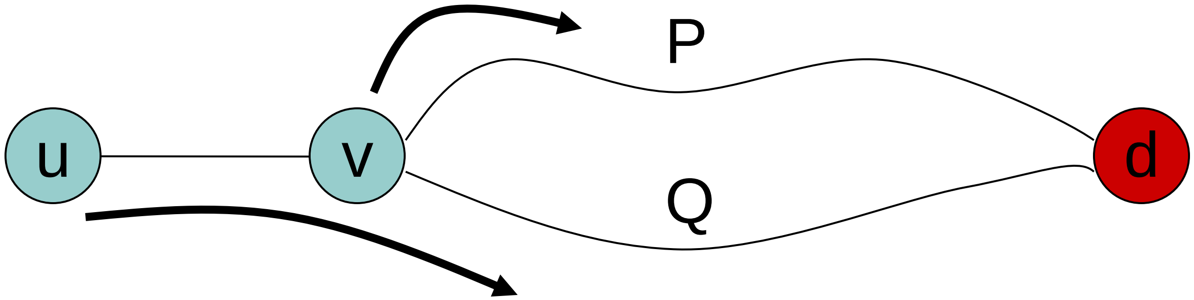
BGP Divergence

Computer Networks
Dr. Jorge A. Cobb

Overview

- We will show how BGP diverges
- Provide a formal model for the study of this divergence
- Detecting if BGP will diverge is intractable (probably will not go through this)
- We present a sufficient condition to prevent divergence
- There are dynamic solution to prevent divergence at run time, at the expense of efficiency (probably will not go through this)

Conflicting Policies

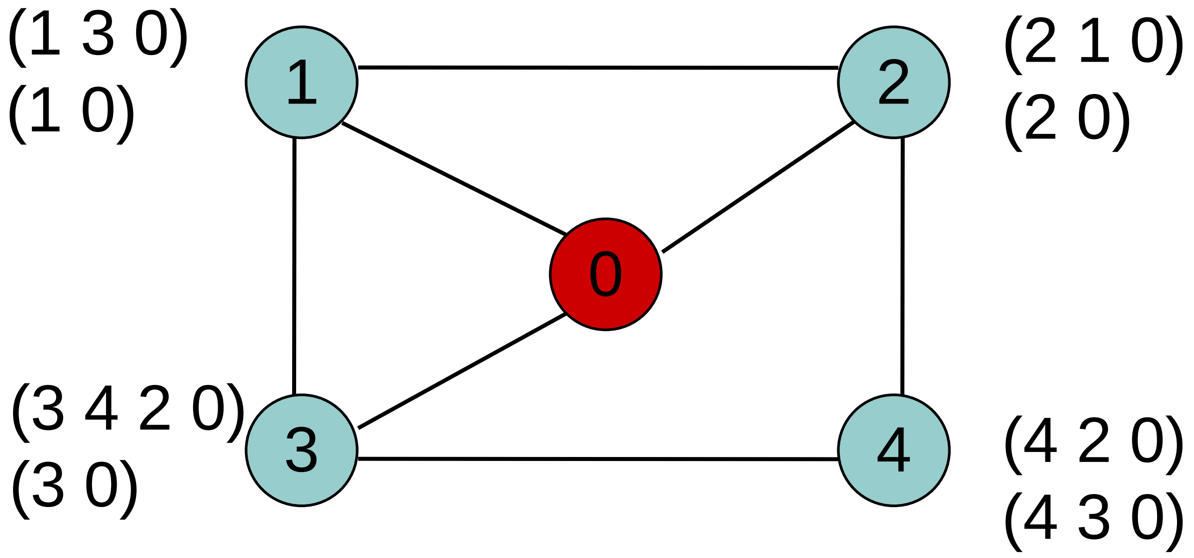


Router v prefers	P	over	Q
Router u prefers	(u v)Q	over	(u v)P

Conflicting policies can cause divergence!

Stable Path Problem (SPP)

- Provides a formalization of BGP.
- Designed by T. Griffin, B. Shepherd, G. Wilfong.
- Each node represents an **entire AS**

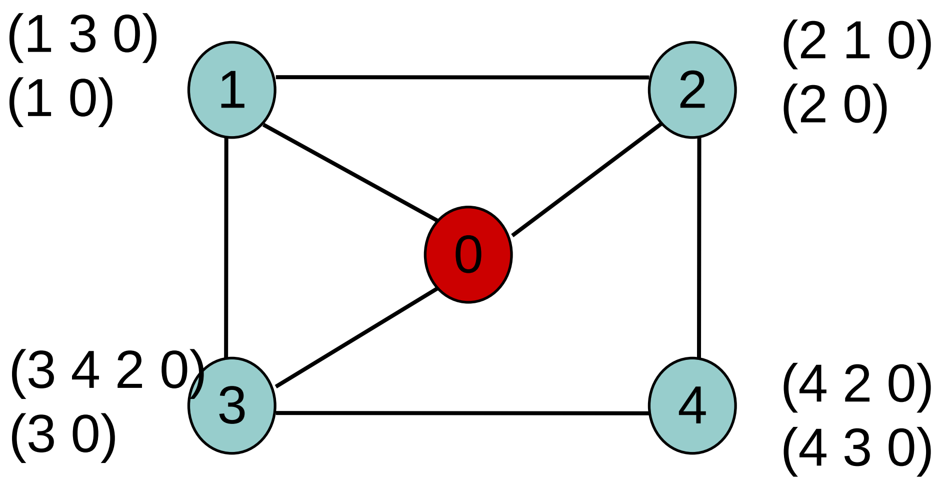


Stable Paths Problem (formally)

- An instance S of the stable paths problem is a triple,

  S = (G, $\mathcal{P}$,Λ)

  - G = network graph
  - $\mathcal{P}$ is the set of permitted paths
  - Λ is the ranking of the permitted paths
- We overview next each of the components.

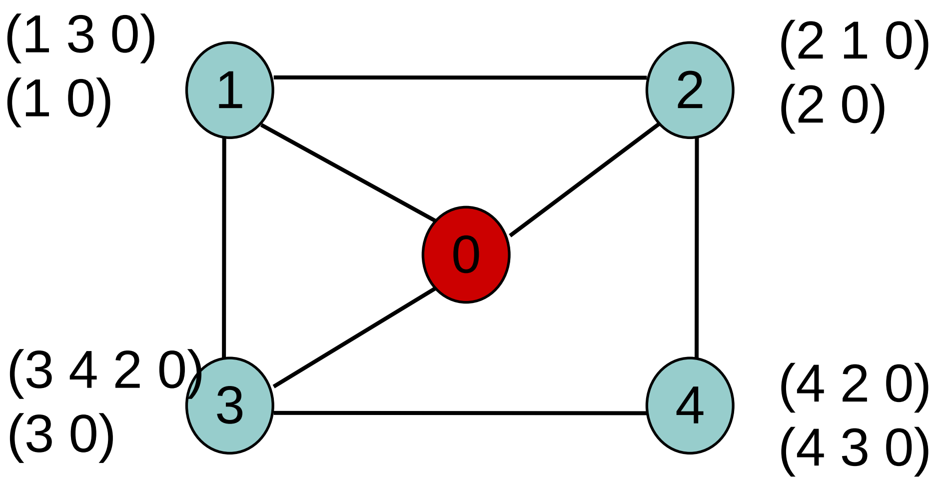


Graph G

- The network is a graph G = (V, E)
  - V = {0, 1, 2, . . . , n} is the set of nodes
  - 0 is the origin (destination)
  - E is the set of undirected edges
- Peers(u) = $\{v \mid \{u,v\} \in E\}$
- A path is a (possibly empty) sequence of nodes
- ε denotes the empty path

Permitted Paths

- For each $u \in V$, $\mathcal{P}^u$ is the set of *permitted paths* of u.
- $\mathcal{P}^0$ = { (0) } (always, by definition, 0 can reach 0 ☺)
- $\mathcal{P}^1$ = { (1, 3, 0), (1, 0), ε }
- $\mathcal{P}^2$ ={ (2, 1, 0), (2, 0), ε }
- $\mathcal{P}^3$ = { (3, 4, 2, 0), (3, 0), ε }

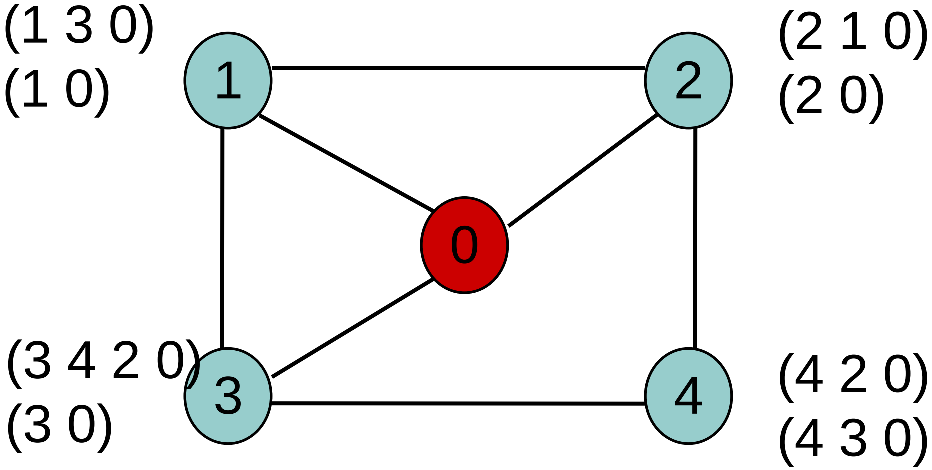


Permitted Paths: details

- $\mathcal{P}^0 = \{ (0) \}$
- For each u, $u \neq 0$,
  - $\varepsilon \in \mathcal{P}^u$
  - for each $P \in \mathcal{P}^u$,
    - the first node in P is u
    - the last node in P is 0.
    - P is a simple path
- If $P = (u, v, w, \ldots, 0) \in \mathcal{P}^u$, then v is the *next hop* of P
- $\mathcal{P}$ = union over of all u of $\mathcal{P}^u$

Ranking Function

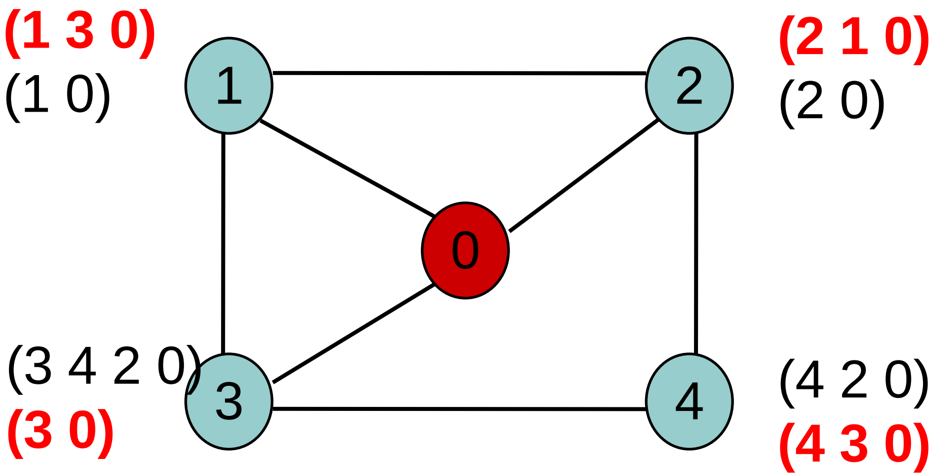
- λ^u is a ranking function over $\mathcal{P}^u$
- λ^u (P) represents how desirable P is to u.
- If $P_1, P_2 \in \mathcal{P}^u$ and $\lambda^u(P_1) < \lambda^u(P_2)$, then
  - P_2 is said to be *preferred* over P_1
- For every u and P $\in \mathcal{P}^u$, where P $\neq \varepsilon$,
  - $\lambda^u(\varepsilon) < \lambda^u(P)$



- $\lambda(3, 0) < \lambda(3, 4, 2, 0)$
  - $\lambda(\varepsilon) < \lambda(3, 0)$
- $\lambda(1, 0) < \lambda(1, 3, 0)$
  - $\lambda(\varepsilon) < \lambda(1, 0)$

Path Assignments

- A path assignment is a function π that gives a path to each node u (think of it as the "state" of the system)
- $\pi(u) \in \mathcal{P}^u$
- Below:
  - $\pi(2) = (2\ 1\ 0)$, obviously this is a problem since 1 is not taking path (1 0) at this moment



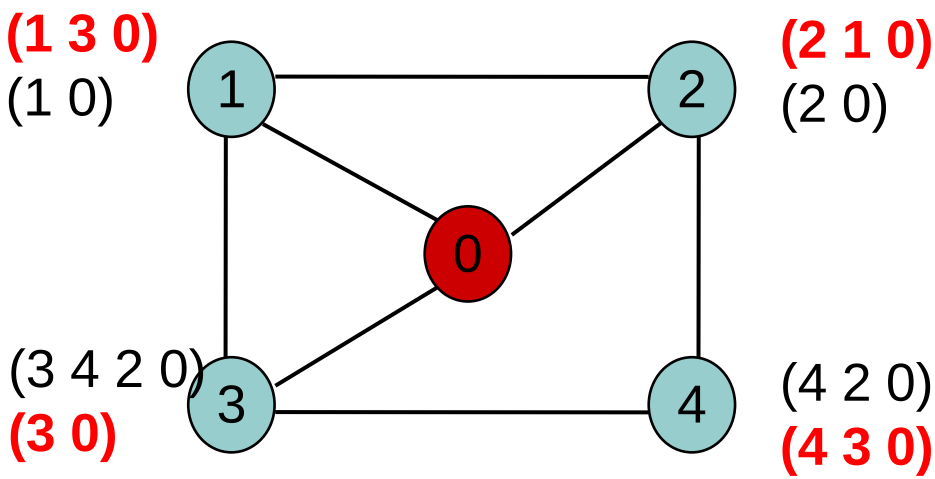
Path Assignments

- Given a path assignment π, choices(π,u) lists the possible new paths of u.

$$\text{choices}(\pi,u) = \{ (u ; \pi(v)) \;\;|\;\; \{u,v\} \in E \} \cap \mathcal{P}^u \text{ if } u \neq 0$$

$$\{ (0) \} \text{ if } u = 0$$

note: empty path is always a choice although I did not explicitly include it above



choices(1) = {ε, (1, 3, 0), (1, 0)}

choices(2) = {ε, (2, 0)}

choices(3)? choices(4)?

Stable Path Assignments

- best(π,u) = best possible choice for u, **given** a path assignment π

  best(π,u) = P iff

  $P \in \text{choices}(\pi,u) \wedge$

  $(\forall P', P' \in \text{choices}(\pi,u), \lambda^u(P) \geq \lambda^u(P'))$

- A path assignment is stable iff for all u,

  $\pi(u) = \text{best}(\pi,u)$

  - Stable, NOT optimal!
  - Nodes may not end up with their highest ranking path

Example

- best(0) = { (0) }
  best(1) = { (1,3,0) }
  best(2) = { (2,0) }
  best(3) = { (3,0) }

best(4) = { (4,3,0) }

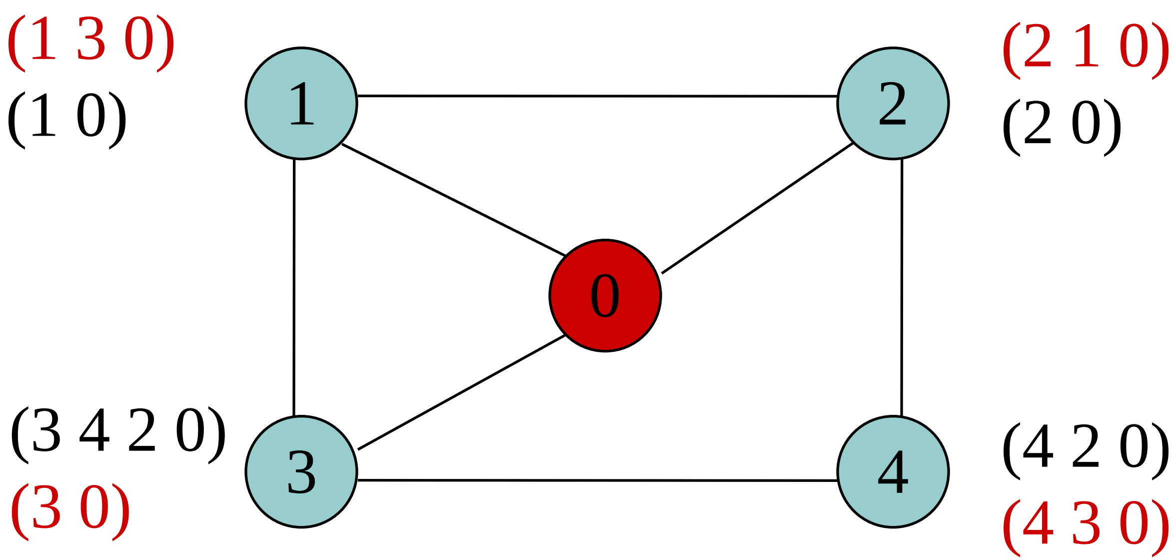
Choices(0) = { (0) }

Choices(1) = { (1,3,0), (1,0), ε}

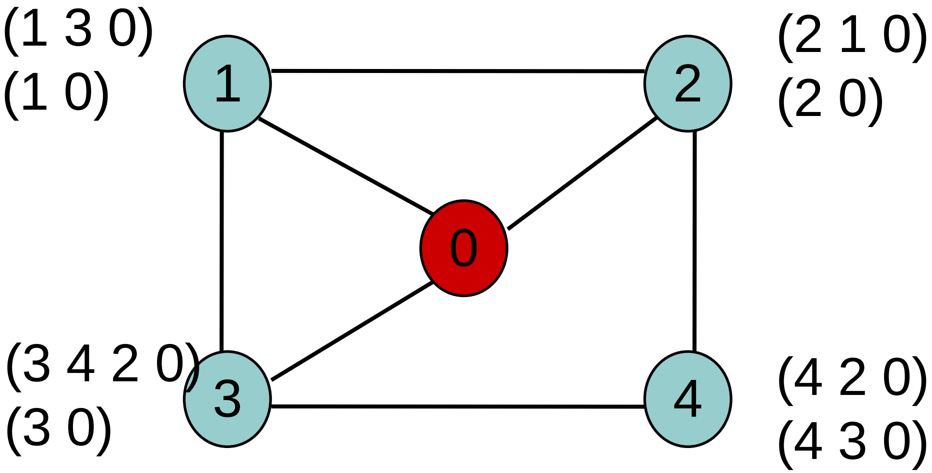
Choices(2) = { (2,0), ε }

Choices(3) = { (3,0), ε }

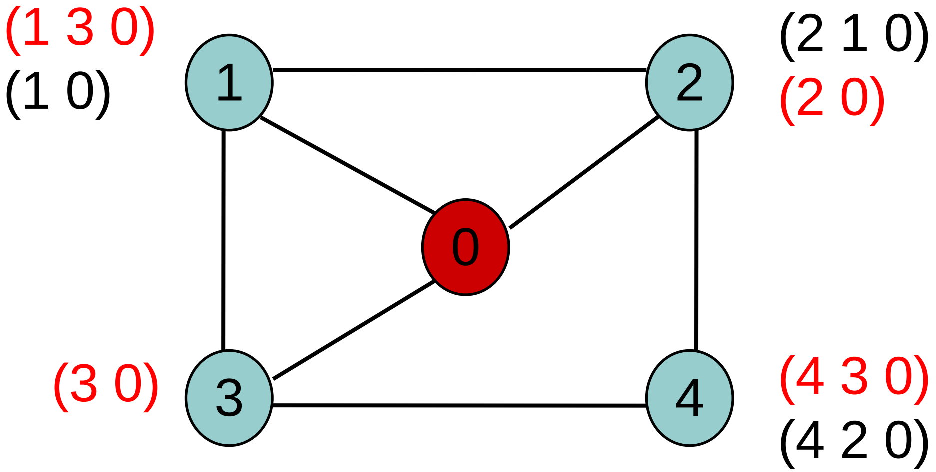
Choices(4) = { (4,3,0), ε}



SPP Examples



Bad gadget: this SPP instance has no stable path assignment



Good gadget: this SPP instance has a stable path assignment (in red)

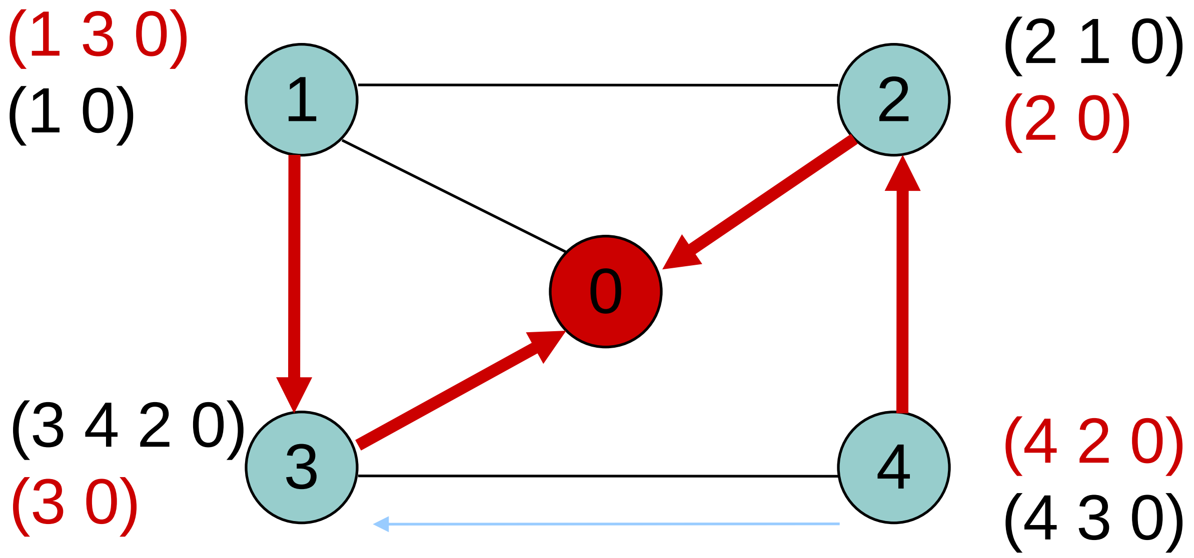
Note: 2 is not taking its highest ranked path

Executing The System

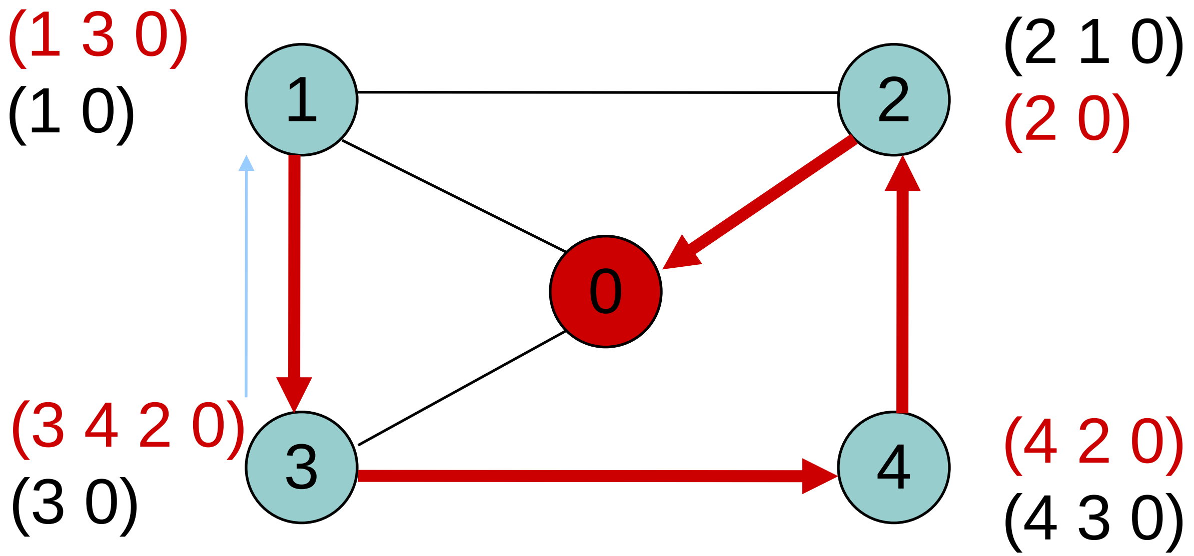
- An execution step of the system consists of the following
  - Pick any node u arbitrarily (with some fairness, of course, don't ignore some nodes forever)
  - Compute best(π,u), where π is the current path assignment
  - π(u) := best(π,u)

- Repeat forever until can't do another step (i.e. a stable path assignment is found)

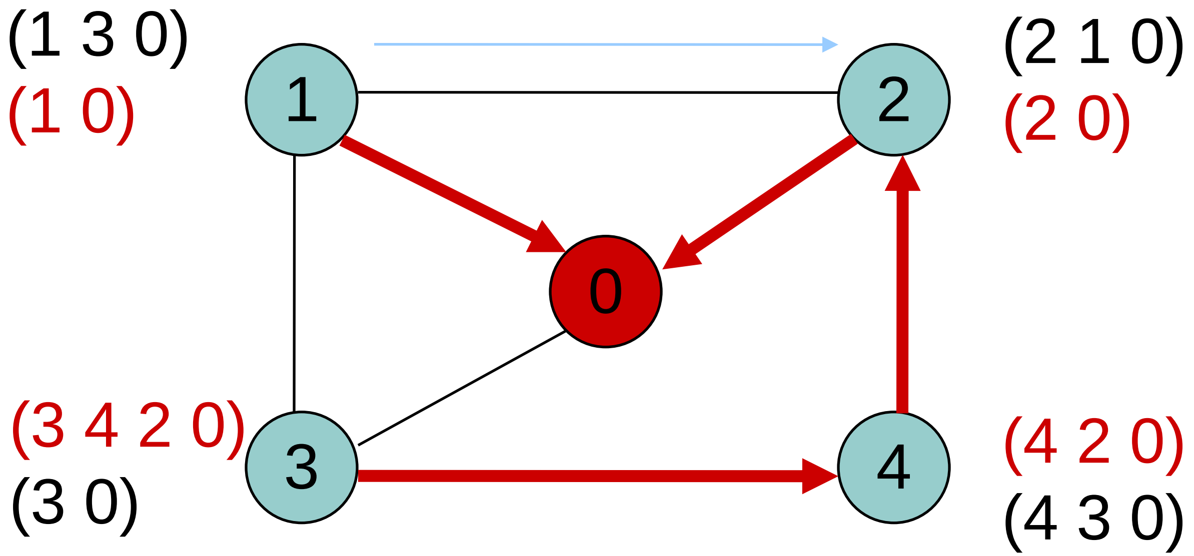
Executing Bad Gadget



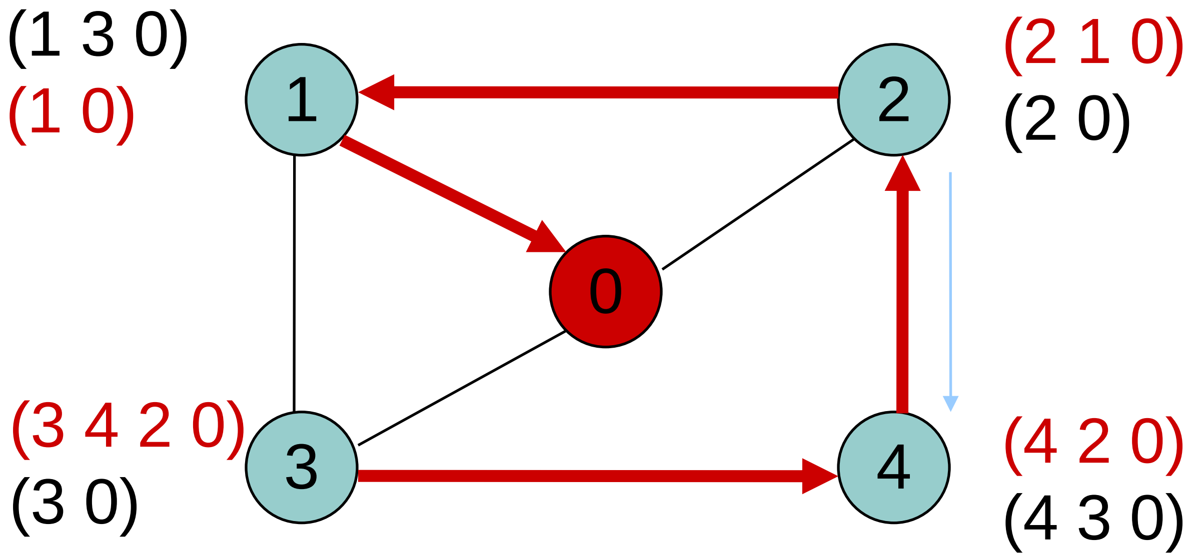
Executing Bad Gadget



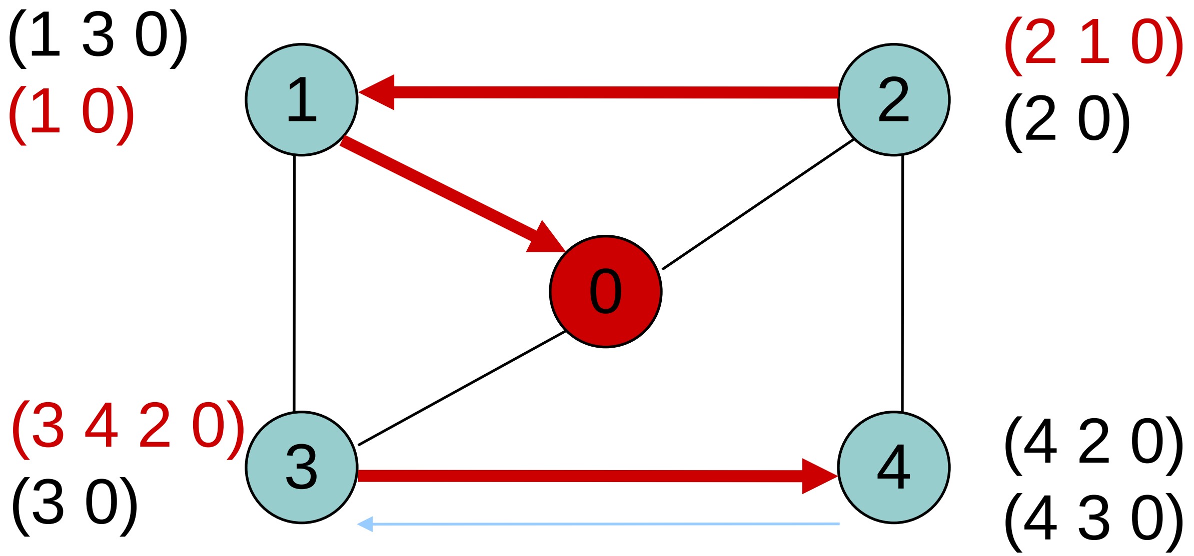
Executing Bad Gadget



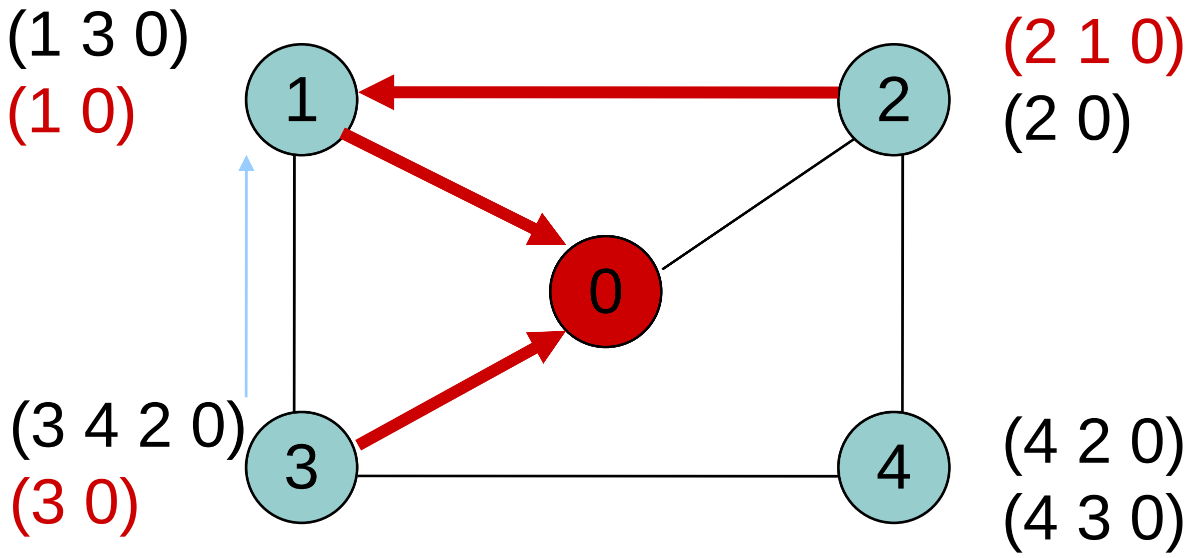
Executing Bad Gadget



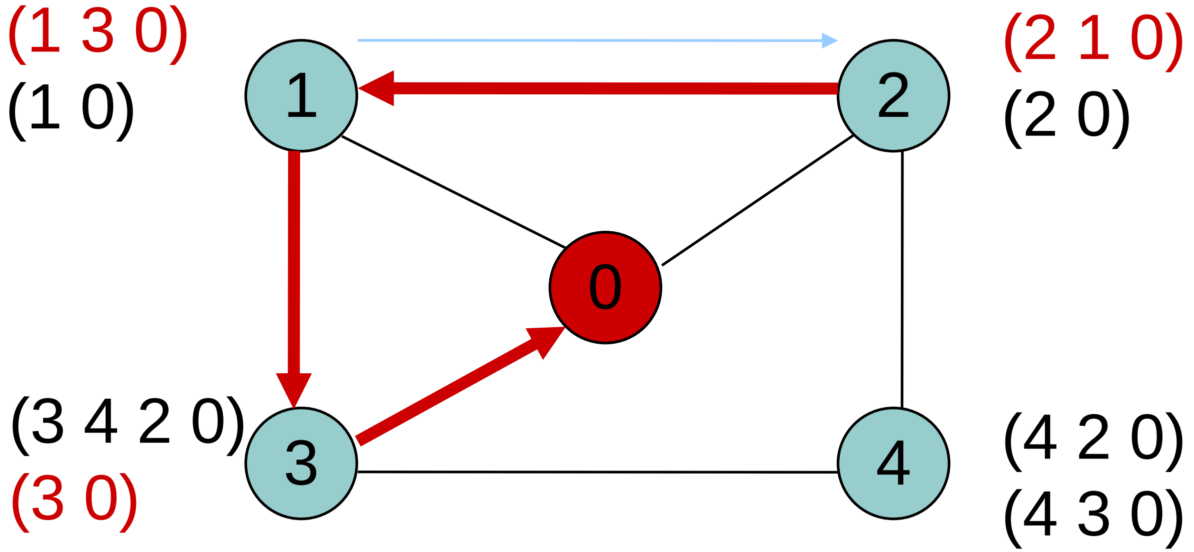
Executing Bad Gadget



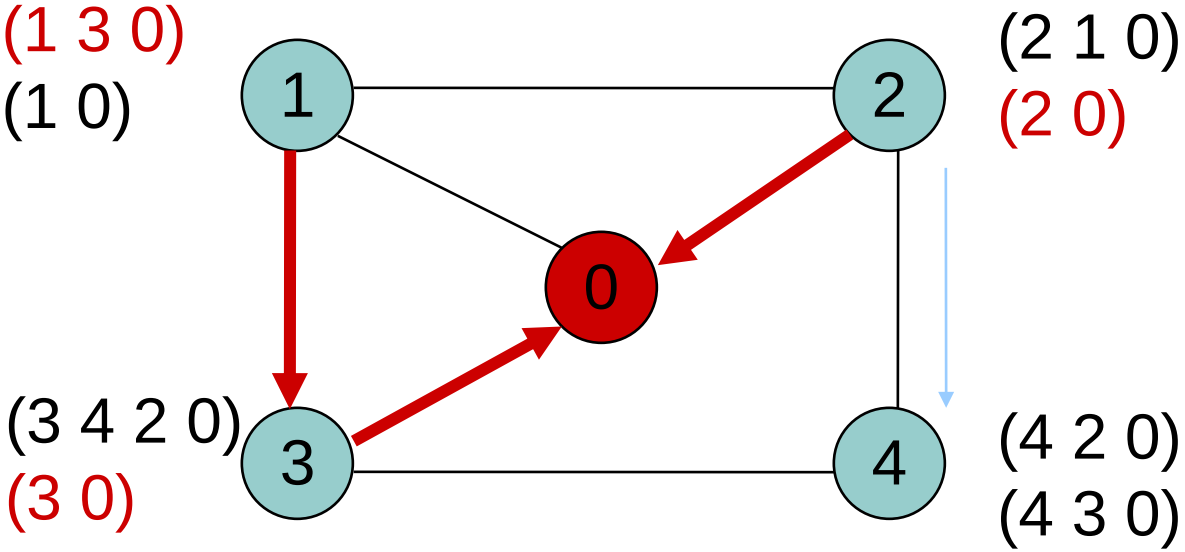
Executing Bad Gadget



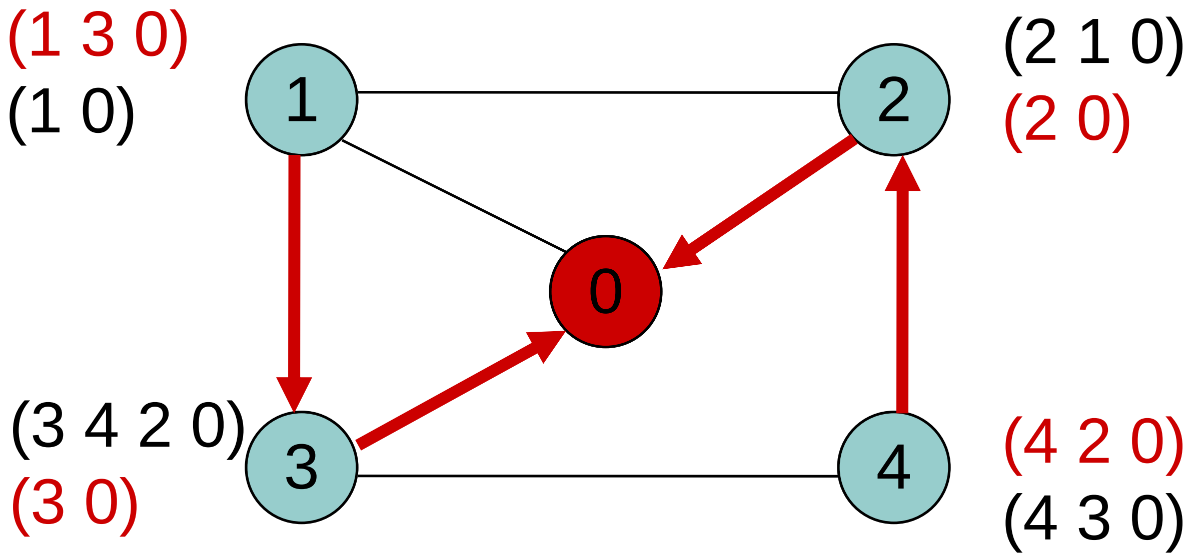
Executing Bad Gadget



Executing Bad Gadget



Executing Bad Gadget

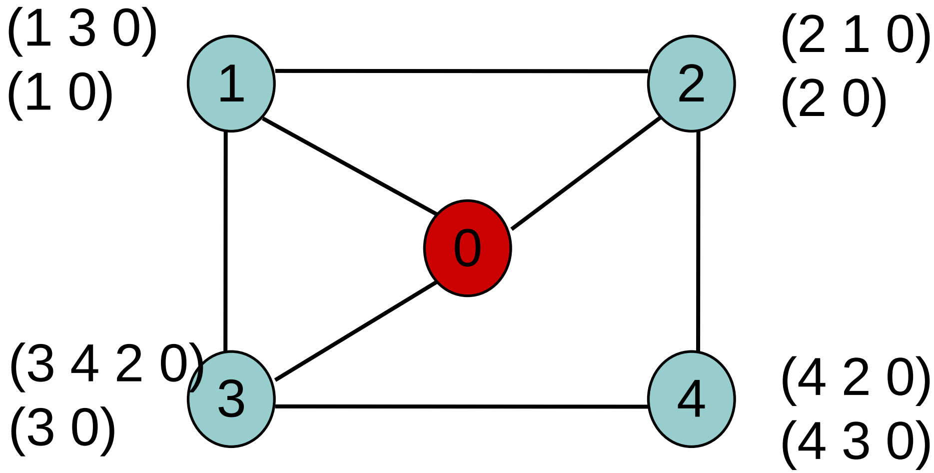


Possible divergence!

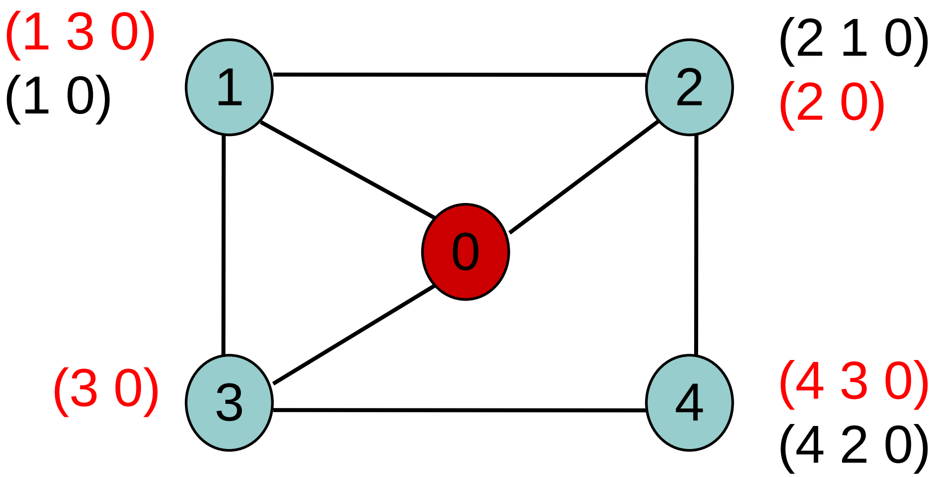
Solvable SPP

- A SPP S = (G, $\mathcal{P}$,Λ) is solvable iff there is a stable path assignment of S.
  - This path assignment is known as a "solution".
- Note that some nodes may have an empty path in a given solution.
  - if choices(π,u) only has empty path then best(π,u) is the empty path
- Some SPP instances may have more than one solution

Example of SPP solutions



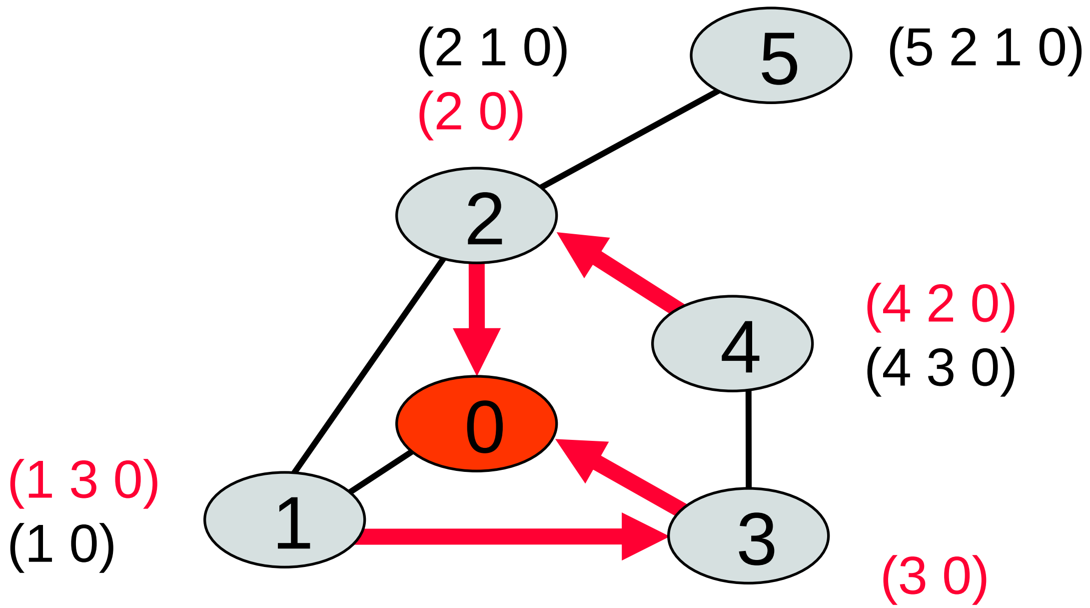
Bad gadget: has no solution



Good gadget: has a <u>single</u> solution (in red)

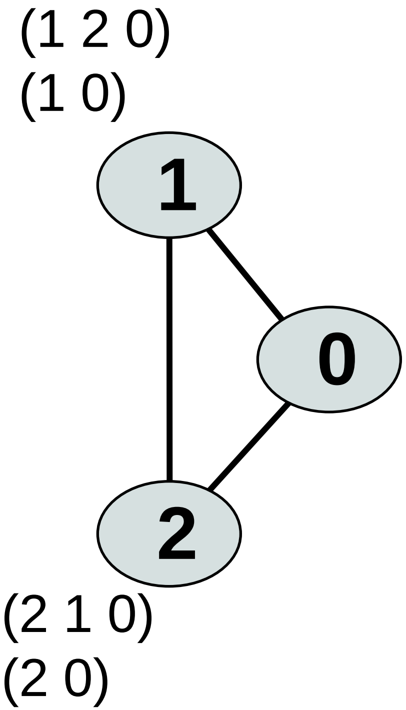
Note: 2 is not taking its highest ranked path

Single Solution Example

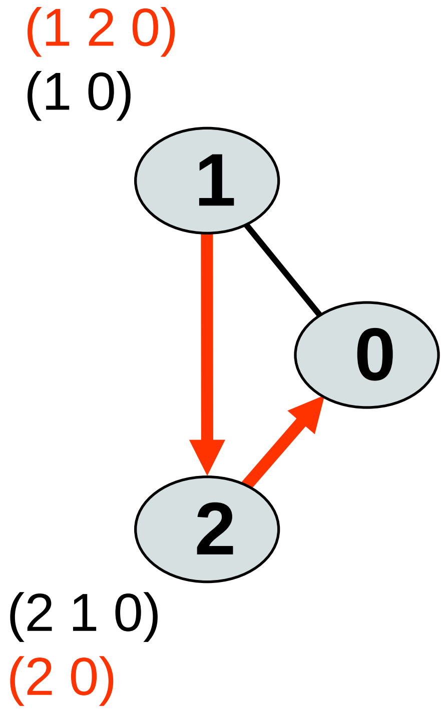


- Has a single solution (in red above)
- Notice node 5 (empty path)
- A solution need not represent a shortest path tree, or a spanning tree.

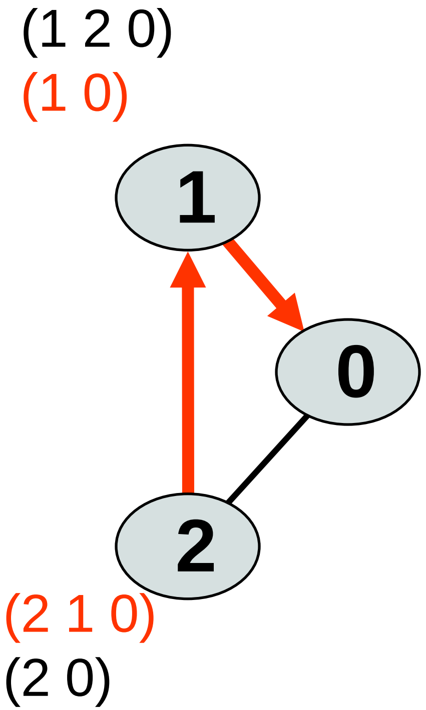
Multiple Solutions



DISAGREE

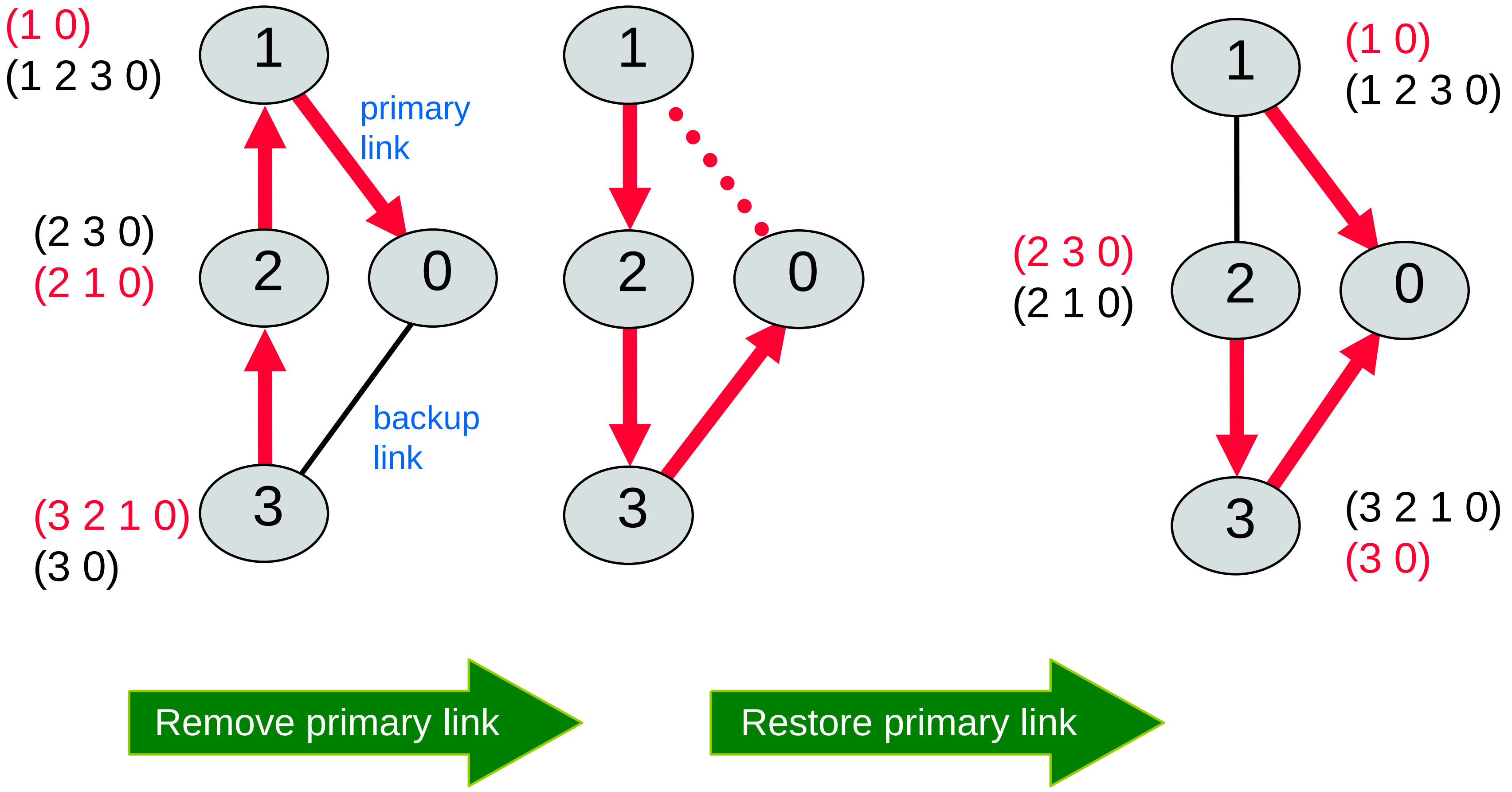


First solution



Second solution

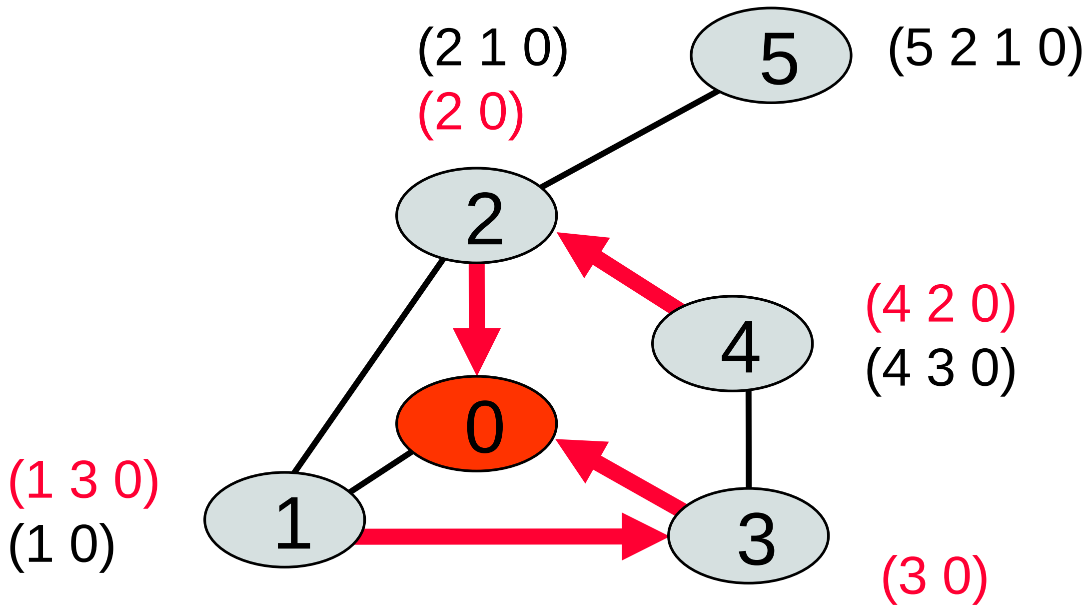
Multiple solutions can result in "Route Triggering"



Convergence

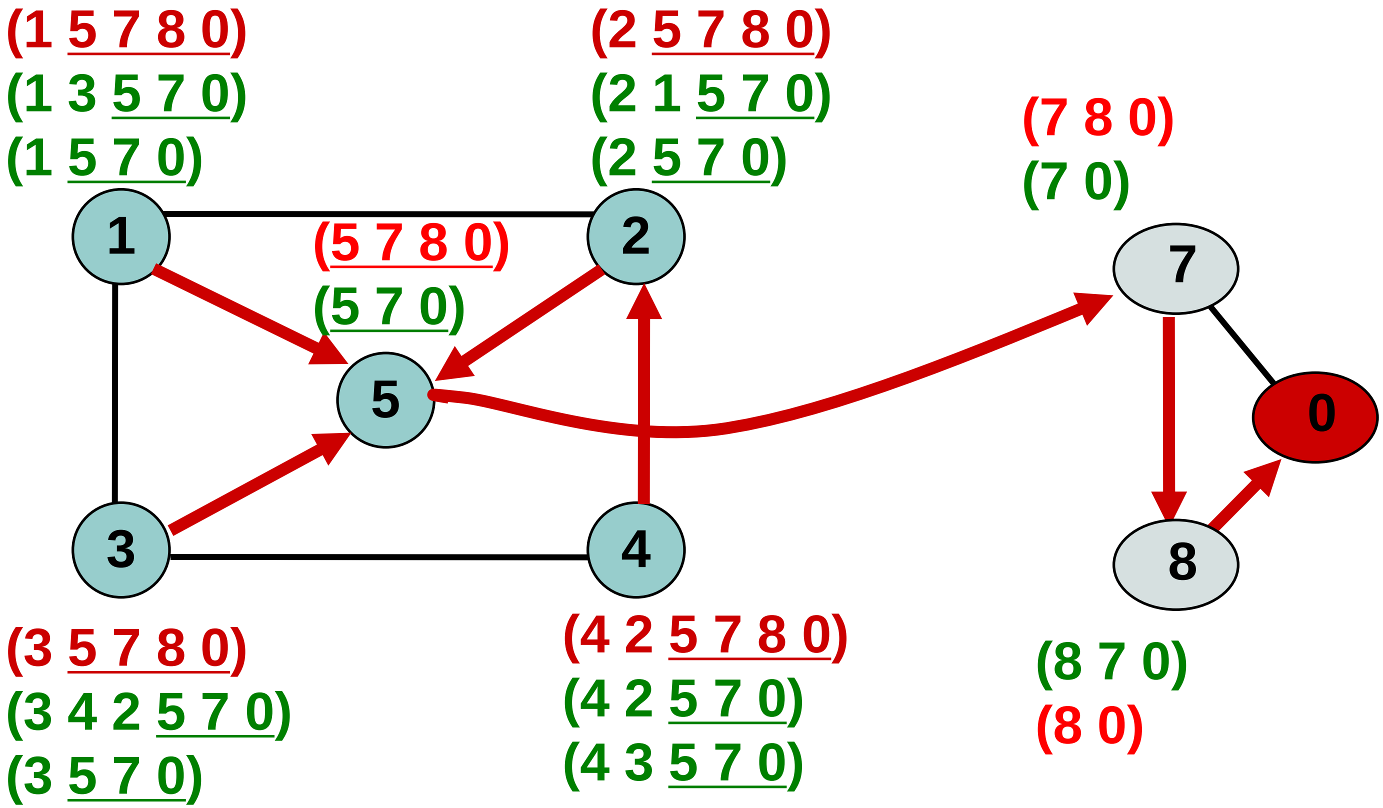
- An SPP instance is said to convergence iff from all possible initial states, and all possible executions from those states, a solution is always reached
- Solvable **does not** imply convergence
  - Some initial states may not lead to a solution.
  - I.e., even if a solution exists you may never reach it
- Convergence of course implies solvable

Single Solution Example



- Has a single solution (in red above), and it ALWAYS reaches this solution (converges) from any initial state!
- Good gadget also converges, and also has a single solution

PRECARIOUS:
Solvable, but may get trapped (impossible to converge)



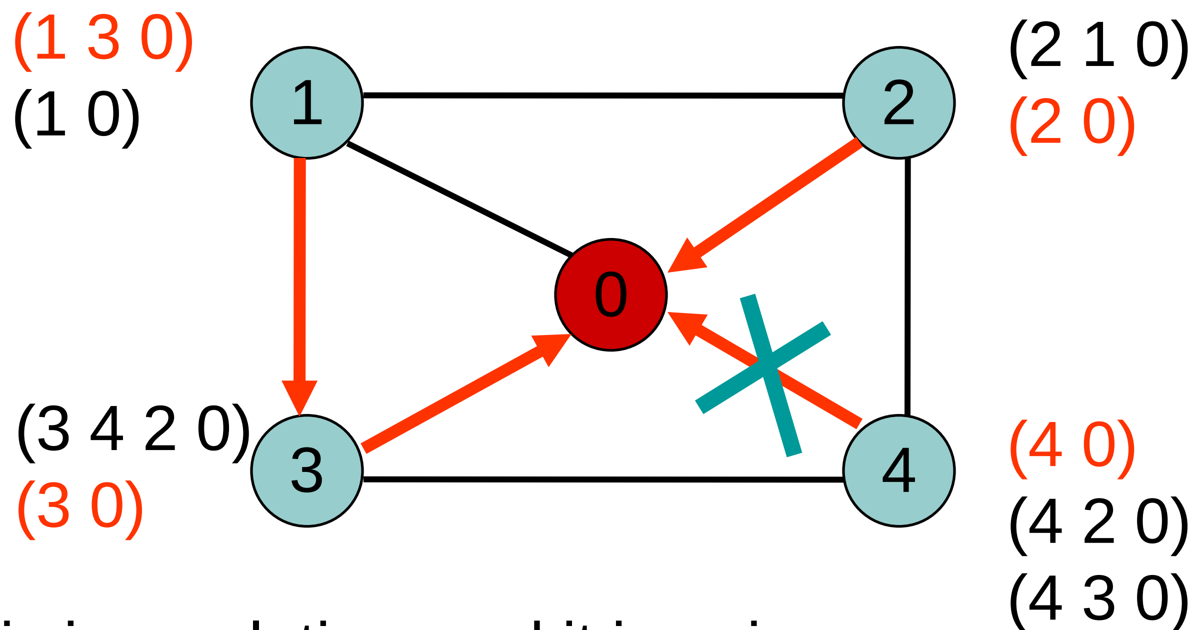
This part has a solution only when node 8 is assigned the direct path (8 0).

As with DISAGREE, this part has two distinct solutions

Safe systems

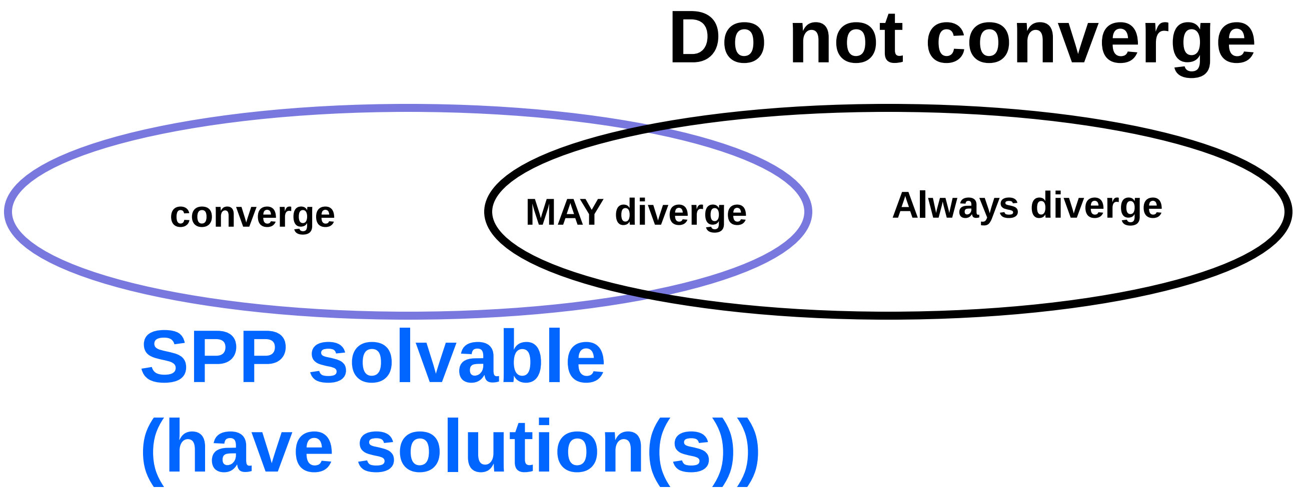
- A sub-instance of an SPP instance is obtained by removing either nodes, links, or paths, from the original sub-instance.
- An SPP instance is **safe** iff, all its subinstances:
  - Have a unique solution
  - They converge
- Good gadget is safe, and so is the single-solution example of slide 31.

SURPRISE : Beware of Backup Policies



- This is a solution, and it is unique
- Also, it converges!
- What if link (4, 0) goes down?
  - Bad gadget! (no solution!)
  - Not robust to link failures, hence not safe!

Another Picture



Problem: Find a stable state

- Problem: find a stable system state
  - i.e., for all u, $\pi(u) = \text{best}(\pi,u)$
- Two approaches: static and dynamic
  - Static: given $S = (G, \mathcal{P},\Lambda)$ find a stable path assignment (i.e. write an algorithm whose input is S and output is π)
  - Dynamic: let each AS (i.e. node) continue to execute

    $$\pi(u) := \text{best}(\pi,u)$$

    until a stable path assignment is found

Network Algorithms for Solving SPP

- Centralized (static):
  - All nodes learn the SPP S = (G, $\mathcal{P}$,Λ)
  - Solve SPP locally
  - Exponential worst case (NP-Hard)

- Distributed (dynamic):
  - Pick the best path from the set of your neighbor's paths, tell your neighbors when you change your mind
  - Can diverge
    - Not guaranteed to find a solution, <u>even when one exists</u>
    - No bound on convergence time

What to do!

Static Approach

Automated Analysis of Routing Policies (check for a sufficient, but not necessary, Condition for a solution).

Inter-AS coordination

Dynamic Approach

Extend BGP with a dynamic means of detecting and suppressing policy-based oscillations?

These approaches are <u>complementary</u>