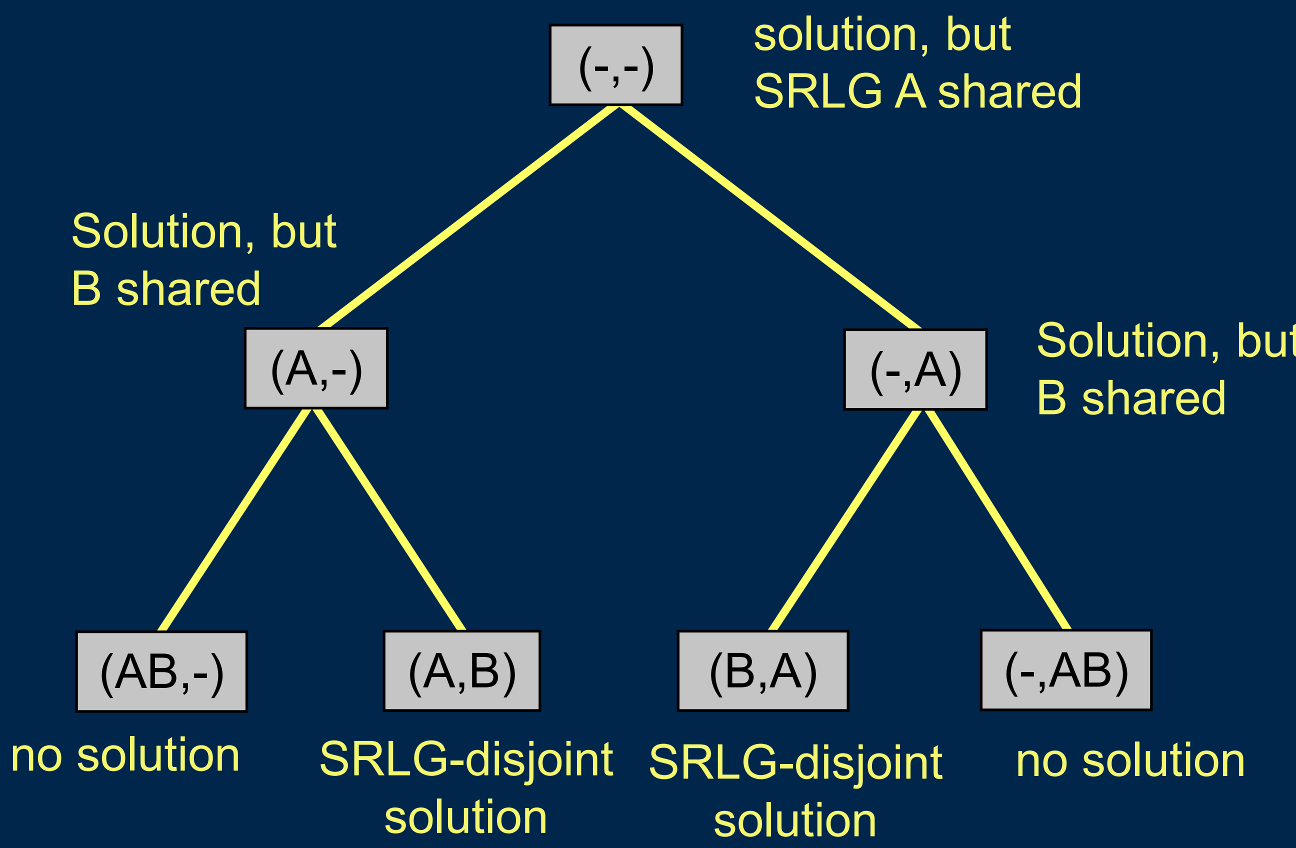
(AB,-) does not have a solution



(A,B) gives a SRLG-disjoint solution

introduction
protection
SRLG
problem definition
complexity
SRLG-tree
SRLG-exclusion
perform. SRLG-tree
demo
conclusions

Example: parsing tree



Tree properties

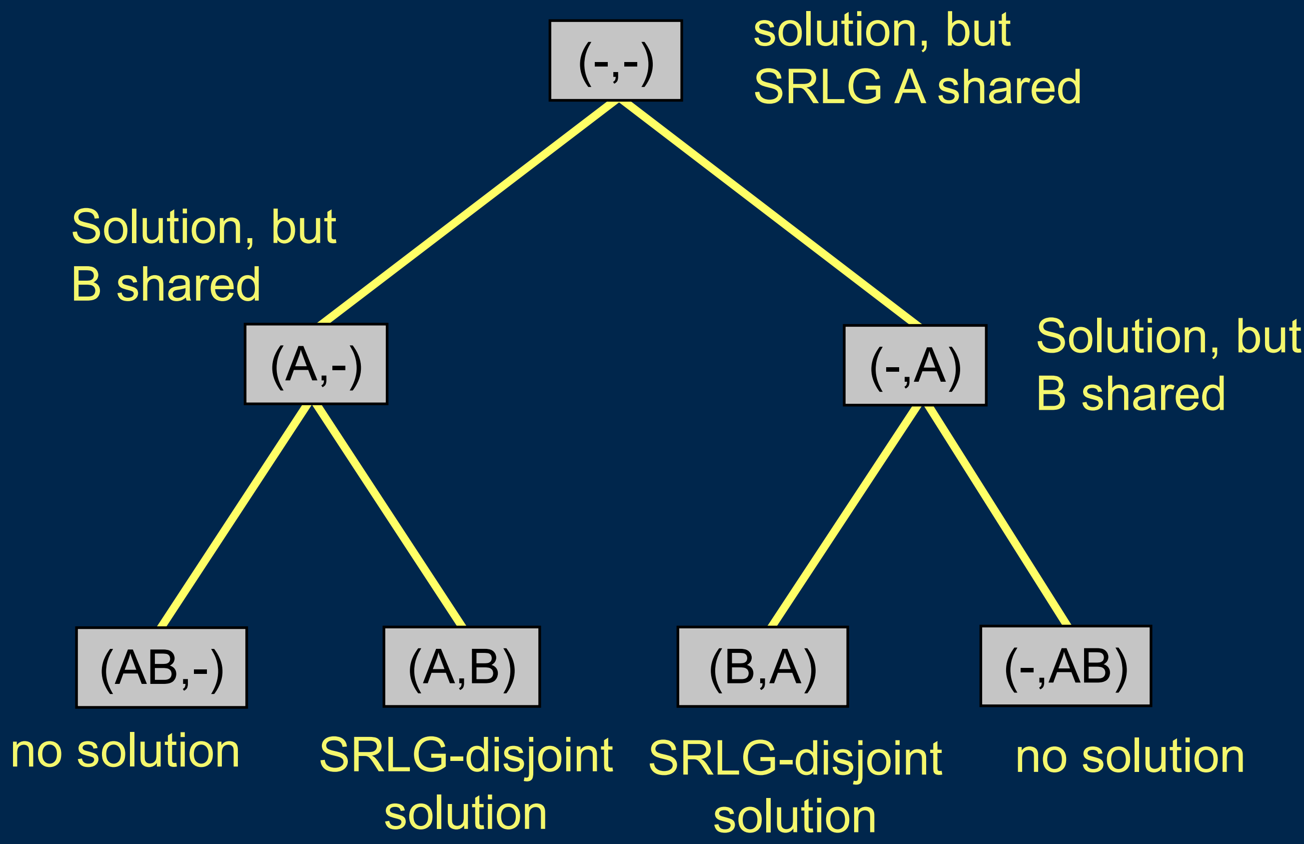
- Starting from the tree source (-,-) the cost of solutions increase

- If tree node does not have solution then child nodes do not have solutions

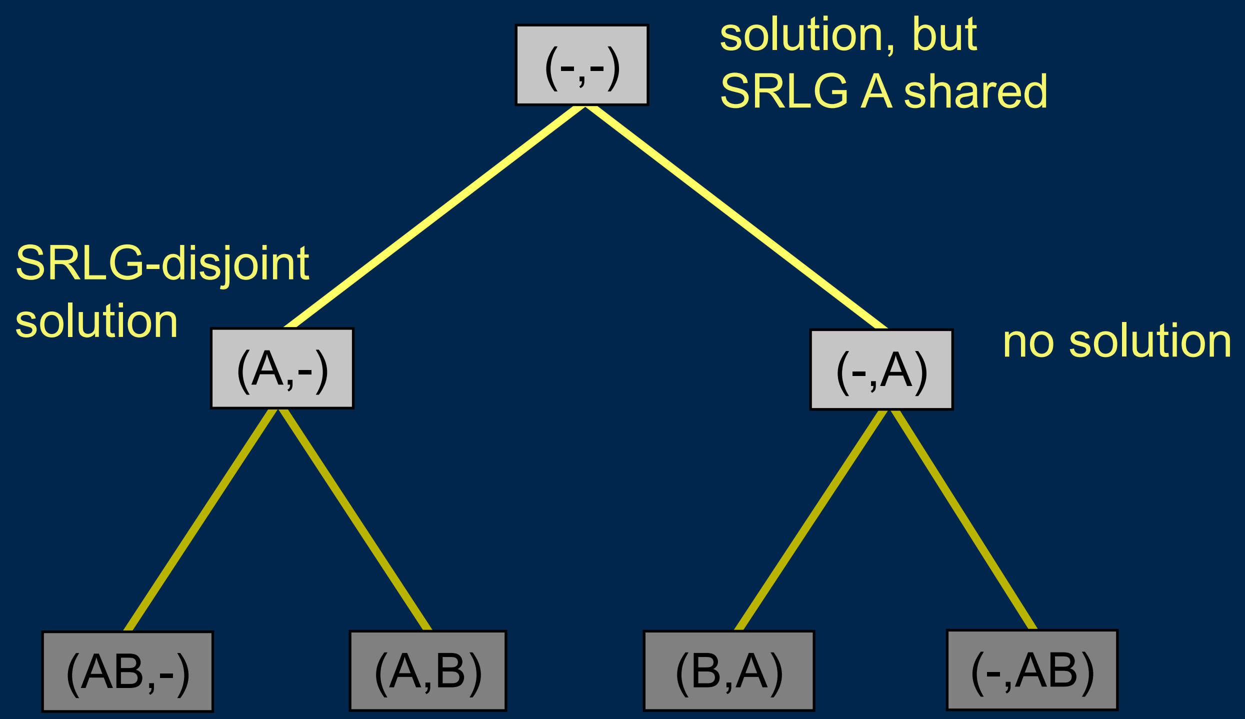
Result: if a tree node has a SRLG-disjoint solution or no solution then further branching stops

introduction
protection
SRLG
problem definition
complexity
SRLG-tree
SRLG-exclusion
perform. SRLG-tree
demo
conclusions

Example: parsing tree



Example: parsing tree

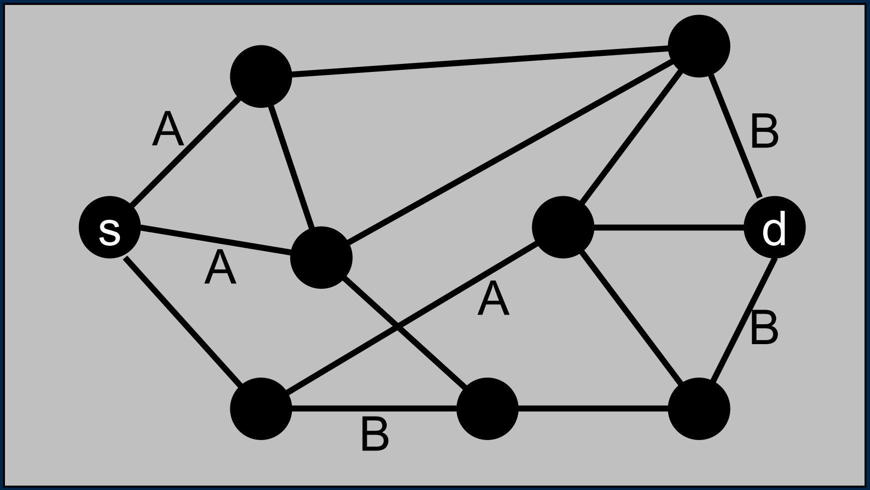


SRLG-exclusion algorithm

How do we calculate the nodes in the parsing tree?

For example: tree node (A,B)

SRLG-exclusion (A,B) returns a protected path with primary path excluded from A, backup path excluded from B

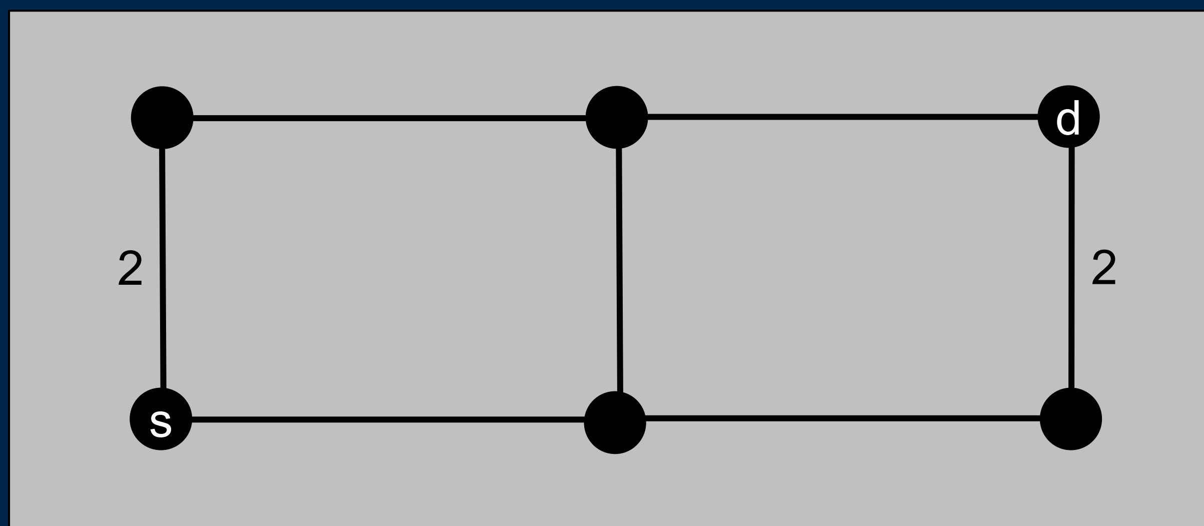


Unfortunately, this does not always work...

Problem

Sometimes approach with two-times Dijkstra algorithm does not give a solution

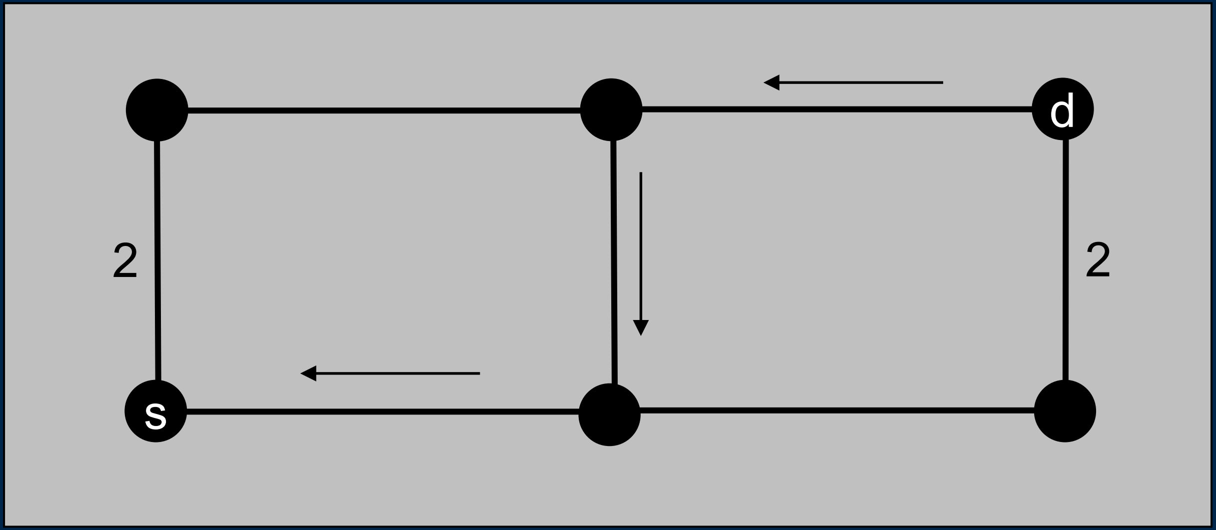
Simple example without SRLGs



Solution for this is for example Bhandari's algorithm

Bhandari's algorithm

Simple example without SRLGs

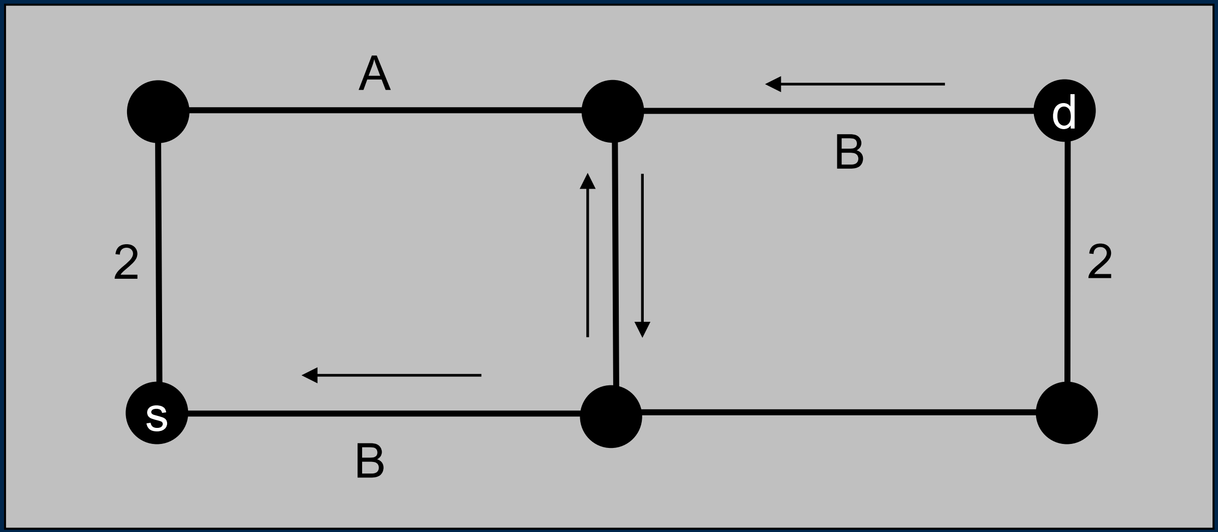


Initial paths are mixt!

New problem

Applied to SRLG network

We calculate (A,B)



Backup path not excluded from B, caused by mixing of paths

How can we make SRLG-exclusion to work correctly?

Idea

Assume we interested in calculating (A,B).

Primary path is easy

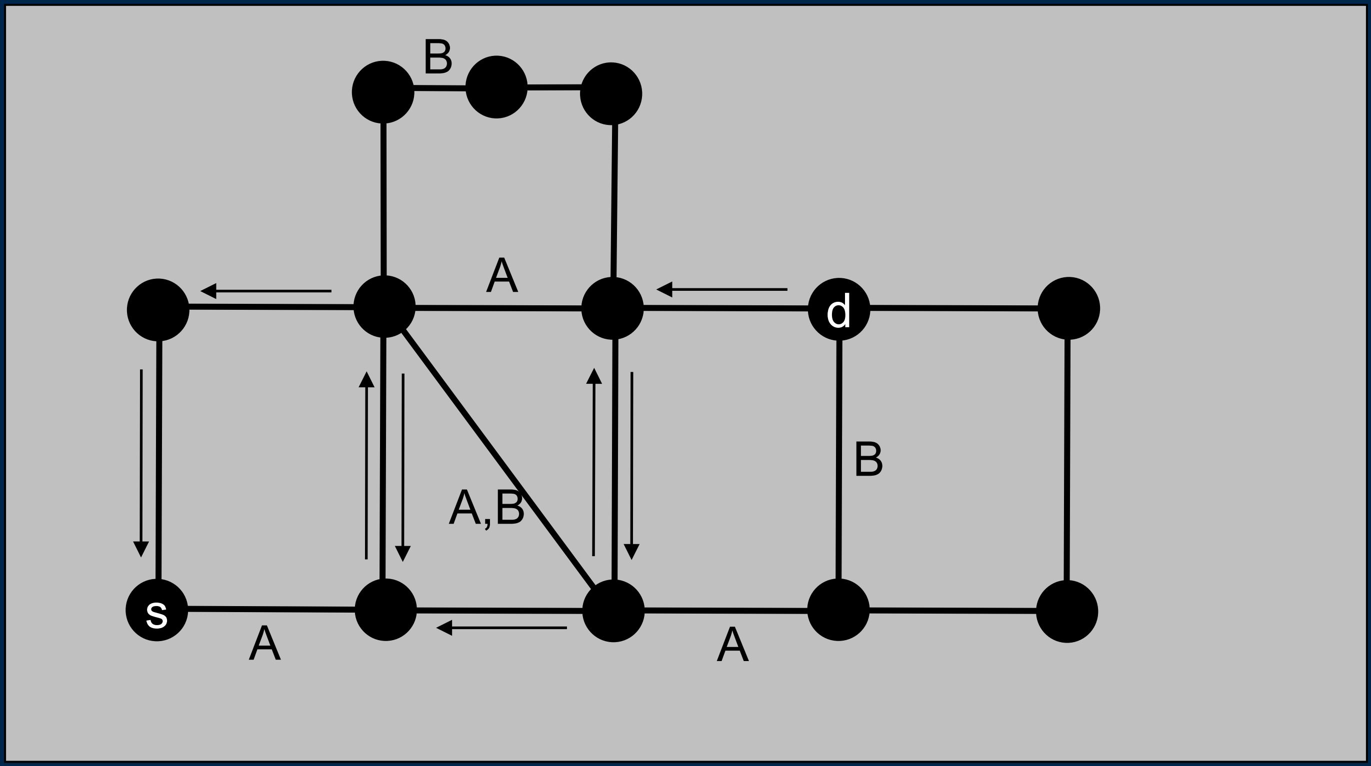
Backup path:

• first search in network without B

• if backup path uses a directed link, continu search in network without A

Example

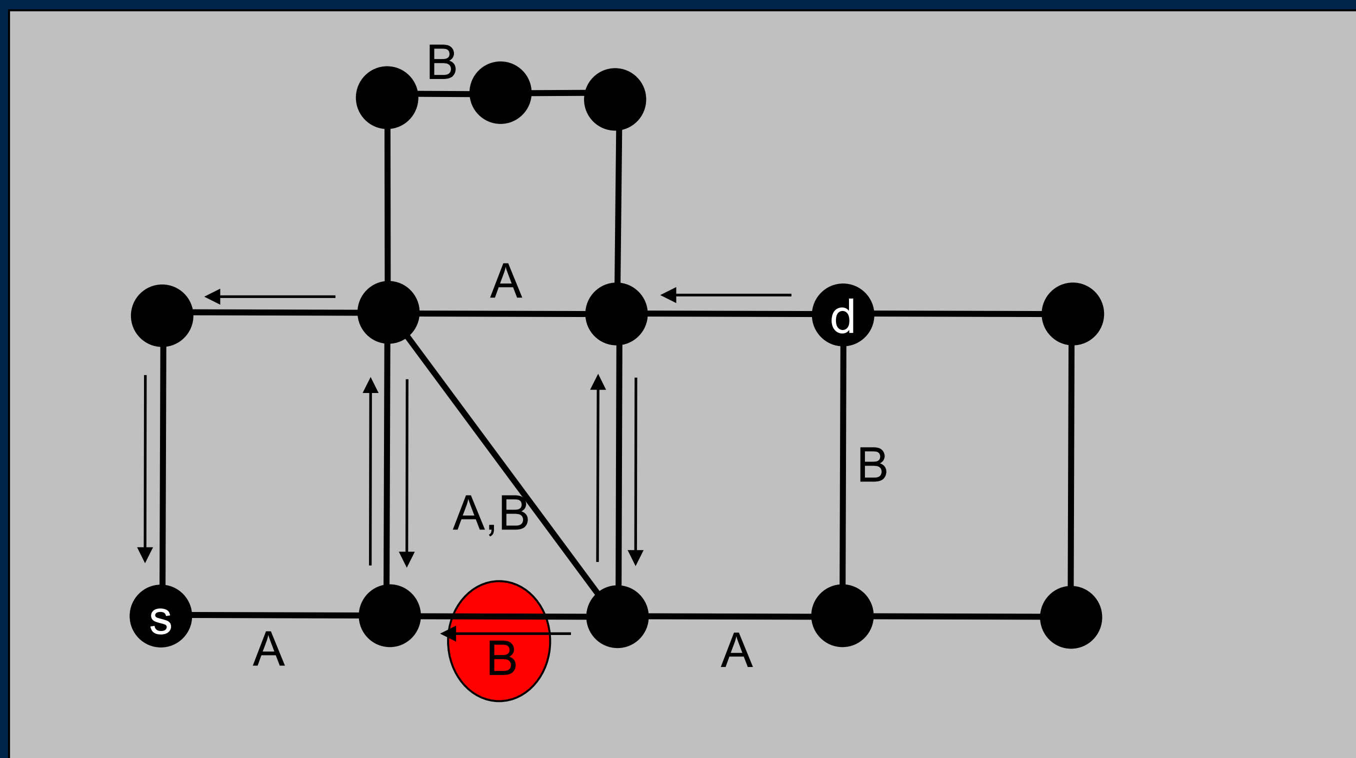
We calculate (A,B)



But also this does not always work...

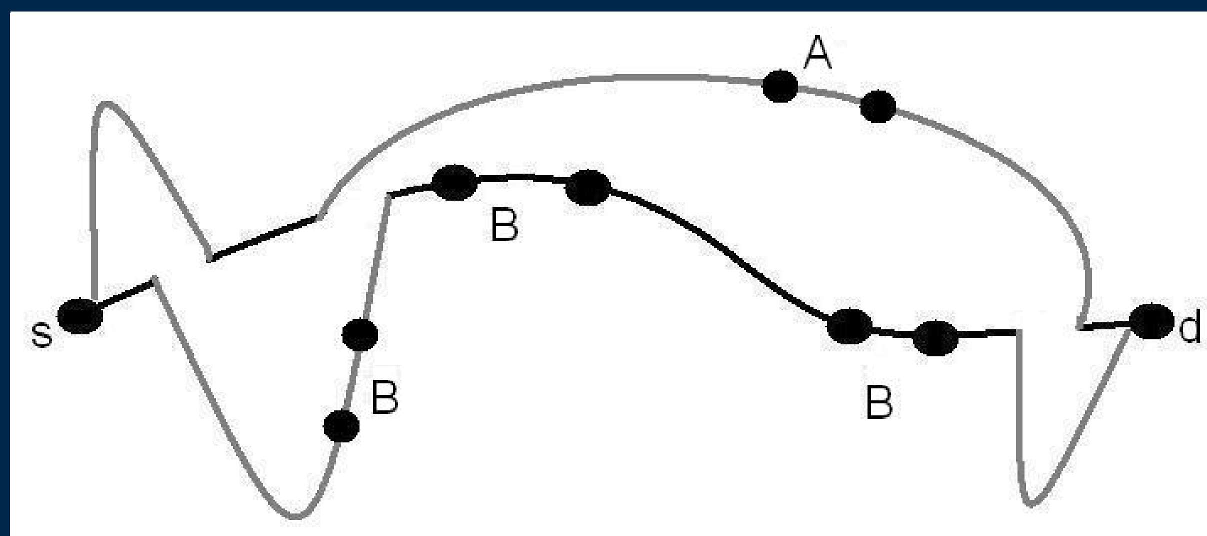
Example: problem

Again we calculate (A,B)



Example: solution

- Find initial primary path for (A,B)
- Remove B > one or more parts, ignore last
- For initial backup path:
  cross each part an even number of times



SRLG-exclusion concluded

SRLG-exclusion results in correct protected path if exists

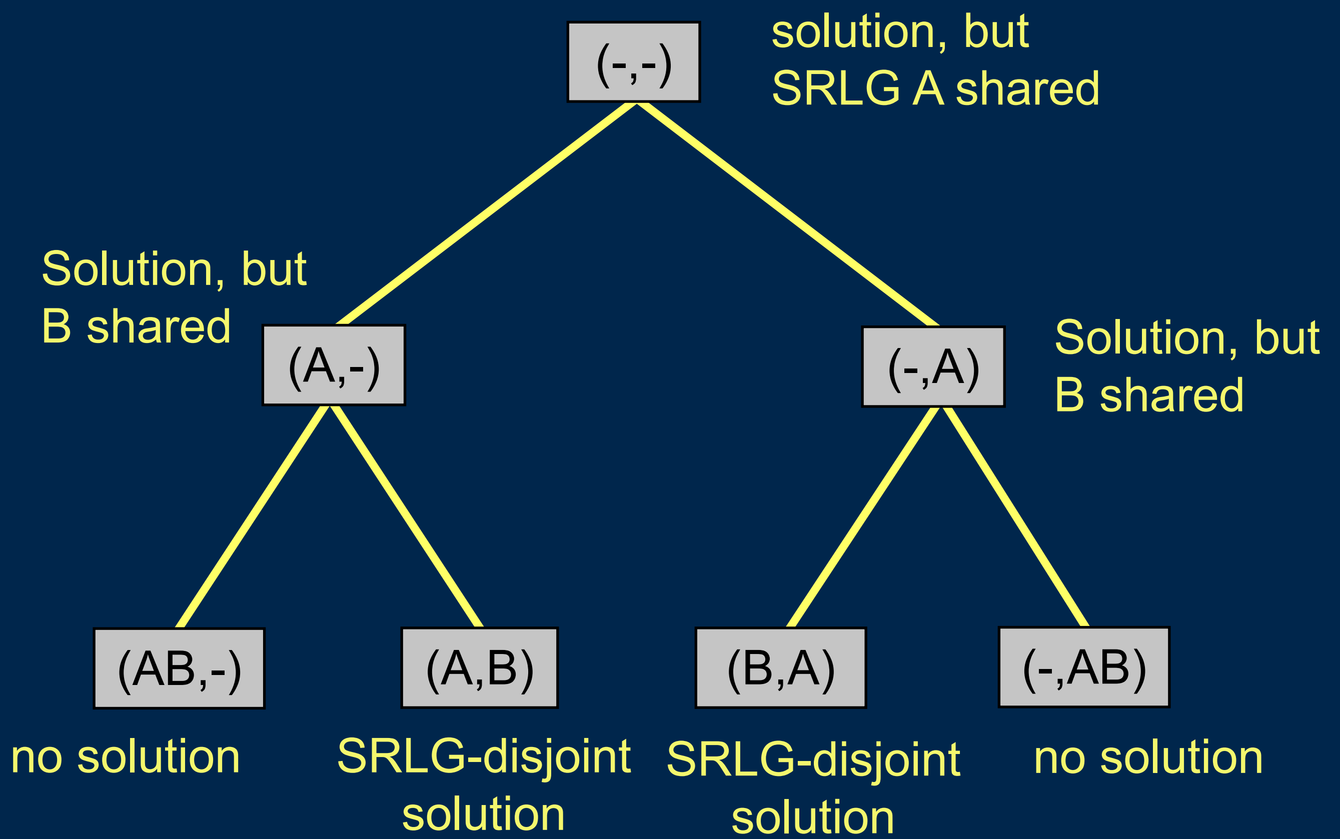
SRLG-exclusion does not give optimal solution, but...

For example, we look for a protected path with one path without A and another path without B then

SRLG-exclusion (A,B) or SRLG-exclusion (B,A) gives the optimal protected path.

SLRG-tree

Now we are able to calculate the tree nodes



Pick SRLG-disjoint solution with least cost for optimal solution

introduction

protection

SRLG

problem definition

complexity

SRLG-tree

SRLG-exclusion

perform. SRLG-tree

demo

conclusions

Timecomplexity SRLG-tree

Considering the worst-cast scenario

#L : number of links in network

#N : number of nodes in network

#R : number of SRLGs

SRLG-exclusion:

Initial primary path: Dijkstra algorithm

➢ $O(\#L + \#N \log(\#N))$

Initial backup path: Two networks

➢ $O(\#2L + \#2N \log(\#2N))$

$$O(3\#L + 3\#N \log(\#N) + 2\#N \log(2))$$

SRLG-tree:

Maximum number of treenodes: $2^{\#R+1} - 1$

$$O((2^{\#R+1} - 1)(3\#L + 3\#N \log(\#N) + 2\#N \log(2)))$$

As expected, SRLG-tree has exponential complexity

Demonstration PathPlanner

SURFnet6 network:

- Over 8.800 km fiber
- Cross border fibers
  - Hamburg
  - Münster
  - Aachen

Over 200 lightpaths

Customers:
- Universities
- Hospitals
- Institutes for higher professional education
- Research institutes
- Corporate R & D dep.
- Scientific libraries

introduction
protection
SRLG
problem definition
complexity
SRLG-tree
SRLG-exclusion
perform. SRLG-tree
demo
conclusions

Conclusions

SRLG-tree has near-polynomial running time for SRLG-sparse networks

Although running time is exponential, only a fraction of the complete solution space is parsed.

Questions?