Almost Polynomial Hardness of NDP-Grids

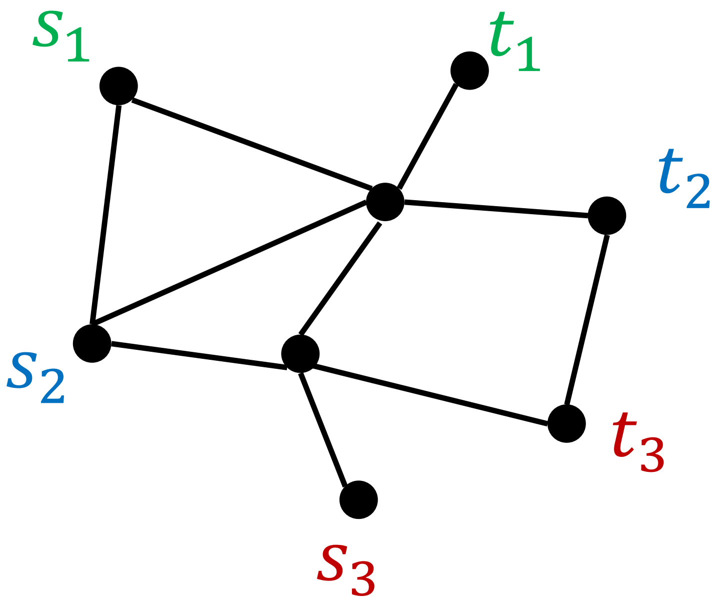
JULIA CHUZHOY DAVID KIM RACHIT NIMAVAT

TOYOTA TECHNOLOGICAL INSTITUTE AT CHICAGO

Banff, Nov 2017

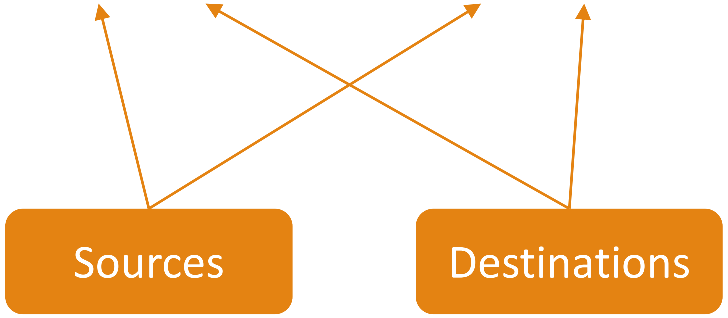
Node-Disjoint Paths (NDP)

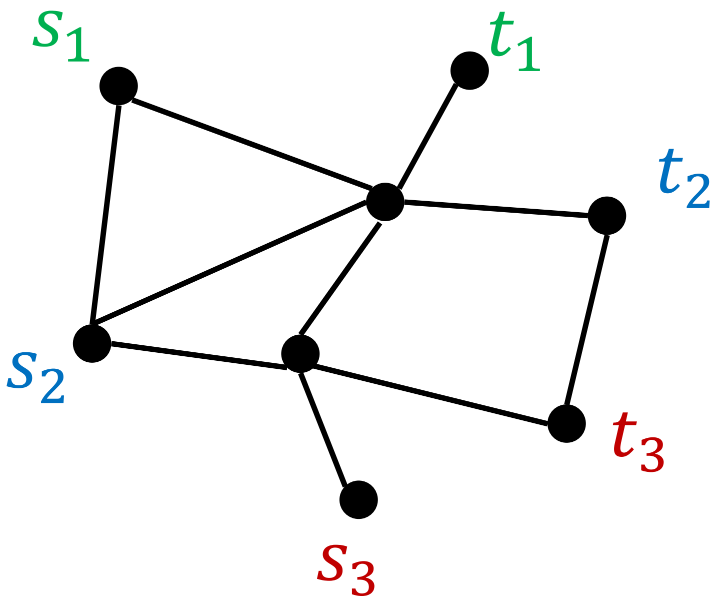
Input: Graph and demand pairs $(s_1, t_1), \dots, (s_k, t_k)$



Node-Disjoint Paths (NDP)

Input: Graph and demand pairs $(s_1, t_1), \ldots, (s_k, t_k)$



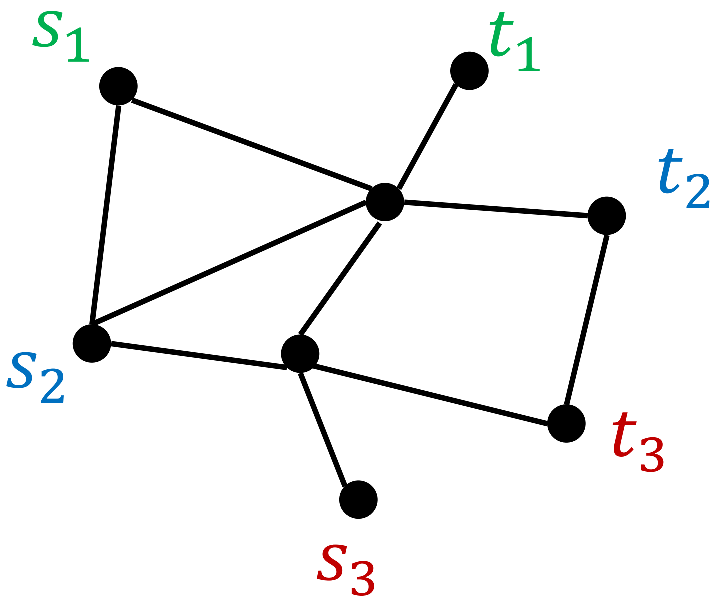


Node-Disjoint Paths (NDP)

Input: Graph and demand pairs $(s_1, t_1), \dots, (s_k, t_k)$

Sources

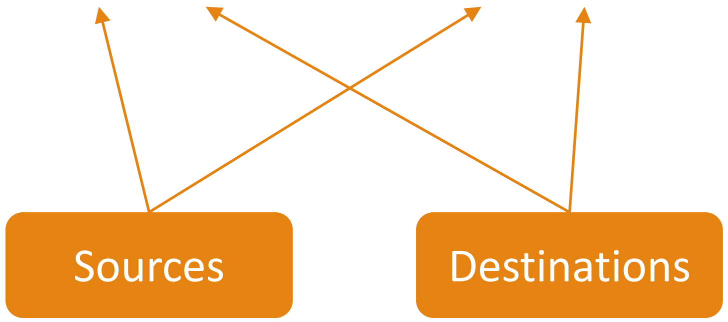
Destinations

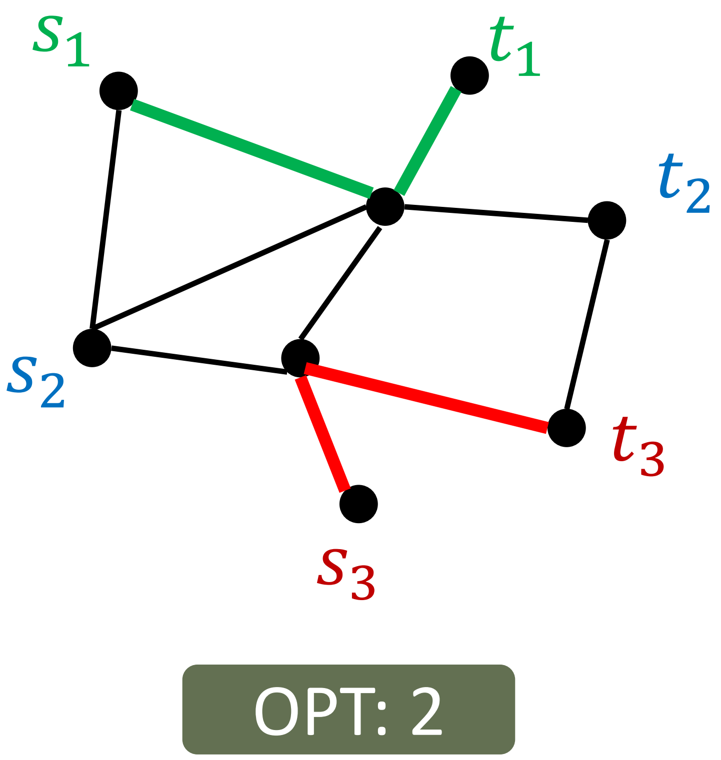


Goal: Route as many pairs as possible via node-disjoint paths

Node-Disjoint Paths (NDP)

Input: Graph and demand pairs $(s_1, t_1), \dots, (s_k, t_k)$





Goal: Route as many pairs as possible via node-disjoint paths

Known Results

Constant $k \Rightarrow$ Efficient algorithm [Robertson, Seymour '90]

k part of input $\Rightarrow$ NP-Hard [Karp '72]

Known Results

Constant $k \Rightarrow$ Efficient algorithm [Robertson, Seymour ’90]

k part of input $\Rightarrow$ NP-Hard [Karp ’72]

$O(\sqrt{n})$ −approx. [Kolliopoulos, Stein ‘98]

Roughly $\Omega\left(\sqrt{\log n}\right)$ −hardness of approx. [Andrews, Zhang ‘05], [Andrews, Chuzhoy, Guruswami, Khanna, Talwar, Zhang ’10]

Known Results

Constant $k \Rightarrow$ Efficient algorithm [Robertson, Seymour '90]

k part of input $\Rightarrow$ NP-Hard [Karp '72]

$O(\sqrt{n})$ −approx. [Kolliopoulos, Stein '98]

Roughly $\Omega\left(\sqrt{\log n}\right)$ −hardness of approx. [Andrews, Zhang '05], [Andrews, Chuzhoy, Guruswami, Khanna, Talwar, Zhang '10]

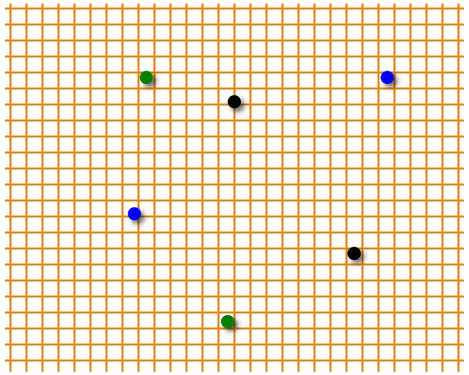
What about *simpler* cases?

Known Results

	Upper Bound	Lower Bound
General NDP	$O(\sqrt{n})$	$\Omega\left(\sqrt{\log n}\right)$

NDP-Grid:

- $O(n^{1/4})$ −approx. for NDP-Grid [Chuzhoy, Kim '15]
- APX-hardness [Chuzhoy, Kim '15]



Known Results

	Upper Bound	Lower Bound
General NDP	$O(\sqrt{n})$	$\Omega\left(\sqrt{\log n}\right)$
NDP-Grid	$O(n^{1/4})$	APX-hardness

NDP-Planar:

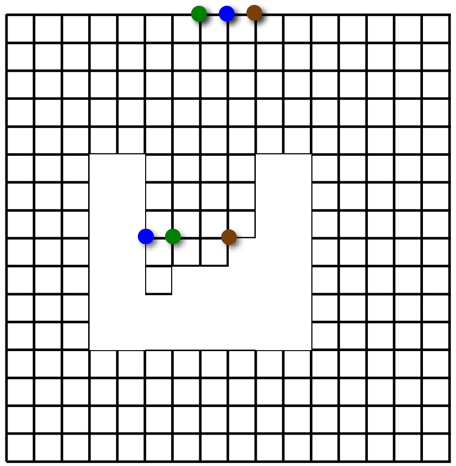
- $O(n^{9/19})$ —approx. for NDP-Planar [Chuzhoy, Kim, Li '16]
- $2^{\Omega(\sqrt{\log n})}$ —hardness [Chuzhoy, Kim, N '16]

Known Results

	Upper Bound	Lower Bound
General NDP	$O(\sqrt{n})$	$2^{\Omega(\sqrt{\log n})}$
NDP-Grid	$O(n^{1/4})$	APX-hardness
NDP-Planar	$O(n^{9/19})$	$2^{\Omega(\sqrt{\log n})}$

"grids with holes" are hard

(even with sources on top)

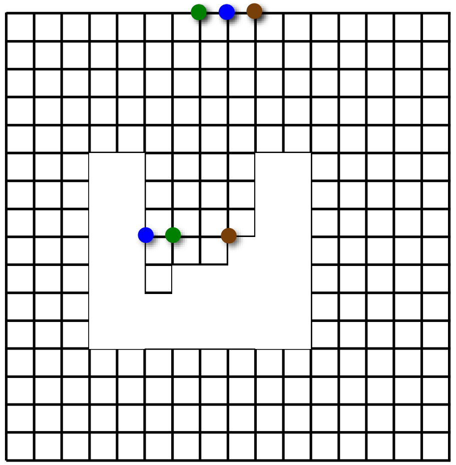


Matching Upper Bound?

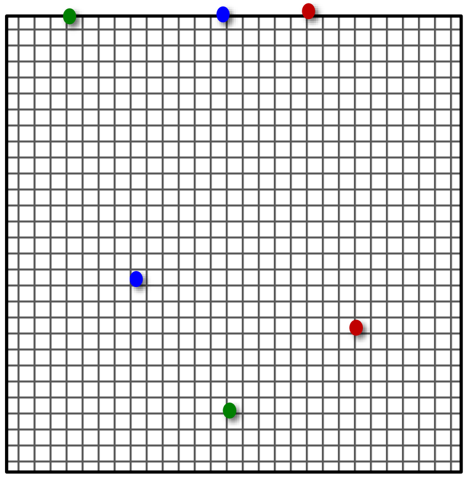
NDP-Grid with Sources on Boundary

$2^{O(\sqrt{\log n})}$ −approx. if sources on boundary [Chuzhoy, Kim, N '17]

- First sub-polynomial approx. algorithm!



$2^{\Omega(\sqrt{\log n})}$ −hard



$2^{O(\sqrt{\log n})}$ −approx.

Generalize both to NDP-Grid?

Main Theorem

NDP-Grid is

$2^{\Omega(\log^{1-\epsilon} n)}$−hard assuming NP $\not\subseteq$ RTIME($n^{poly \log n}$)

For all $\epsilon > 0$

Main Theorem

Weaker than rETH

NDP-Grid is

$2^{\Omega(\log^{1-\epsilon} n)}$–hard assuming NP $\not\subseteq$ RTIME($n^{poly\log n}$)

$n^{\Omega(1/(\log\log n)^2)}$ –hard assuming NP $\not\subseteq$ RTIME(2^{n^δ})

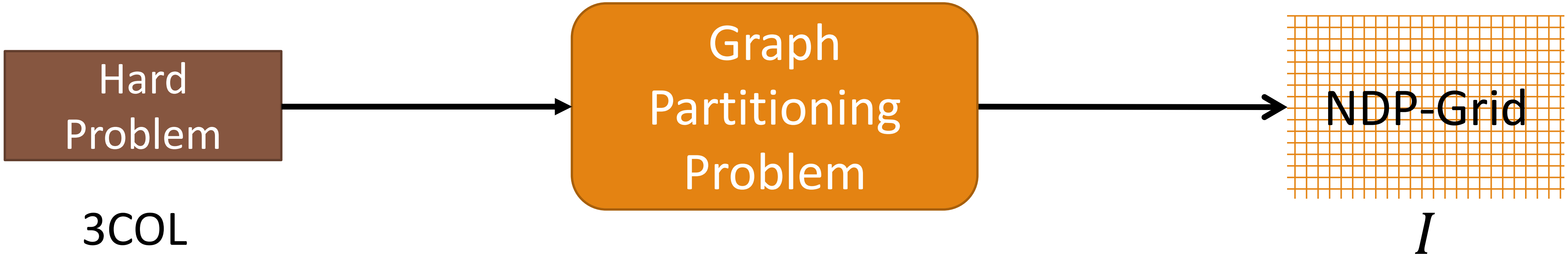
For all $\epsilon > 0$

For some $\delta > 0$

The Updated Picture

	Upper Bound	Lower Bound
General NDP	$O(\sqrt{n})$	$n^{\Omega(1/(\log\log n)^2)}$
NDP-Planar	$O(n^{9/19})$	$n^{\Omega(1/(\log\log n)^2)}$
NDP-Grid	$O(n^{1/4})$	$n^{\Omega(1/(\log\log n)^2)}$

A Proxy Problem

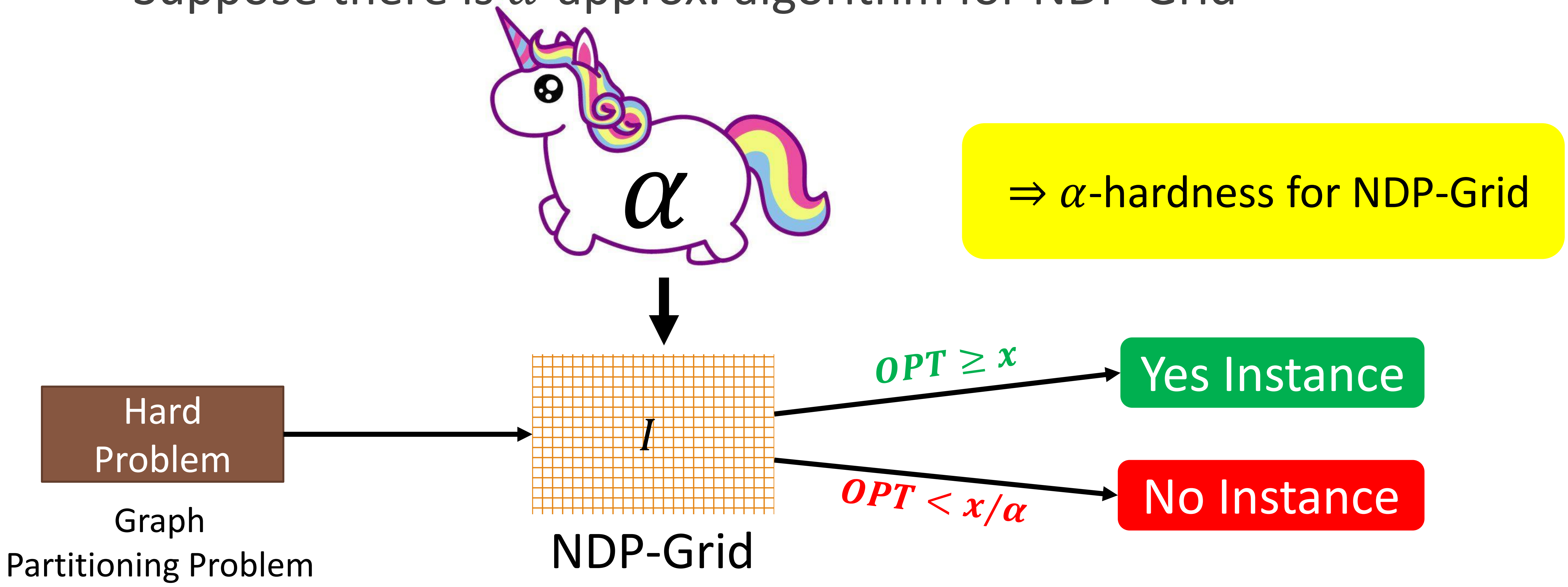


Part 1: Graph Partitioning ⇒ NDP-Grid

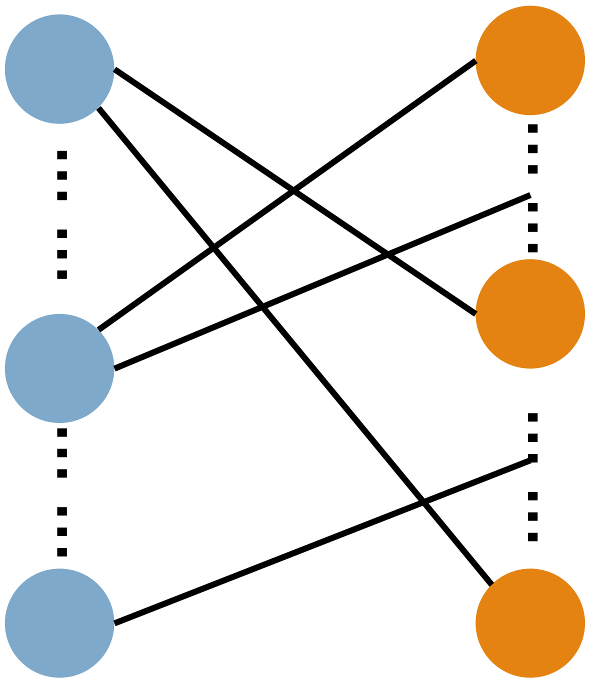
Part 2: 3COL ⇒ Graph Partitioning

How to Show Hardness? : The Karp Way

Suppose there is α-approx. algorithm for NDP-Grid

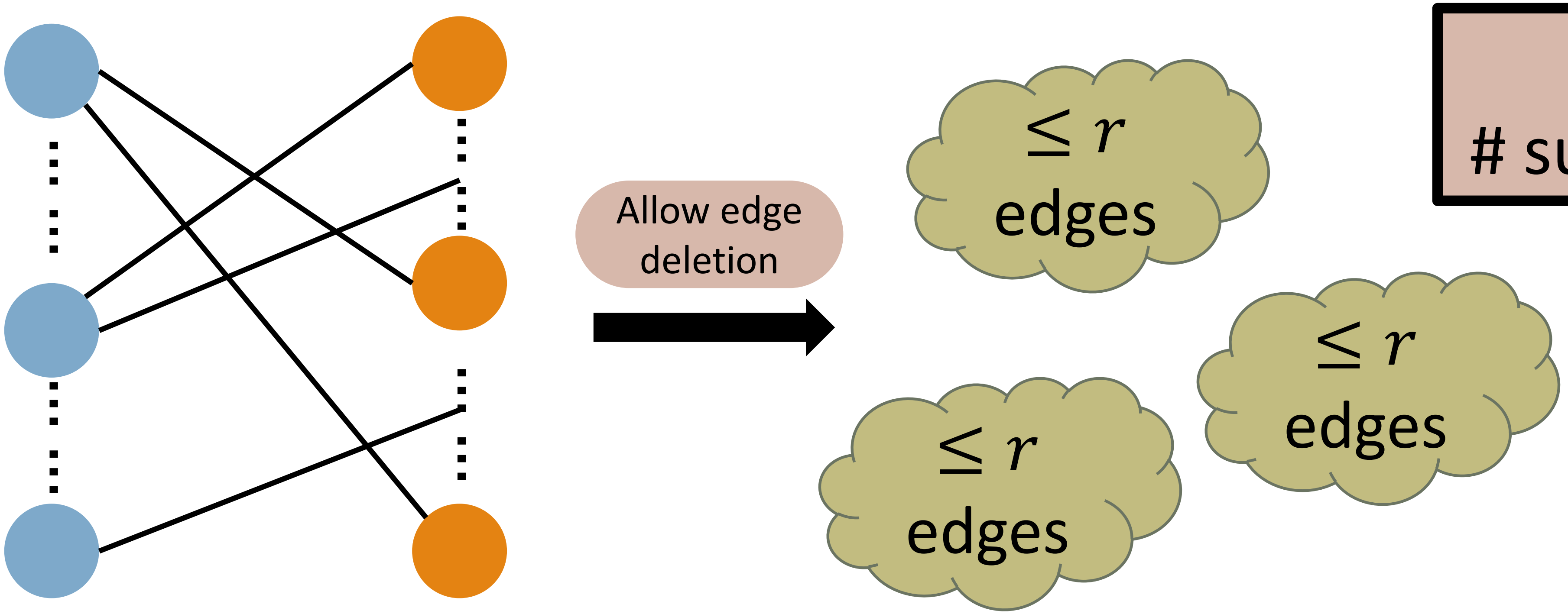


Graph Partitioning Problem



Bipartite graph B
Parameters: p, r

Graph Partitioning Problem

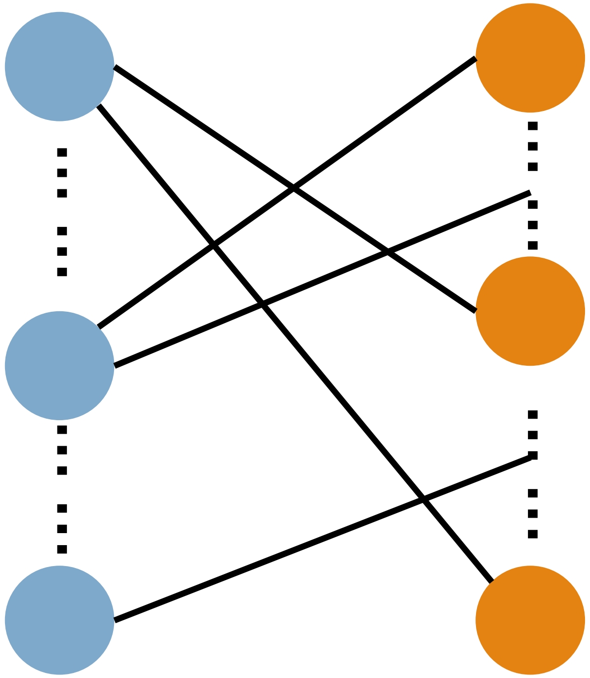


Bipartite graph B
Parameters: p, r

p Pieces

Maximize:
surviving edges

Graph Partitioning Problem



Bipartite graph B
Parameters: p, r

Allow edge deletion

$\leq r$ edges

$\leq r$ edges

$\leq r$ edges

p Pieces

Maximize:
surviving edges

Looks like
Densest k-Subgraph
Problem…

Graph Partitioning ⇒ NDP on Grids

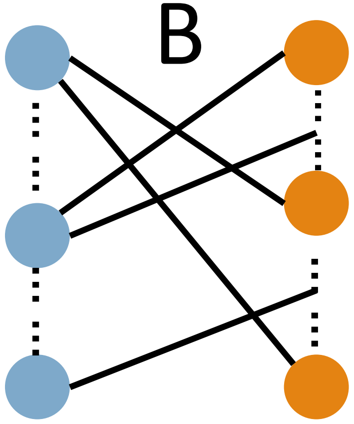
Theorem: Can construct NDP-Grid instance I s.t

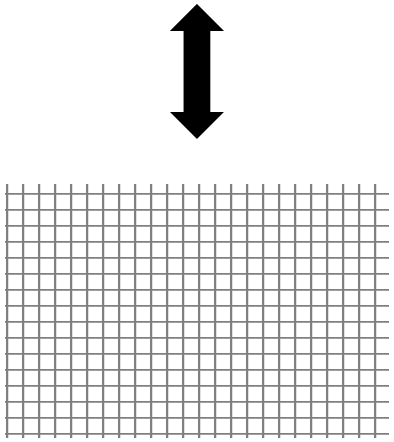
Partitioning with many surviving edges <=>
Routing a large number of demand pairs

Step 1: Construction

Step 2: Partitioning ⇒ Routing [Skipped]

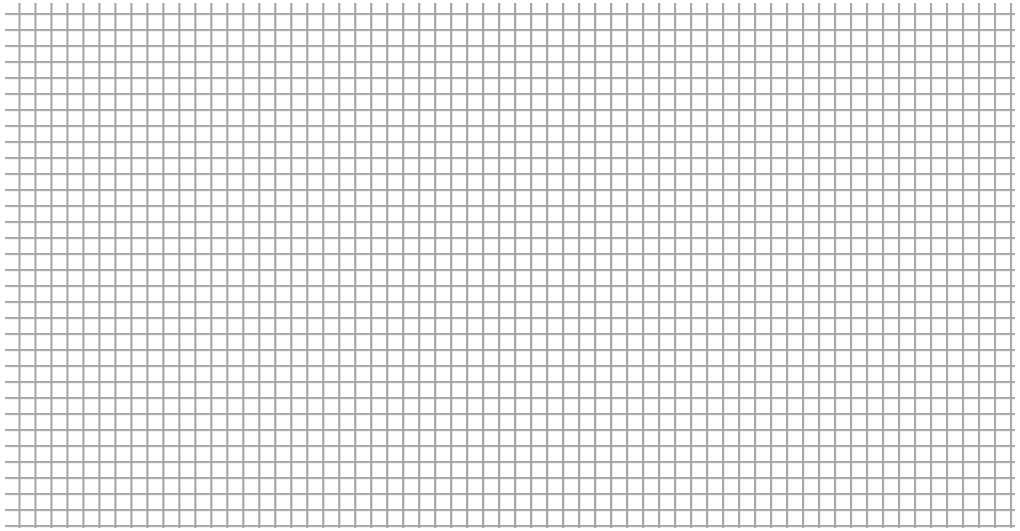
Step 3: Partitioning ⇐ Routing

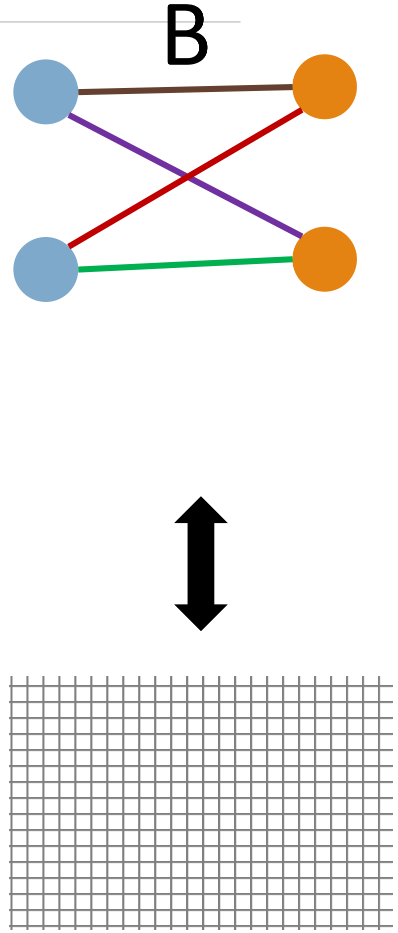




NDP-Grid I

Constructing NDP-Grid

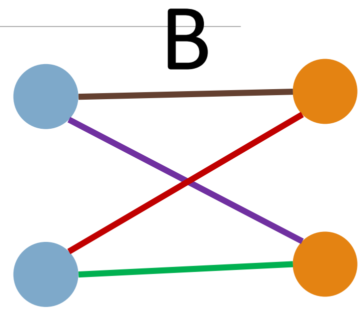


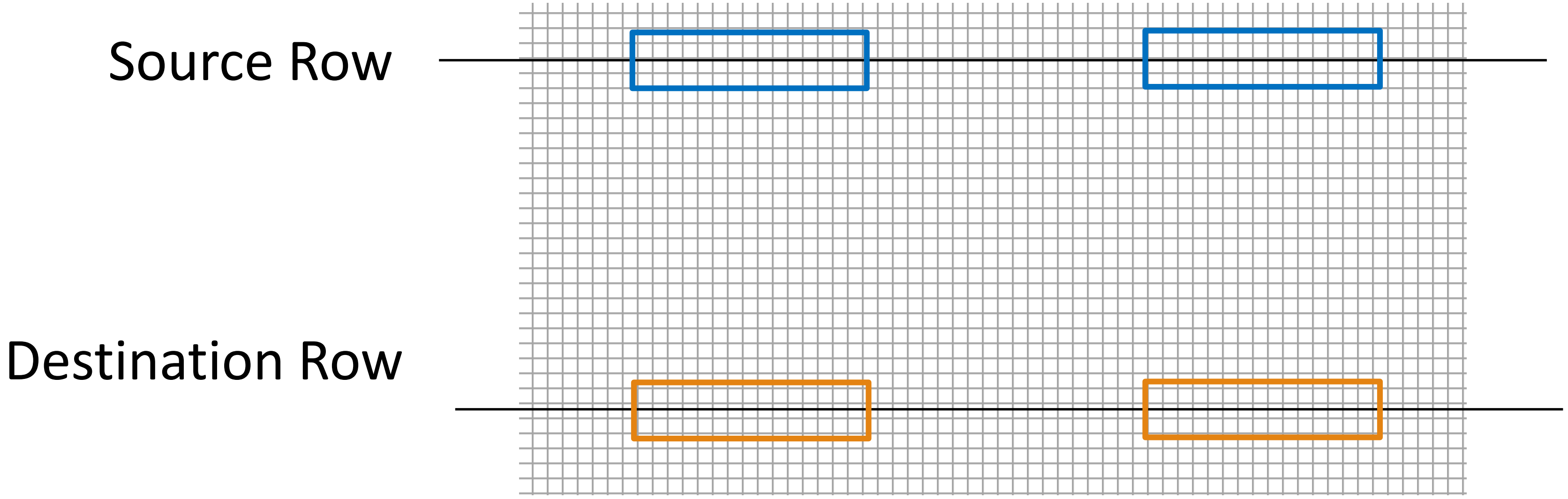


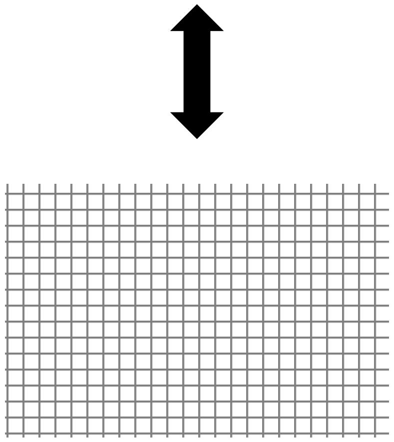
NDP-Grid I

Constructing NDP-Grid

- Box for each vertex of B in source/destination row

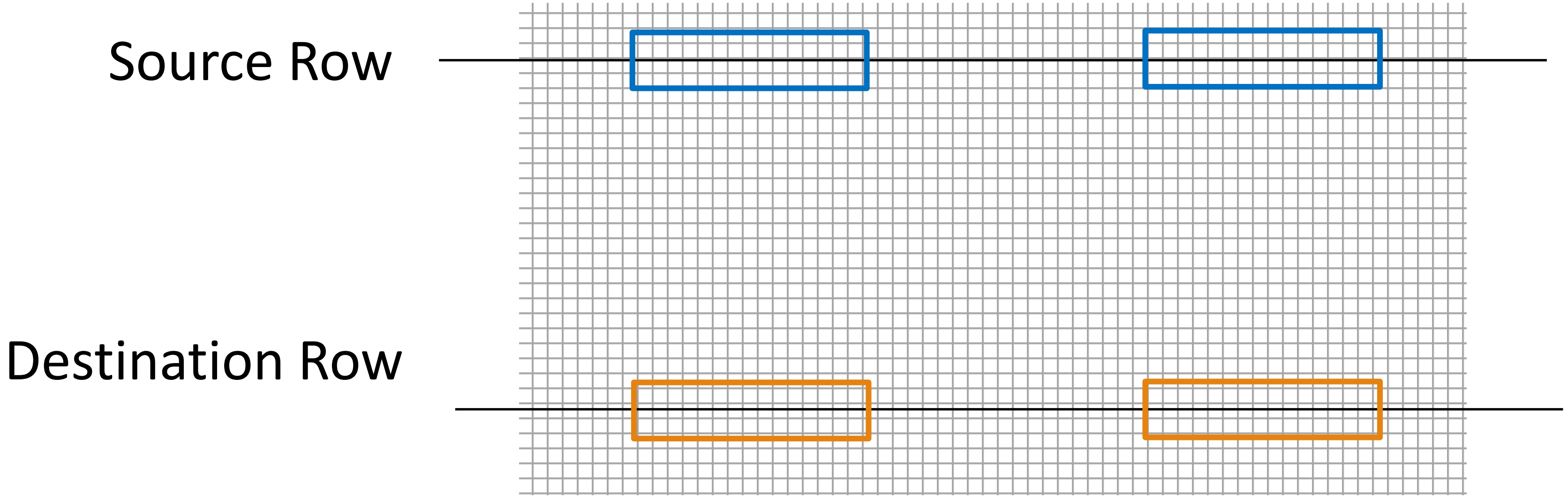


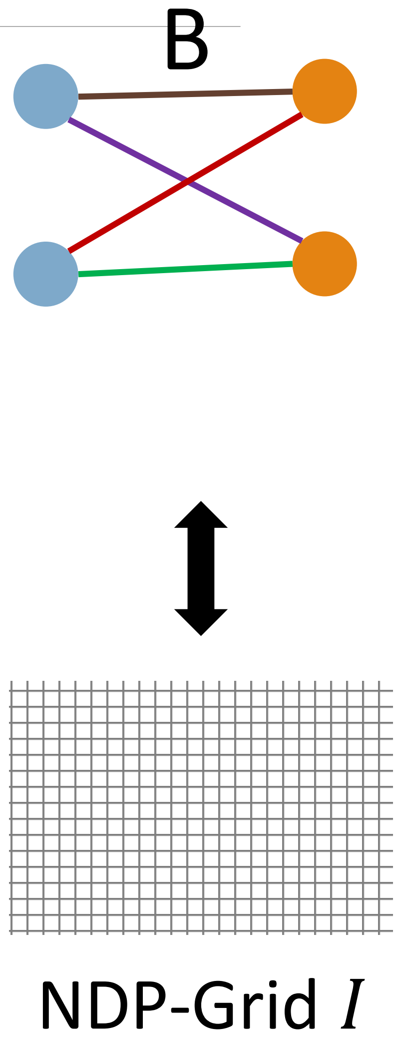




Constructing NDP-Grid

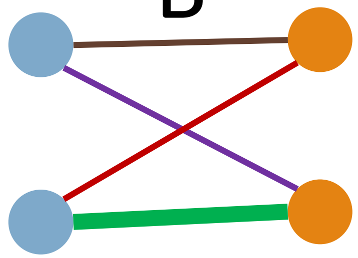
- Box for each vertex of B in source/destination row
- Demand pair (s_e, t_e) for each edge e of B
- Place s_e and t_e inside boxes of endpoints of e

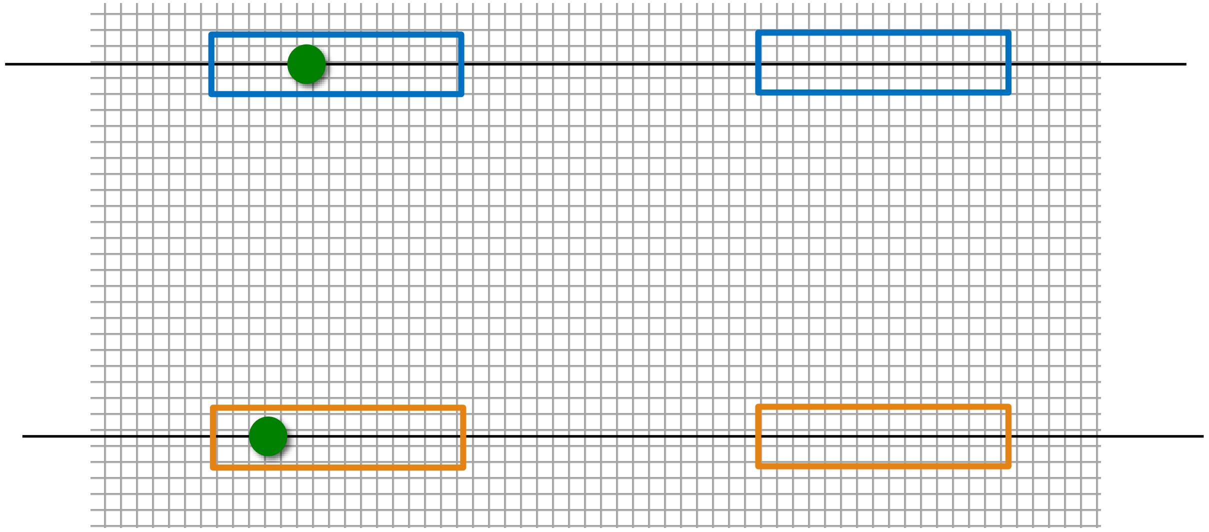


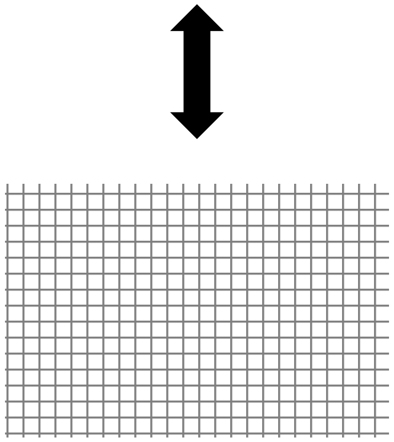


Constructing NDP-Grid

- Box for each vertex of B in source/destination row
- Demand pair (s_e, t_e) for each edge e of B
- Place s_e and t_e inside boxes of endpoints of e

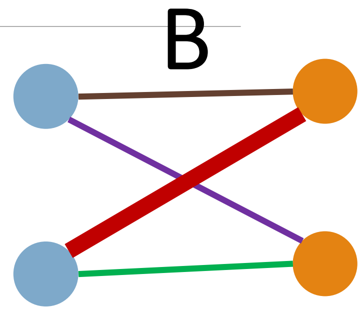


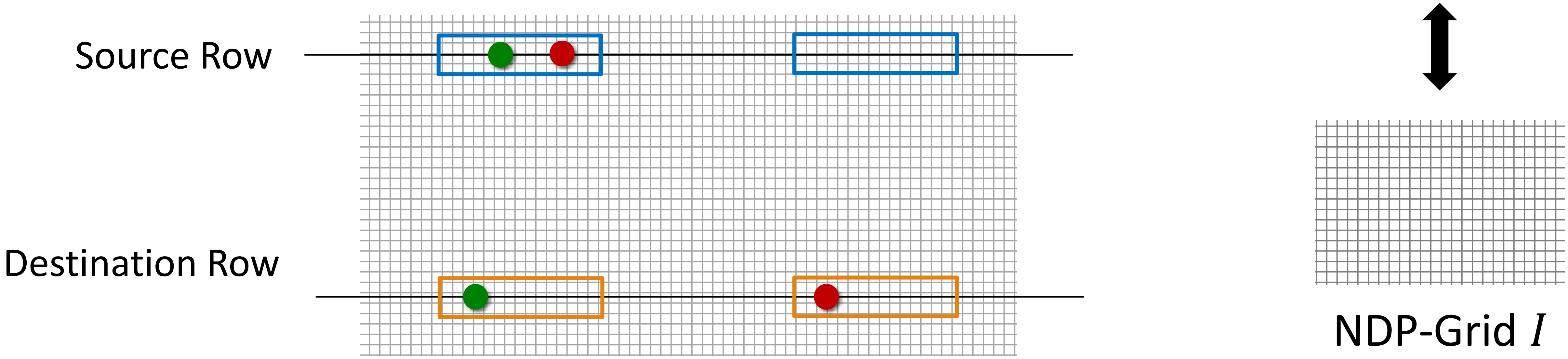




Constructing NDP-Grid

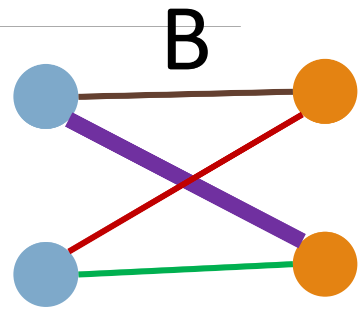
- Box for each vertex of B in source/destination row
- Demand pair (s_e, t_e) for each edge e of B
- Place s_e and t_e inside boxes of endpoints of e

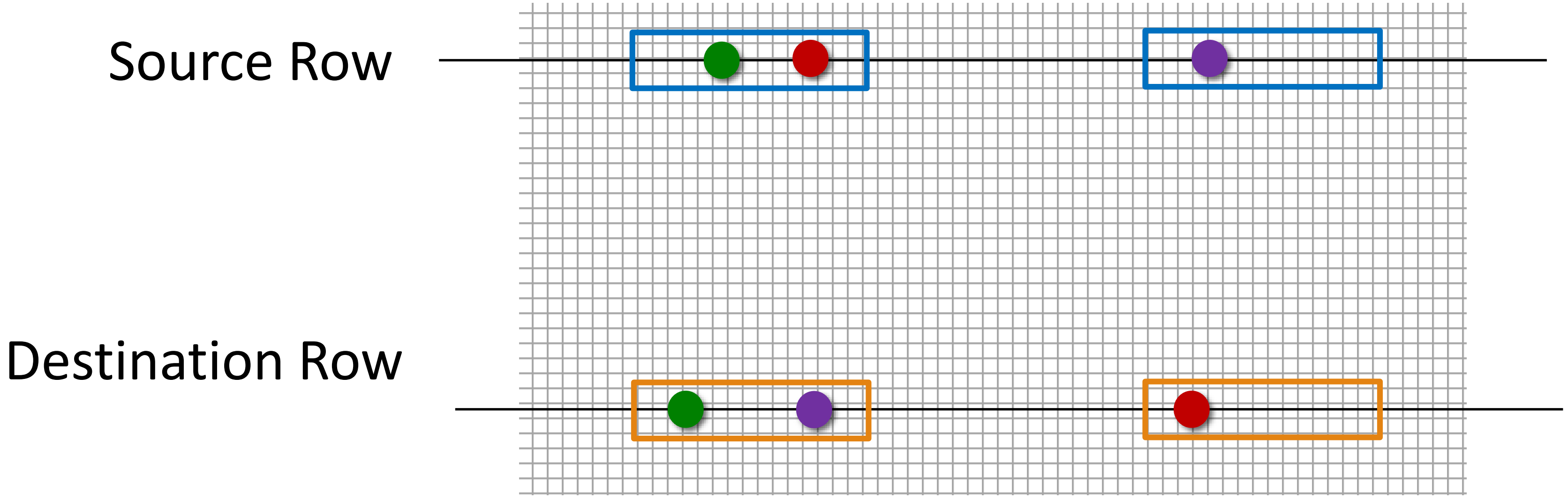


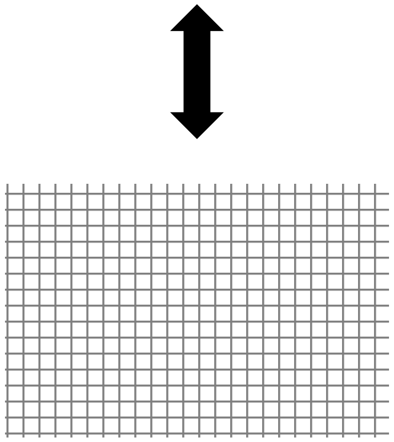


Constructing NDP-Grid

- Box for each vertex of B in source/destination row
- Demand pair (s_e, t_e) for each edge e of B
- Place s_e and t_e inside boxes of endpoints of e



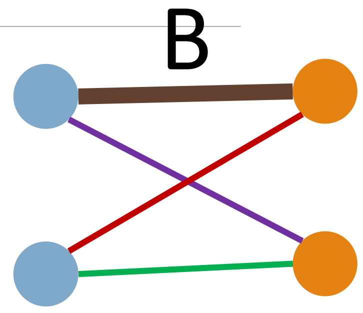


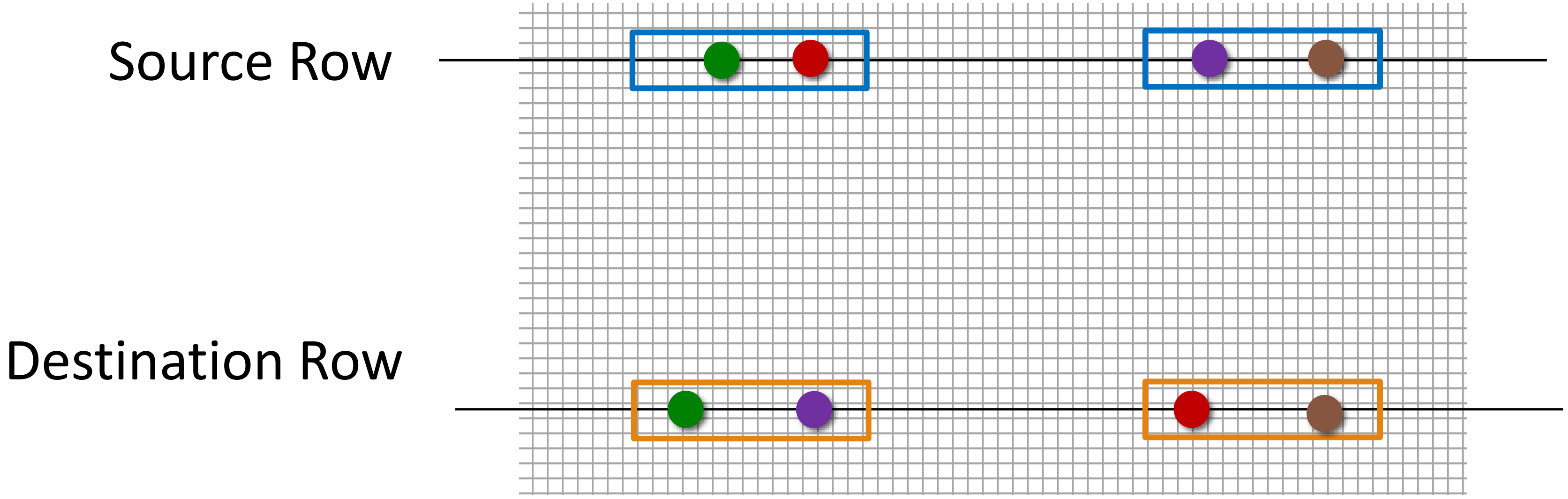


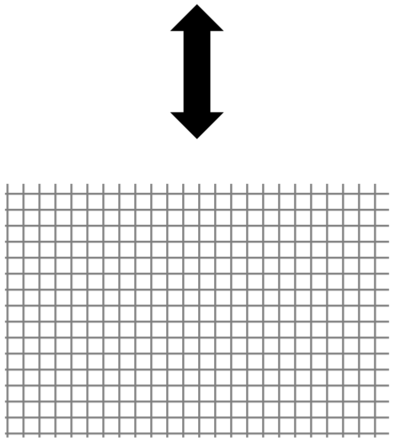
NDP-Grid I

Constructing NDP-Grid

- Box for each vertex of B in source/destination row
- Demand pair (s_e, t_e) for each edge e of B
- Place s_e and t_e inside boxes of endpoints of e



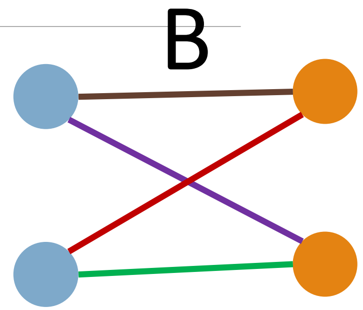


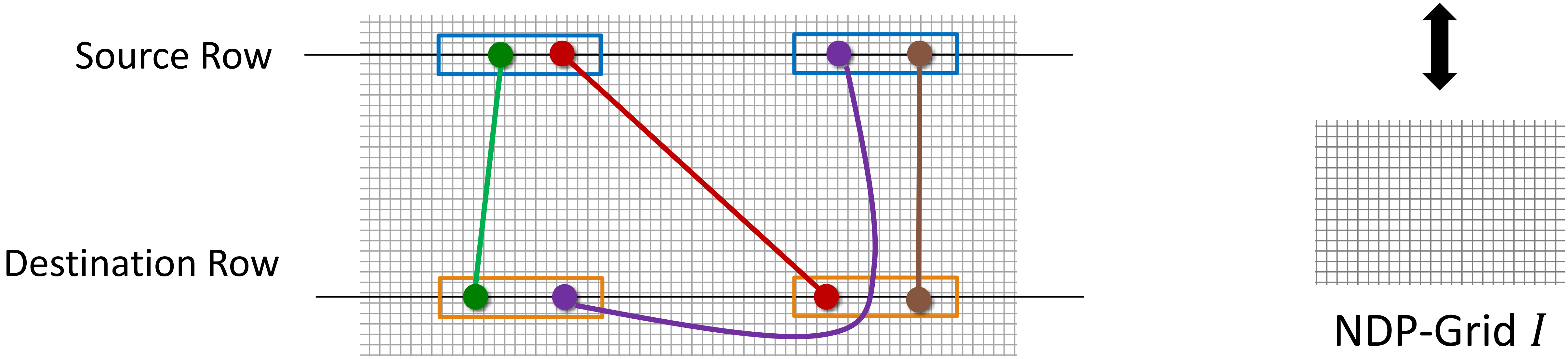


NDP-Grid I

Graph Partitioning $\Leftarrow$ NDP on Grids

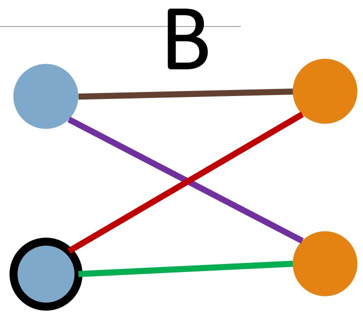
- Consider routing of a large subset of demand pairs

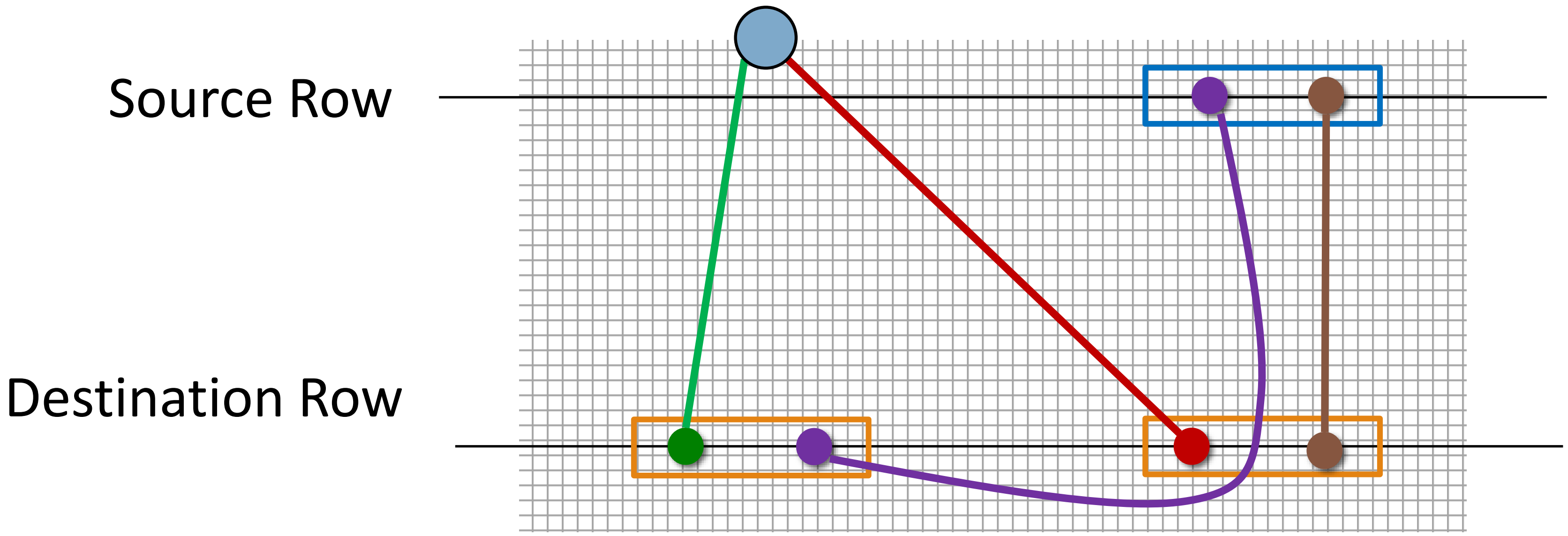


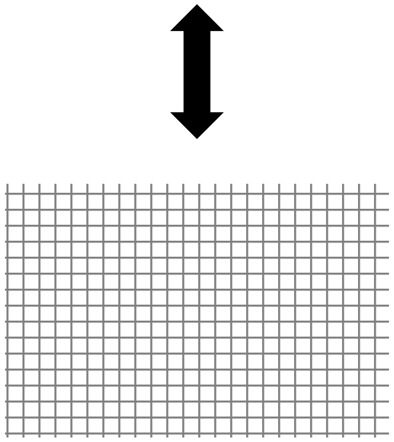


Graph Partitioning ⇐ NDP on Grids

- Consider routing of a large subset of demand pairs
- Contract the blocks



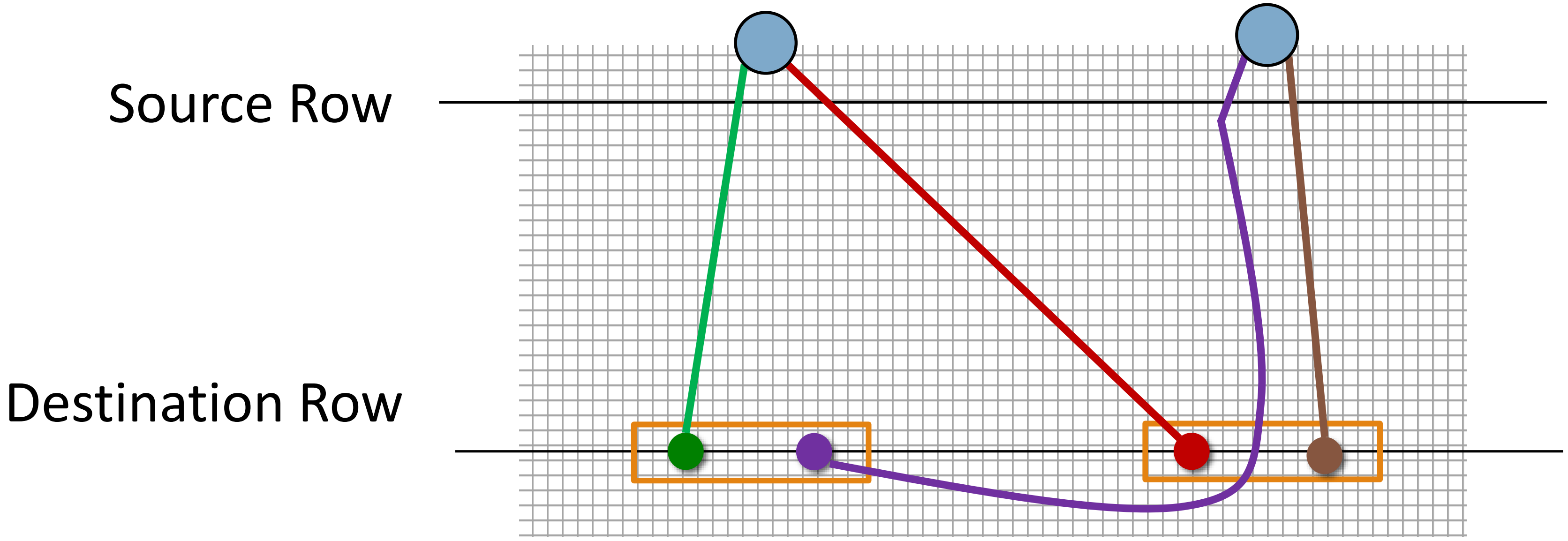


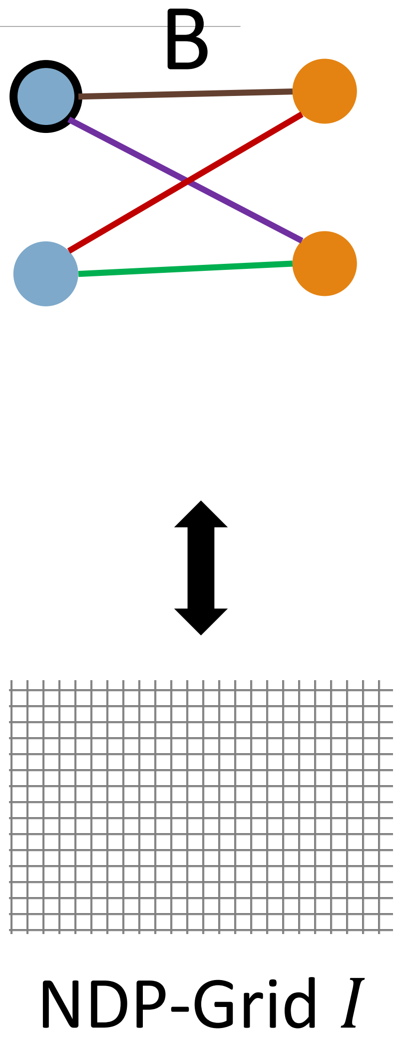


NDP-Grid I

Graph Partitioning ⇐ NDP on Grids

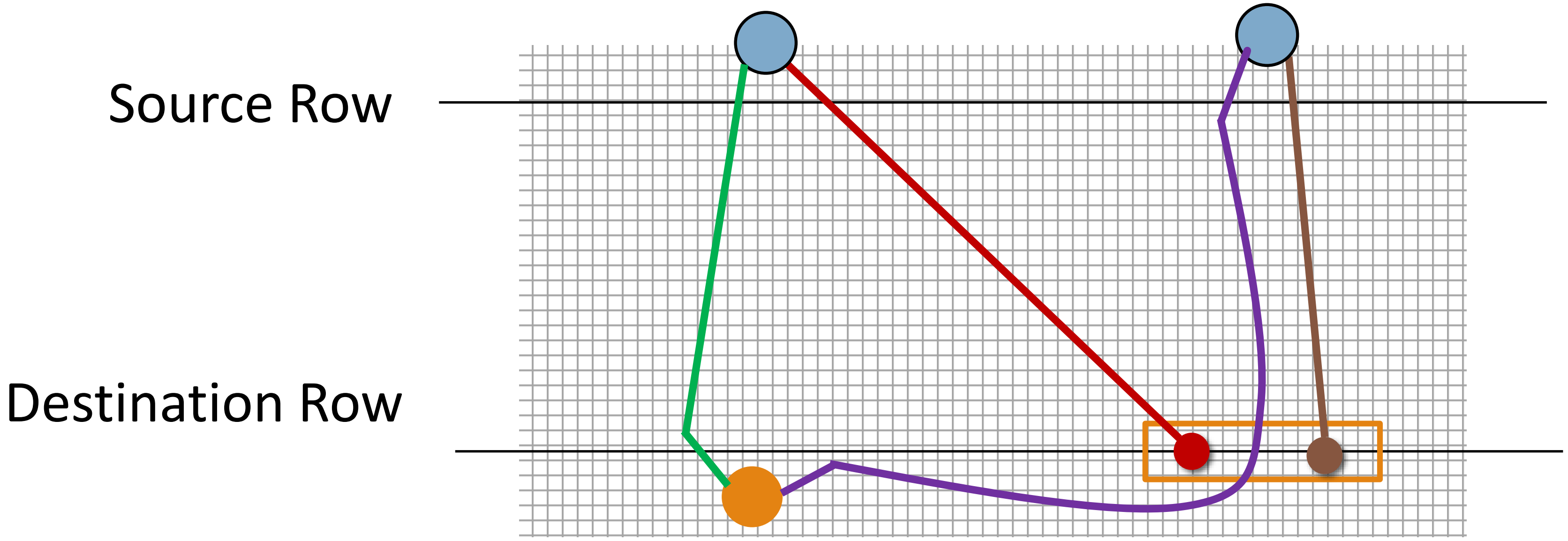
- Consider routing of a large subset of demand pairs
- Contract the blocks

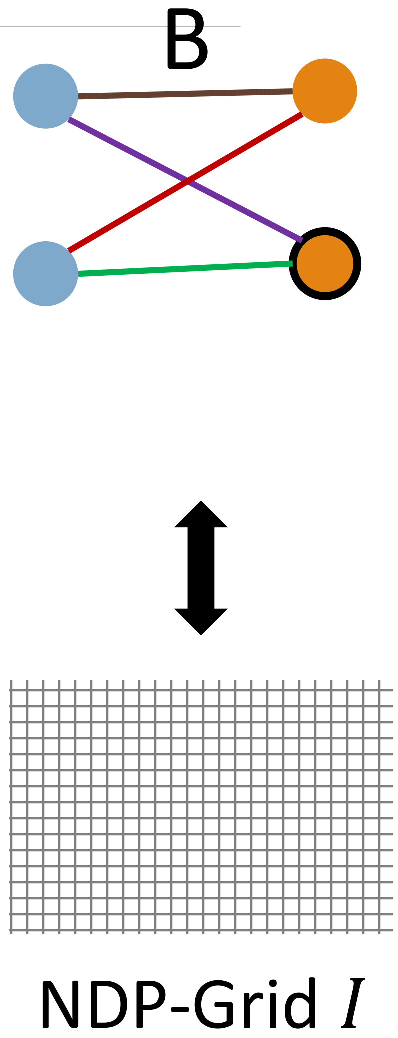




Graph Partitioning ⇐ NDP on Grids

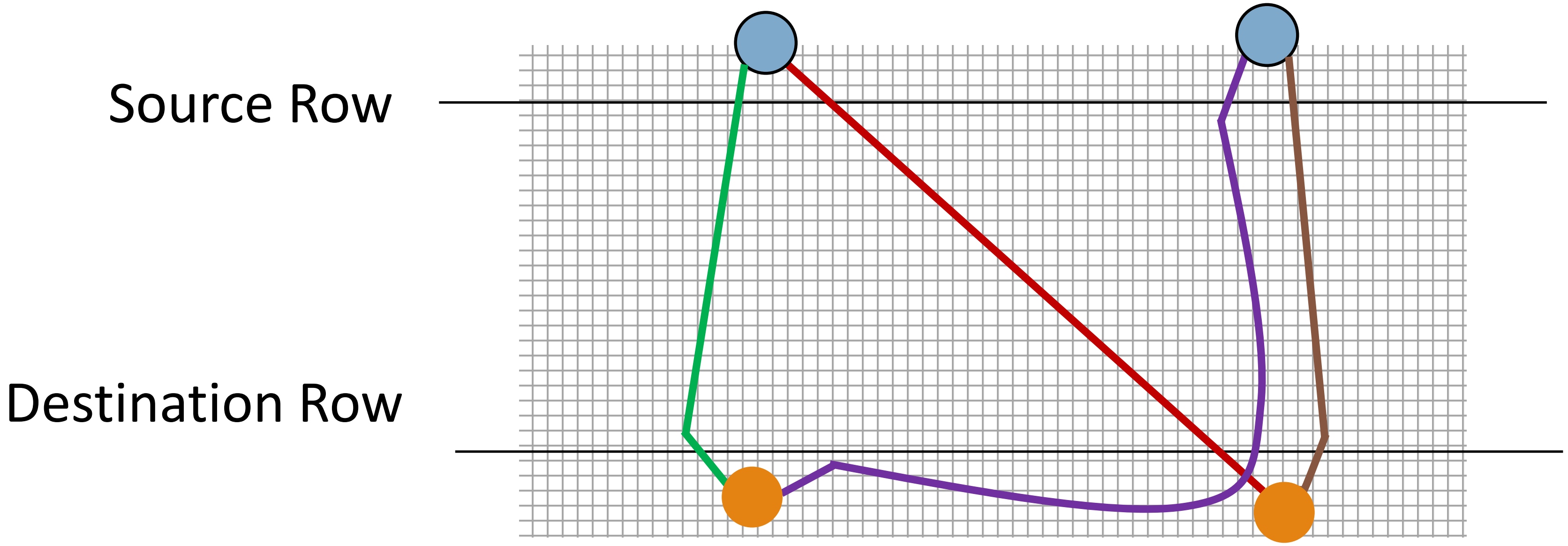
- Consider routing of a large subset of demand pairs
- Contract the blocks

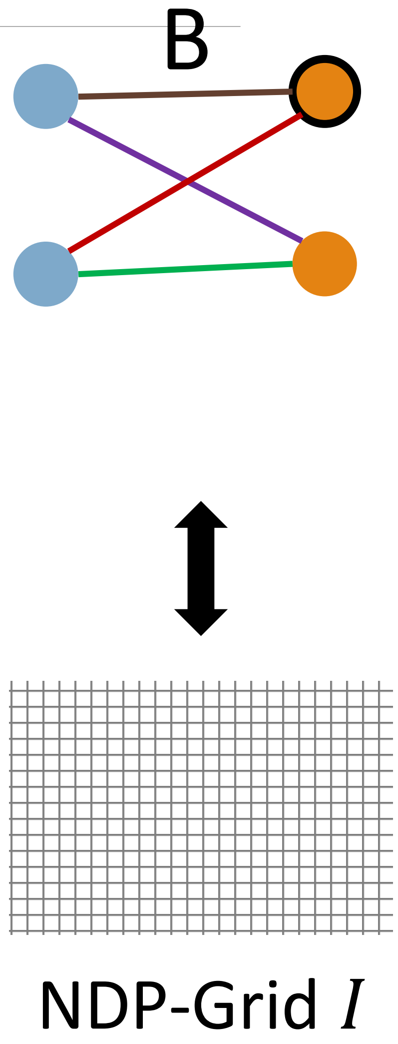




Graph Partitioning ⇐ NDP on Grids

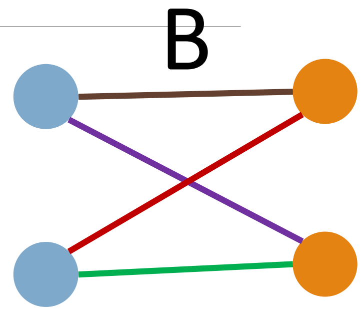
- Consider routing of a large subset of demand pairs
- Contract the blocks



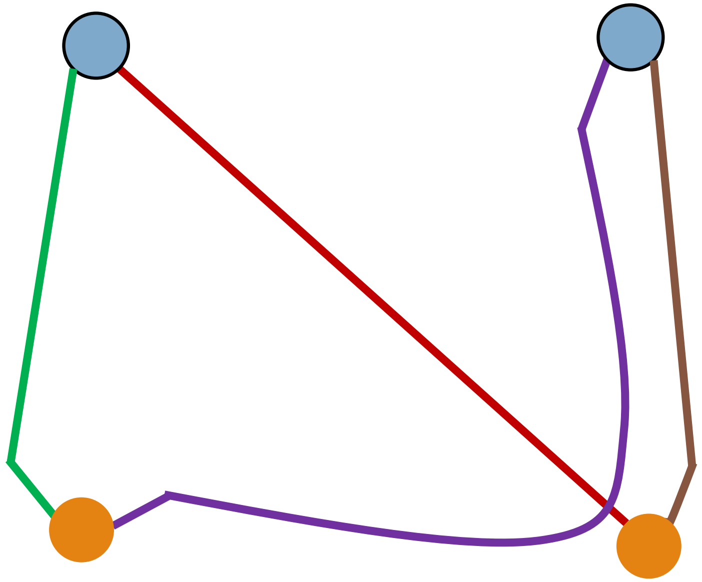


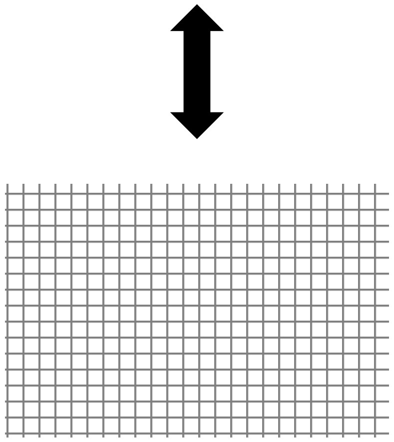
Graph Partitioning ⇐ NDP on Grids

- Consider routing of a large subset of demand pairs
- Contract the blocks



Drawing with low crossing number

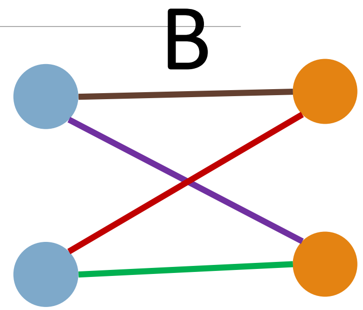




NDP-Grid I

Graph Partitioning $\Leftarrow$ NDP on Grids

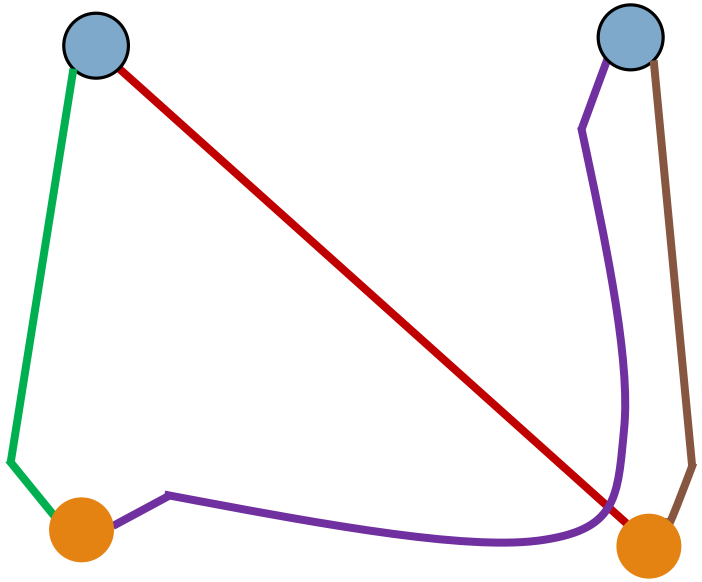
- Consider routing of a large subset of demand pairs
- Contract the blocks

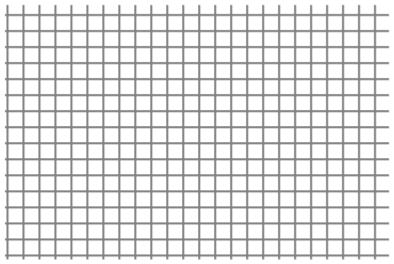


Drawing with low crossing number

Almost planar

Good balanced cut





NDP-Grid I

Graph Partitioning $\Leftarrow$ NDP on Grids

- Consider routing of a large subset of demand pairs
- Contract the blocks

Almost planar

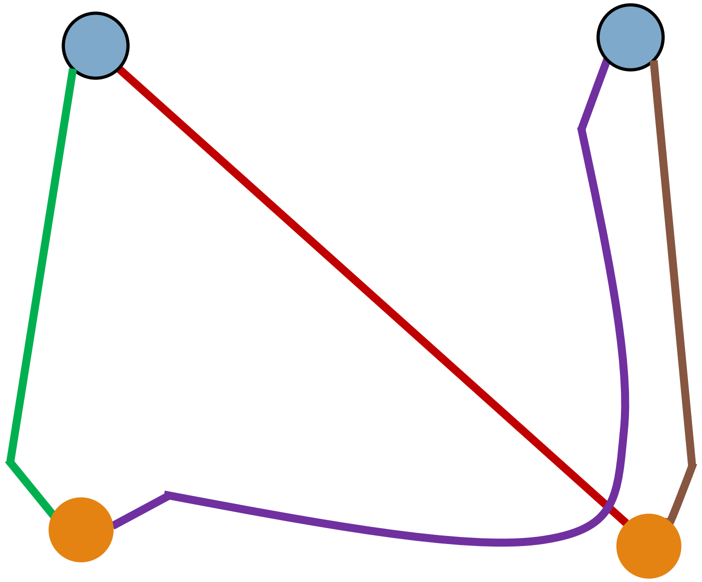
↓

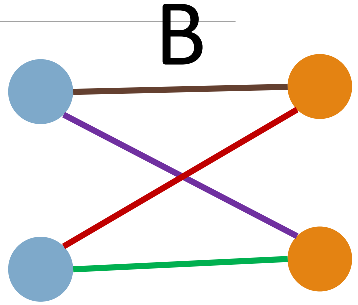
Good balanced cut

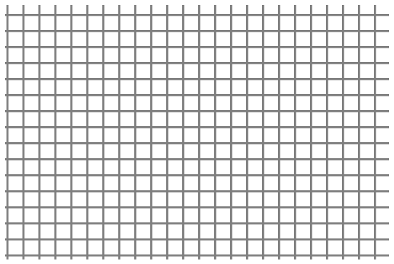
↓

Good partitioning

Drawing with low crossing number

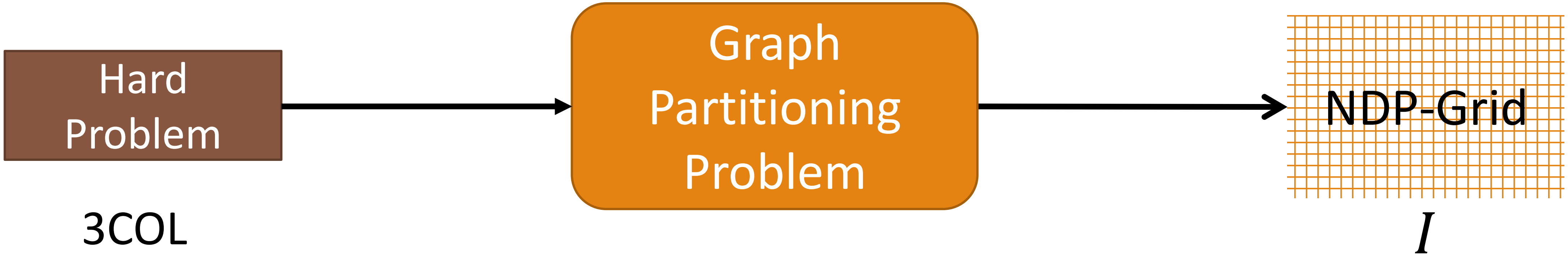






NDP-Grid I

A Proxy Problem



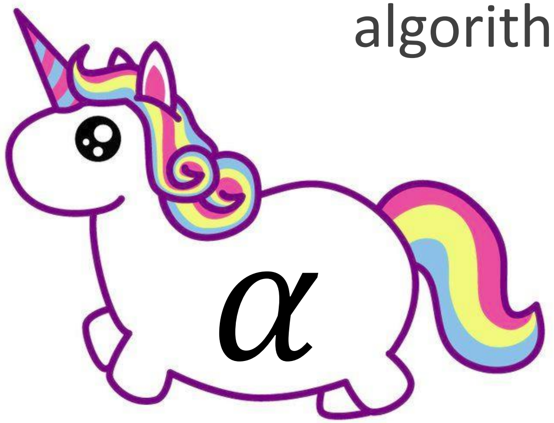
Part 1: Graph Partitioning ⇒ NDP-Grid

Part 2: 3COL ⇒ Graph Partitioning

How to Show Hardness? : The Cook Way

Suppose there is α-approx. algorithm for NDP-Grid

I_1



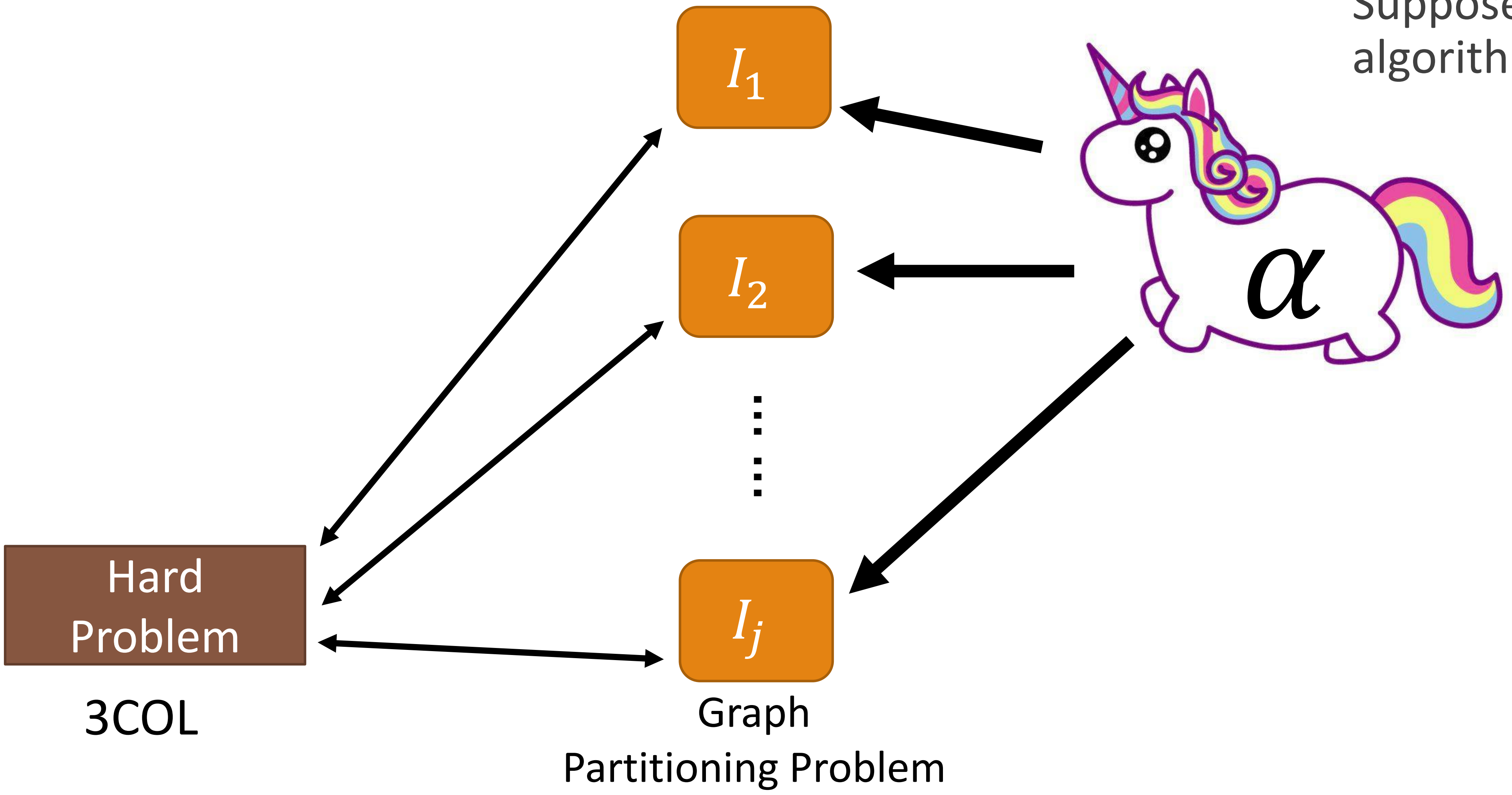
Hard Problem

3COL

Graph Partitioning Problem

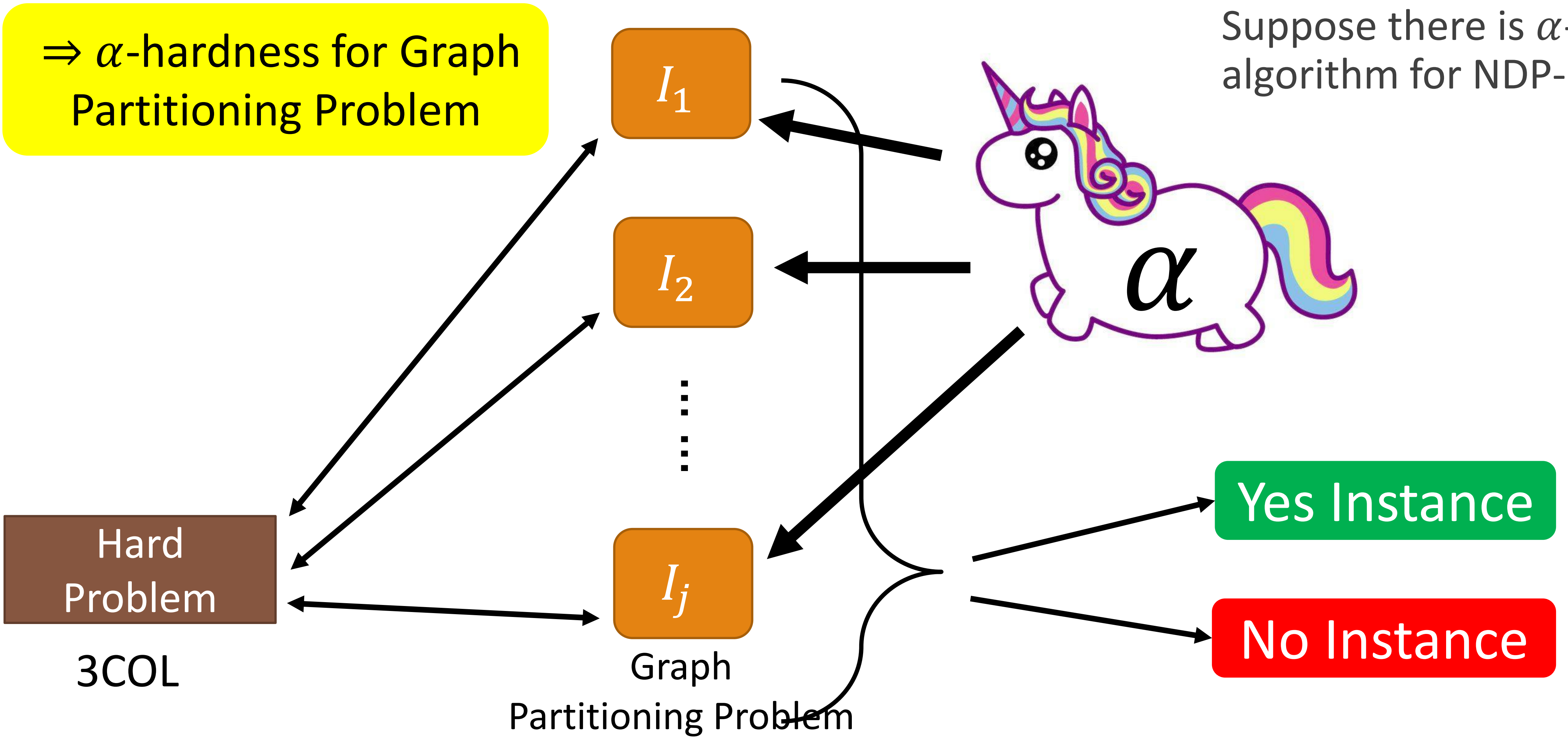
How to Show Hardness? : The Cook Way

Suppose there is α-approx. algorithm for NDP-Grid



How to Show Hardness? : The Cook Way

Suppose there is α-approx. algorithm for NDP-Grid

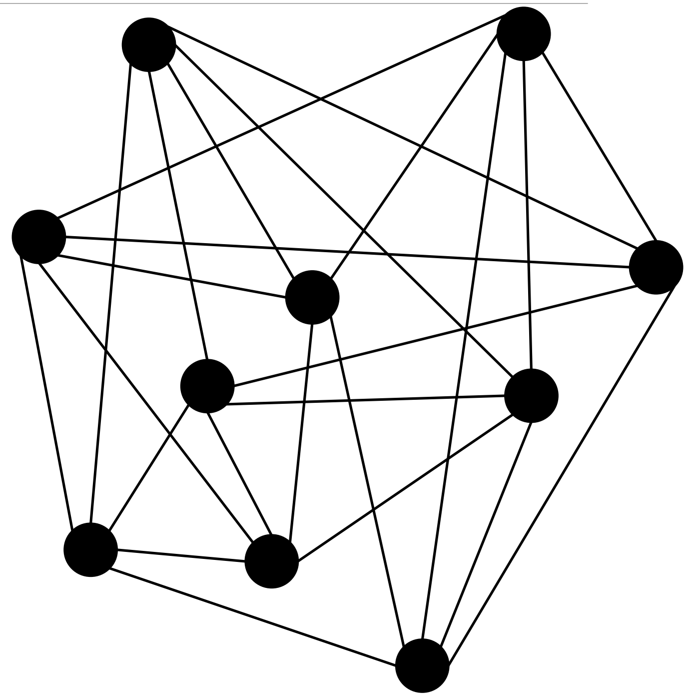


3COL5

Graph G: n vertices, m edges

Each vertex degree is exactly 5

Color vertices by $\{RGB\}$ such that no edge connects a pair of same color



NP-Hard

Coloring Possible

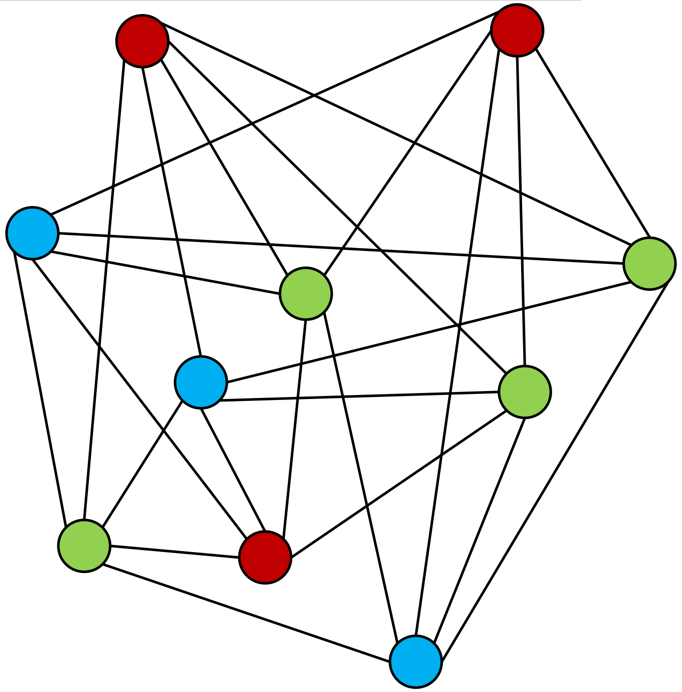
Every coloring violates at least ϵ- fraction of edges

3COL5

Graph G: n vertices, m edges

Each vertex degree is exactly 5

Color vertices by $\{RGB\}$ such that no edge connects a pair of same color



NP-Hard

Coloring Possible

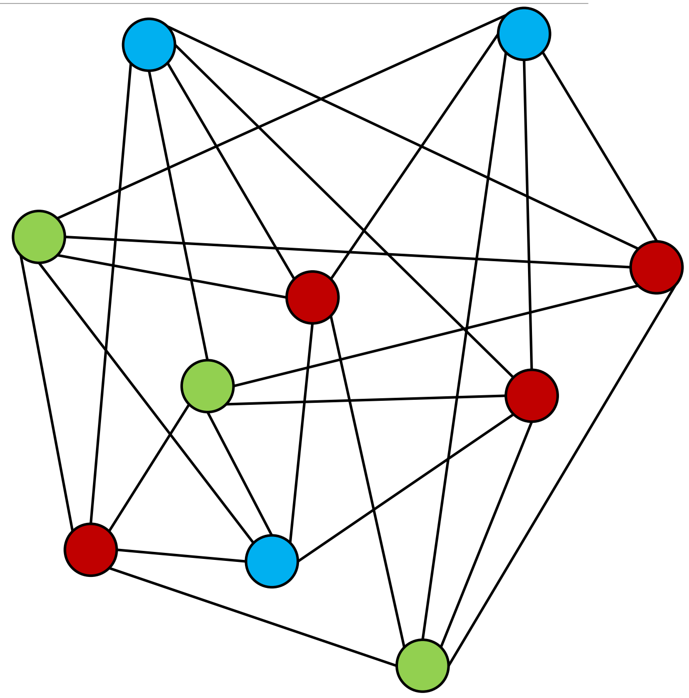
Every coloring violates at least ϵ- fraction of edges

3COL5

Graph G: n vertices, m edges

Each vertex degree is exactly 5

Color vertices by $\{RGB\}$ such that no edge connects a pair of same color



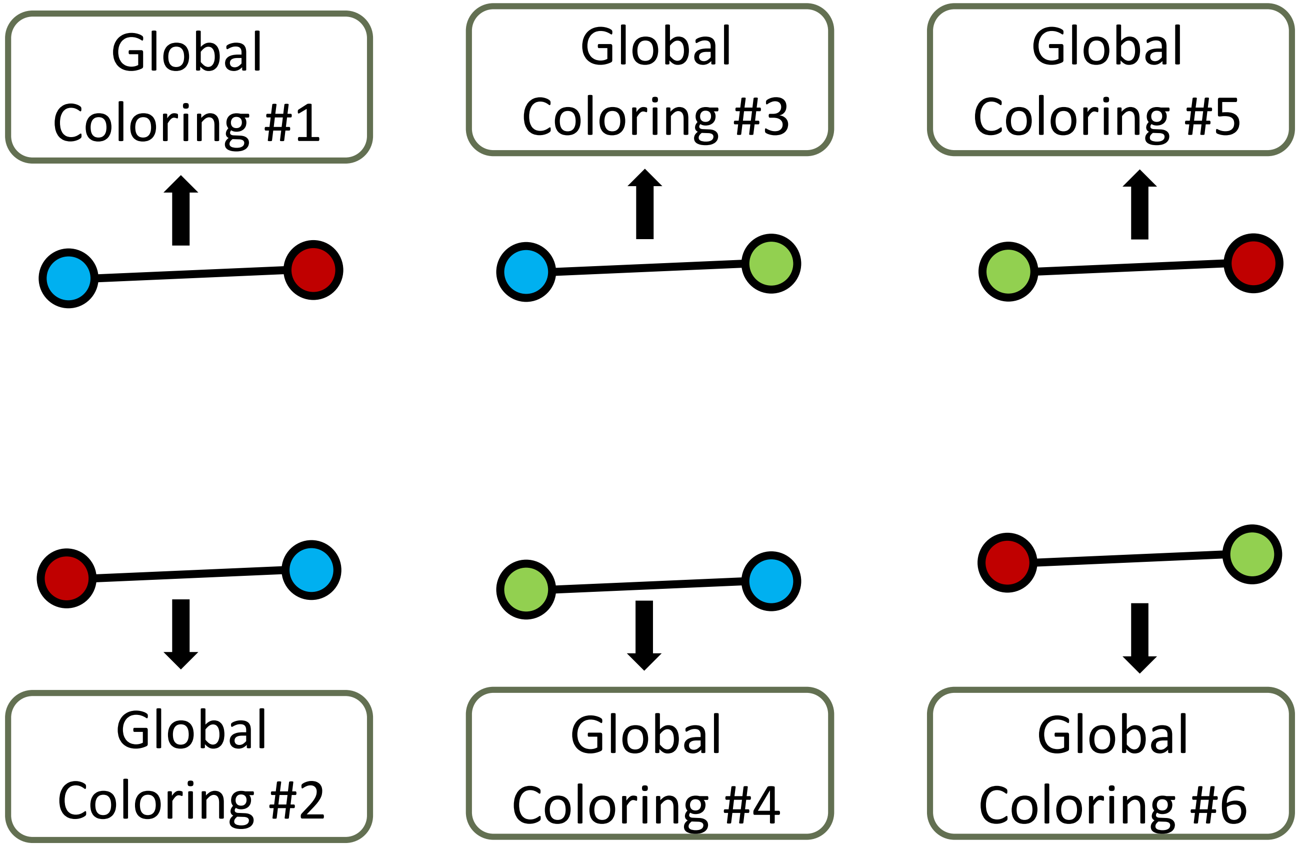
NP-Hard

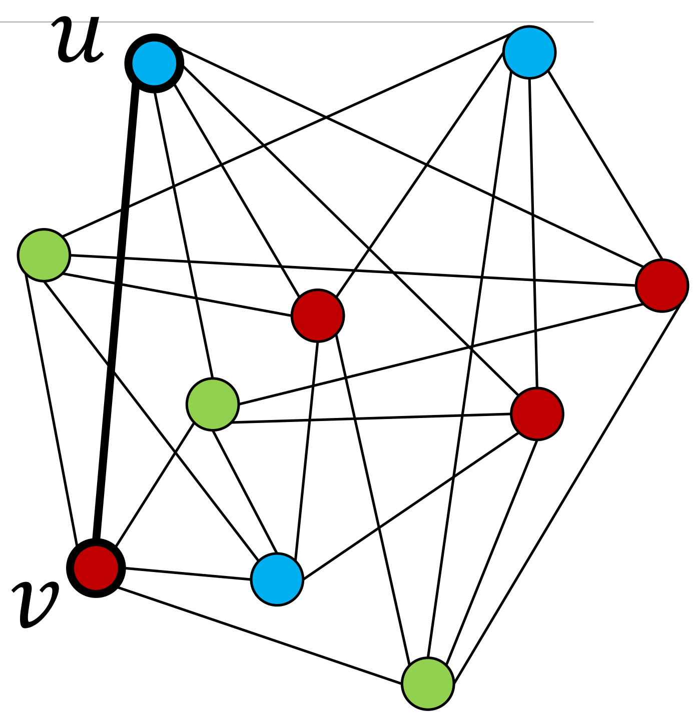
Coloring Possible

Every coloring violates at least ϵ- fraction of edges

1 legal coloring $\Rightarrow$ 6 legal colorings!

3COL5

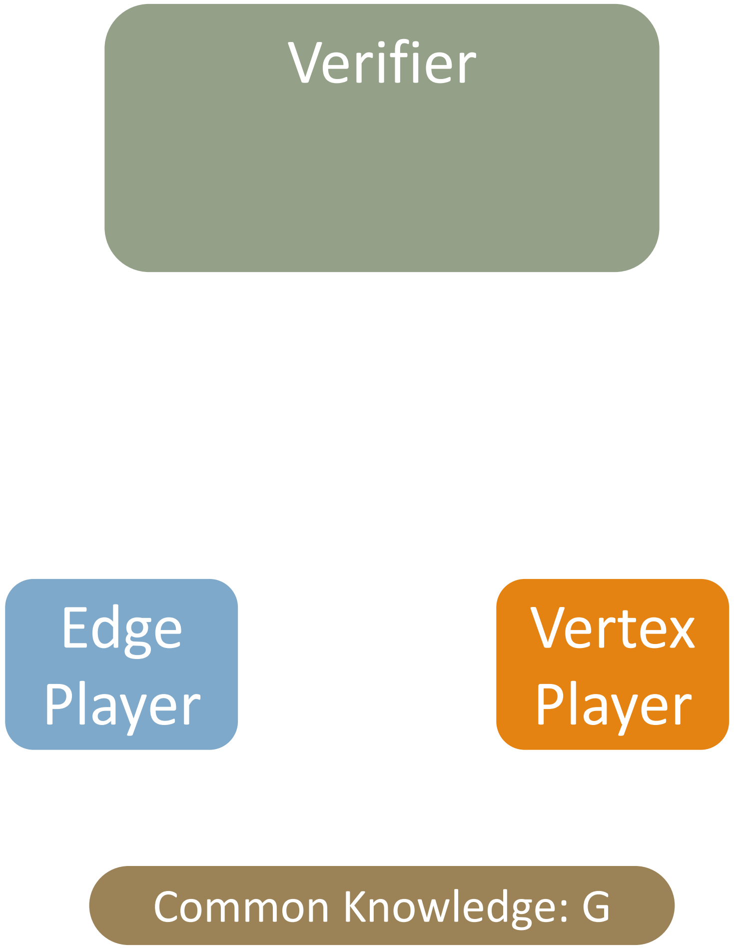




1 legal coloring ⇒ 6 legal colorings!

2 Prover Protocol

Edge-Player, Vertex-Player



2 Prover Protocol

Edge-Player, Vertex-Player

Verifier

$(u, v) \in_r E(G)$

Edge Player

Vertex Player

Common Knowledge: G

2 Prover Protocol

Edge-Player, Vertex-Player

Verifier
$(u, v) \in_r E(G)$
$u \in_r \{u, v\}$

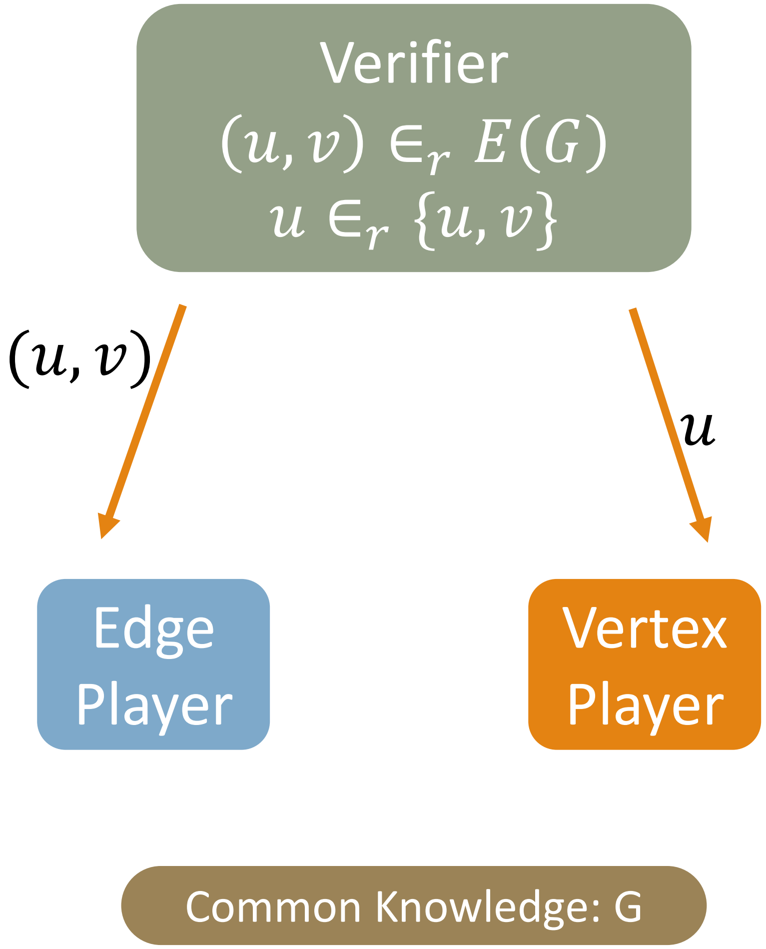
Edge Player

Vertex Player

Common Knowledge: G

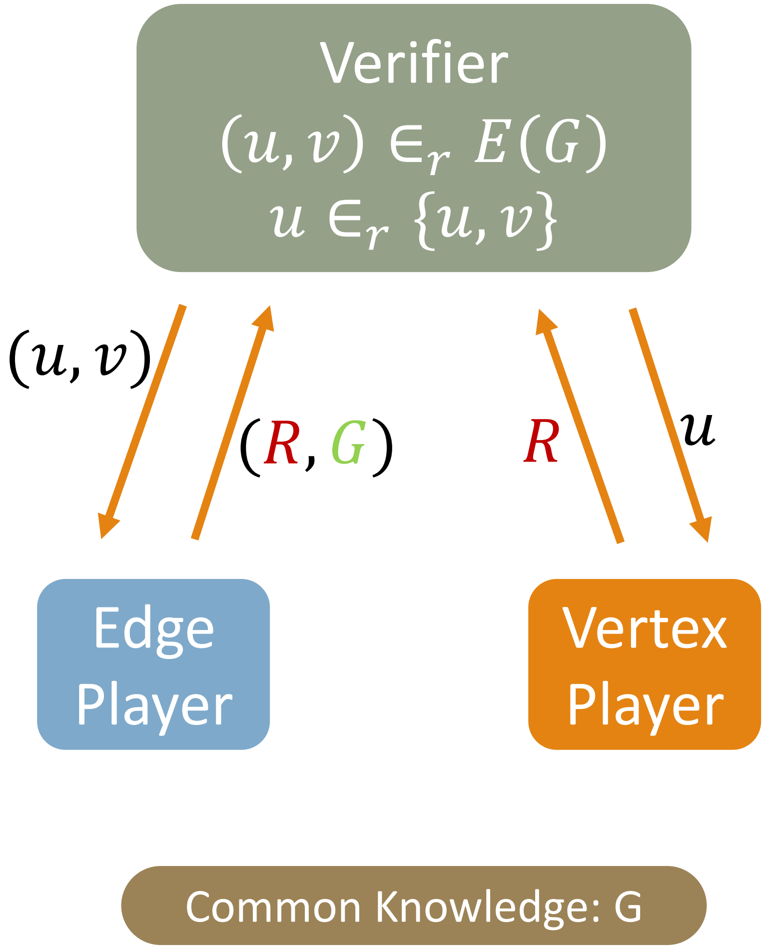
2 Prover Protocol

Edge-Player, Vertex-Player



2 Prover Protocol

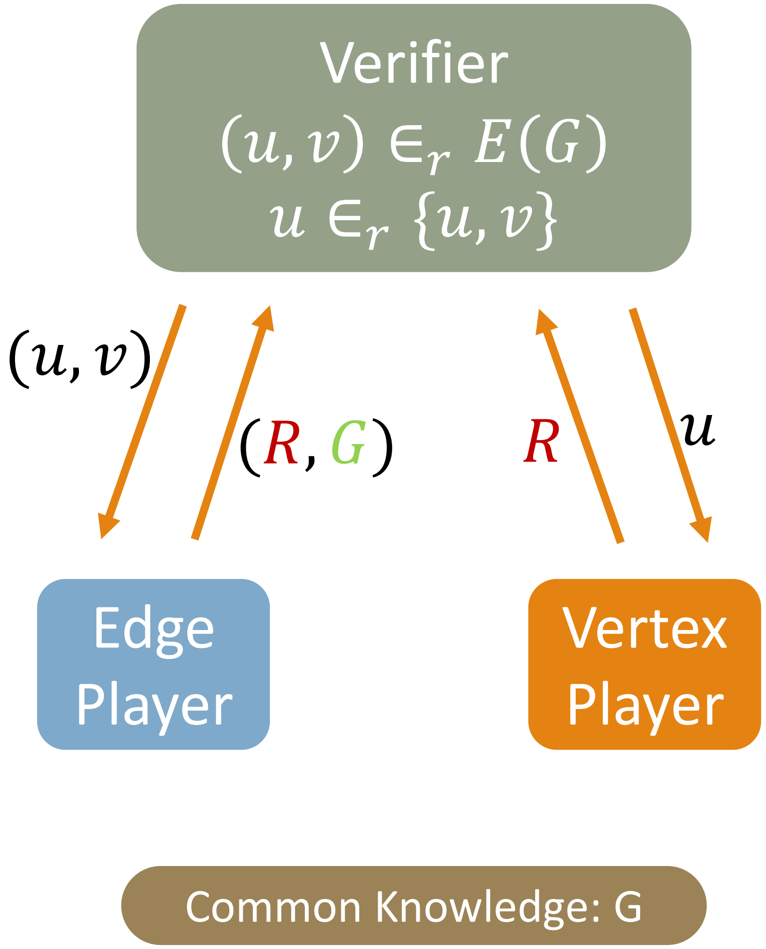
Edge-Player, Vertex-Player



2 Prover Protocol

Edge-Player, Vertex-Player

Verifier accepts iff colors match



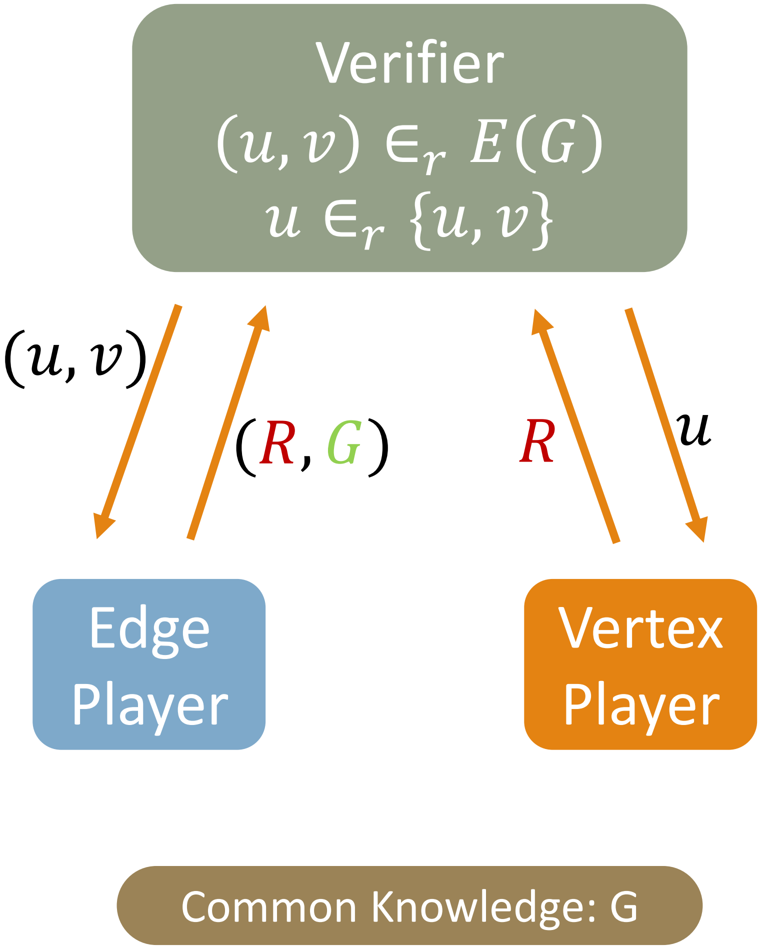
2 Prover Protocol

Edge-Player, Vertex-Player

Verifier accepts iff colors match

G=YI ⇒ 6 prover strategies where Verifier always accepts

G=NI ⇒ For any strategy of provers, Verifier accepts with probability $\leq 1 - \frac{\epsilon}{2}$



Parallel Repetition

l rounds

Think of l as $\log^{100} n$

$(e_1, e_2, \ldots, e_l) \longrightarrow$ **Edge Player** $\longrightarrow (RB, GR, \ldots, RB)$

$(v_1, v_2, \ldots, v_l) \longrightarrow$ **Vertex Player** $\longrightarrow (B, R, \ldots, B)$

Parallel Repetition

l rounds

Think of l as $\log^{100} n$

Accept iff all answers match

$(e_1, e_2, \ldots, e_l) \longrightarrow$ **Edge Player** $\longrightarrow (RB, GR, \ldots, RB)$

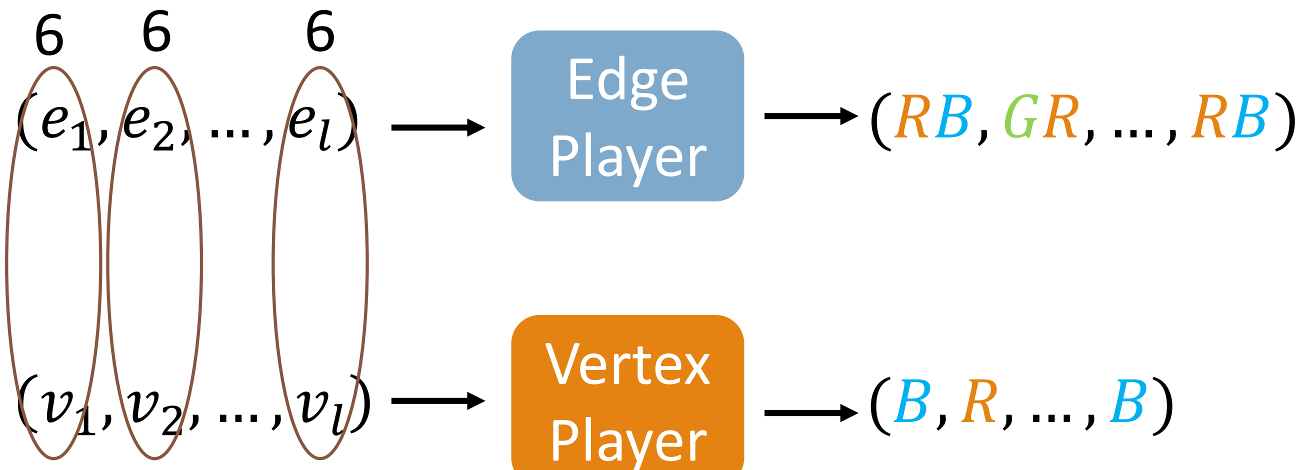
$(v_1, v_2, \ldots, v_l) \longrightarrow$ **Vertex Player** $\longrightarrow (B, R, \ldots, B)$

Parallel Repetition

l rounds

Think of l as $\log^{100} n$

Accept iff all answers match



G=YI $\Rightarrow 6^l$ prover strategies where Verifier always accepts

Parallel Repetition

l rounds

Think of l as $\log^{100} n$

$(e_1, e_2, \dots, e_l) \longrightarrow$ Edge Player $\longrightarrow (RB, GR, \dots, RB)$

Accept iff all answers match

$(v_1, v_2, \dots, v_l) \longrightarrow$ Vertex Player $\longrightarrow (B, R, \dots, B)$

G=YI $\Rightarrow$ 6^l prover strategies where Verifier always accepts

G=NI $\Rightarrow$ Verifier accepts with prob. $\leq 2^{-\gamma l}$

Parallel Repetition Theorem [Raz '98]

Parallel Repetition

l rounds

Think of l as $\log^{100} n$

$(e_1, e_2, \ldots, e_l) \longrightarrow$ Edge Player $\longrightarrow (RB, GR, \ldots, RB)$

Accept iff all answers match

$(v_1, v_2, \ldots, v_l) \longrightarrow$ Vertex Player $\longrightarrow (B, R, \ldots, B)$

G=YI $\Rightarrow 6^l$ prover strategies where Verifier always accepts

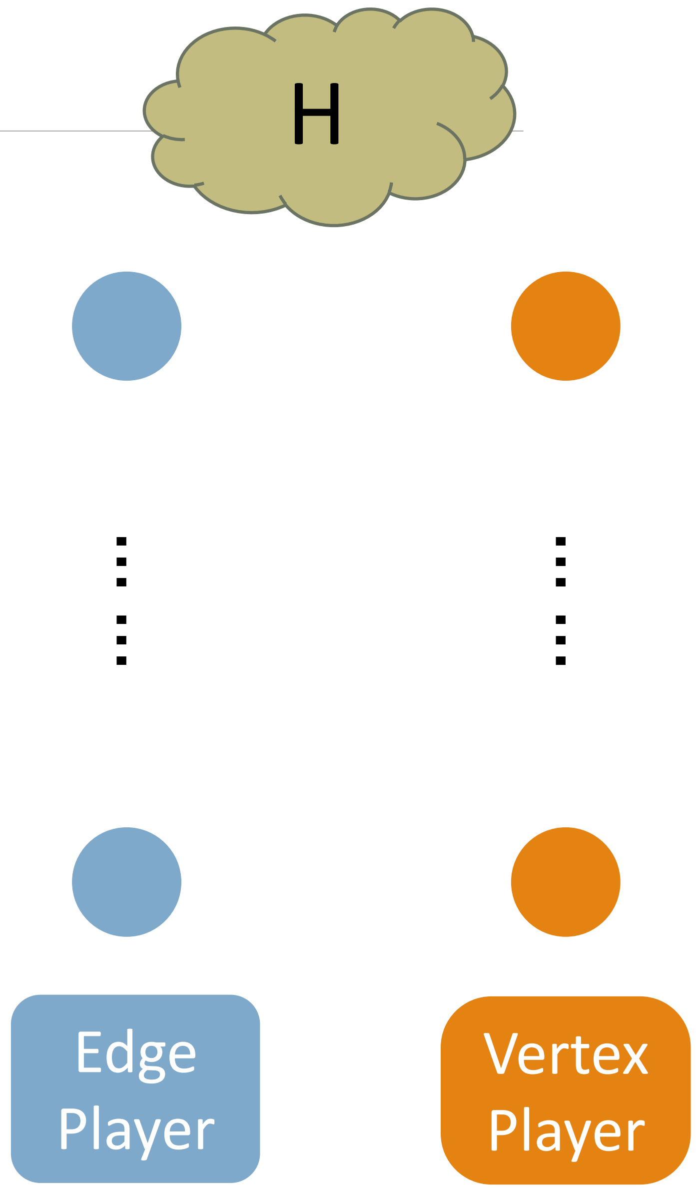
Good prover strategy certifies that G is YI

G=NI $\Rightarrow$ Verifier accepts with prob. $\leq 2^{-\gamma l}$

Parallel Repetition Theorem [Raz '98]

The Constraint Graph

- Bipartite graph
- Edge-Player queries on one side
- Vertex-Player queries on other

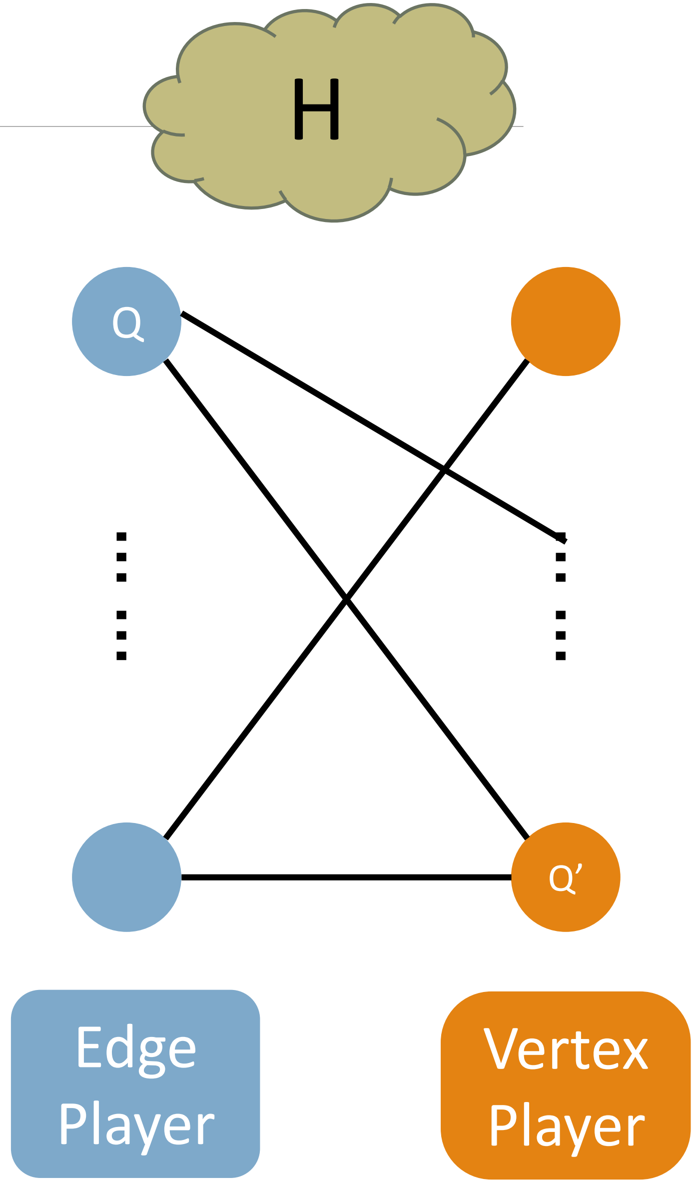


The Constraint Graph

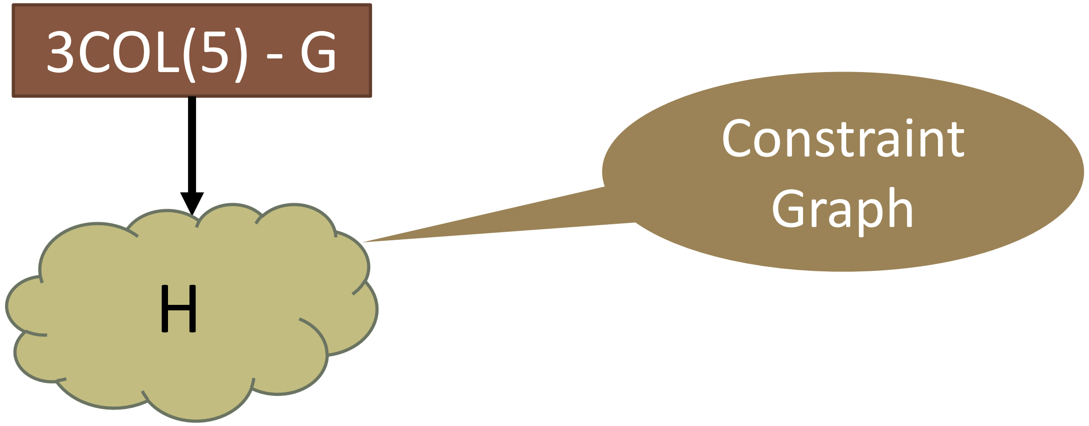
- Bipartite graph
- Edge-Player queries on one side
- Vertex-Player queries on other

- Edge iff **compatible** queries

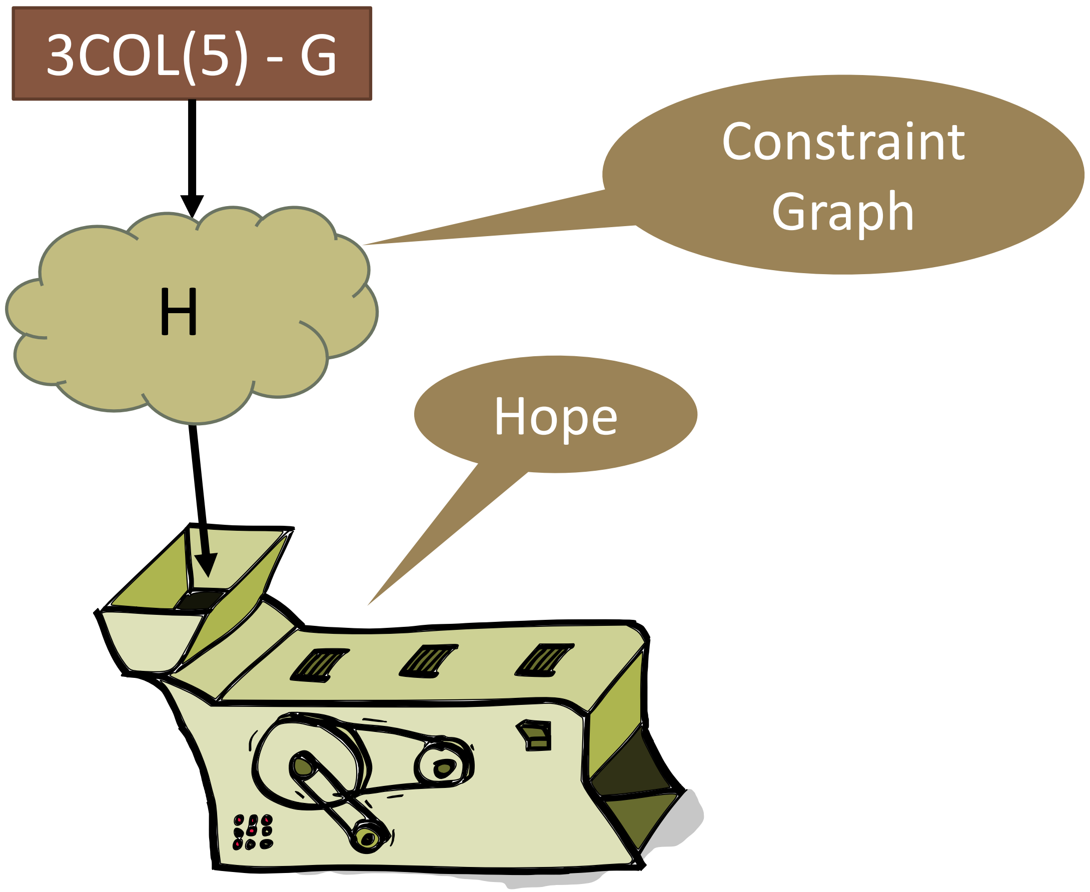
Verifier asks that pair of queries



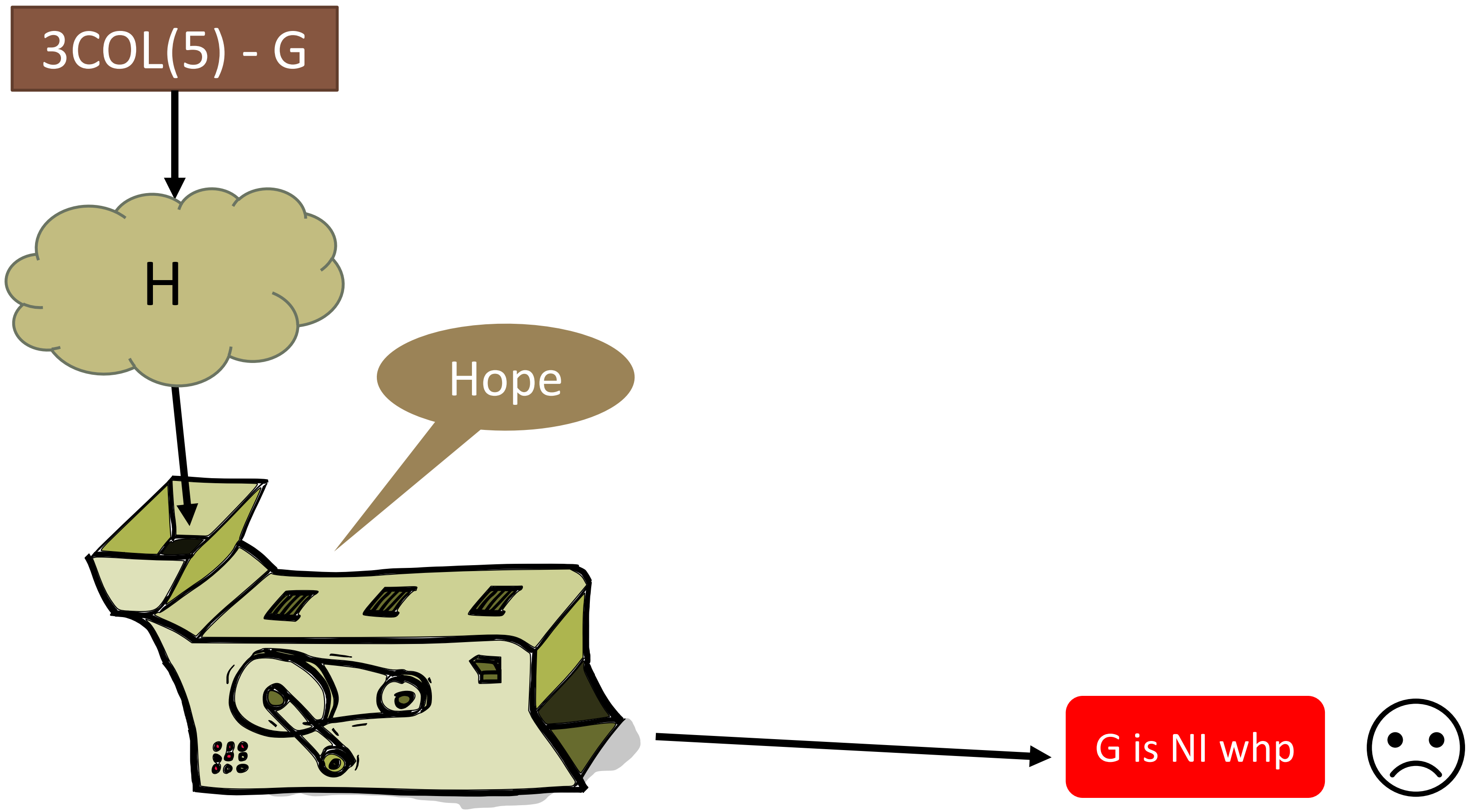
The Reduction



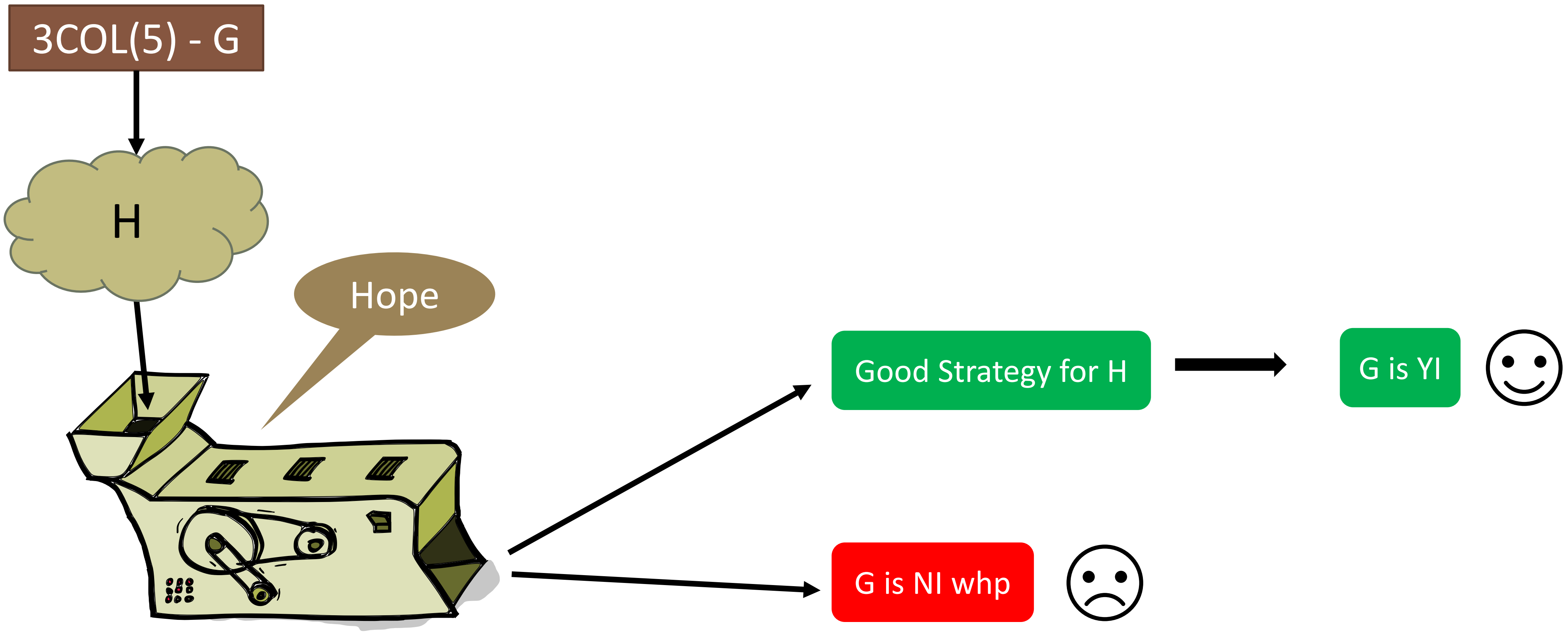
The Reduction



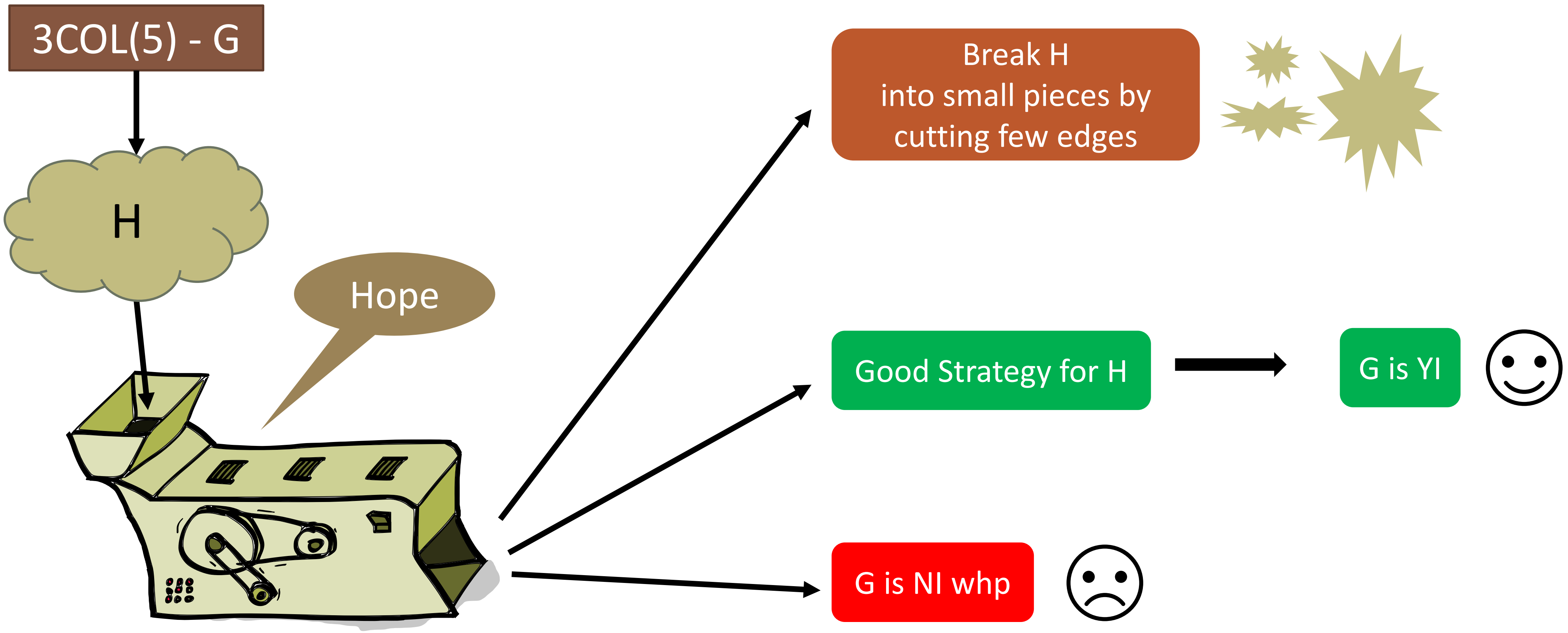
The Reduction



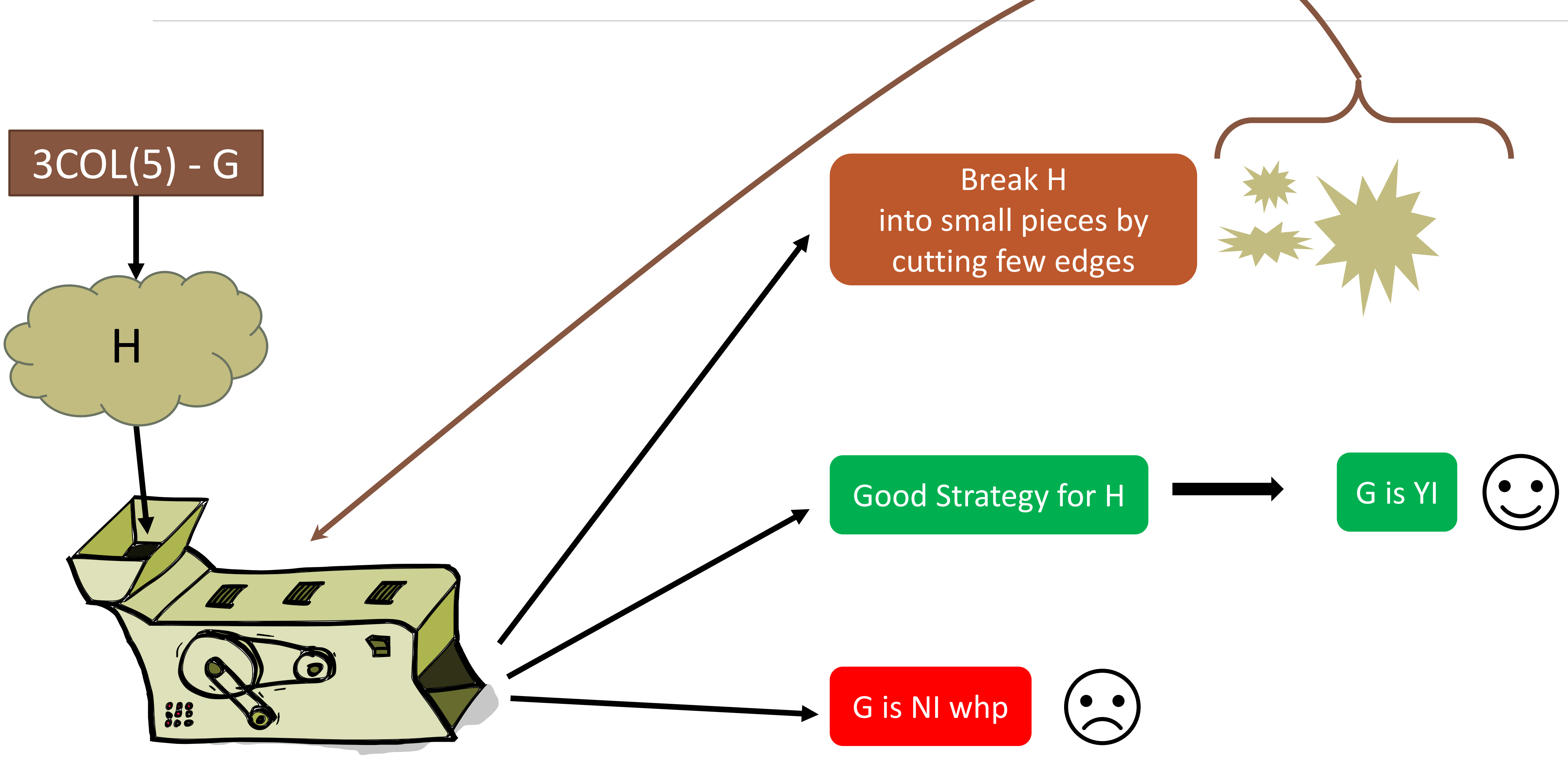
The Reduction



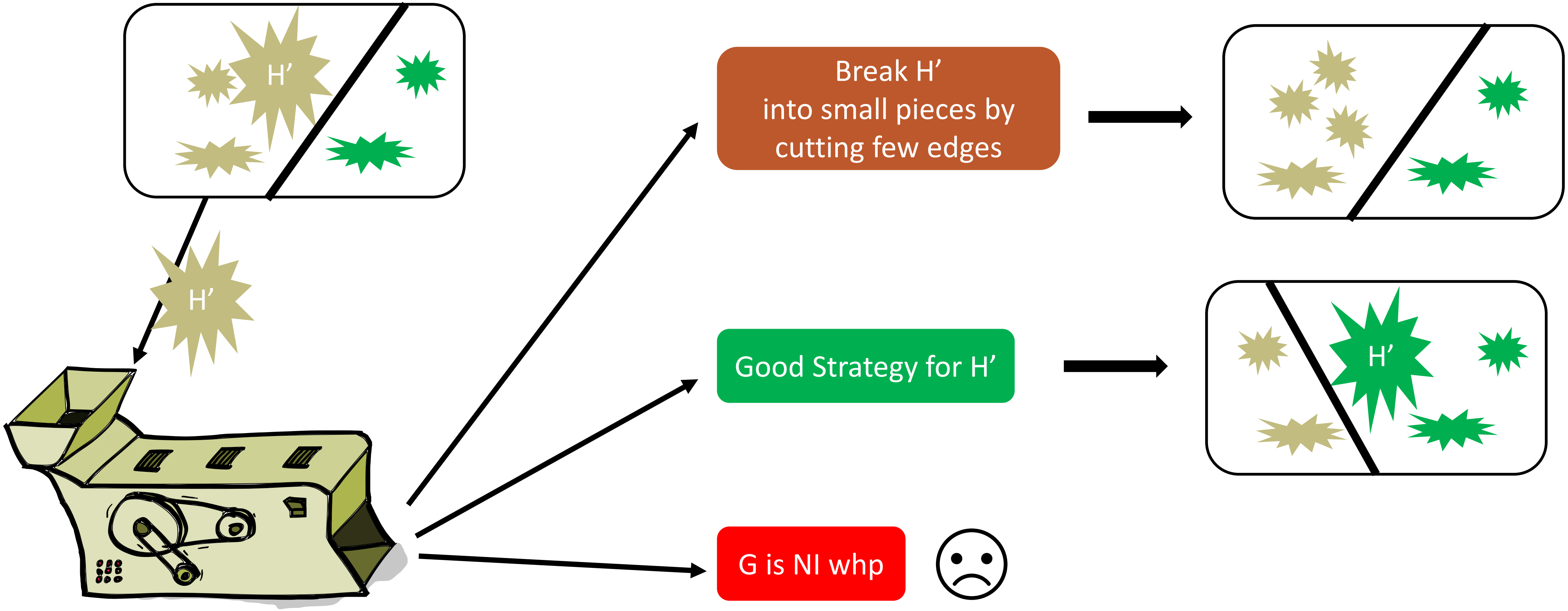
The Reduction



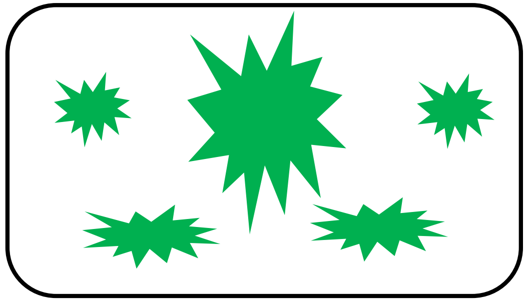
The Reduction



The Reduction

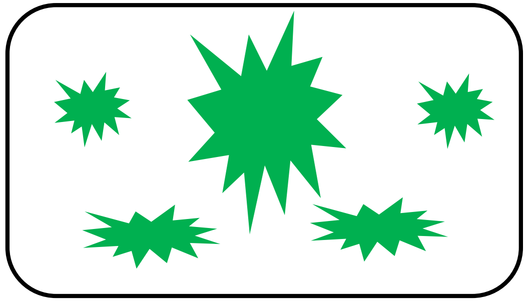


Eventually....

Either: 

Or: G is NI whp 

Eventually....

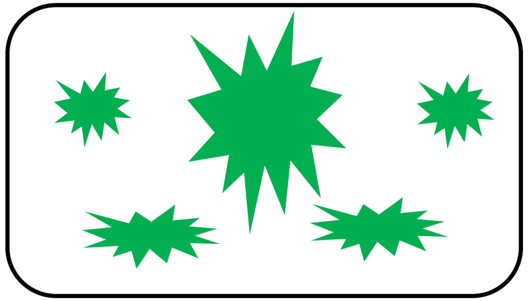
Either: 

Good Strategy for most query-pairs to provers

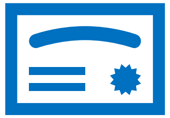
Or: G is NI whp ☹

Eventually....

Either:



Good Strategy for most query-pairs to provers

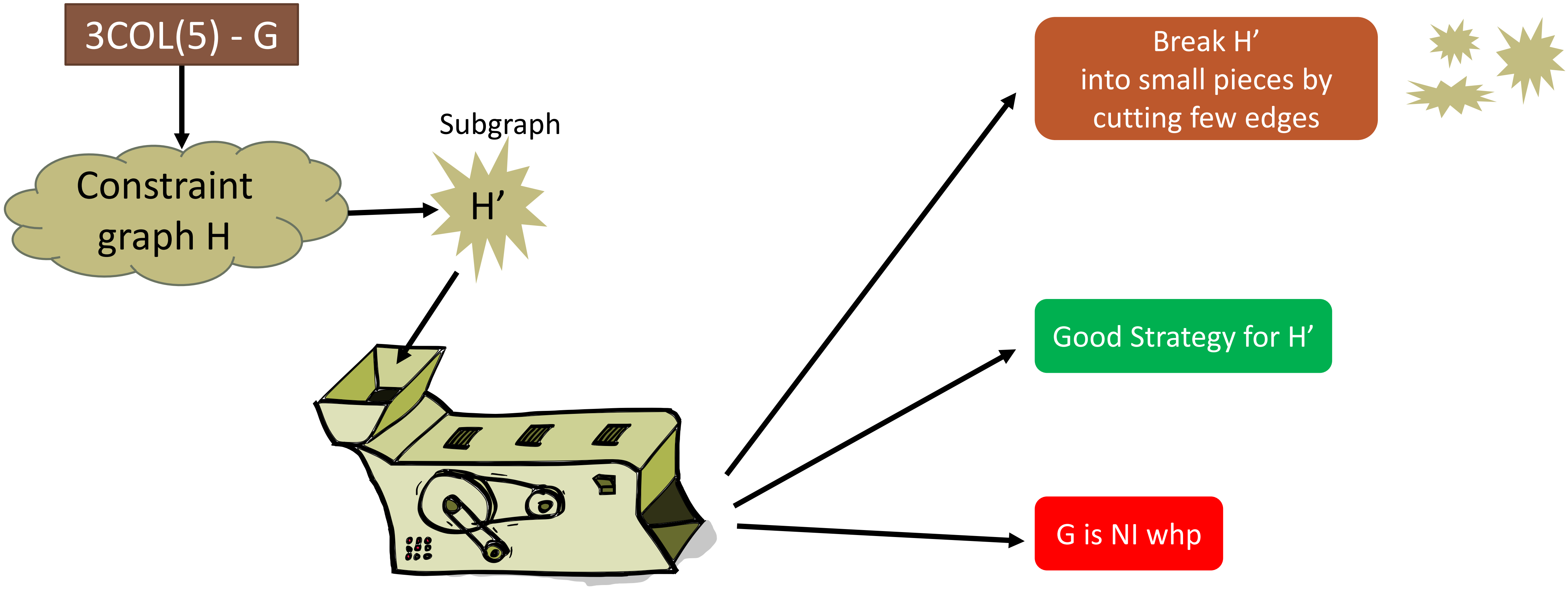


G is YI ☺

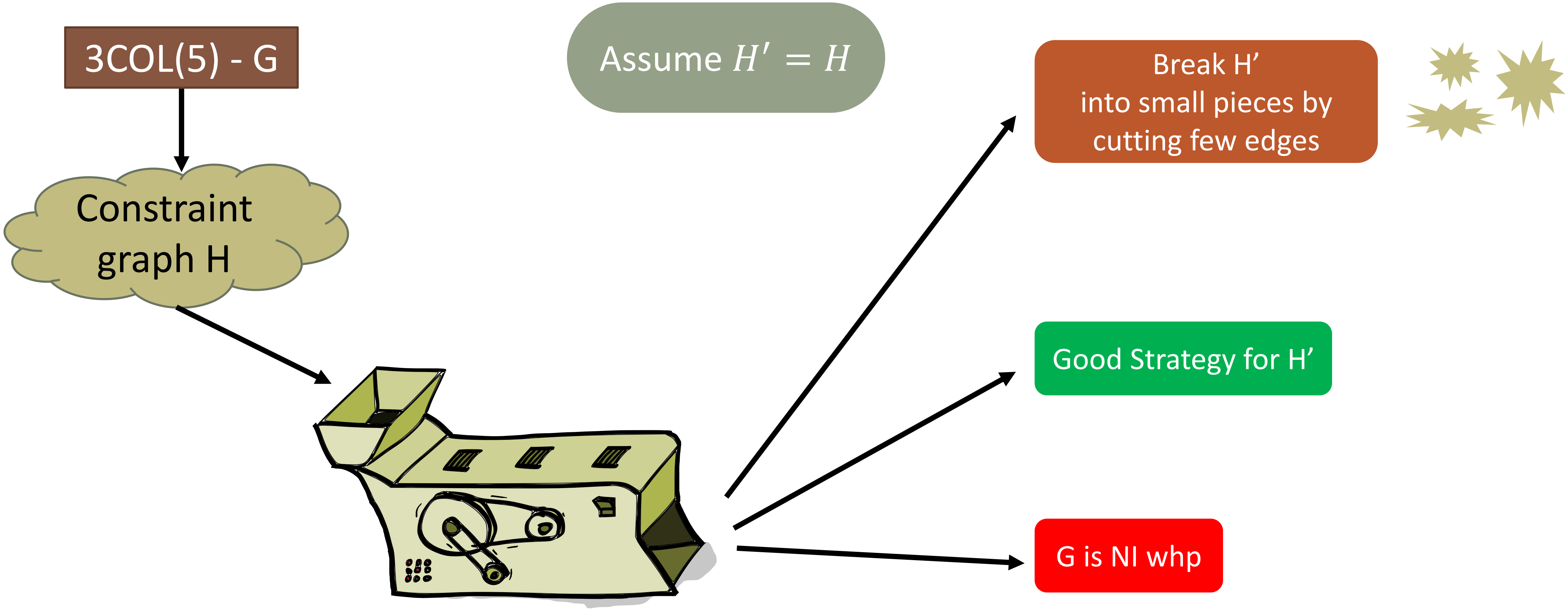
Or:

G is NI whp ☹

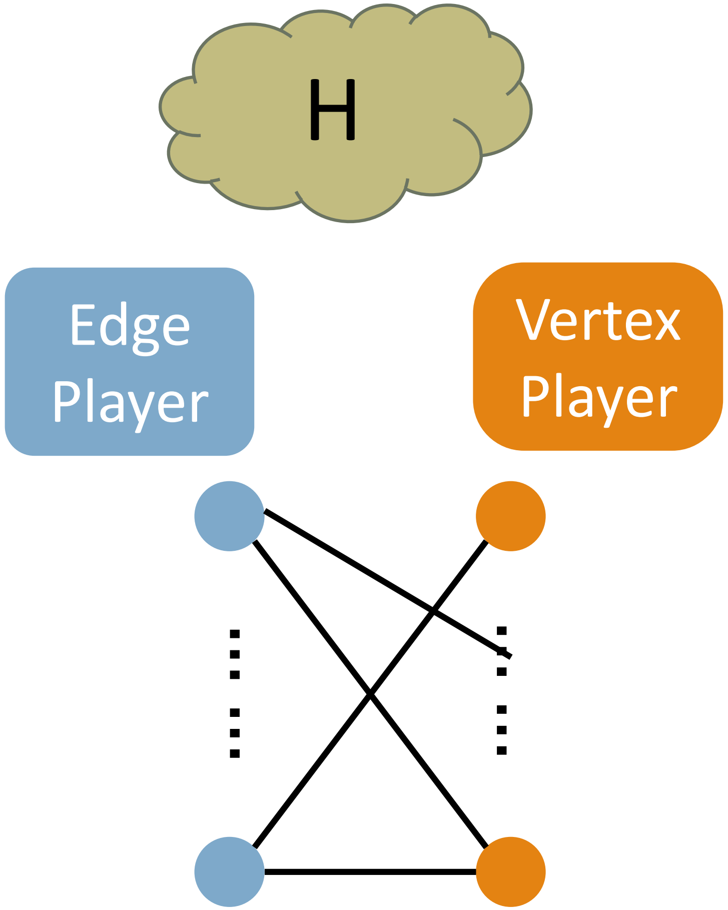
Core Algorithm



Core Algorithm: Simplified



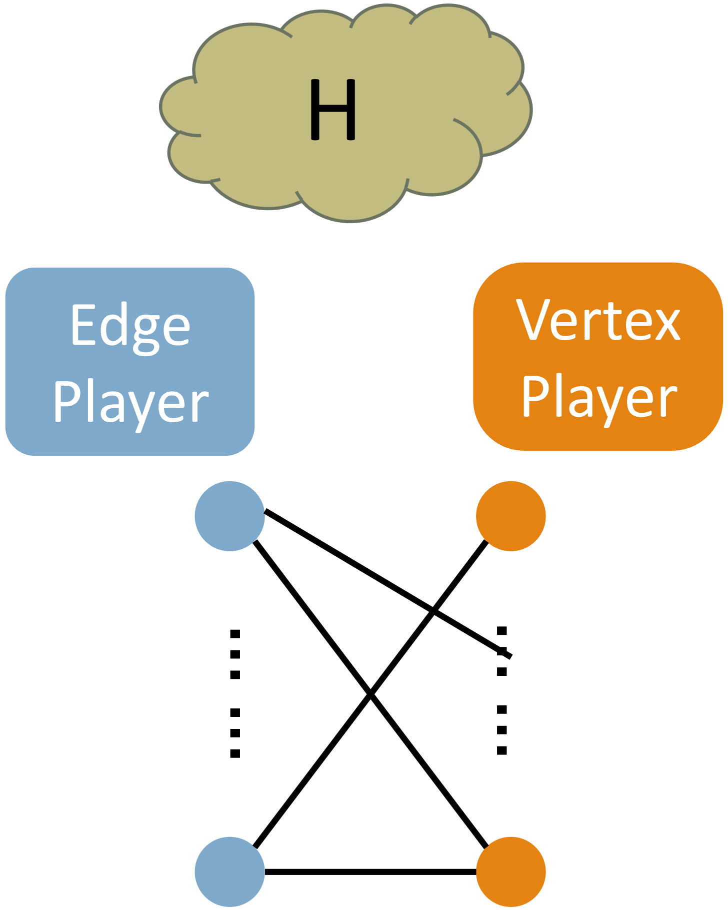
The Constraint Graph



The Constraint Graph

$(e_1, e_2, \ldots, e_l) \longrightarrow$ **Edge Player** $\longrightarrow (RB, GR, \ldots, RB)$

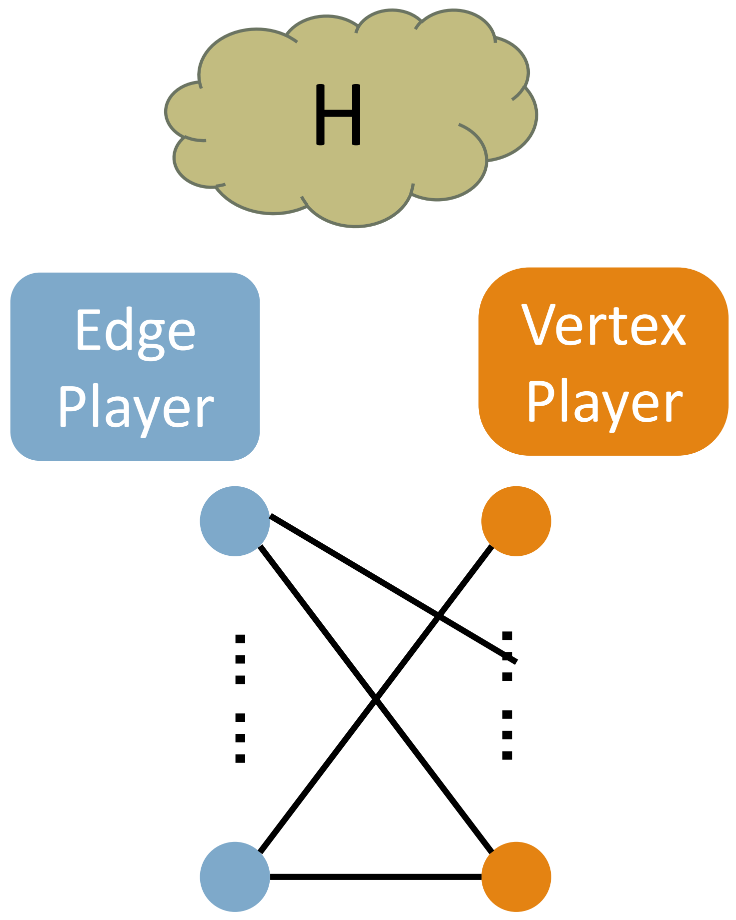
$(v_1, v_2, \ldots, v_l) \longrightarrow$ **Vertex Player** $\longrightarrow (B, R, \ldots, B)$



The Constraint Graph

$(e_1, e_2, \dots, e_l) \longrightarrow$ **Edge Player** $\longrightarrow (RB, GR, \dots, RB)$

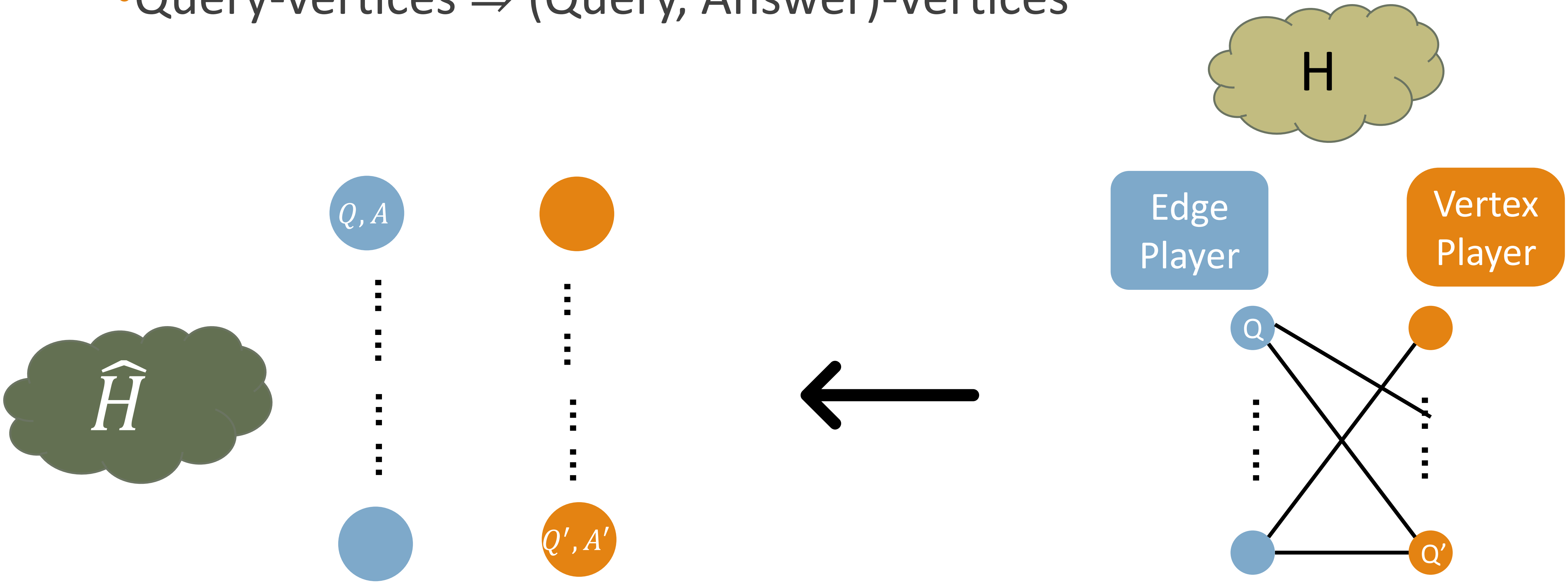
$(v_1, v_2, \dots, v_l) \longrightarrow$ **Vertex Player** $\longrightarrow (B, R, \dots, B)$



- Only 6^l responses of edge-player
- Only 3^l responses of vertex-player

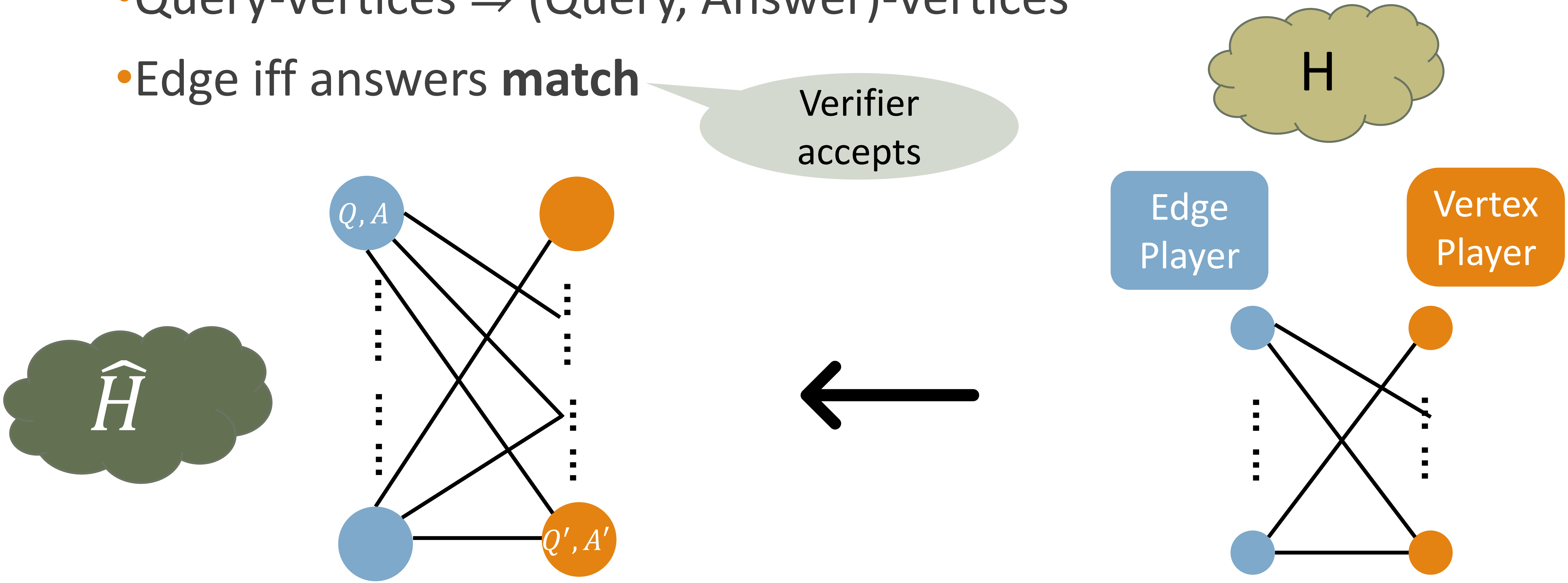
The Cover Graph

- Query-vertices ⇒ (Query, Answer)-vertices



The Cover Graph

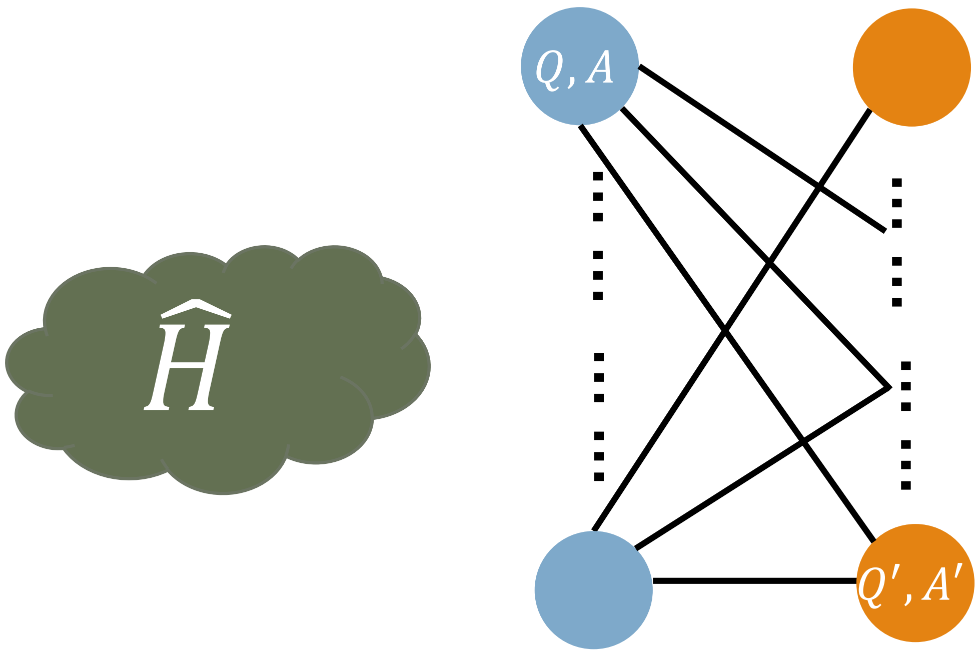
- Query-vertices ⇒ (Query, Answer)-vertices
- Edge iff answers **match**



The Cover Graph

What if G is YI?

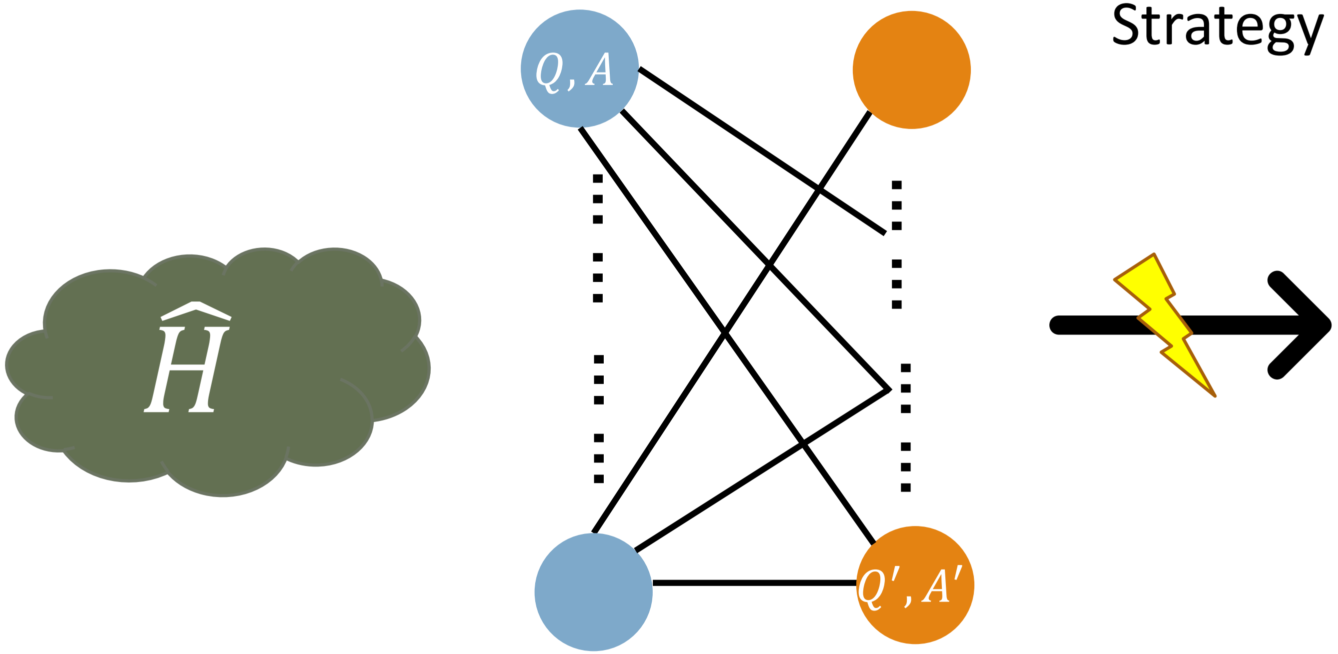
- 6^l strategies for H

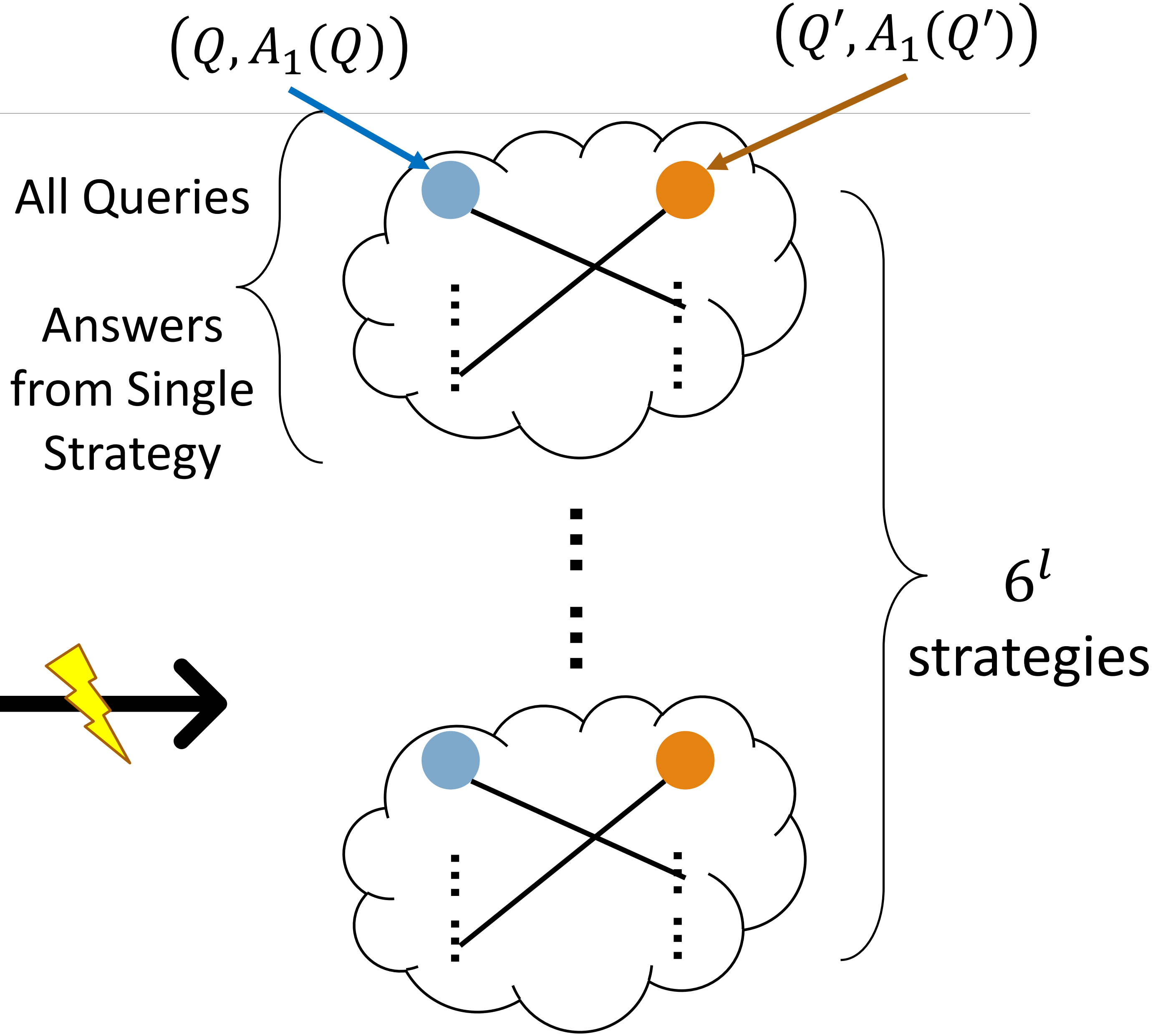


The Cover Graph

What if G is YI?

- 6^l strategies for H

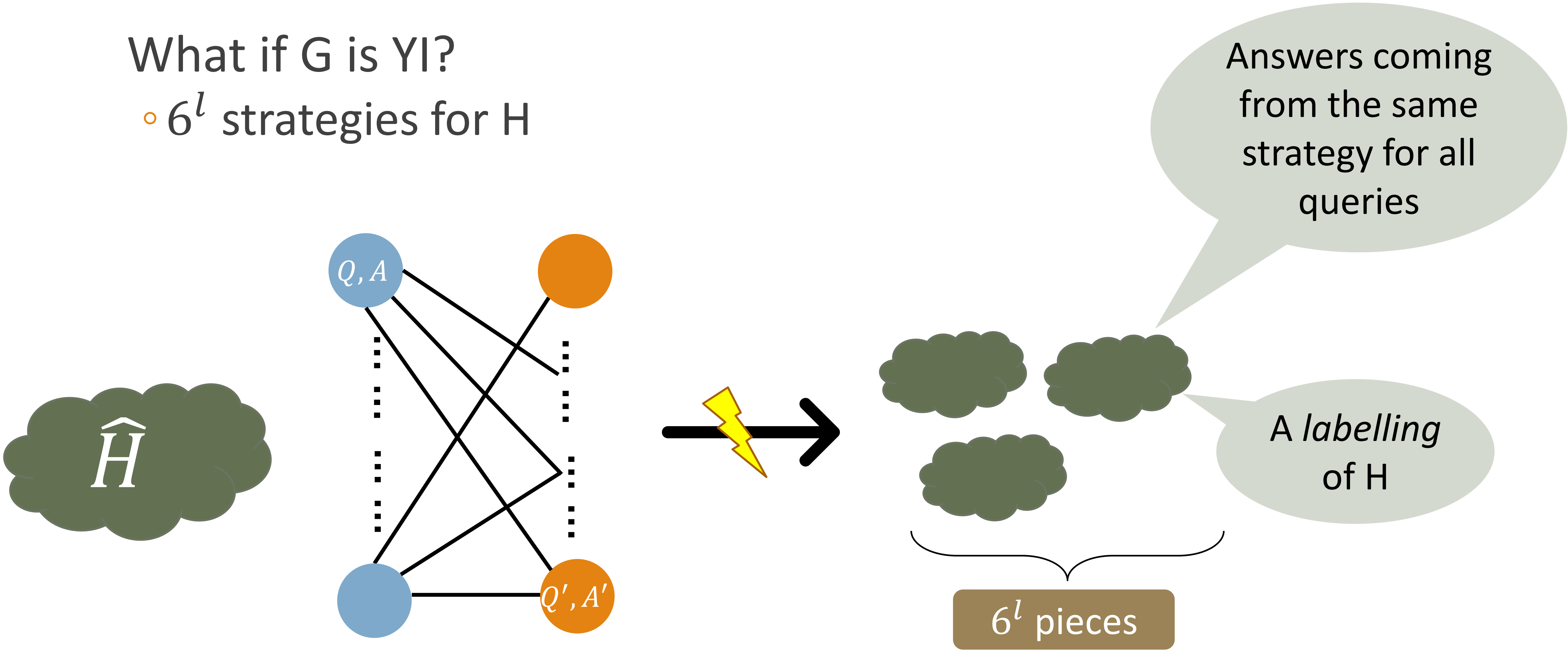




The Cover Graph

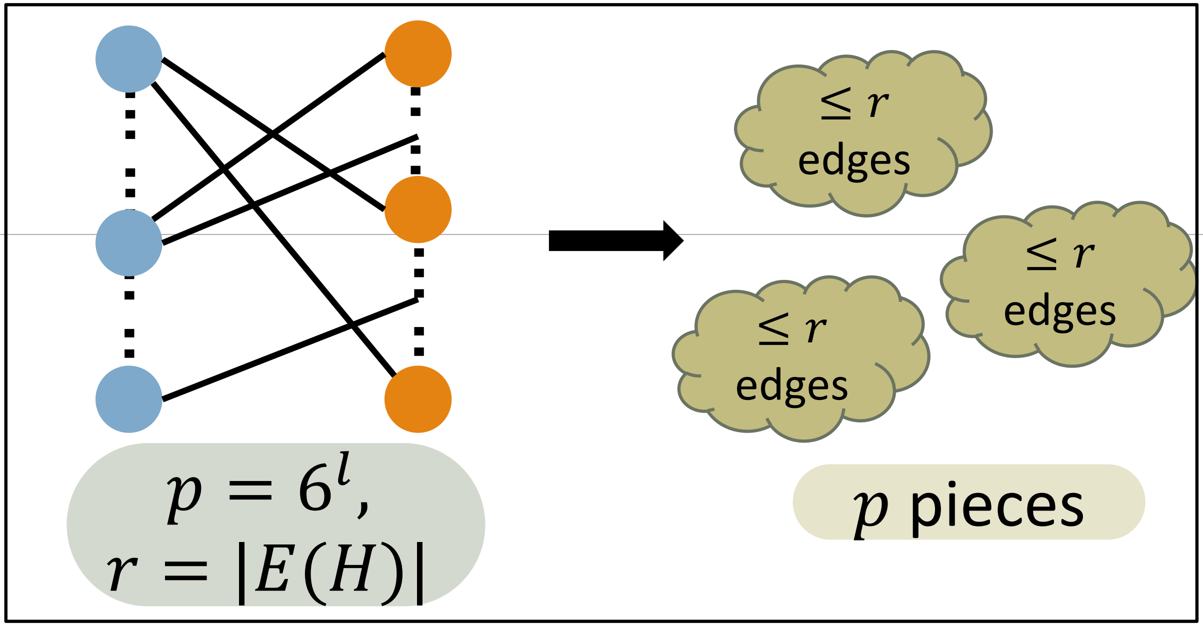
What if G is YI?

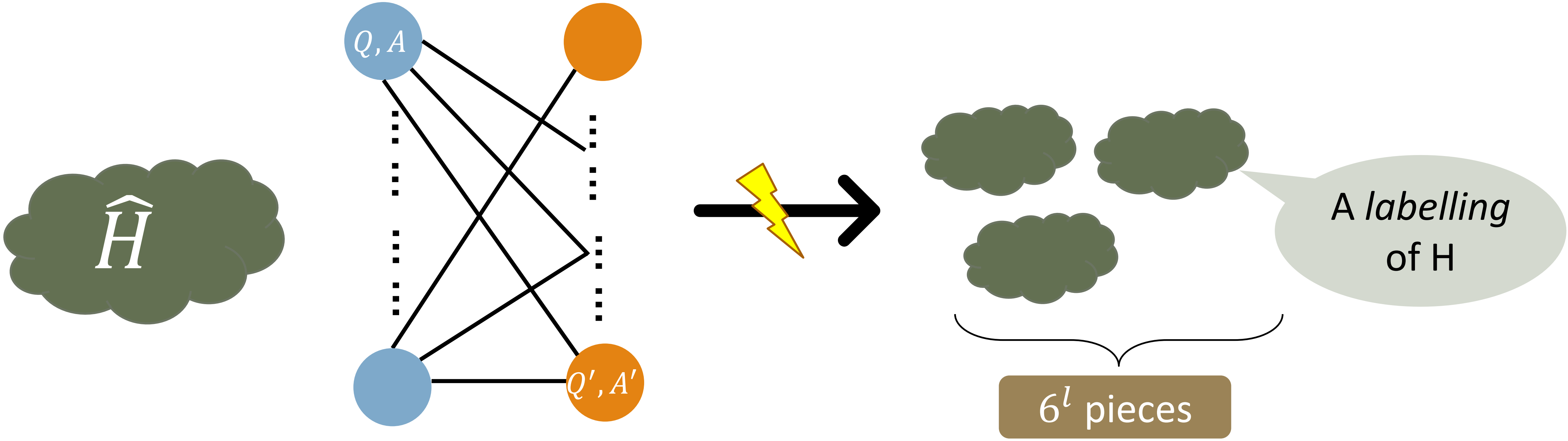
- 6^l strategies for H



G is YI

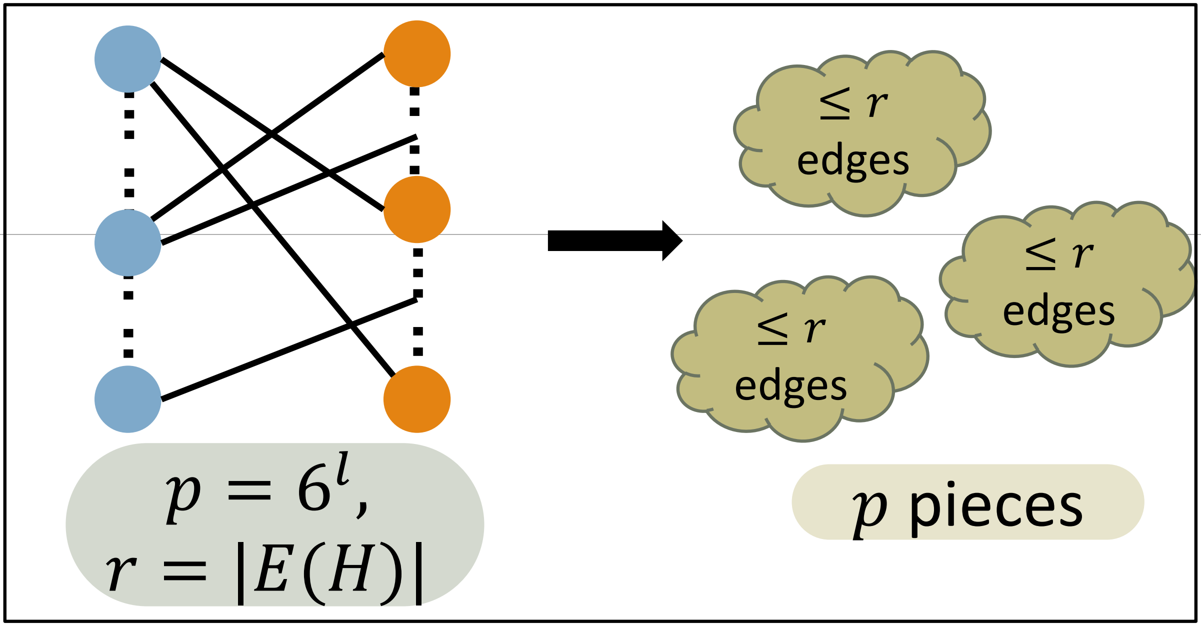
Partitioning Problem!

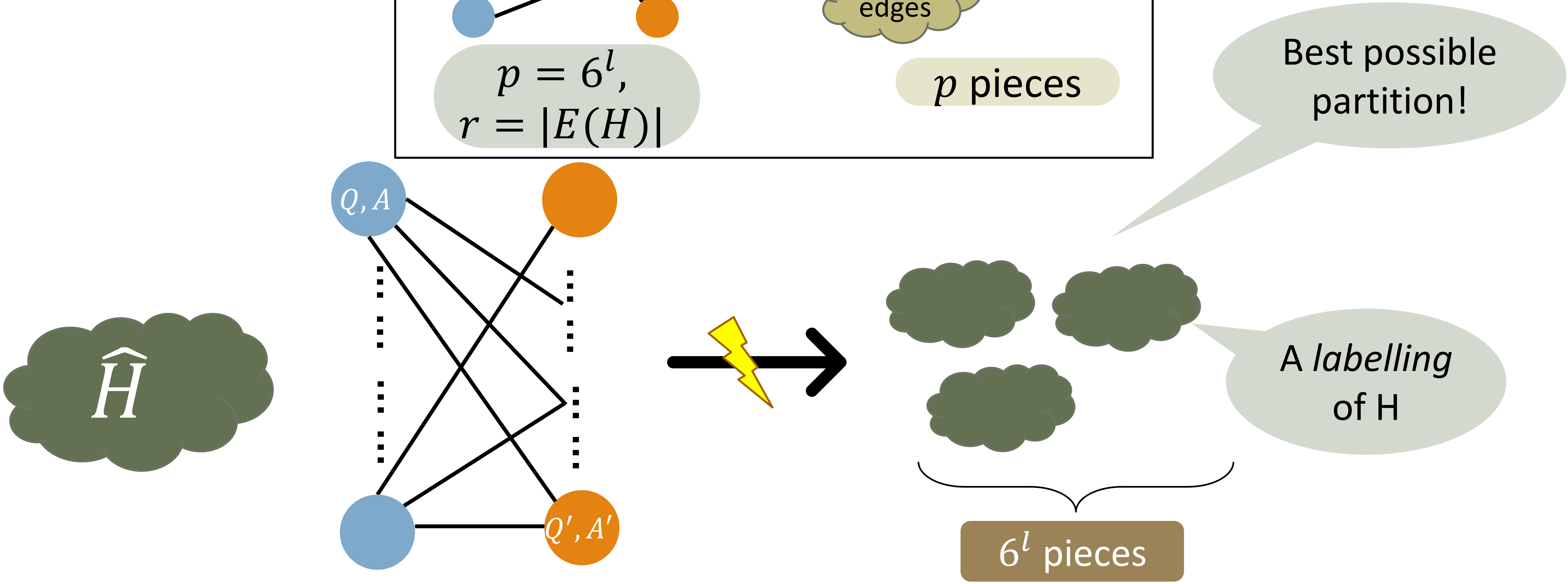




G is YI

Partitioning Problem!





Partitioning Problem

G is YI $\longleftrightarrow$ Good Strategy for H $\longrightarrow$ Partitioning with Many Surviving Edges

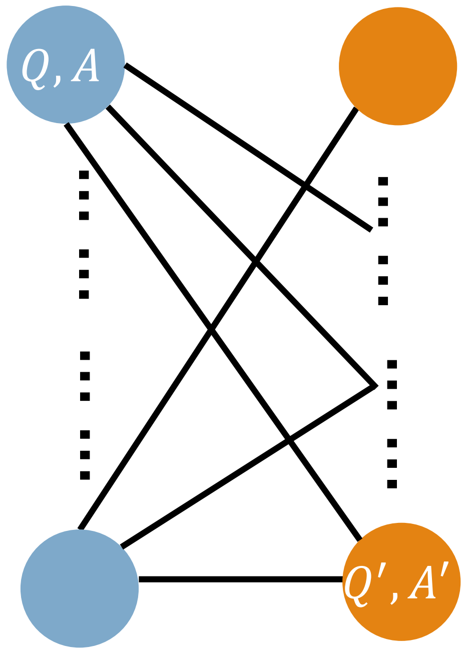
Partitioning Problem

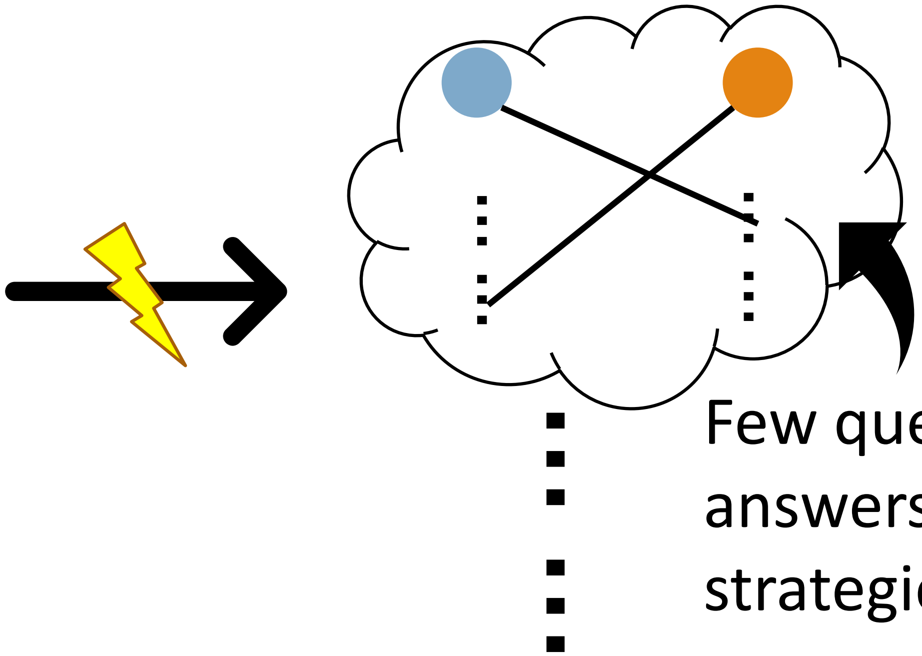
G is YI $\longleftrightarrow$ Good Strategy for H $\underset{?}{\overset{\longrightarrow}{\longleftarrow}}$ Partitioning with Many Surviving Edges

Extract strategy from partitioning with many surviving edges?

Partitioning Problem

G is YI ⟷ Good Strategy for H ⇄ Partitioning with Many Surviving Edges





Extract strategy from partitioning with many surviving edges?

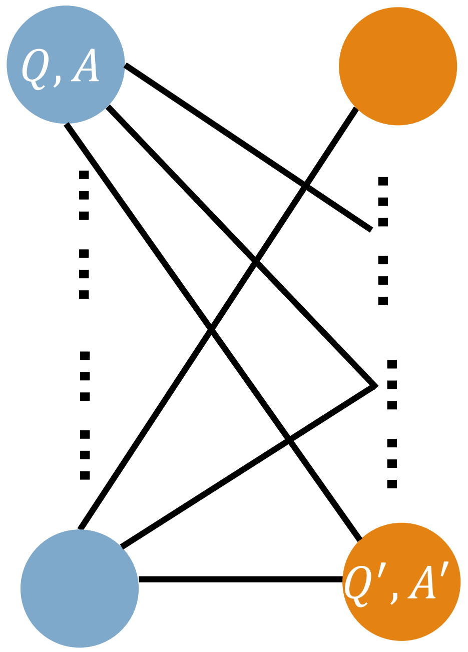
NO! ☹

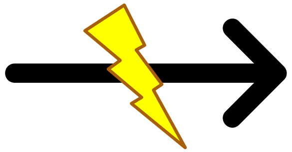
Few queries, but answers from all strategies

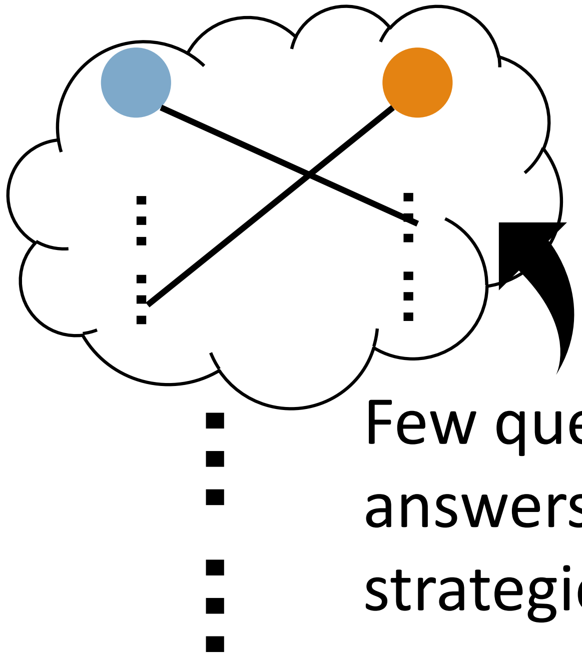
Partitioning Problem

G is YI ⟷ Good Strategy for H ⇄ Partitioning with Many Surviving Edges

Cheating partition must exploit the structure of $\widehat{H}$...







Few queries, but answers from all strategies

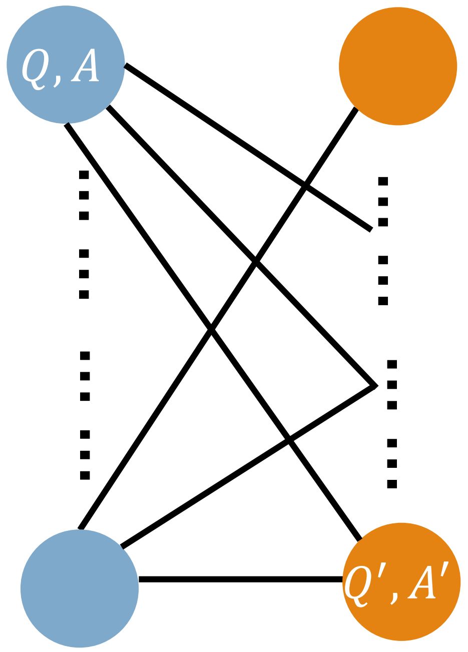
Partitioning Problem

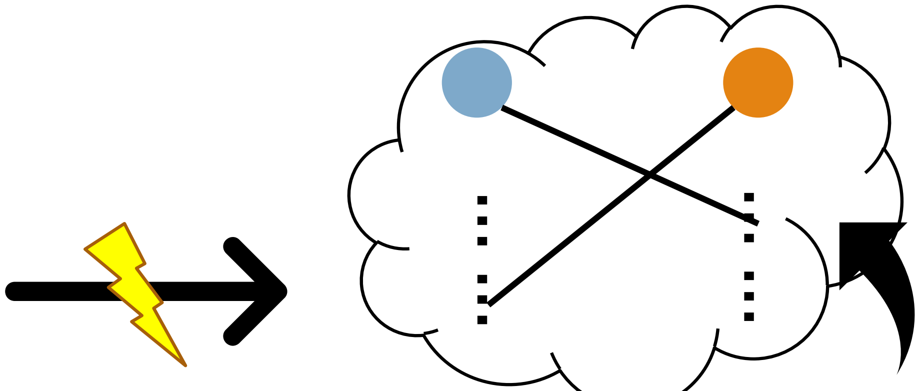
G is YI ⟷ Good Strategy for H ⇄ Partitioning with Many Surviving Edges

?

Cheating partition must exploit the structure of $\widehat{H}$...

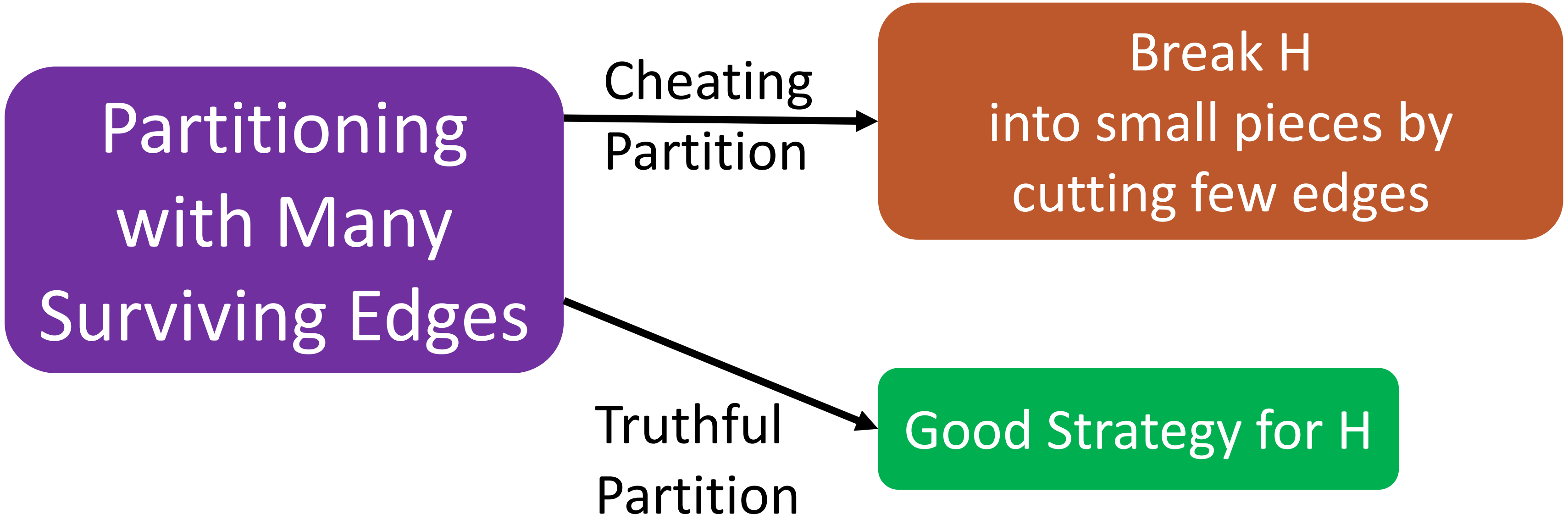
Leverage it to break H into pieces!



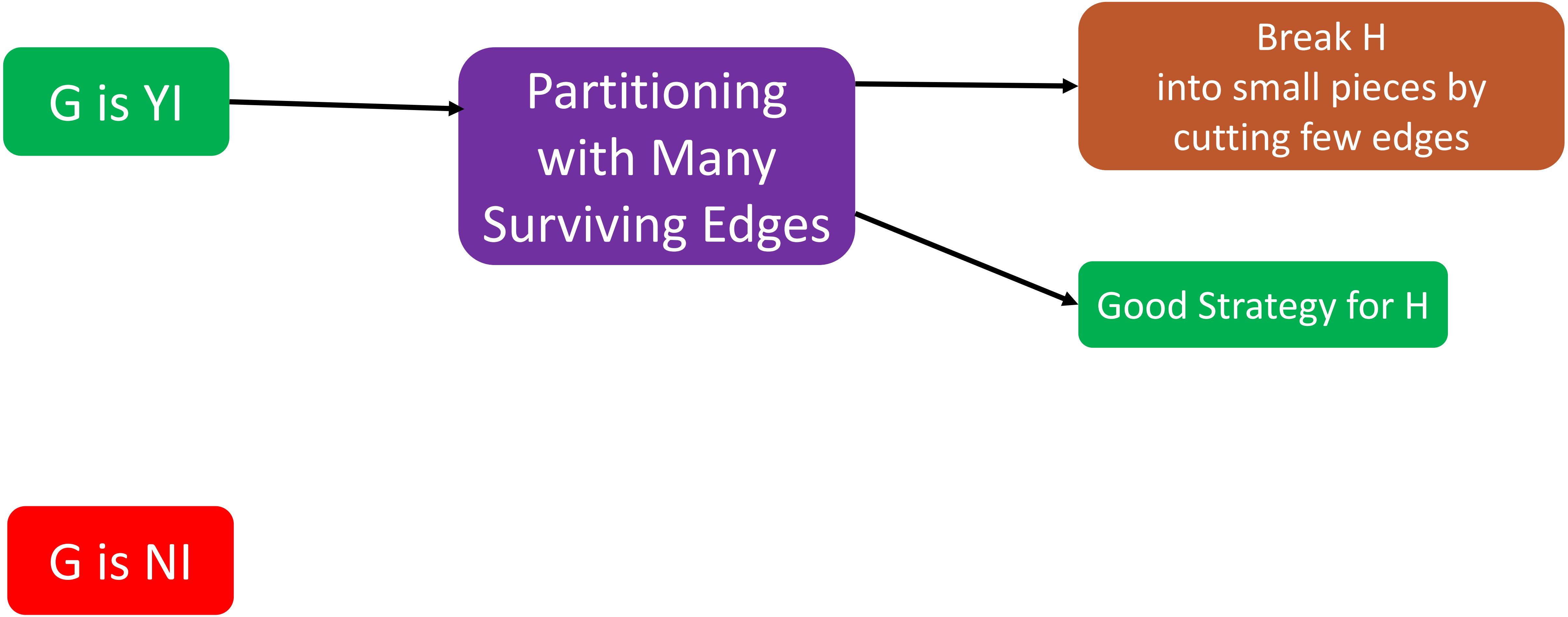


Few queries, but answers from all strategies

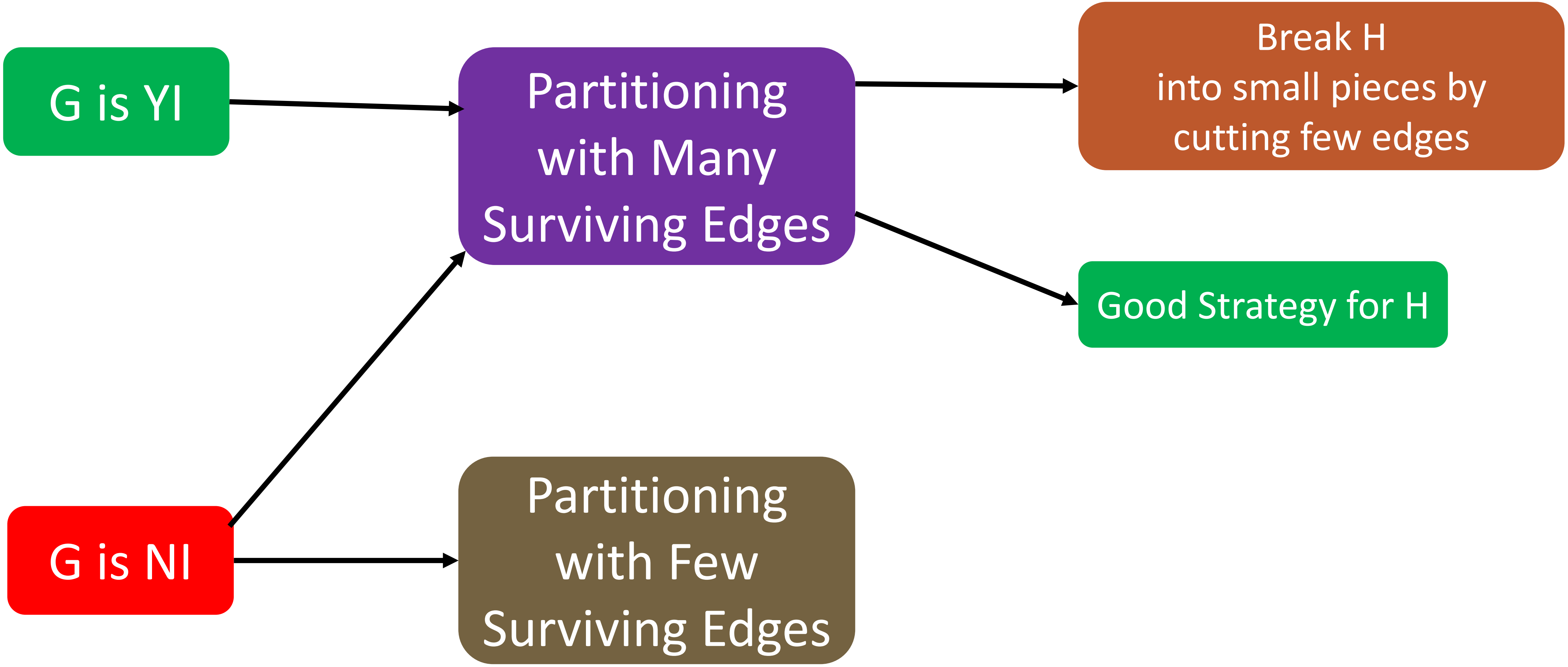
Partitioning Problem : Main Theorem



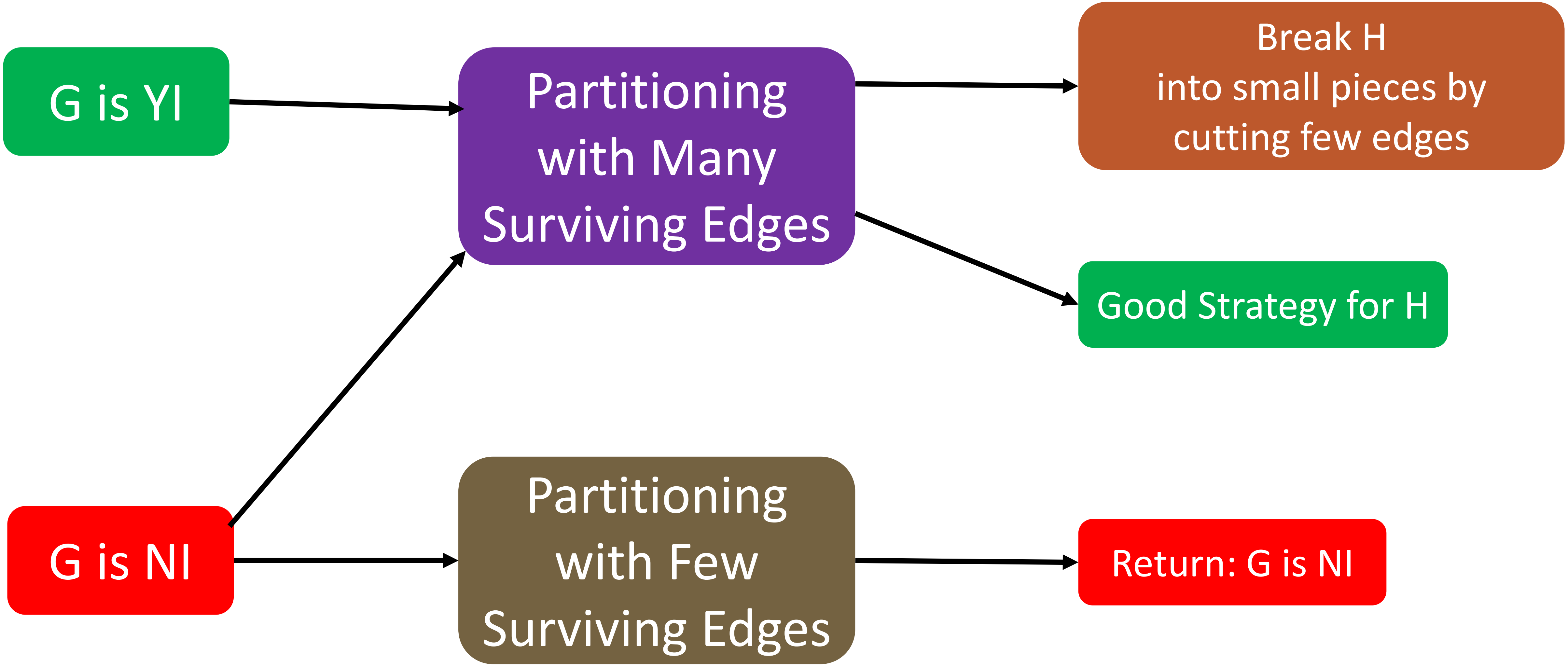
Partitioning Problem



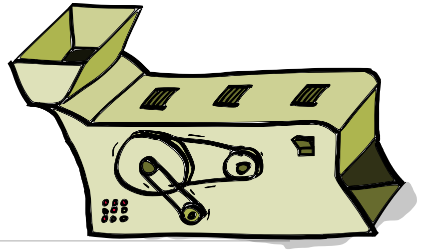
Partitioning Problem

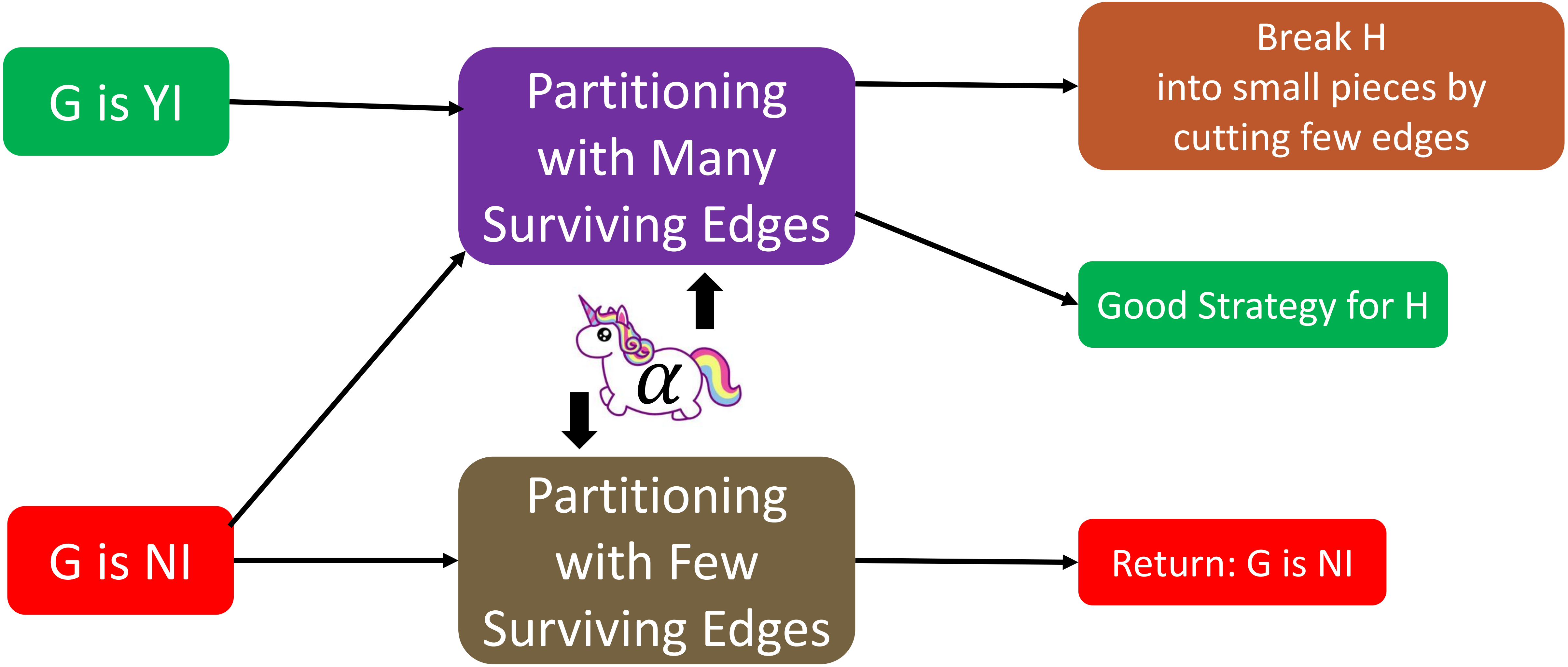


Partitioning Problem

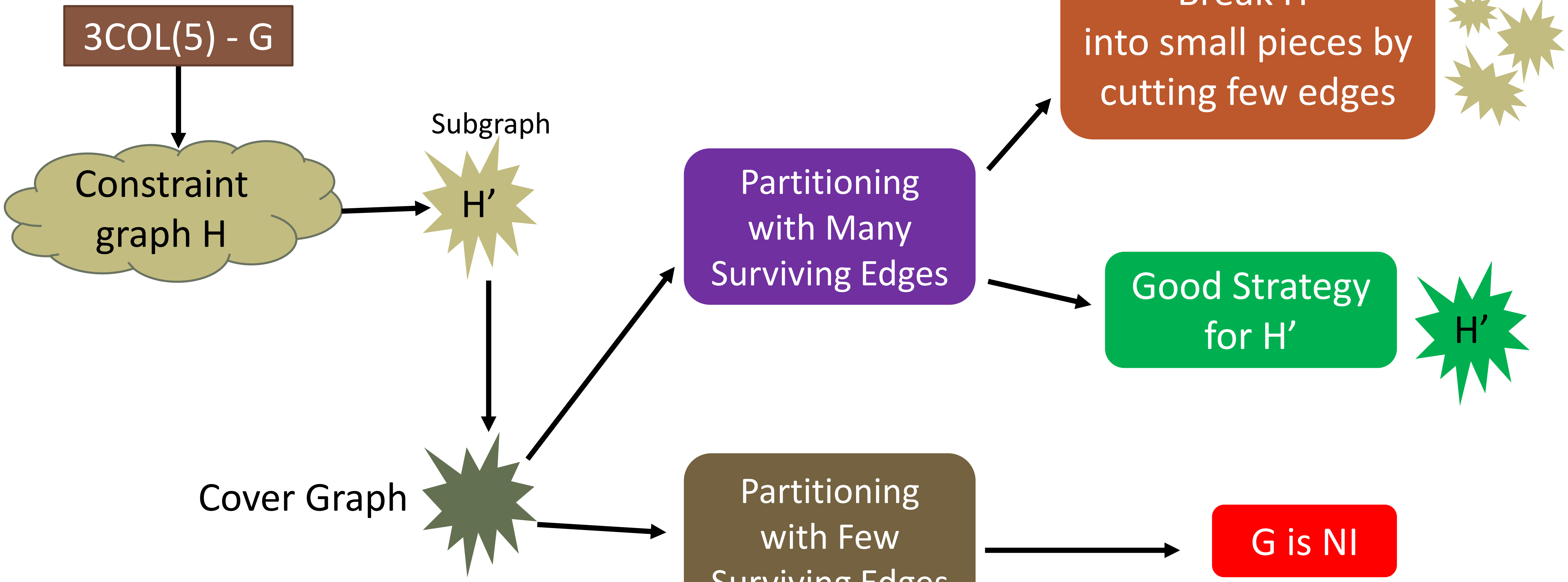


Partitioning Problem





The Core Algorithm



Result

Size of 3COL5: n

Parallel repetition parameter: l

Size of the constraint graph: $n^{O(l)}$

Result

Size of 3COL5: n

Parallel repetition parameter: l

Size of the constraint graph: $n^{O(l)}$

Setting parameters,

$2^{\log^{1-\epsilon} n}$ −hard assuming NP $\not\subseteq$ RTIME($n^{poly \log n}$)

$n^{\Omega(1/(\log\log n)^2)}$ −hard assuming NP $\not\subseteq$ RTIME($2^{n^{\delta}}$)

For all $\epsilon > 0$

For some $\delta > 0$

Conclusion

- Upper and lower bounds for both, general-NDP and NDP-Grids are now either polynomial or near polynomial
- Polynomial hardness for general-NDP?
- Congestion minimization?
  When paths are allowed to share nodes
- Can get something for Densest k-Subgraph from this approach?

Conclusion

- Upper and lower bounds for both, general-NDP and NDP-Grids are now either polynomial or near polynomial
- Polynomial hardness for general-NDP?
- Congestion minimization?
  When paths are allowed to share nodes
- Can get something for Densest k-Subgraph from this approach?

Thank You!