Transiently Consistent SDN Updates: Being Greedy is Hard

Saeed Akhoondian Amiri[1] Arne Ludwig[1]
Jan Marcinkowski[2] Stefan Schmid[3]

[1]TU Berlin
[2]University of Wroclaw
[3]Aalborg University

Table of contents

1 Introduction

2 Greedy is Hard

3 Polynomial Time Algorithm for Special Cases

Plan

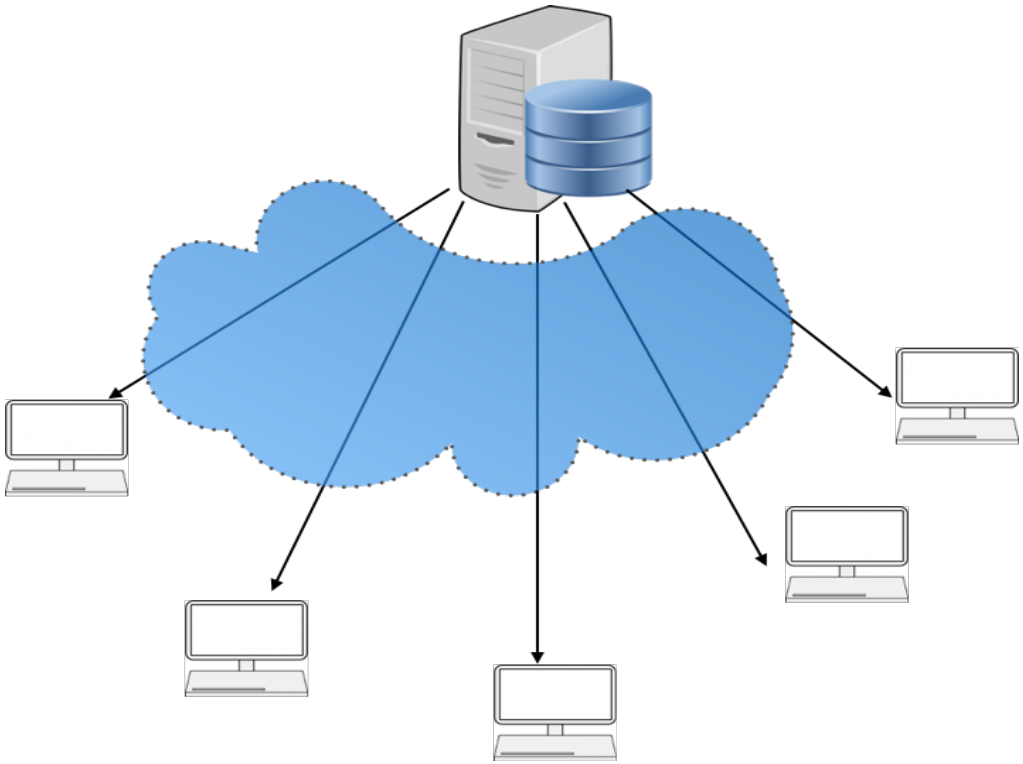
1 Introduction

2 Greedy is Hard

3 Polynomial Time Algorithm for Special Cases

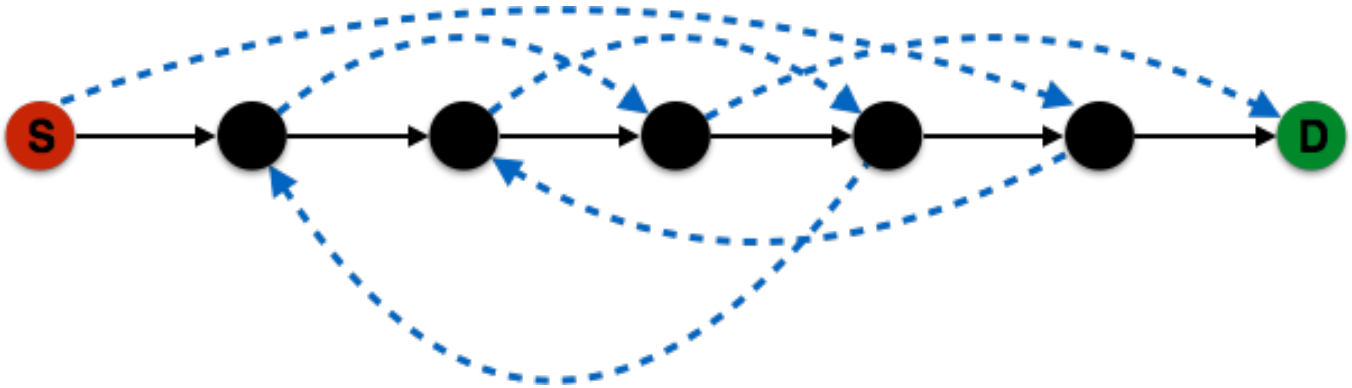
Software Defined Networks

1. Centralised Controller,
2. Asynchronous Updates,
3. Consistent Updates



Packet Routing

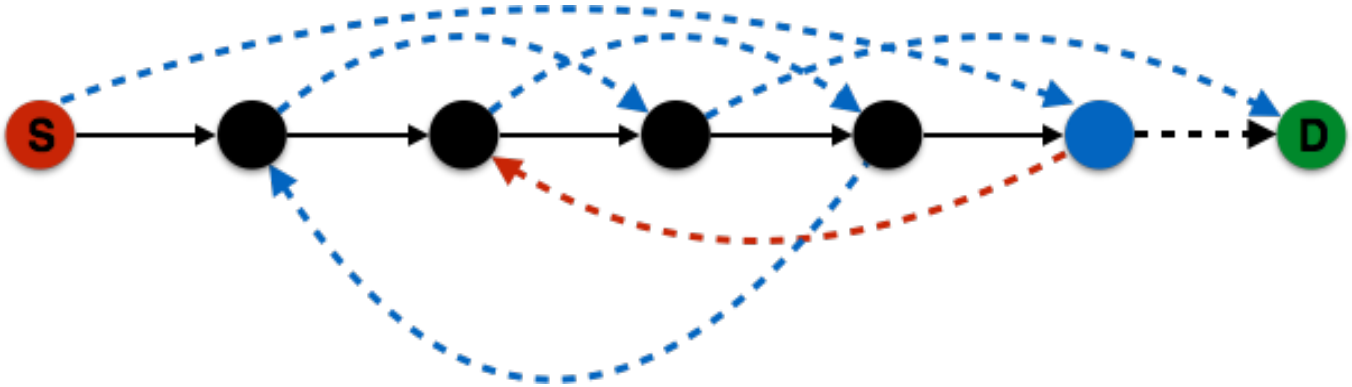
1. Initial routing from source to destination
2. New routing policy by controller
3. *Consistent* updates in few rounds.



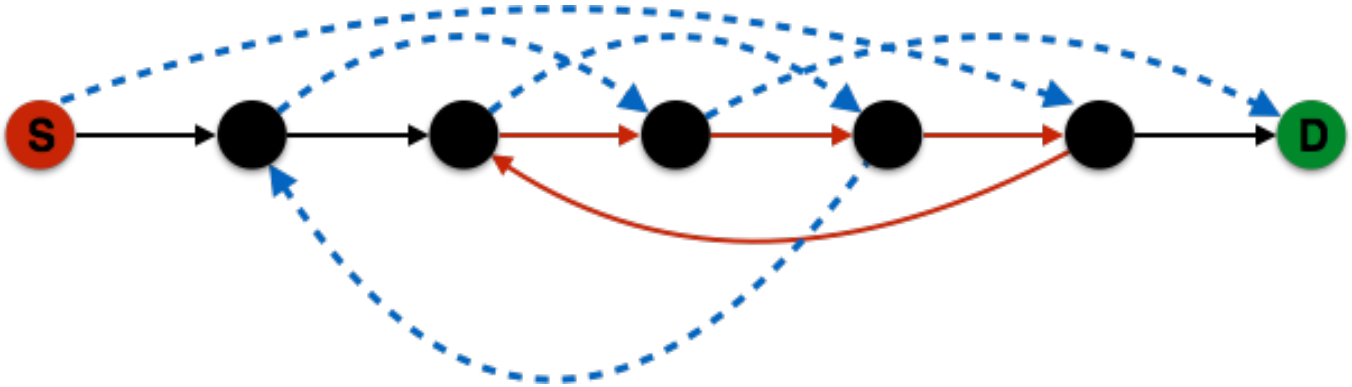
Relaxed Loop Freedom

Consistency:

1. Always there is a path from the source to the destination.
2. There cannot be any loop at any moment in a path from source to destination.

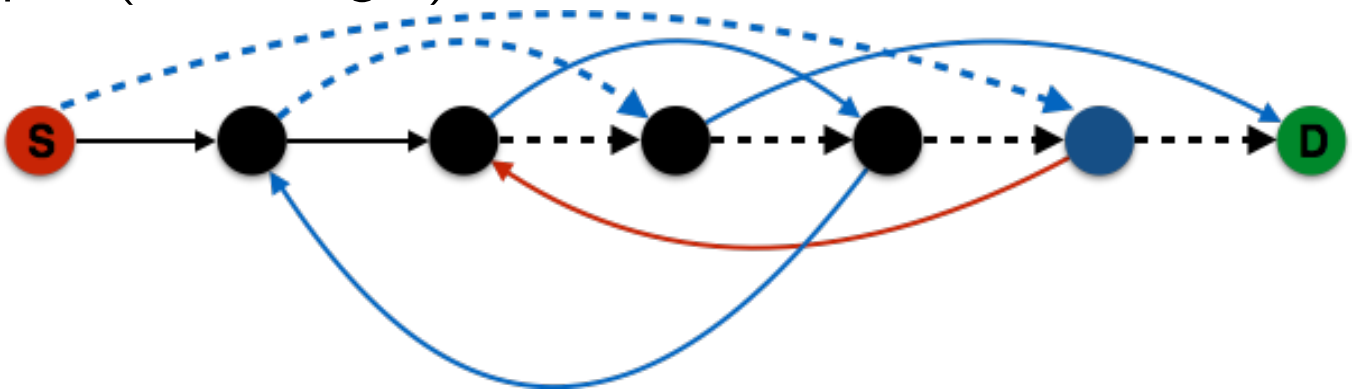


Relaxed Loop Freedom



Relaxed Loop Freedom

The source and the destination are not connected via an active path(solid edges).



Strong Loop Freedom

1. Always there is a path from the source to the destination.
2. There cannot be any loop at any moment w.r.t. active edges.

Example: Outages

Even technically sophisticated companies are struggling to build networks that provide reliable performance.

We discovered a misconfiguration on this pair of switches that caused what's called a "bridge loop" in the network.

A network change was [...] executed incorrectly [...] more "stuck" volumes and added more requests to the re-mirroring storm

Service outage was due to a series of internal network events that corrupted router data tables

Experienced a network connectivity issue [...] interrupted the airline's flight departures, airport processing and reservations systems

Relaxed (Strong) Loop Freedom

Goal: Minimise the number of rounds.

1. Optimal solution?
2. Greedy algorithm: In the round i, find the maximum number of vertices that we can update.

Greedy Algorithm (Ludwing,Marcinkowski,Schmid PODC15):

1. It is at least as hard as Feedback Arc Set (FAS) problem in the update graph.
2. FAS is hard in general graphs but easy for some graph classes.
3. Network update graph is very simple:
   Out degrees at most two
   They must constitute a legal path
   Is greedy algorithm in **P** ?!

Plan

1 Introduction

2 Greedy is Hard

3 Polynomial Time Algorithm for Special Cases

Greedy is Hard

Theorem 1

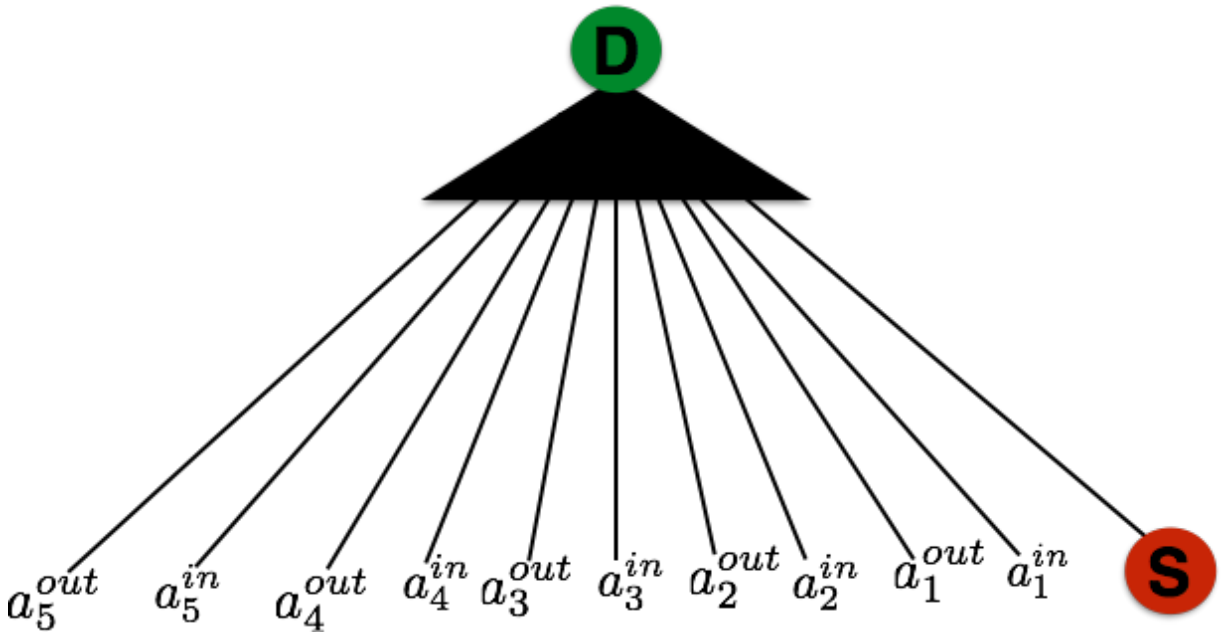
Finding the greedy solution for each round is NP-hard even in update graph.

Reduction from Hitting Set Problem:
Input: A set $\mathcal{S} = \{S_1, \ldots, S_m\}$ of subsets of $U = \{a_1, \ldots, a_n\}$.
Output: Set $S' \subseteq U$ of minimum size such that $S' \cap S_i \neq \emptyset$, For $i \in \{1, \ldots, m\}$

Greedy is Hard: Proof's General Idea

Construction

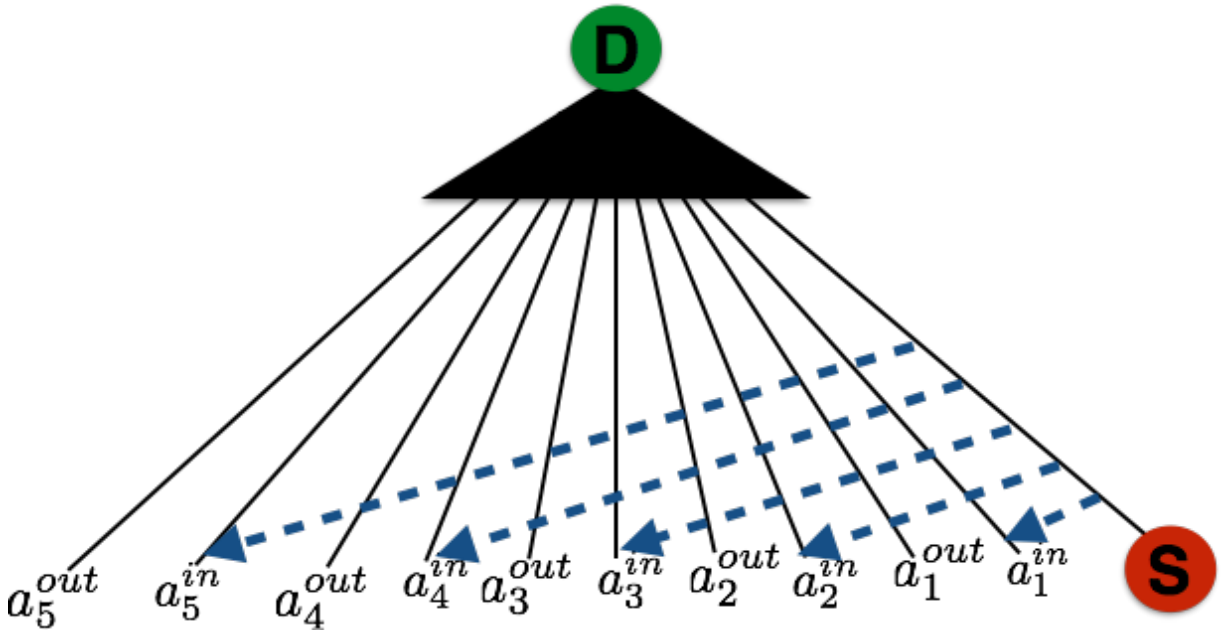
1. Directed rooted tree with $2n + 1$ branches:
   1. One $S - D$ branch
   2. For each element $a_i \in U$ two branches: $a_i^{in} - D$ (input branch) and $a_i^{out} - D$ (output branch)



Greedy is Hard: Proof's General Idea

Construction

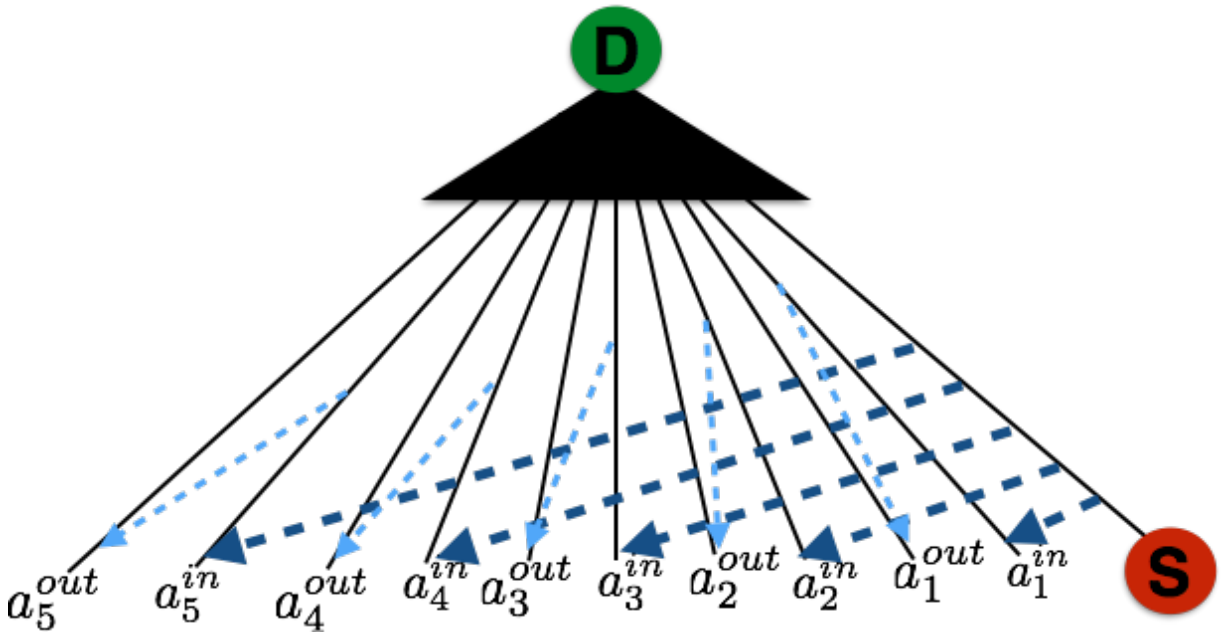
1. Directed rooted tree with $2n + 1$ branches
2. Source Connectors: Thick dashed blue edges



Greedy is Hard: Proof's General Idea

Construction

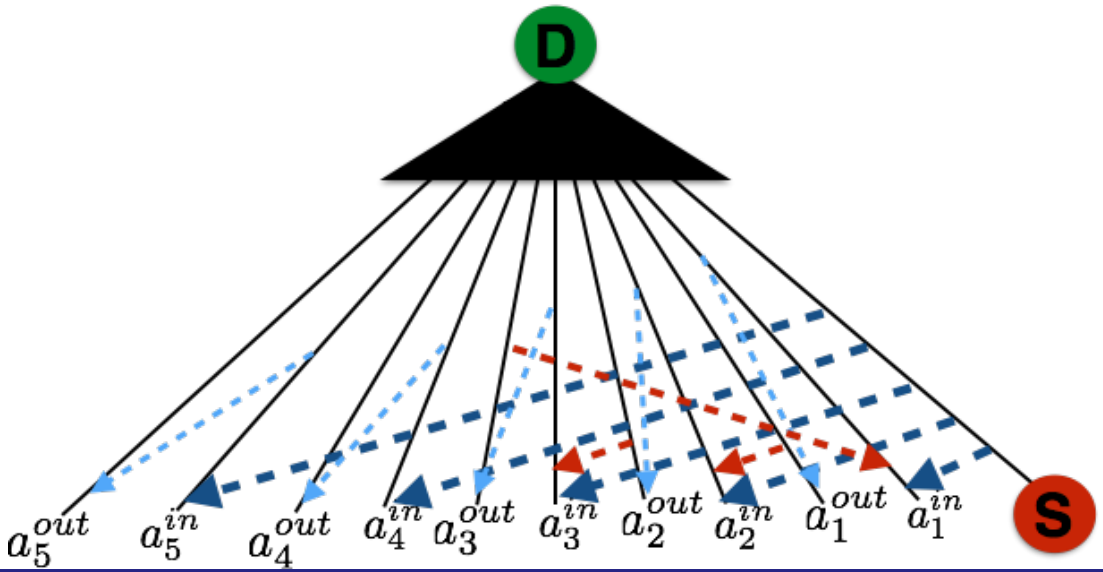
1. Directed rooted tree with $2n + 1$ branches
2. Source Connectors: Thick dashed blue edges
3. Element Selectors: From each a_i^{in} an edge to a_i^{out}, thin blue edges.



Greedy is Hard: Proof's General Idea

Construction

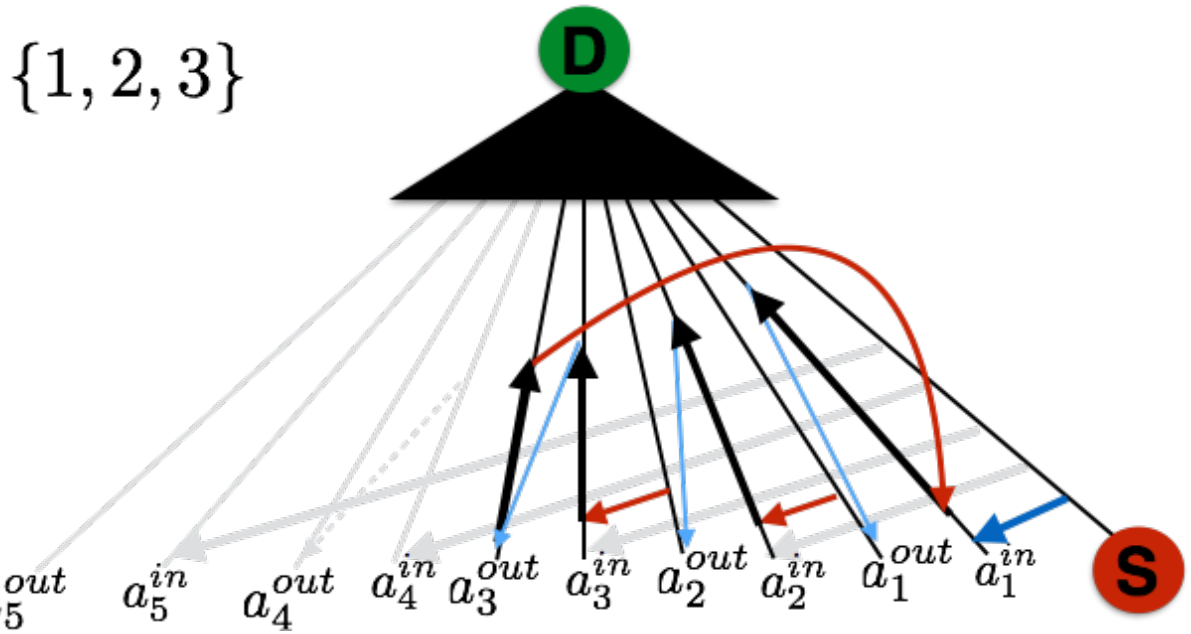
1. Directed rooted tree with $2n + 1$ branches
2. Source Connectors: Thick dashed blue edges
3. Element Selectors: From each a_i^{in} an edge to a_i^{out}, thin blue edges
4. Set Selectors: $n + 1$ edges from element a_i^{out} to a_j^{in} if a_i, a_j appear after each other in sequence S_t, Thick red edges.



Greedy is Hard: Proof's General Idea

$\mathcal{S} = \{S_1, S_2, S_3, S_4\} = \{\{1,2,3\}, \{1,4\}, \{2,5\}, \{3,2,5\}\}$.

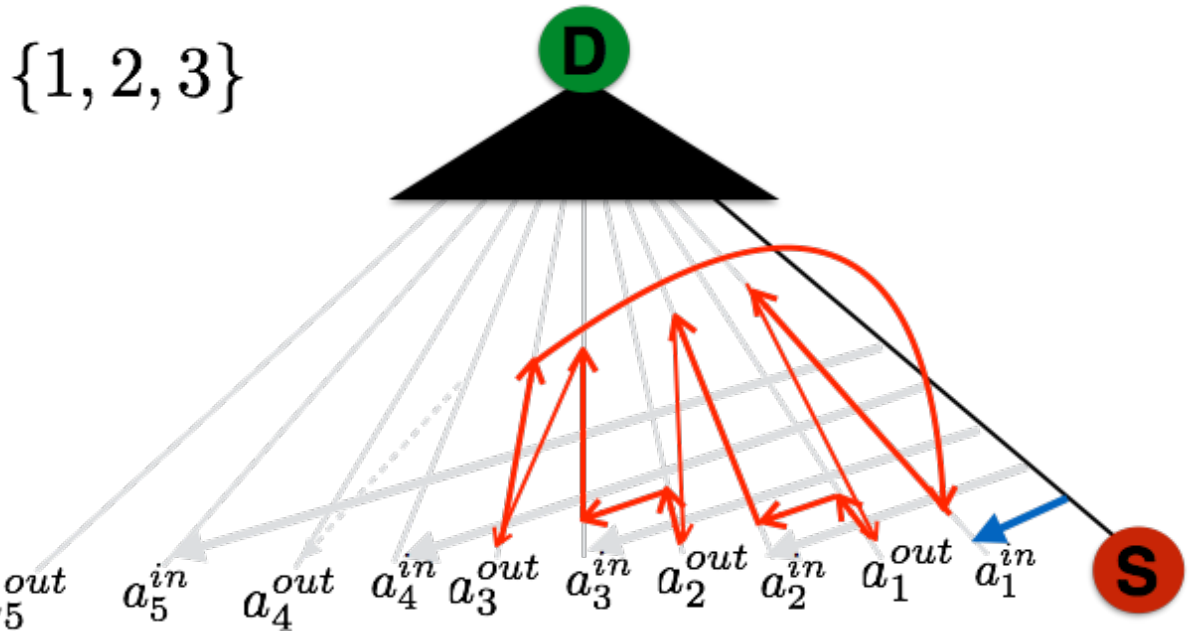
1. Each set S_i corresponds to a cycle,
2. Deleting n element selector edge destroys all cycles,
3. Minimum number of not updatable edges = Minimum hitting set.



Greedy is Hard: Proof's General Idea

$\mathcal{S} = \{S_1, S_2, S_3, S_4\} = \{\{1,2,3\}, \{1,4\}, \{2,5\}, \{3,2,5\}\}$.

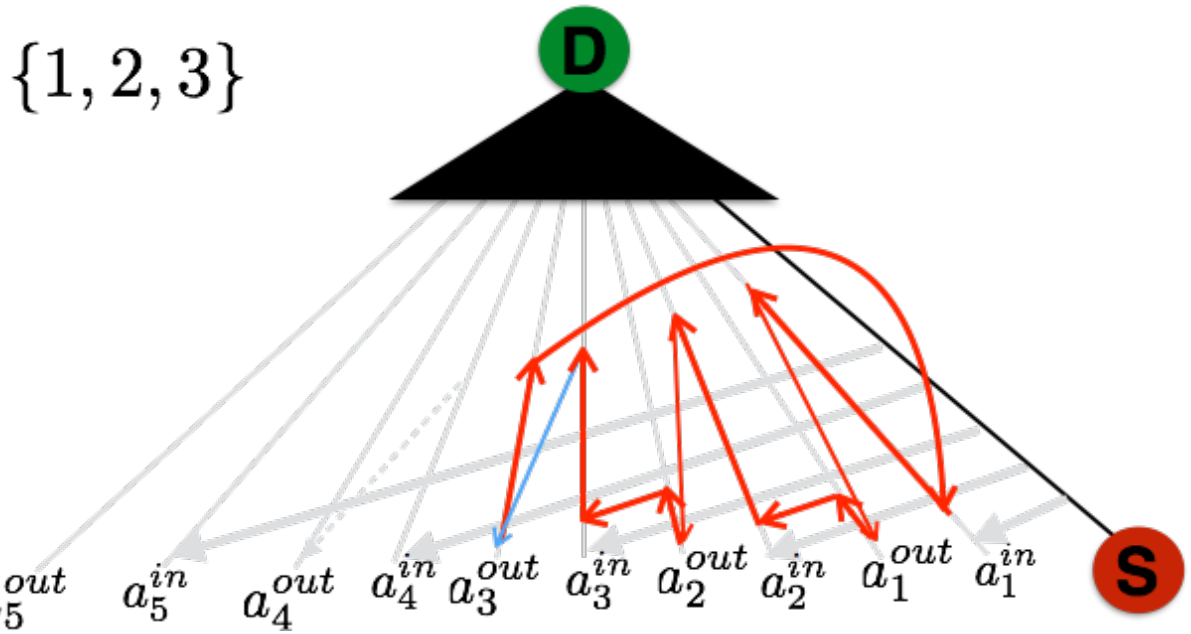
1. Each set S_i corresponds to a cycle,
2. Deleting n element selector edge destroys all cycles,
3. Minimum number of not updatable edges = Minimum hitting set.



Greedy is Hard: Proof's General Idea

$\mathcal{S} = \{S_1, S_2, S_3, S_4\} = \{\{1,2,3\}, \{1,4\}, \{2,5\}, \{3,2,5\}\}$.

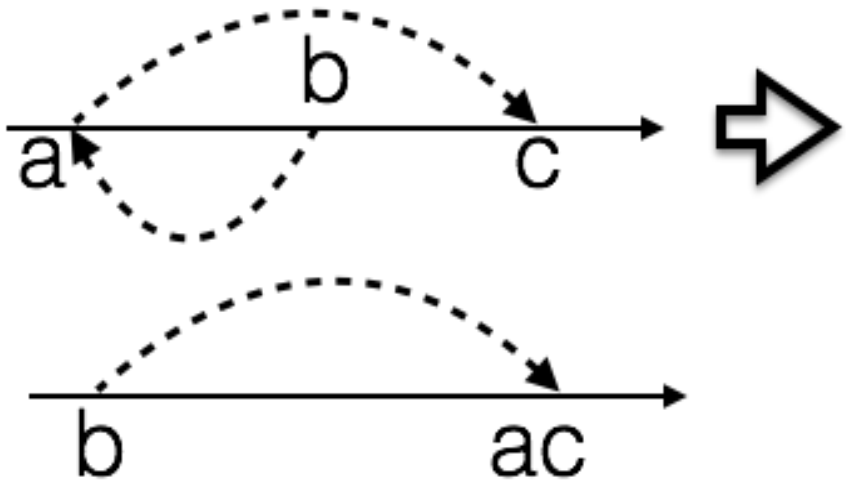
1. Each set S_i corresponds to a cycle,
2. Deleting n element selector edge destroys all cycles,
3. Minimum number of not updatable edges = Minimum hitting set.



Greedy is Hard: Special Update Graph?

Is it possible to reach that special update graph from some initial architecture?
Long Story Short: Yes it is possible, in one greedy step we are there

Greedy is Hard: Forward Edges



Plan

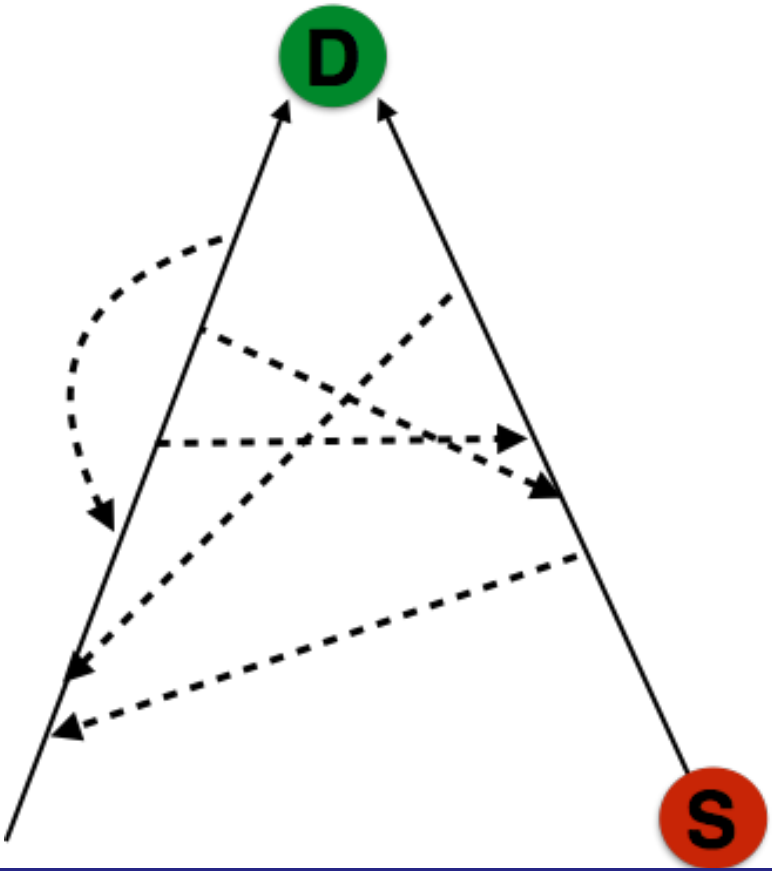
1 Introduction

2 Greedy is Hard

3 Polynomial Time Algorithm for Special Cases

Two Branches

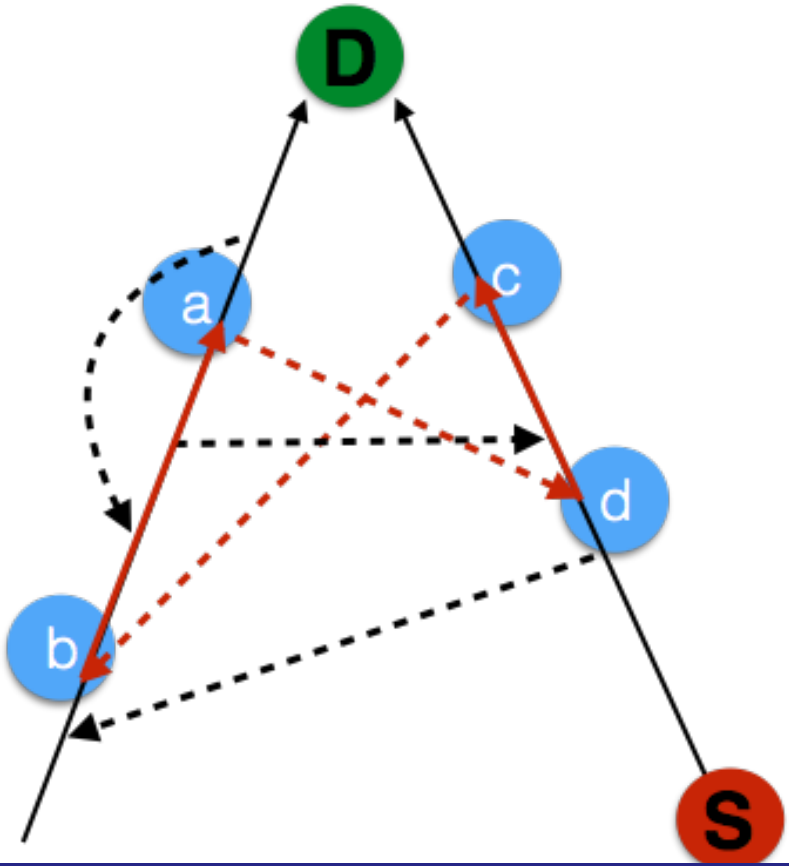
Update graph:
Active Edges: Rooted directed tree with two branches
Update edges: Rest of edges (Dashed Edges)



Two Branches

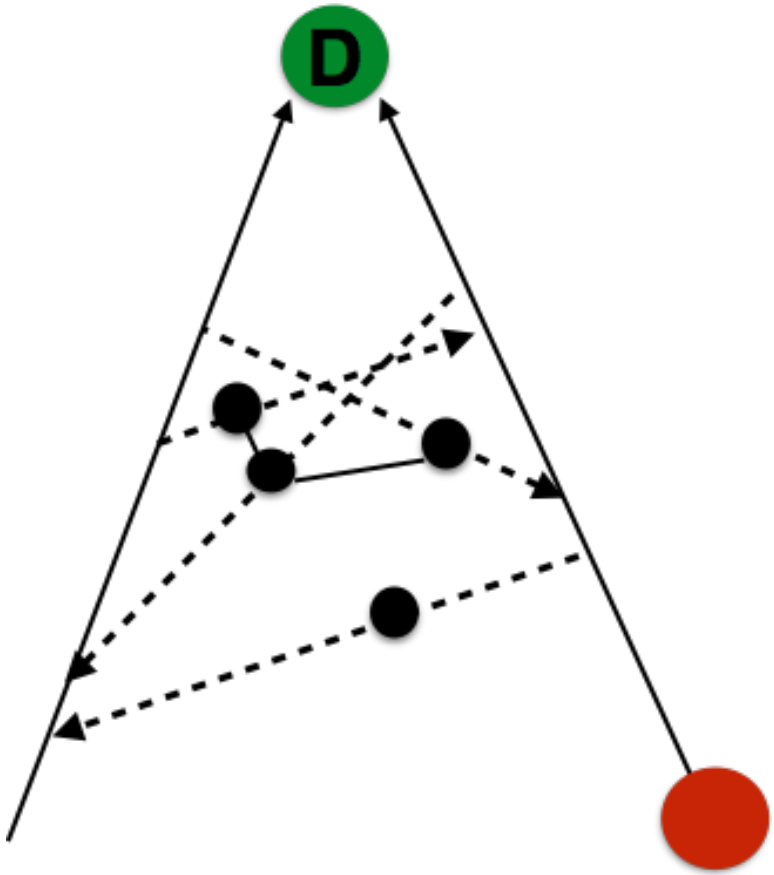
Cycle:

1. If two update edges (e_1, e_2) cross each other, and
2. Head of e_i is below tail of $e_{i'} (i, i' \in \{0, 1\})$.



Two Branches: Strong LF (SLF)

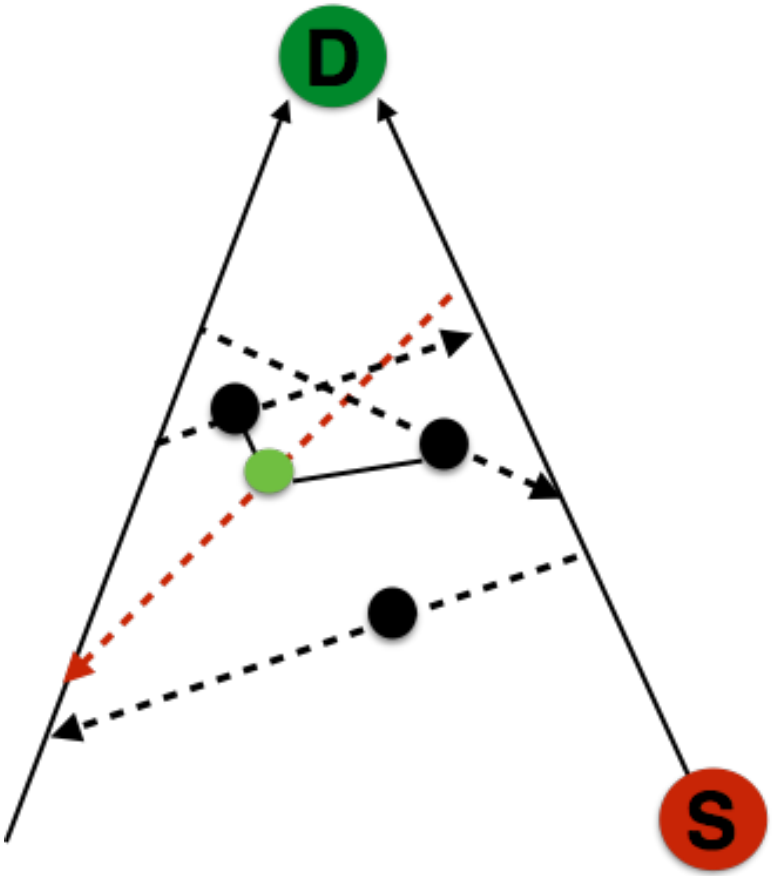
1. Update Edge: Assign a Vertex,
2. Connect two vertices if they correspond to a cycle.



Two Branches: SLF

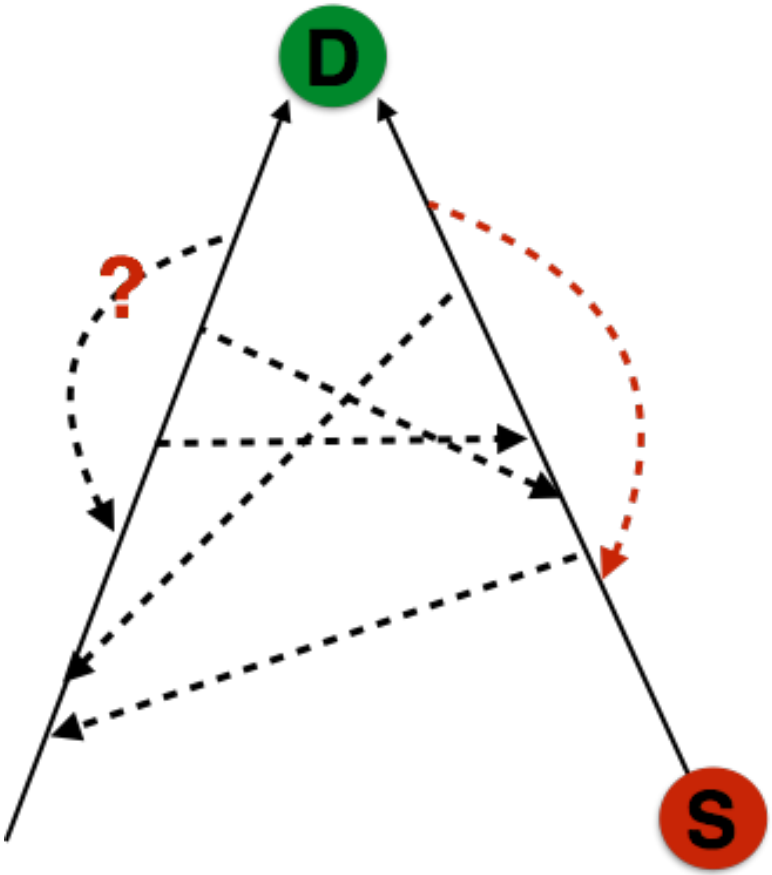
Minimum Vertex Cover $\Leftrightarrow$ Minimum Non Updatable Edges

Graph is Bipartite $\Rightarrow$ Polynomial Time Solvable.



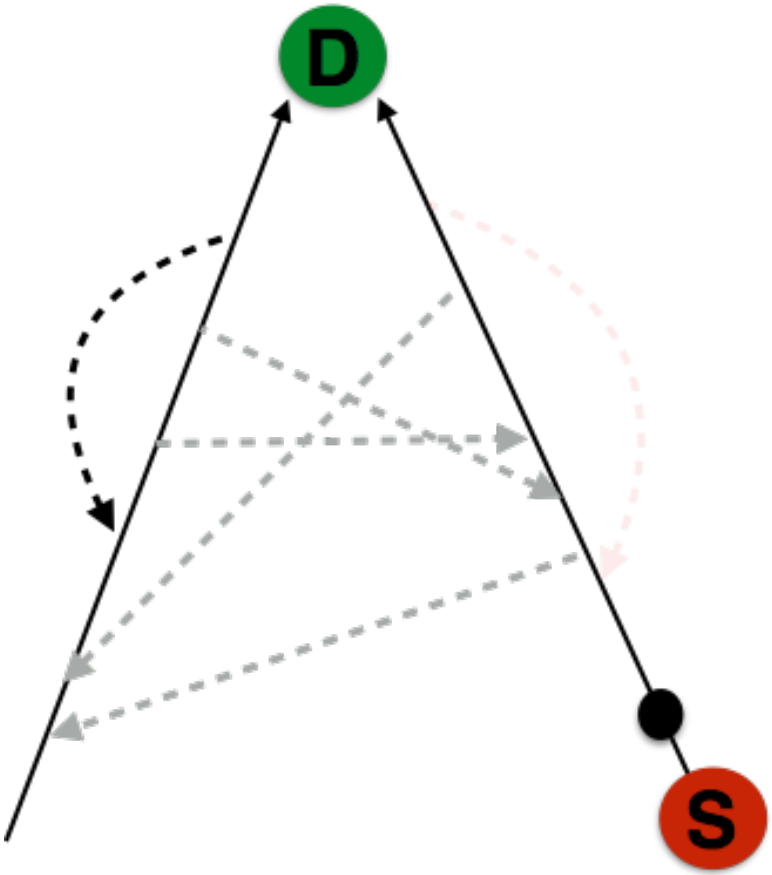
Two Branches: Relaxed LF

Simulate update edges in the second branch to reduce it to SLF



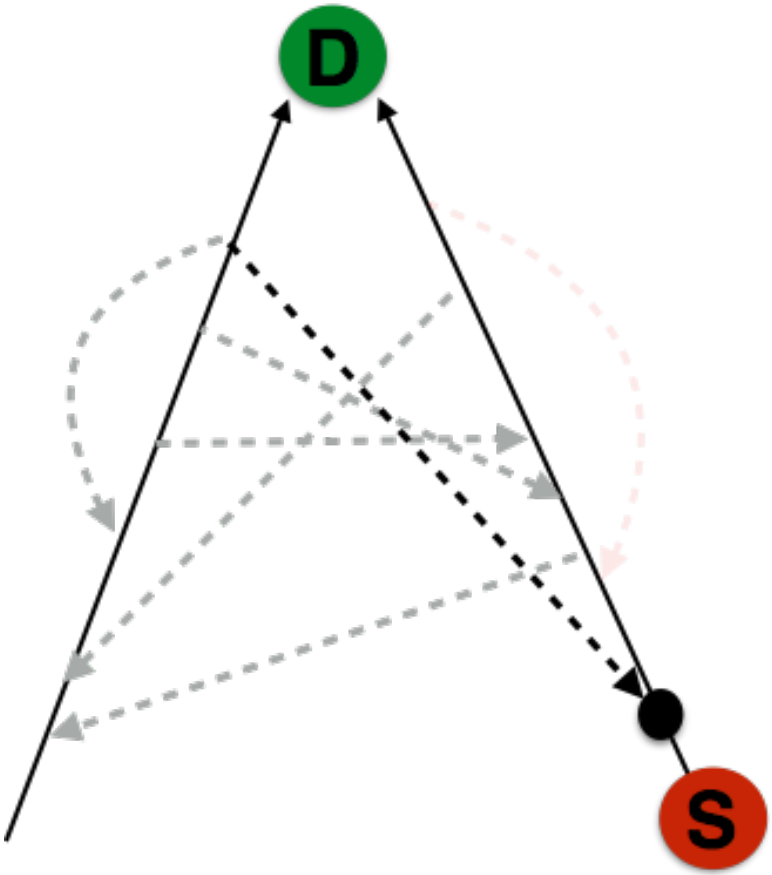
Two Branches: Relaxed LF

Simulate update edges in the second branch to reduce it to SLF



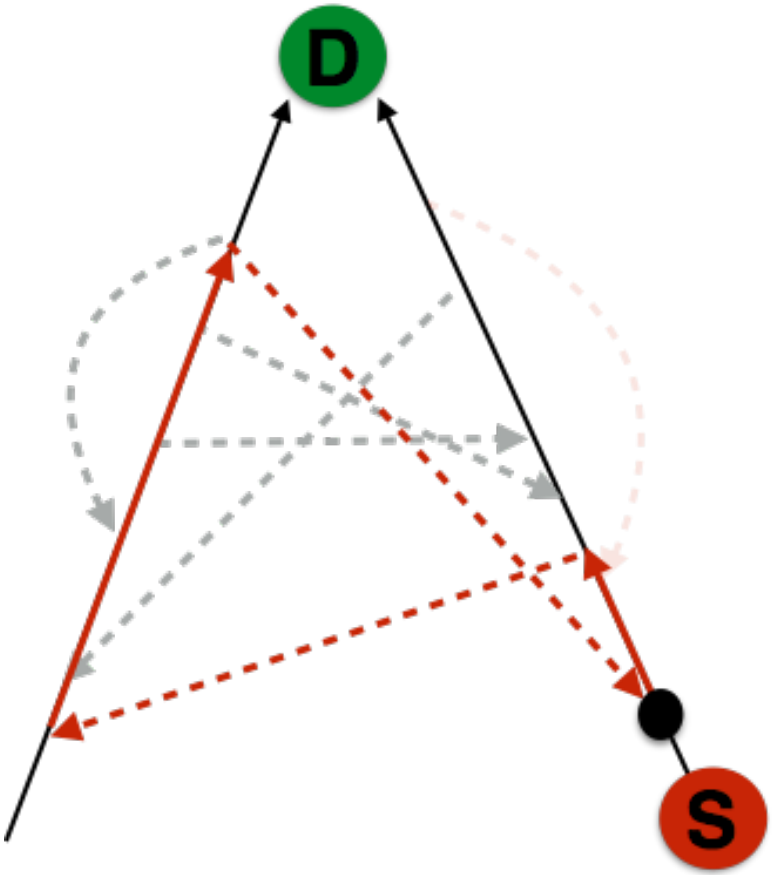
Two Branches: Relaxed LF

Simulate update edges in the second branch to reduce it to SLF



Two Branches: Relaxed LF

Simulate update edges in the second branch to reduce it to SLF



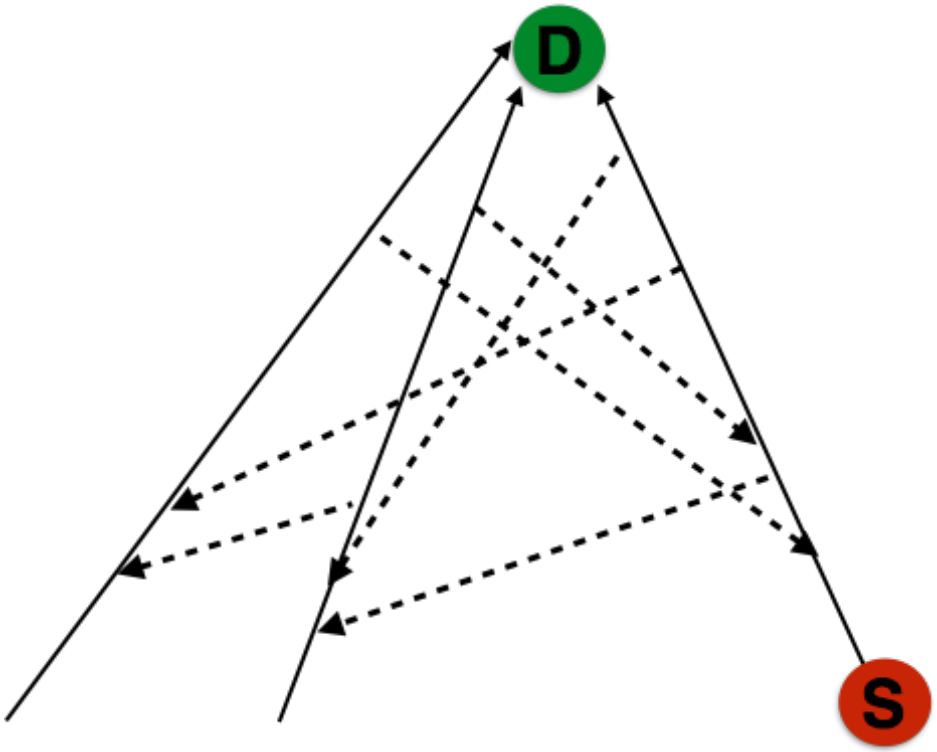
Three Branches

Main Edges: They form three branches

Update edges: Connection between three branches

Construct a similar model as before?

Vertex cover in cubic graphs is NP-Hard (Brooks' Theorem)



More General Graphs?

1. What if the graph has bounded directed tree-width
   Claim: It is NP-hard.
2. Is greedy hard even in update graph with 3 branches?
3. What if Feedback Arc Set is small?

Thank you