Almost Polynomial Hardness for Node-Disjoint Paths in Grids

Julia Chuzhoy
TTIC

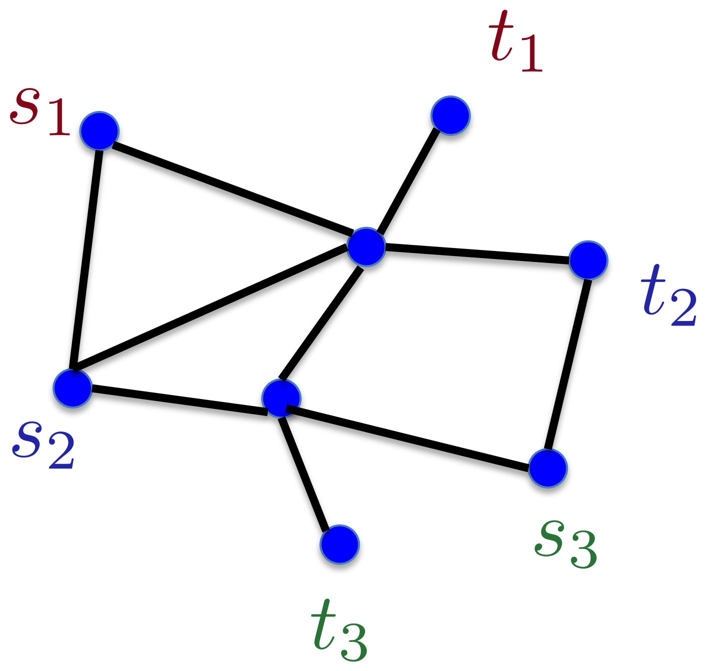
David Kim
U. of Chicago

Rachit Nimavat
TTIC

Node-Disjoint Paths (NDP)

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

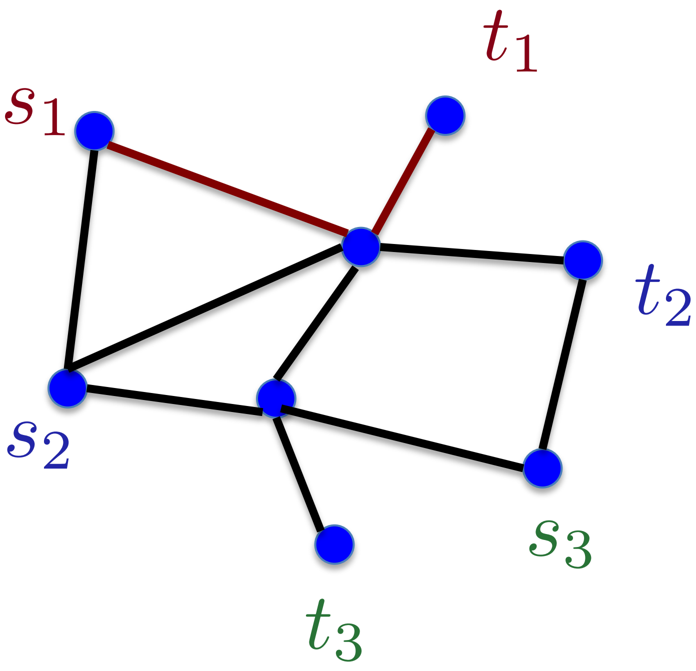
Goal: Route as many pairs as possible via node-disjoint paths



Node-Disjoint Paths (NDP)

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

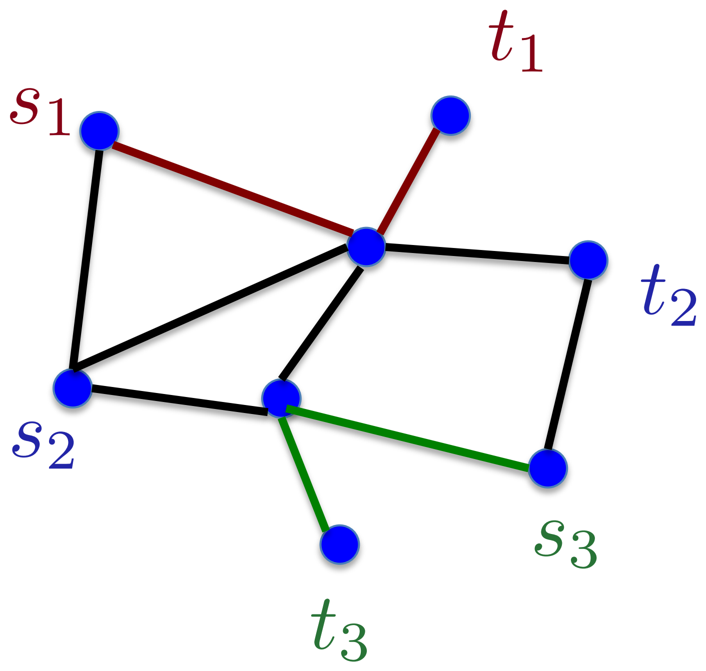
Goal: Route as many pairs as possible via node-disjoint paths



Node-Disjoint Paths (NDP)

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

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



Solution value: 2

Complexity of NDP

demand pairs

- Constant k: efficiently solvable [Robertson, Seymour '90]
- Running time: f(k)•n^2 [Kawarabayashi, Kobayashi, Reed]

$$f(k) = 2^{2^{2^{\cdot^{\cdot^{\cdot^{k}}}}}}$$

Complexity of NDP

- Constant k: efficiently solvable [Robertson, Seymour ’90]
- Running time: $f(k) \bullet n^2$ [Kawarabayashi, Kobayashi, Reed]
- NP-hard when k is part of input [Knuth, Karp ’72]

Approximation Algorithm?

Best Current Approximation Algorithm

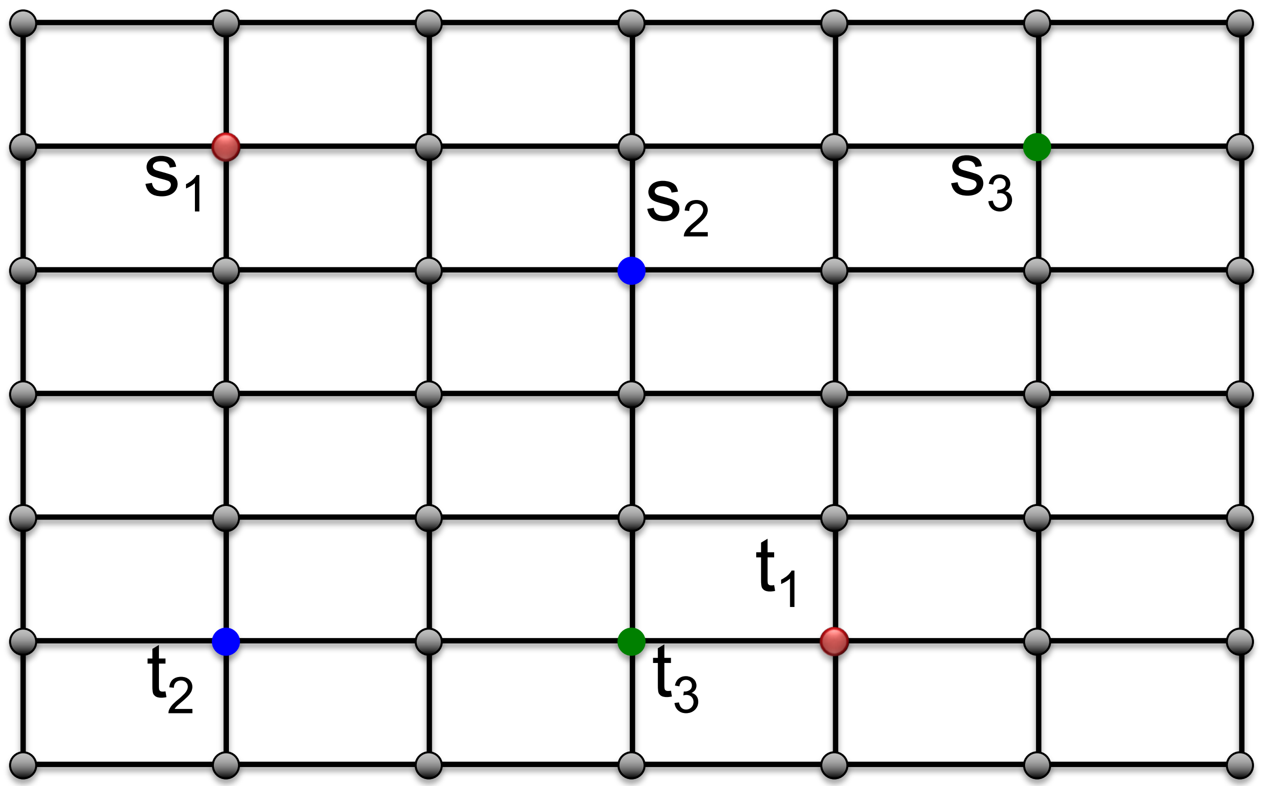
[Kolliopoulos, Stein ‘98]

- Choose a path P of minimum length connecting some demand pair
- Add P to the solution
- Delete vertices of P from the graph
- Repeat

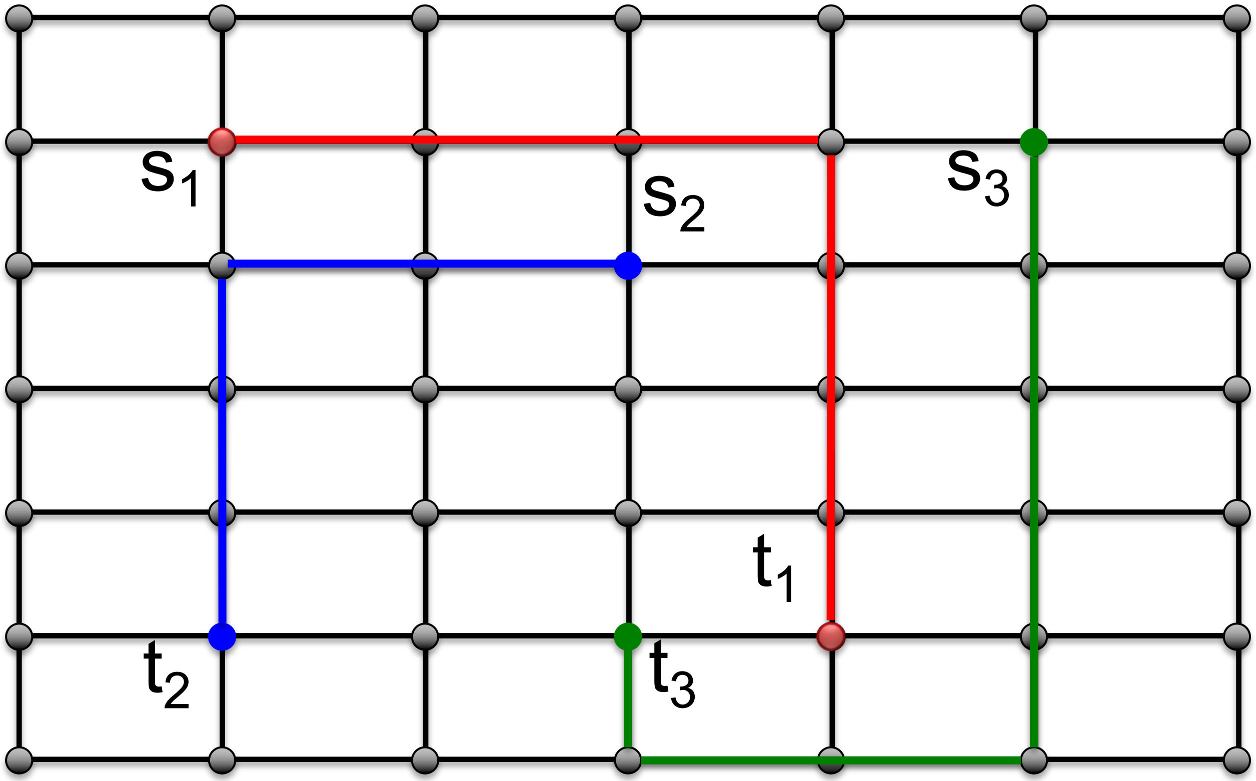
Until recently: nothing better even for planar graphs and grids!

$O(\sqrt{n})$-approximation

NDP in Grids



NDP in Grids



Approximation Status of NDP from 2015

- $O(\sqrt{n})$ -approximation algorithm
  - Even on planar graphs
  - Even on grid graphs

- $\Omega(\log^{1/2-\epsilon} n)$ -hardness of approximation for any ϵ [Andrews, Zhang '05], [Andrews, C, Guruswami, Khanna, Talwar, Zhang '10]

Approximation Status of NDP from 2015

- $O(\sqrt{n})$ -approximation algorithm
  - Even on planar graphs
  - Even on grid graphs

Plan:

- get polylog(n)-approximation for grids
- extend to planar graphs
- look into general graphs

- $\Omega(\log^{1/2-\epsilon} n)$ -hardness of approximation for any ϵ [Andrews, Zhang '05], [Andrews, C, Guruswami, Khanna, Talwar, Zhang '10]

Approximation Status of NDP from 2015

- $O(\sqrt{n})$ -approximation alg
  - Even on planar graphs
  - Even on grid graphs

$\tilde{O}(n^{9/19})$ -approximation [C, Kim, Li '16]

$\tilde{O}(n^{1/4})$-approximation [C, Kim '15]

$2^{O(\sqrt{\log n})}$-approximation for grids with sources on boundary [C, Kim, Nimavat '17]

- $\Omega(\log^{1/2-\epsilon} n)$-hardness of approximation for any ϵ [Andrews, Zhang '05], [Andrews, C, Guruswami, Khanna, Talwar, Zhang '10]

Approximation Status of NDP from 2015

- $O(\sqrt{n})$ -approximation alg
  - Even on planar graphs
  - Even on grid graphs

$\tilde{O}(n^{9/19})$ -approximation [C, Kim, Li '16]

$\tilde{O}(n^{1/4})$-approximation [C, Kim '15]

$2^{O(\sqrt{\log n})}$-approximation for grids with sources on boundary [C, Kim, Nimavat '17]

- $\Omega(\log^{1/2-\epsilon} n)$-hardness of approximation for any ϵ [Andrews, Zhang '05], [Andrews, C, Guruswami, Khanna, Talwar, Zhang '10]
- $2^{\Omega(\sqrt{\log n})}$ -hardness of approximation for subgraphs of grids with all sources on boundary [C, Kim, Nimavat '17]
- Almost polynomial hardness for grids [C, Kim, Nimavat '18]

Approximation Status of NDP from 2015

- $O(\sqrt{n})$ -approximation alg
  - Even on planar graphs
  - Even on grid graphs

$\tilde{O}(n^{9/19})$ -approximation [C, Kim, Li '16]

$\tilde{O}(n^{1/4})$-approximation [C, Kim '15]

$2^{O(\sqrt{\log n})}$-approximation for grids with sources on boundary [C, Kim, Nimavat '17]

- $\Omega(\log^{1/2-\epsilon} n)$-hardness of approximation for any ϵ [Andrews, Zhang '05], [Andrews, C, Guruswami, Khanna, Talwar, Zhang '10]
- $2^{\Omega(\sqrt{\log n})}$ -hardness of approximation for subgraphs of grids with all sources on boundary [C, Kim, Nimavat '17]
- Almost polynomial hardness for grids [C, Kim, Nimavat '18]

Almost polynomial Hardness

Hardness of NDP on grids: [C, Kim,

unless

$\mathrm{NP} \subseteq \mathrm{RTIME}(n^{\mathrm{poly}\log n})$

- $2^{(\log n)^{1-\epsilon}}$-hardness for any constant ϵ
- $n^{\Omega(1/(\log\log n)^2)}$-hardness

Almost polynomial Hardness

Hardness of NDP on grids: [C, K

- $2^{(\log n)^{1-\epsilon}}$-hardness for any constant ϵ
- $n^{\Omega(1/(\log\log n)^2)}$-hardness

unless every problem in NP has randomized quasi-poly-time algorithm

under randomized ETH
(need almost exponential time to solve SAT by randomized alg)

Almost polynomial Hardness

unless

$\mathrm{NP} \subseteq \mathrm{RTIME}(n^{\mathrm{poly}\log n})$

Hardness of NDP on grids: [C

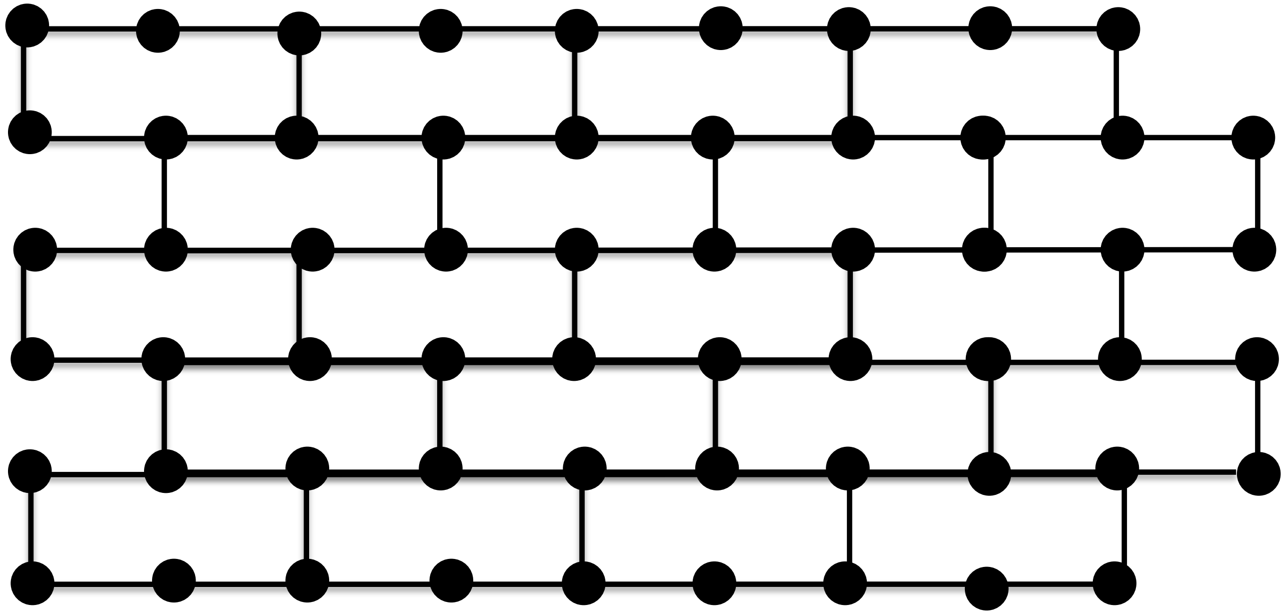
- $2^{(\log n)^{1-\epsilon}}$-hardness for any constant ϵ
- $n^{\Omega(1/(\log\log n)^2)}$-hardness

unless for every δ
$\mathrm{NP} \subseteq \mathrm{RTIME}(2^{n^\delta})$

Edge-Disjoint Paths (EDP)

- Like NDP, only the paths must be disjoint in their edges, may share vertices.
- Same upper/lower bounds as NDP
  - $O(\sqrt{n})$-approximation algorithm [Chekuri, Khanna, Shepherd '06]
  - $\Omega(\log^{1/2-\epsilon} n)$ -hardness of approximation for any ϵ [Andrews, Zhang '05], [Andrews, C, Guruswami, Khanna, Talwar, Zhang '10]
- But: constant approximation on grids [Aumann, Rabani '95], [Kleinberg Tardos '95], [Kleinberg, Tardos '98]

A Wall



All current upper/lower bounds for NDP in grids carry over to EDP in walls

What if we allow paths to share edges/vertices?

routing with congestion

Routing with Congestion

- Congestic
  approxim
- Congestic ...egev '01],
  [Baveja, Srini
- Congestion poly(log log n): polylog(n)-approx
  [Andrews '10
- Congestic ...bayashi,
  Kobayashi '1
- Congestion 14: polylog(k)-approximation [C, '11]
- Congestion 2: polylog(k)-approximation [C, Li '12]
- polylog(k)-approximation for NDP with congestion 2 [Chekuri, Ene '12], [Chekuri, C '16]

If up to 2 paths are allowed to share a vertex/an edge, can get polylog(k)-approximation

Big difference between routing with congestion 1 and 2.

Node-Disjoint Paths in Grid Graphs:

Hardness of Approximation

Main Idea 1

- Define an intermediate graph partitioning problem

Weird Graph
Partitioning Problem
(WGP)

Main Idea 1

- Define an intermediate graph partitioning problem

Weird Graph
Partitioning Problem
(WGP)

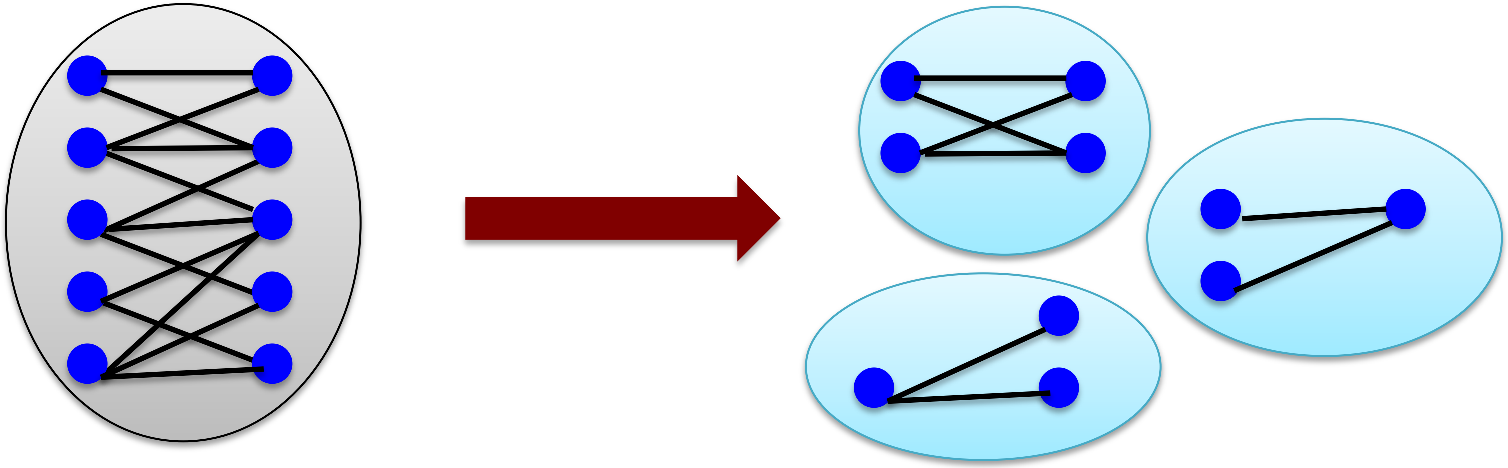
- Prove that NDP in grids is at least as hard as WGP
- Prove hardness of WGP

Weird Graph Partitioning Problem (WGP)

Weird Graph Partitioning Problem (WGP)

- Input: bipartite graph G=(V,E), integers p, L.
- Output:
  - partition G into p vertex-induced subgraphs.

pieces



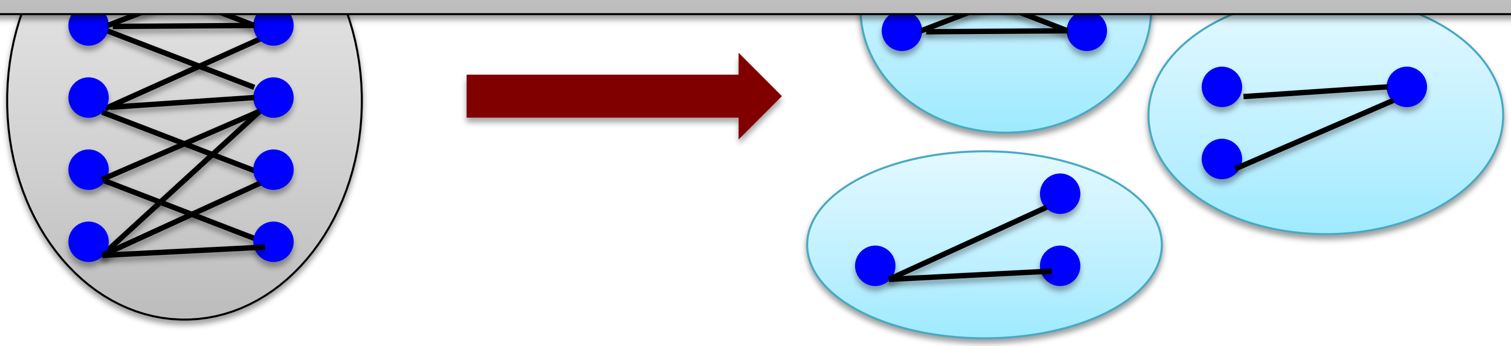
Weird Graph Partitioning Problem (WGP)

Intuition:

- Want to maximize the total number of edges that are not cut
- Don't want one piece to contain all the edges; want a balanced distribution of edges.

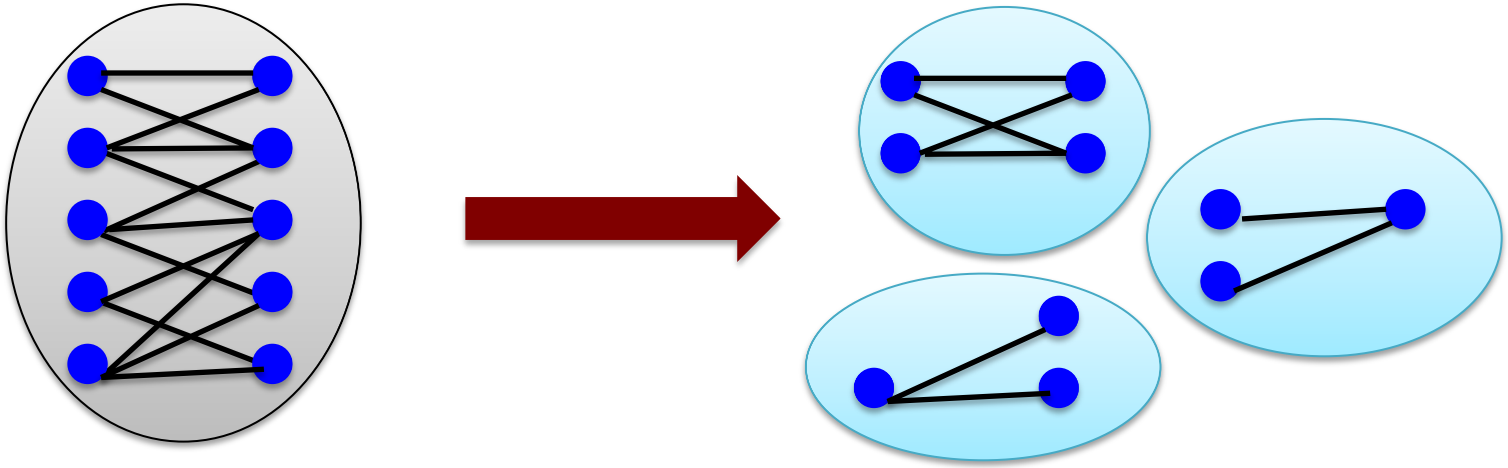
Solution:

- Will count at most L edges from each piece towards the solution



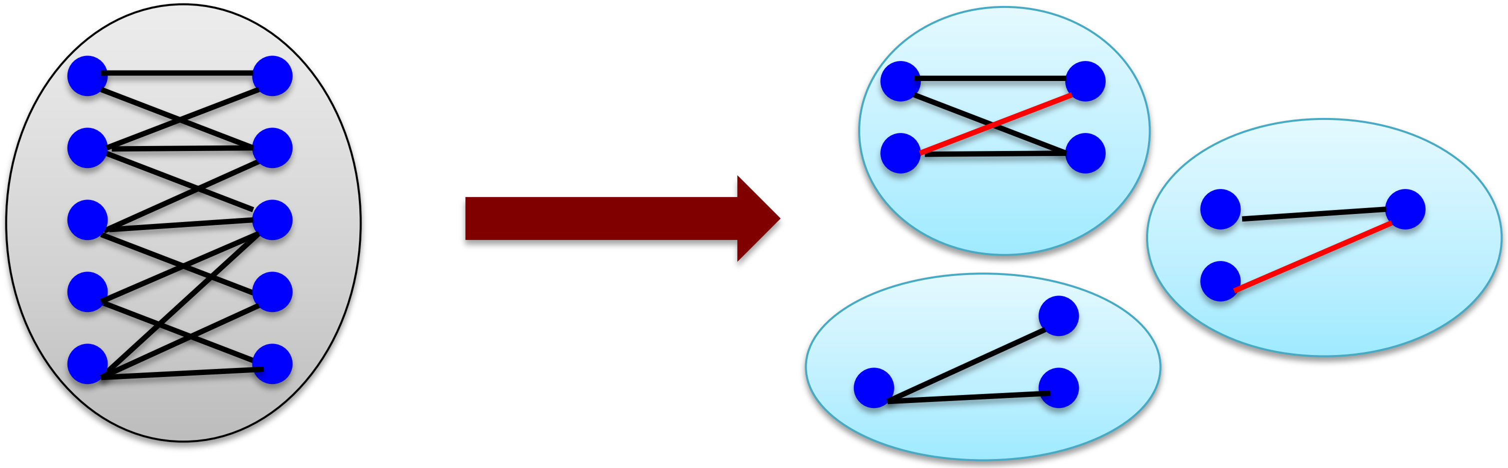
Weird Graph Partitioning Problem (WGP)

- Input: bipartite graph G=(V,E), integers p, L.
- Output:
  - partition G into p vertex-induced subgraphs.
  - for each subgraph G_i, select a subset E_i of at most L edges



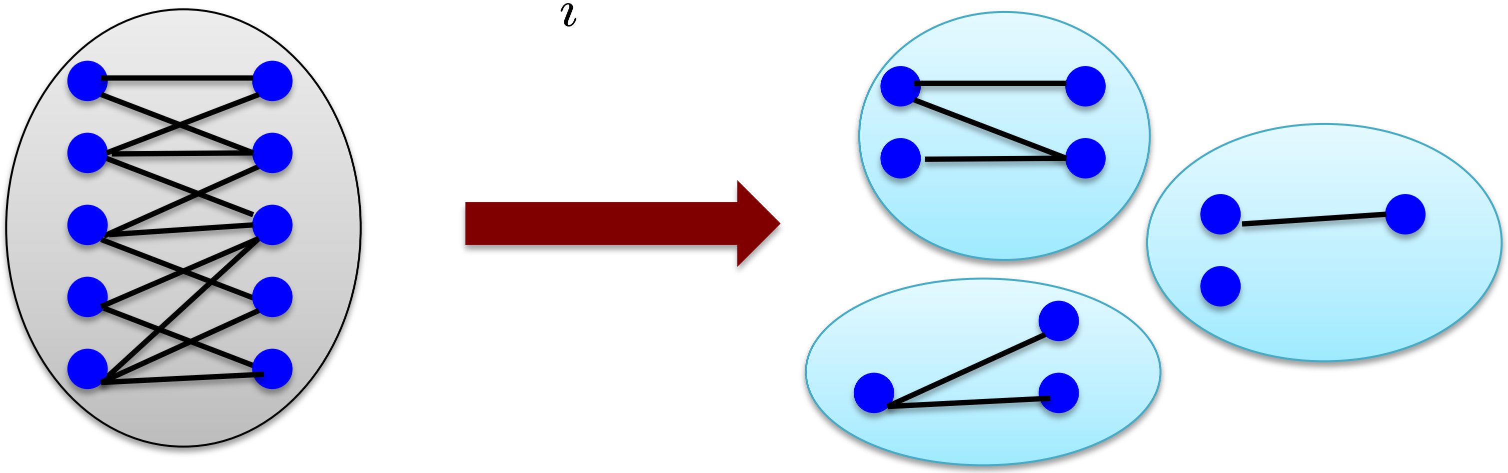
Weird Graph Partitioning Problem (WGP)

- Input: bipartite graph G=(V,E), integers p, L.
- Output:
  - partition G into p vertex-induced subgraphs.
  - for each subgraph G_i, select a subset E_i of at most L edges



Weird Graph Partitioning Problem (WGP)

- Input: bipartite graph G=(V,E), integers p, L.
- Output:
  - partition G into p vertex-induced subgraphs.
  - for each subgraph G_i, select a subset E_i of at most L edges
  - Goal: maximize $\sum_i |E_i|$



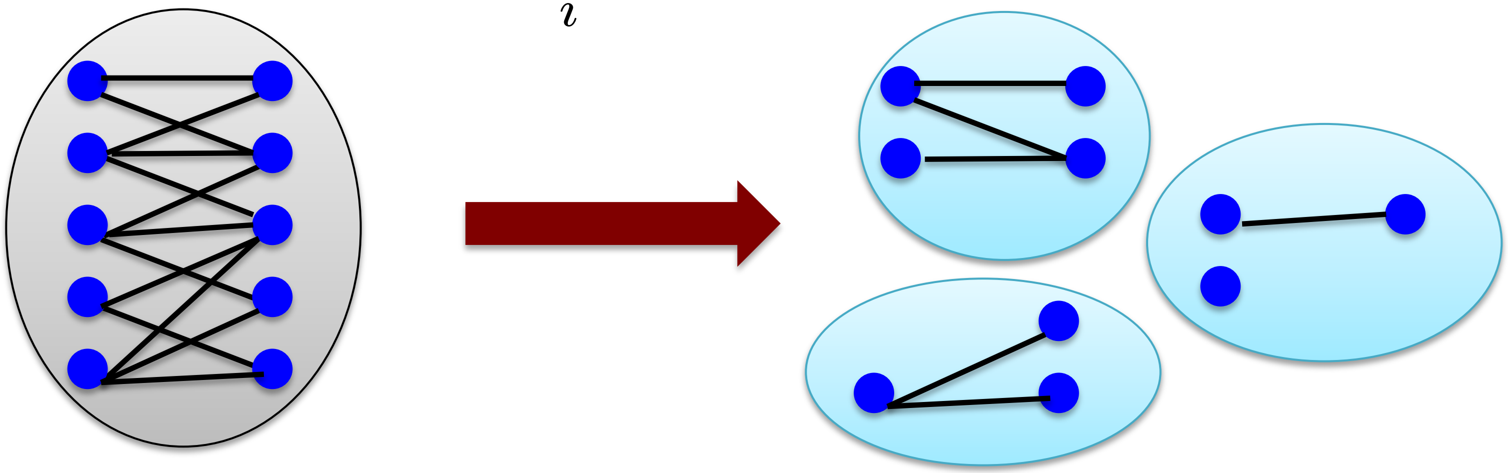
Weird Graph Partitioning Problem (WGP)

Intuitive View 1: Balanced Cut

Except:

- Partition into p and not 2 pieces
- Maximize # of surviving edges.

– Goal: maximize $\sum_i |E_i|$



Weird Graph Partitioning Problem (WGP)

Intuitive View 2: Densest k-Subgraph

– Goal: maximize $\sum |E_i|$

Densest k-subgraph:

Input: graph G, integer k

Output: subgraph G’ of G on k vertices

Goal: maximize |E(G’)|

On Densest k-Subgraph

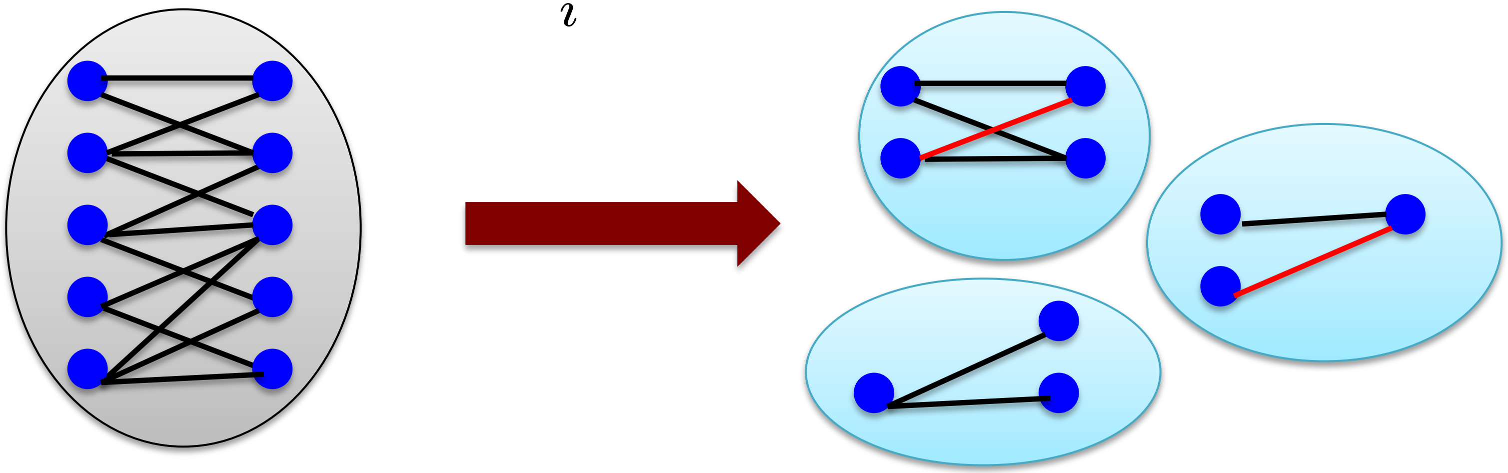
- $O(n^{1/4})$-approximation [Bhaskara, Charikar, Chlamtac, Feige, Vijayaraghavan '10]
- Notoriously hard to prove hardness of approximation
  - APX-hardness [Khot, '06]
  - Constant hardness assuming small-set-expansion conjecture [Raghavendra, Steurer '10]
  - Hardness results based on average-case complexity assumption of SAT of Feige [Alon, Arora, Manokaran, Moshkovitz, Weinstein '11]
  - Almost polynomial hardness using Exponential Time Hypothesis [Manurangsi '16]

Weird Graph Partitioning Problem (WGP)

Intuitive View 2: Densest k-Subgraph

Except:

- Want p dense subgraphs and not one

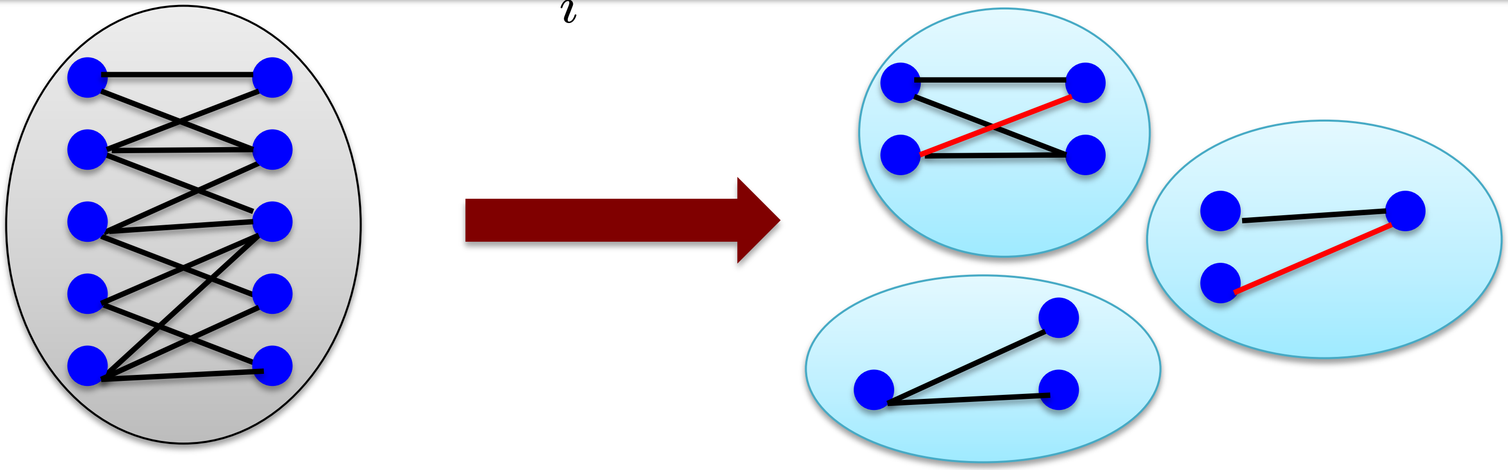


Weird Graph Partitioning Problem (WGP)

Plan:

1. NDP in grids is at least as hard as WGP
2. Prove hardness of WGP

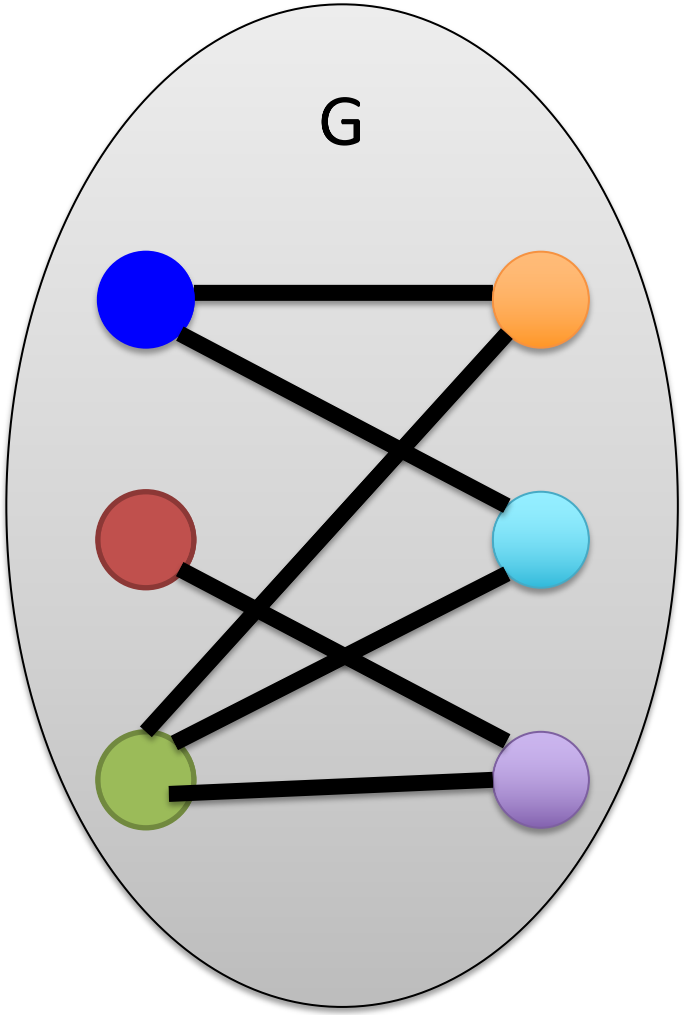
$\imath$



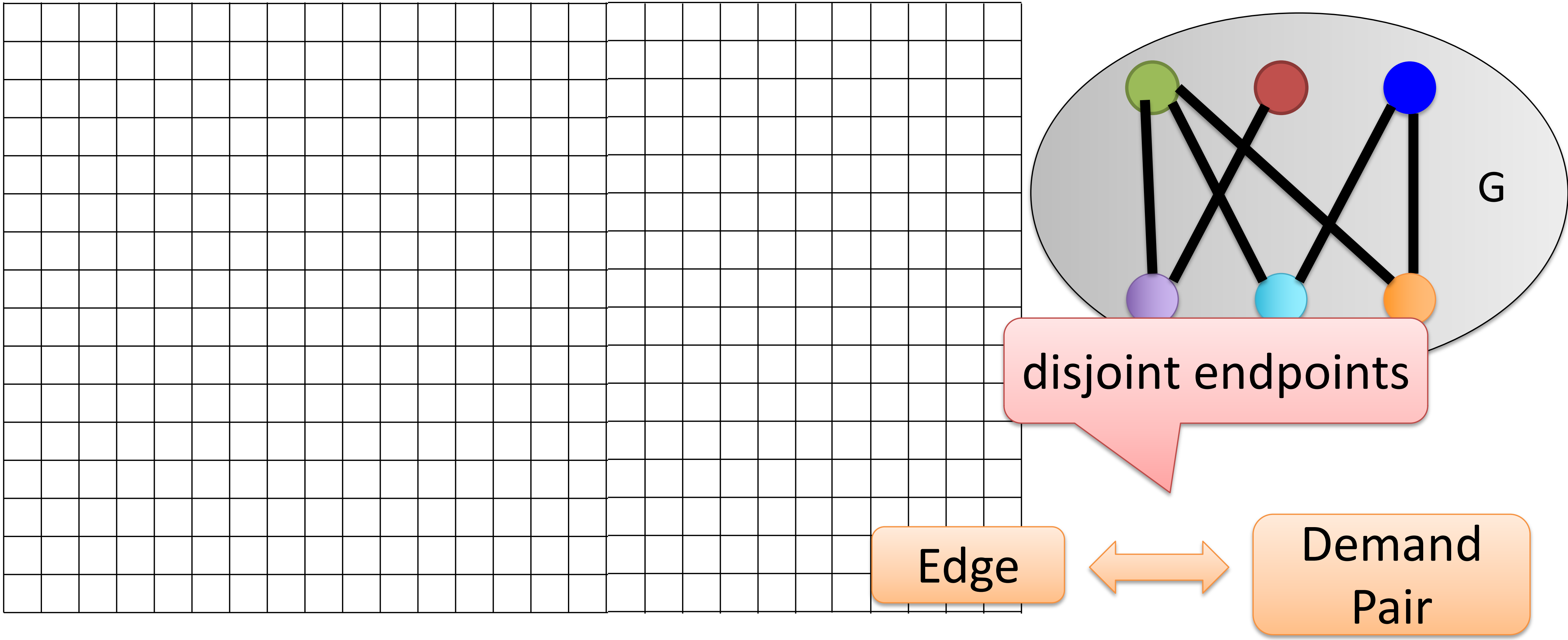
Part 1: NDP in Grids is at Least as Hard as WGP

(up to polylog n factor)

The Reduction

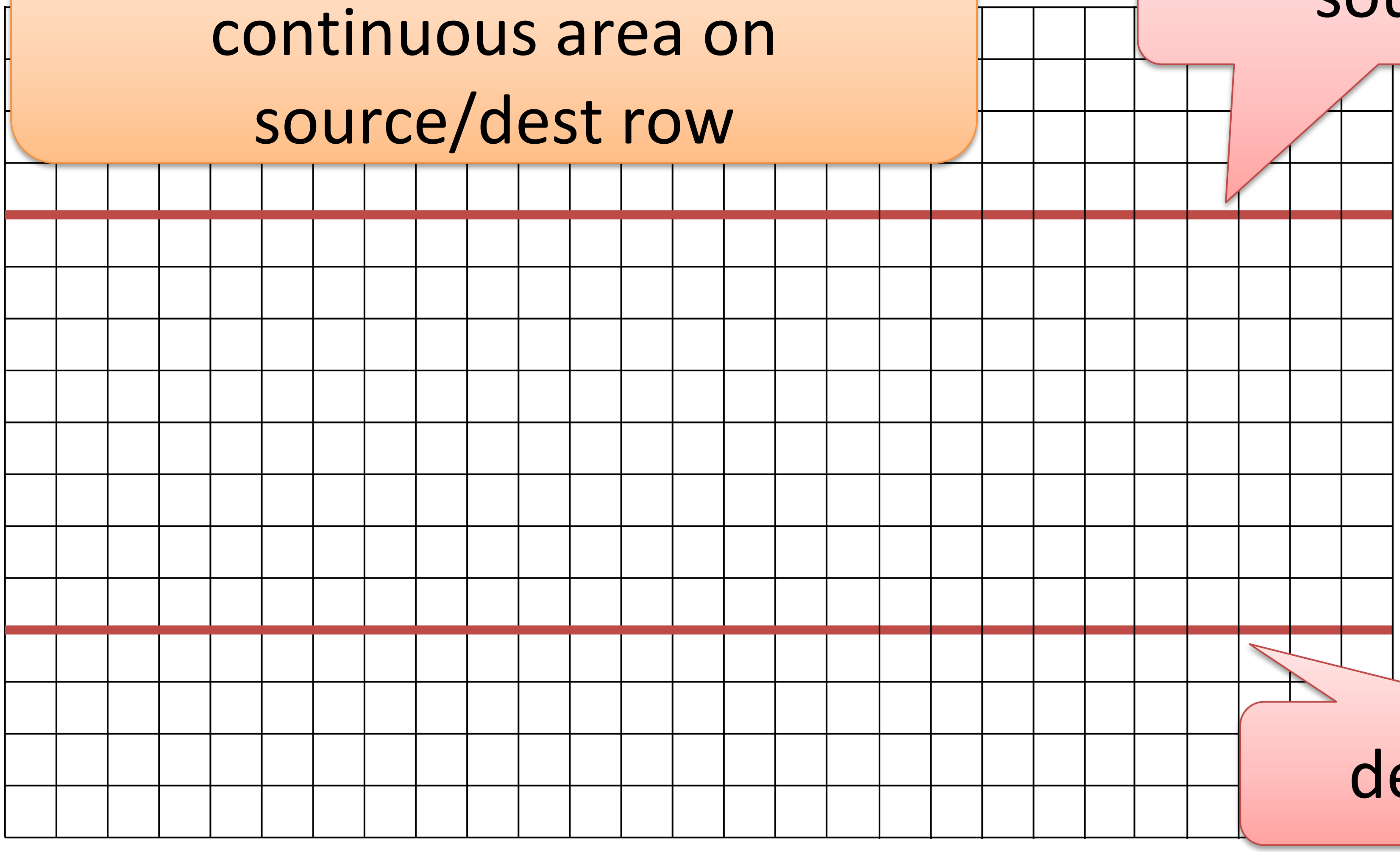


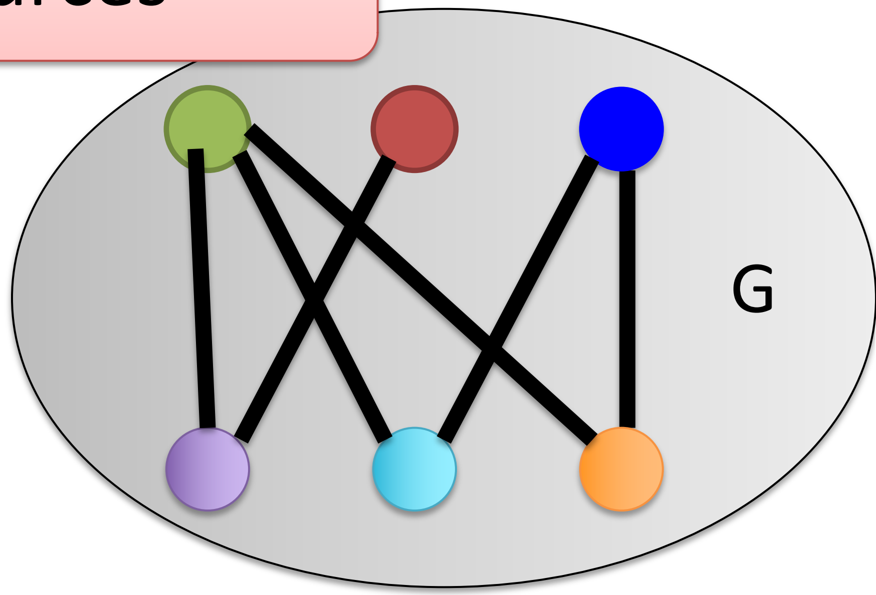
The Reduction



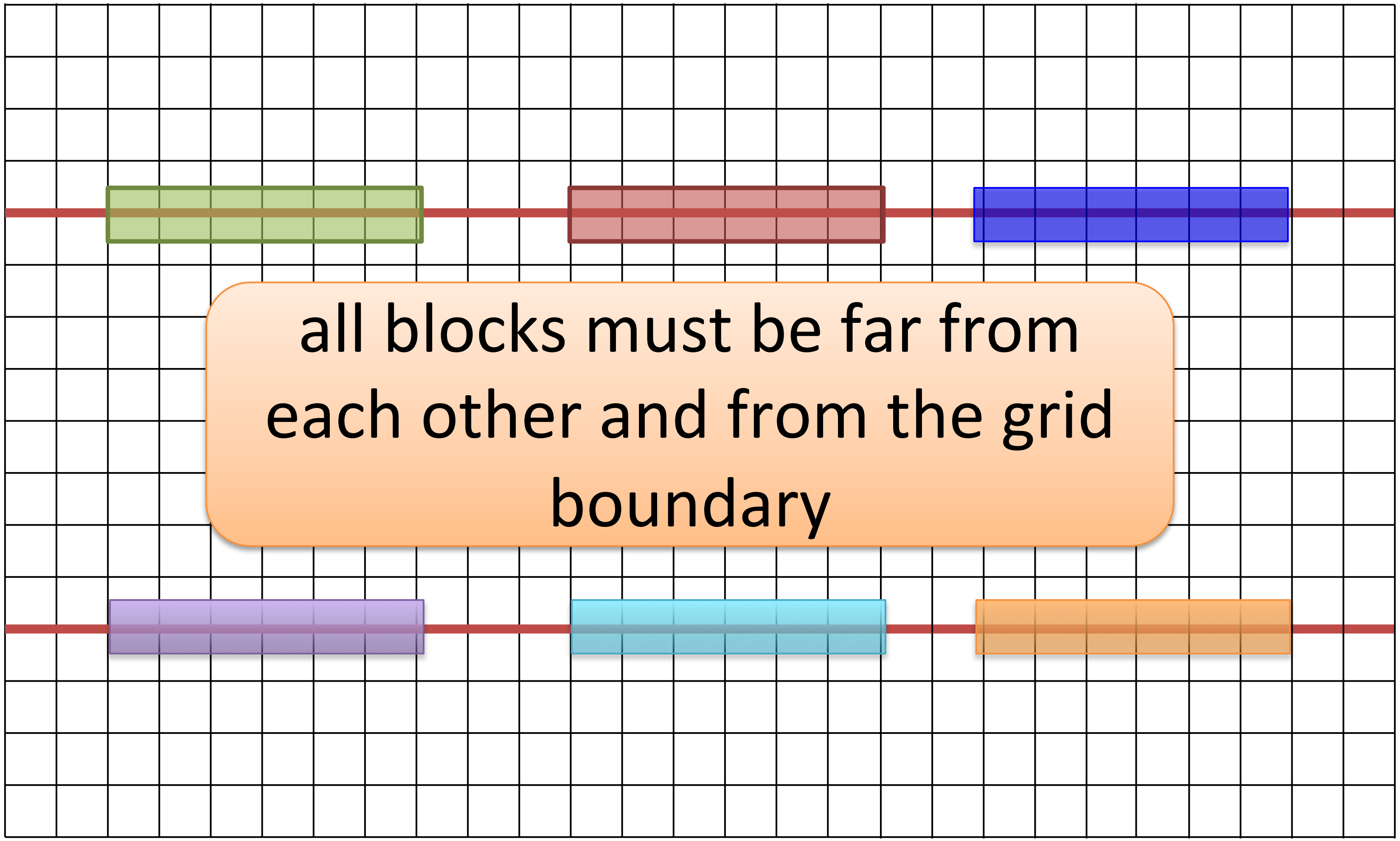
The Reduction

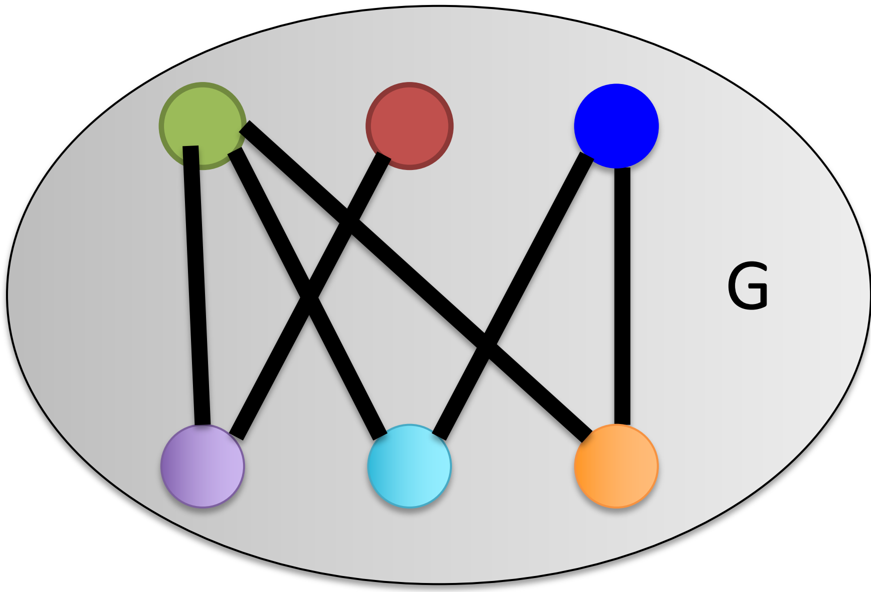
for every vertex, choose a continuous area on source/dest row



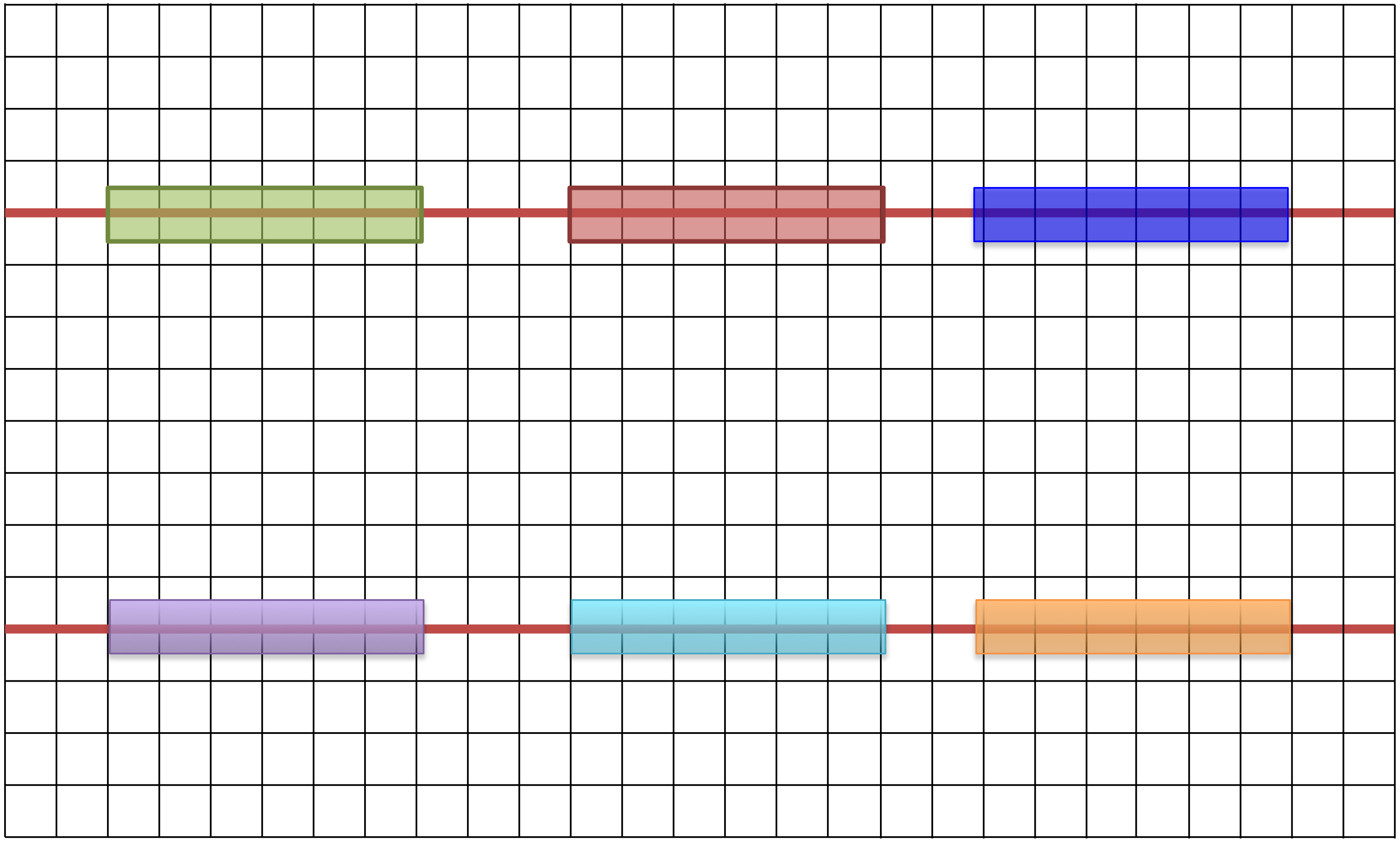


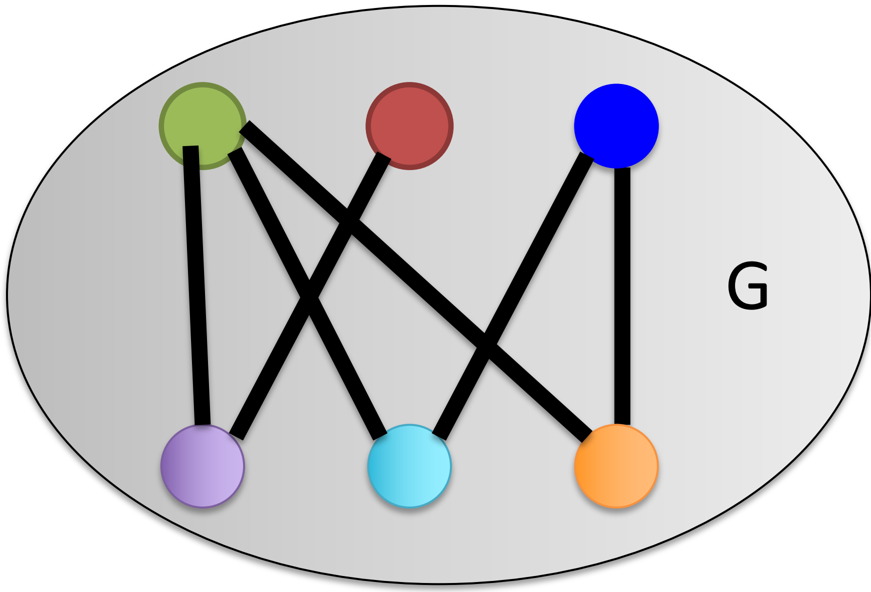
The Reduction



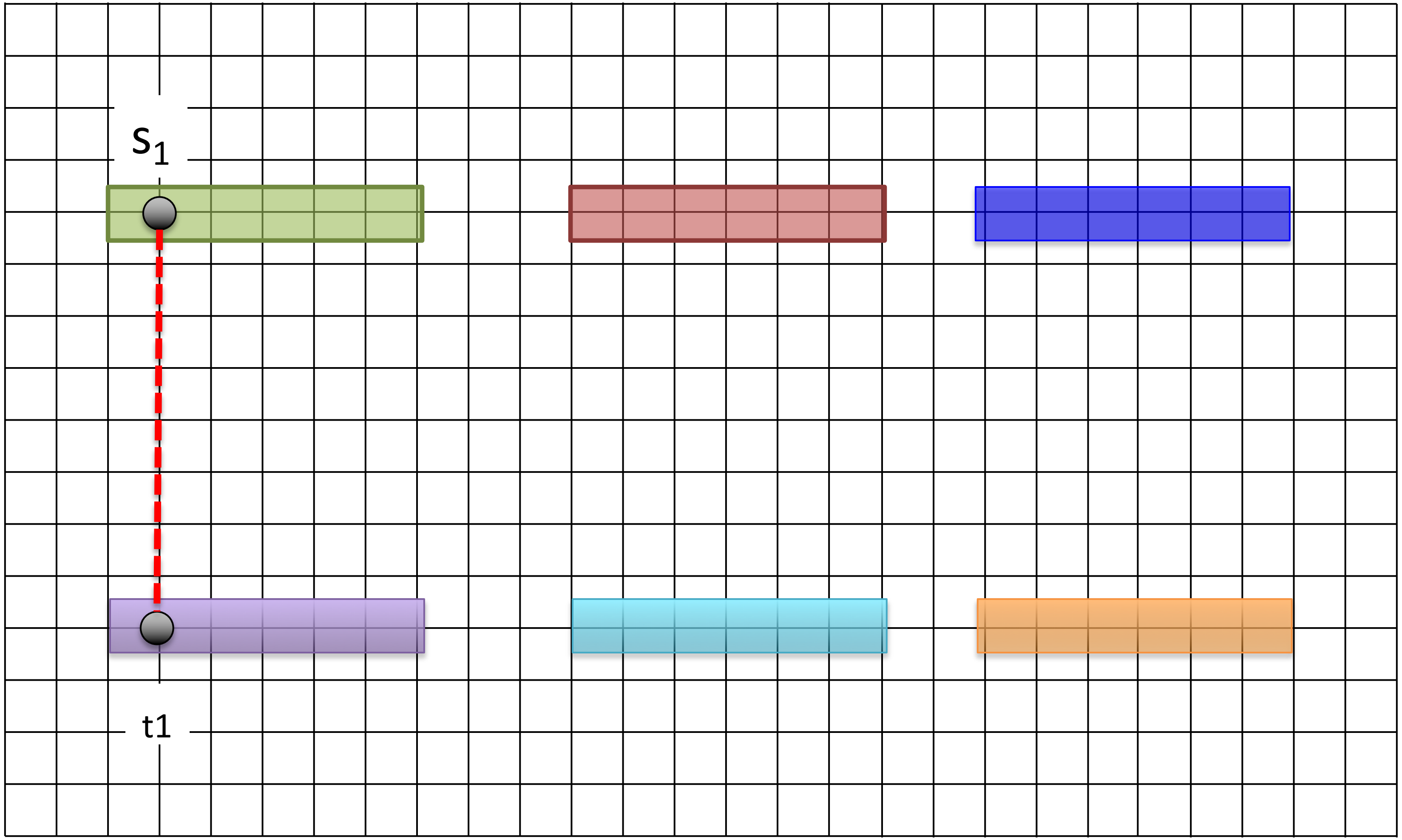


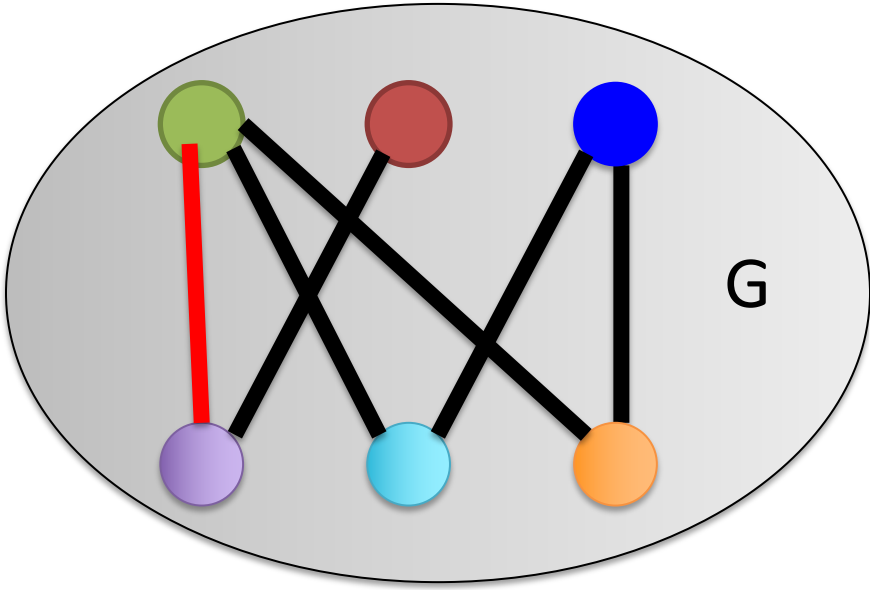
The Reduction



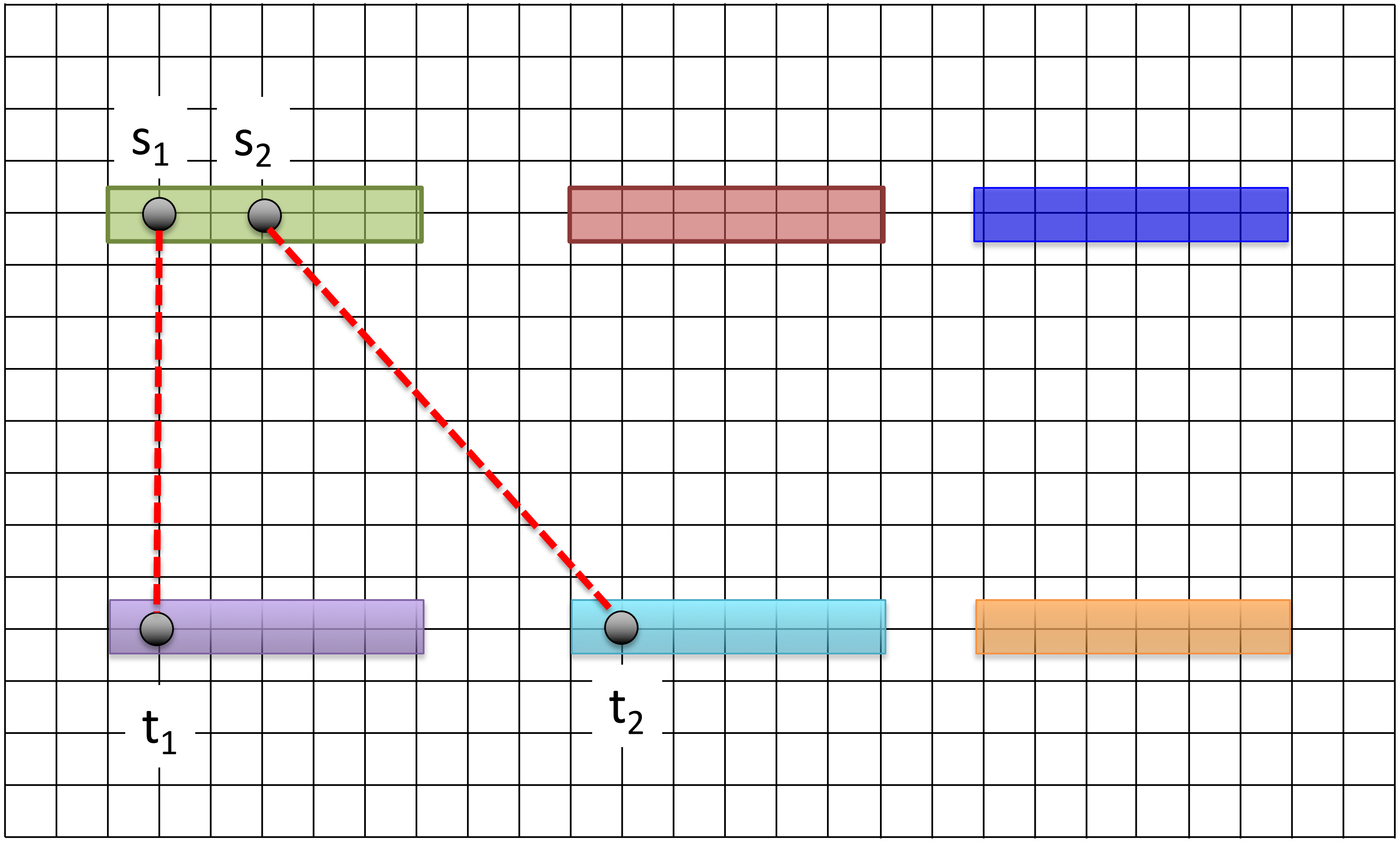


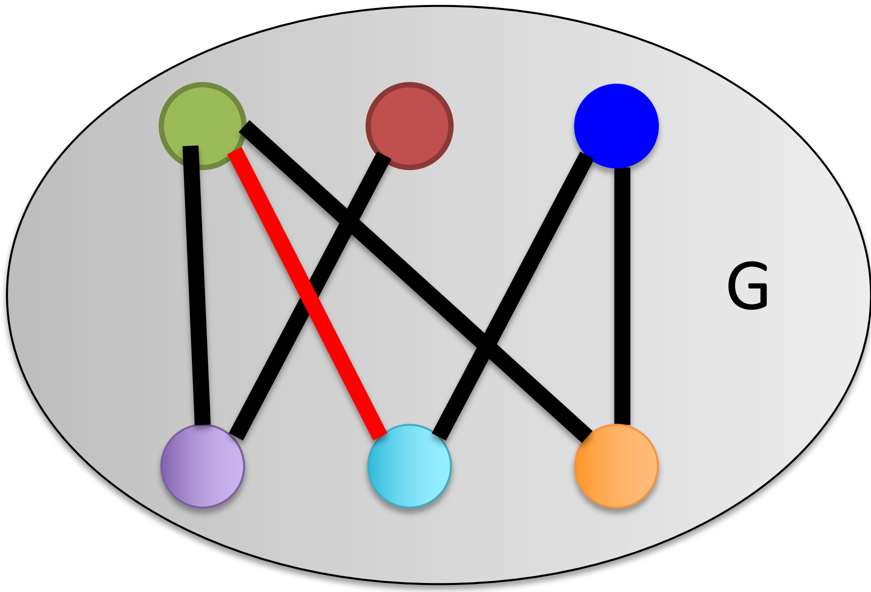
The Reduction



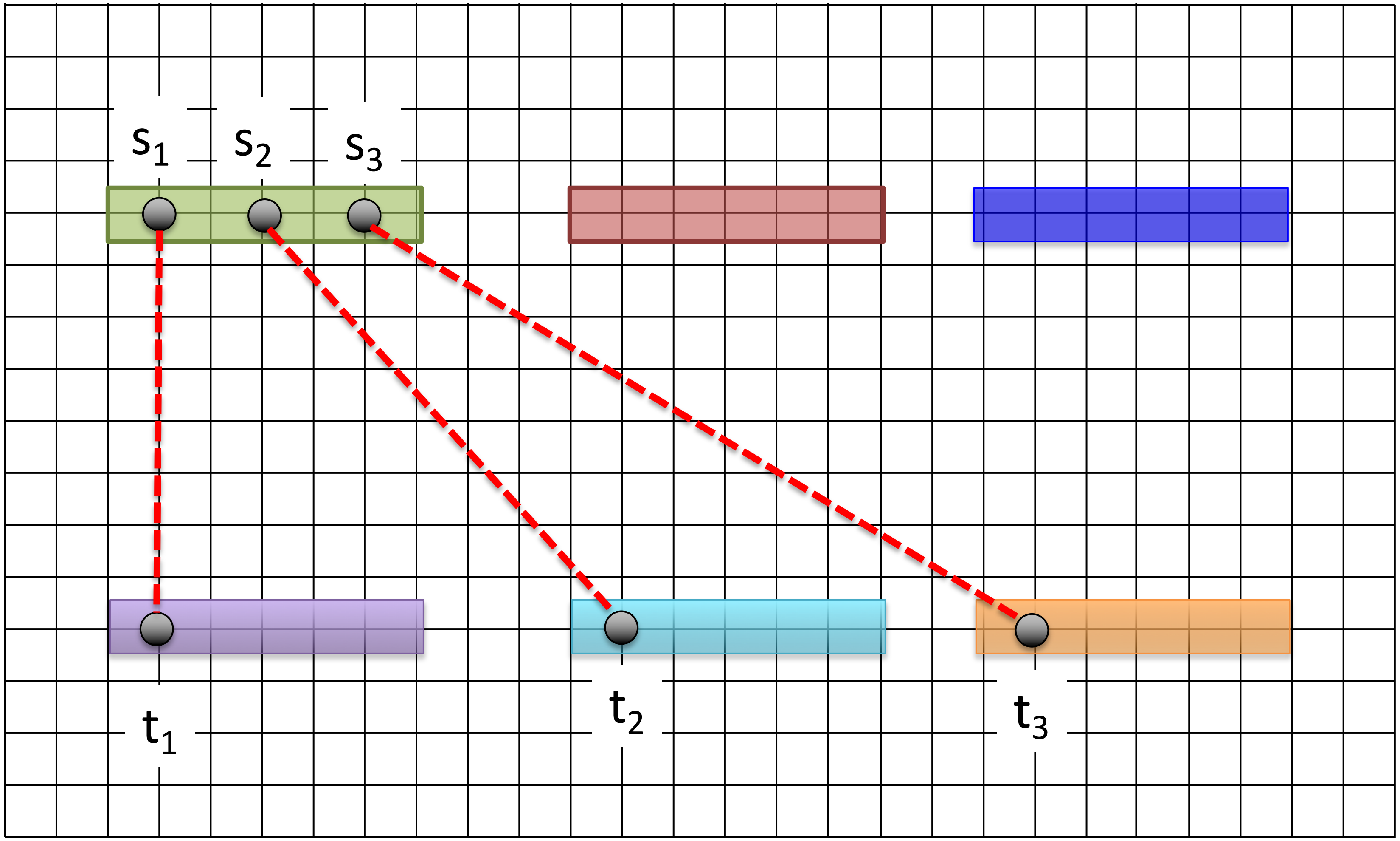


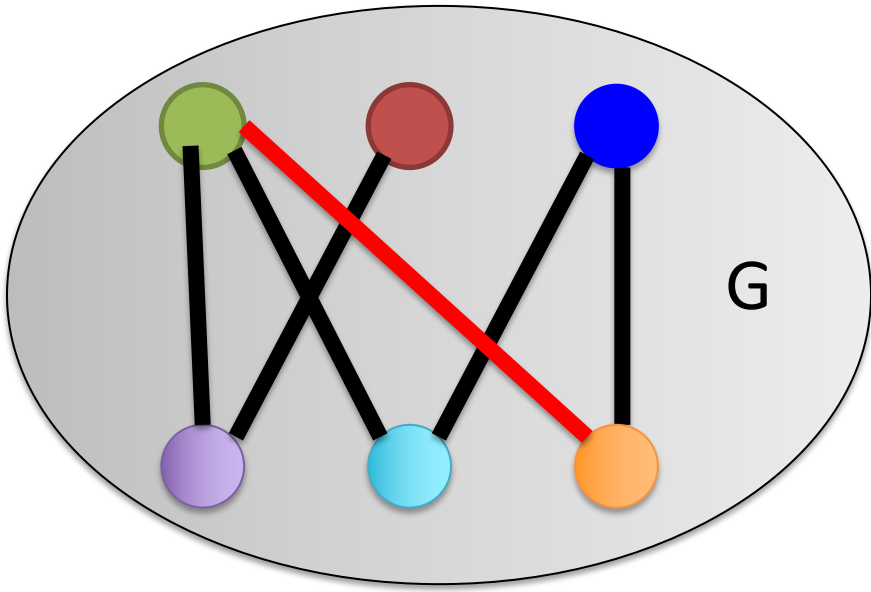
The Reduction



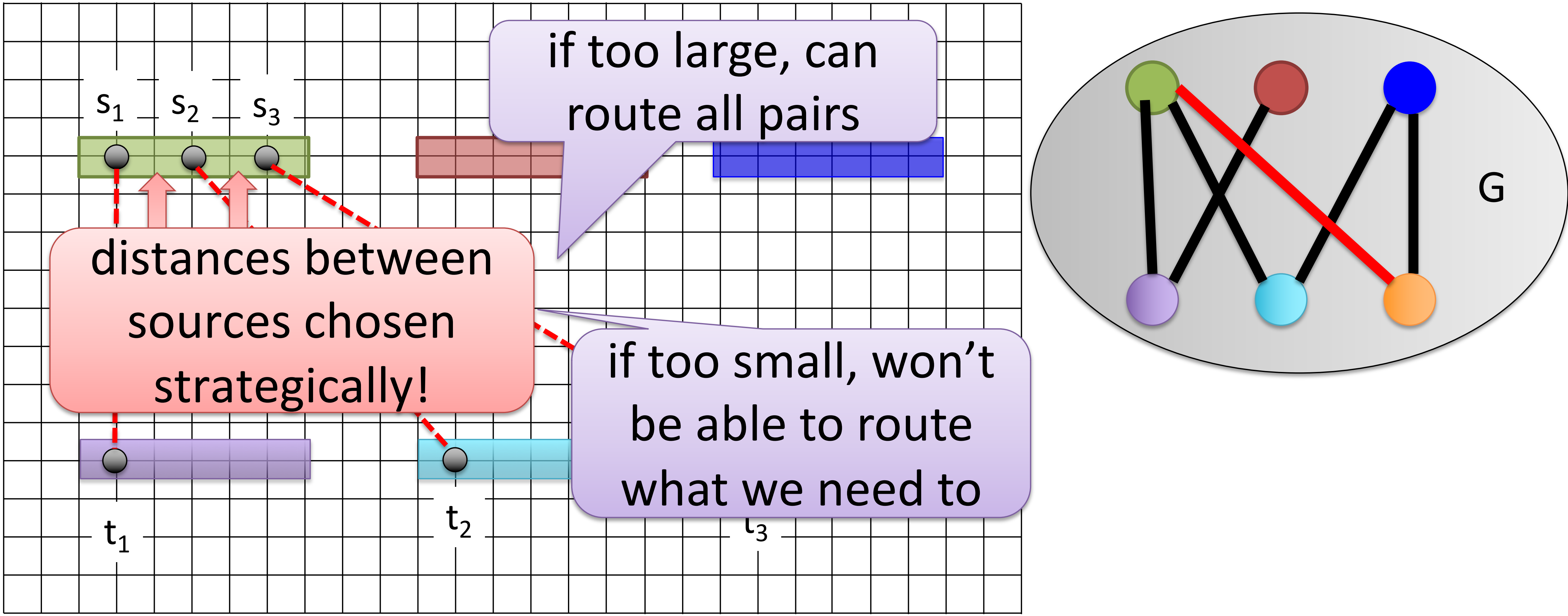


The Reduction

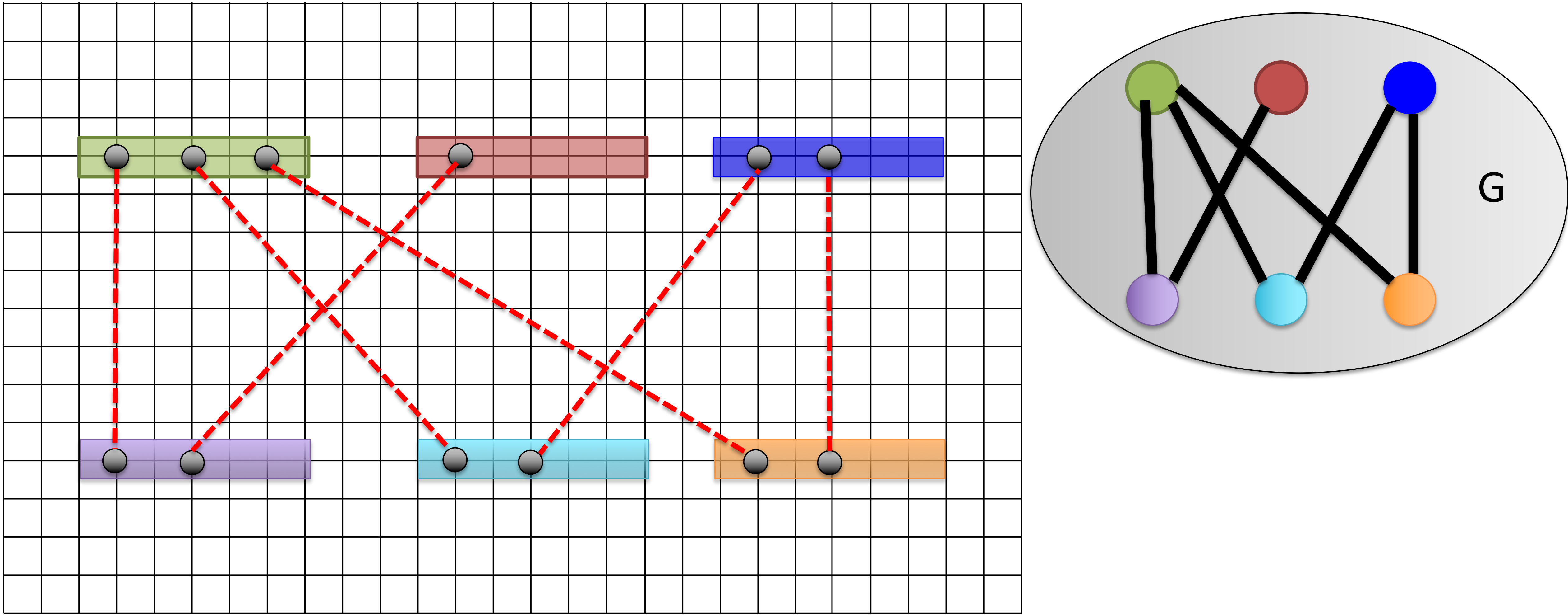




The Reduction



The Reduction



The Reduction

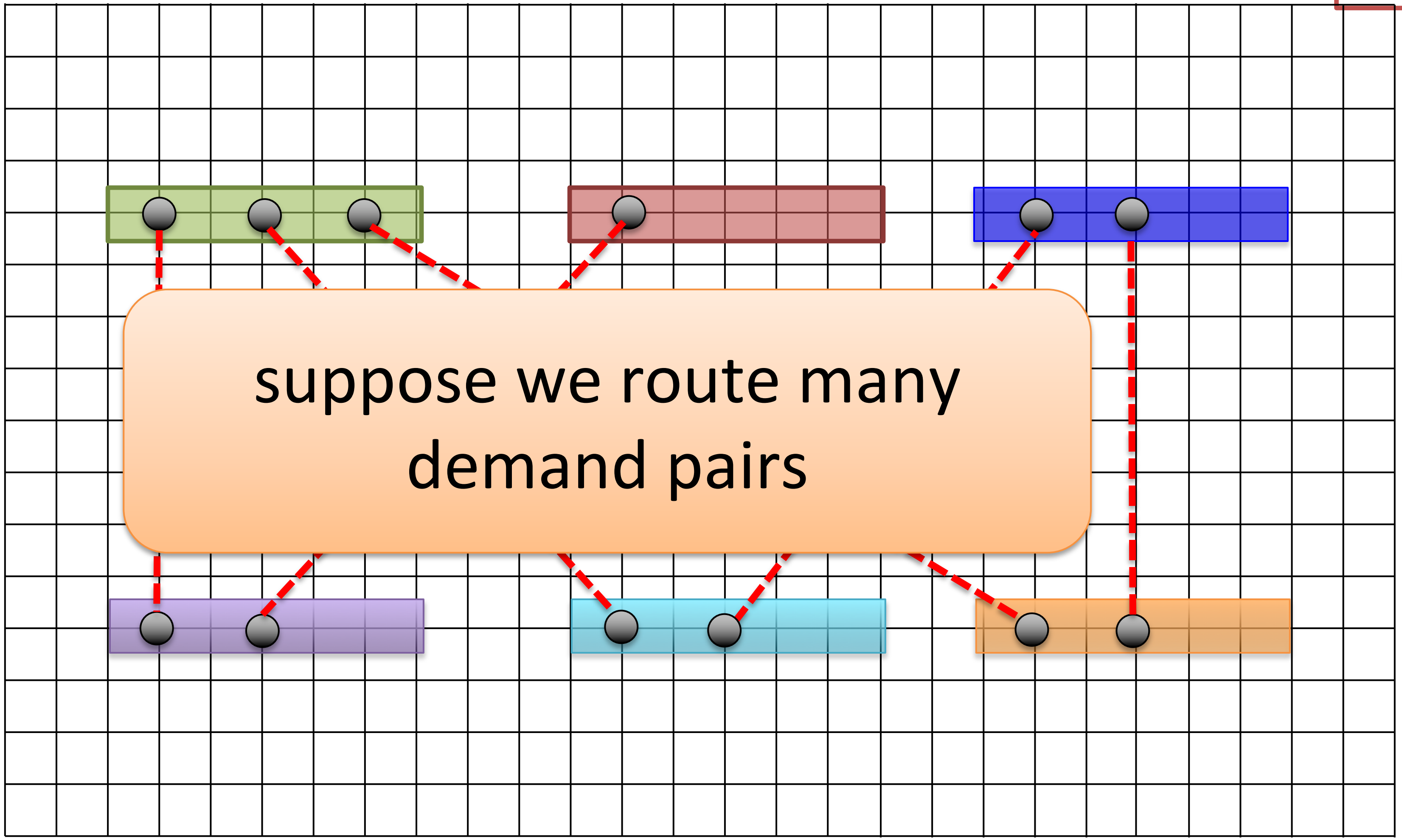
Claim: the reduction preserves solution value to within polylog(n) factor!

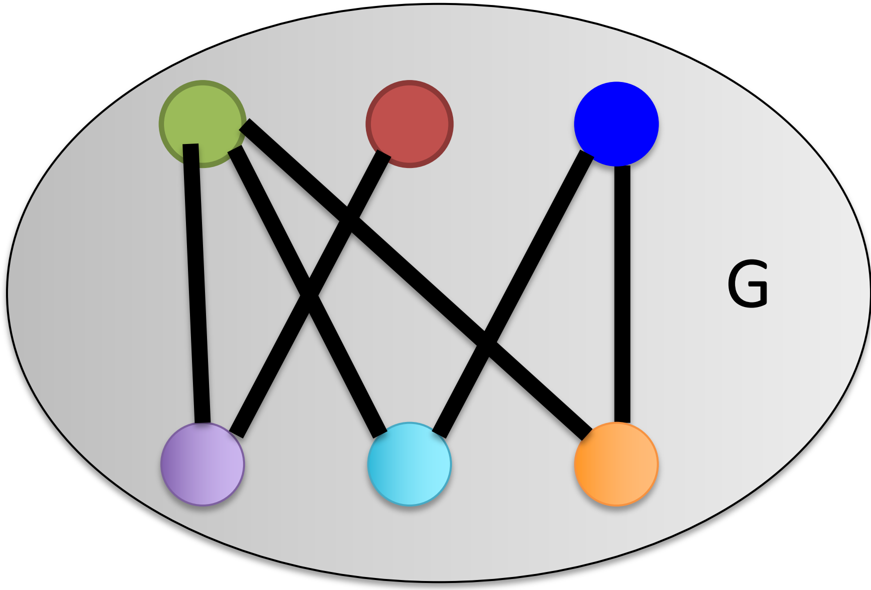
If true, then NDP in grids is at least as hard as WGP

could use an algorithm for NDP to solve WGP

Direction 1

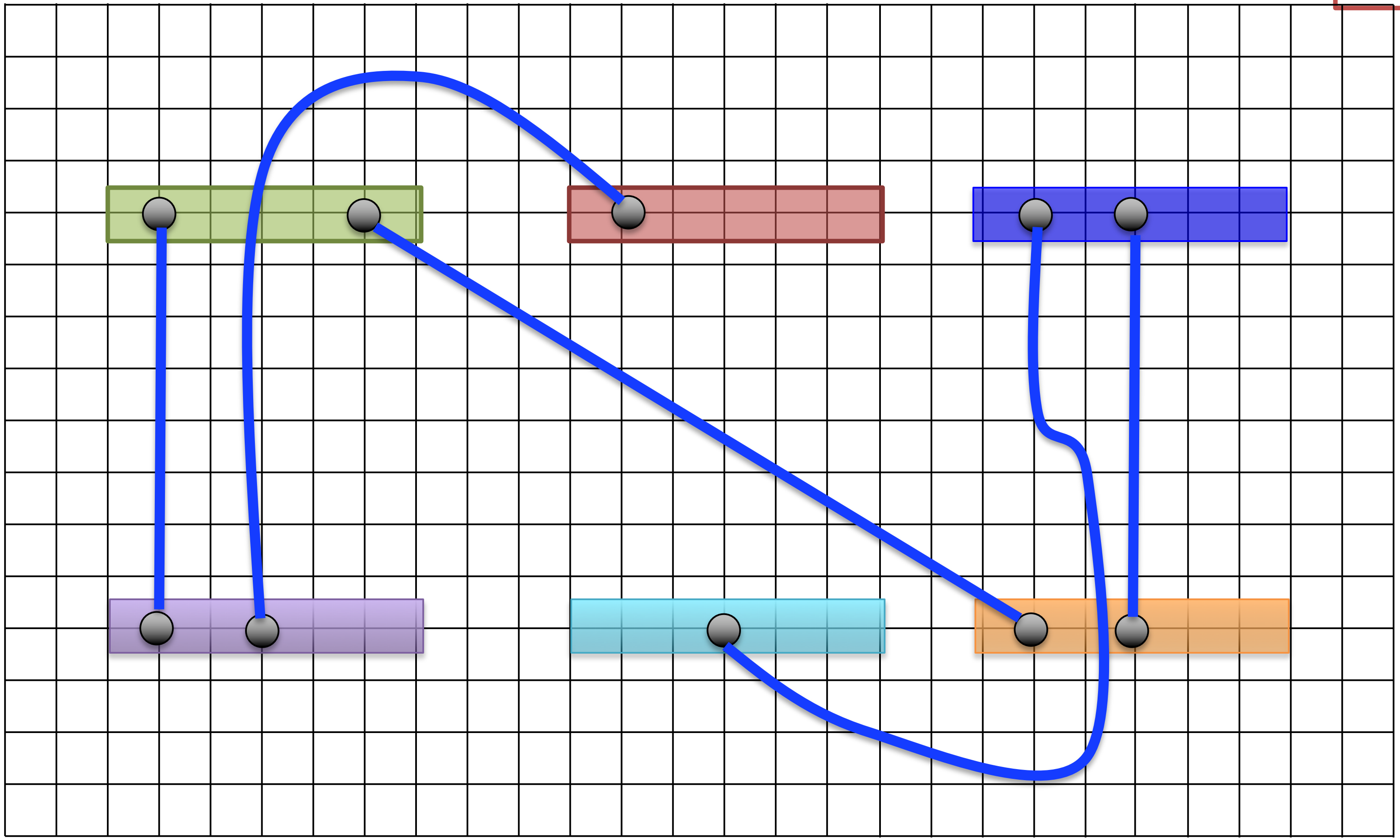
Claim: the reduction preserves solution value to within polylog(n) factor!

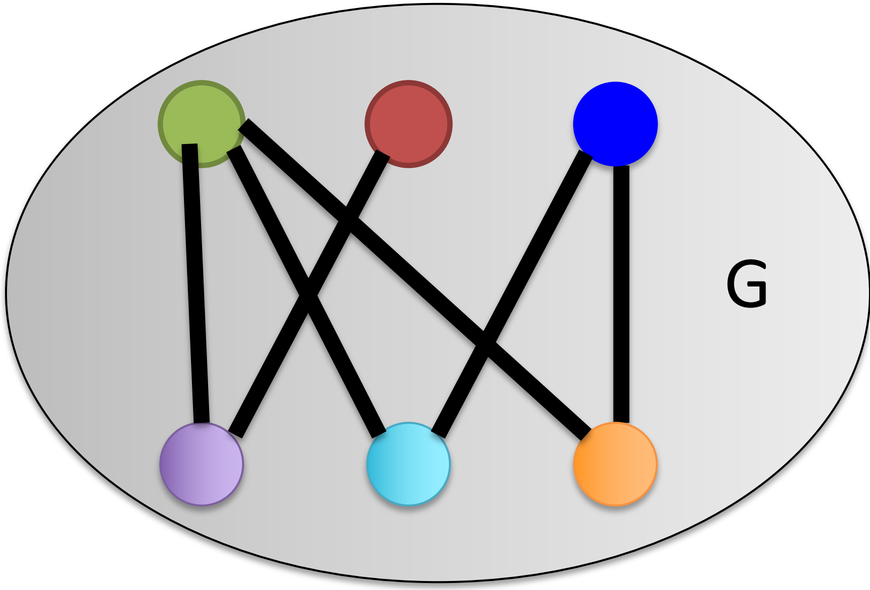




Direction 1

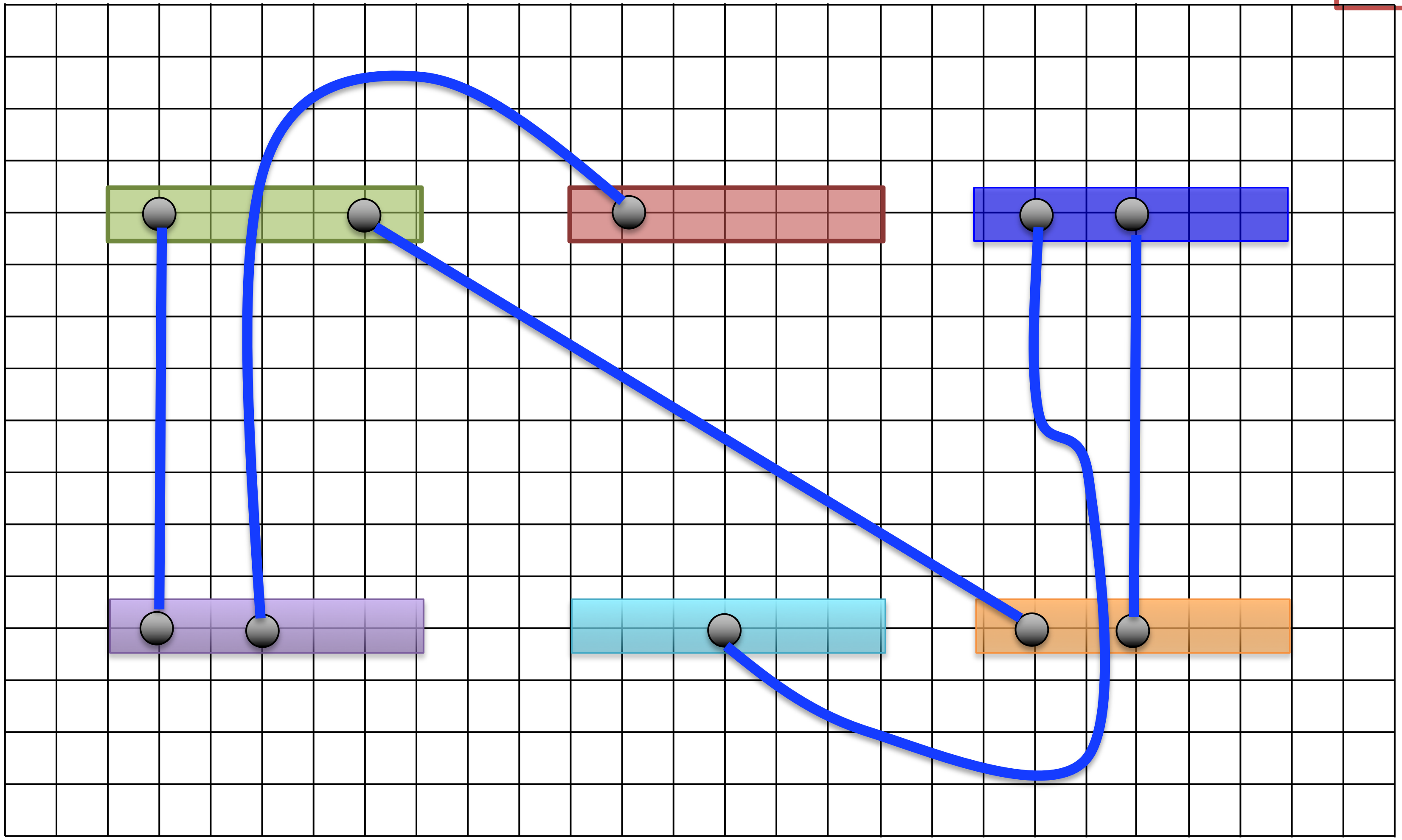
Claim: the reduction preserves solution value to within polylog(n) factor!

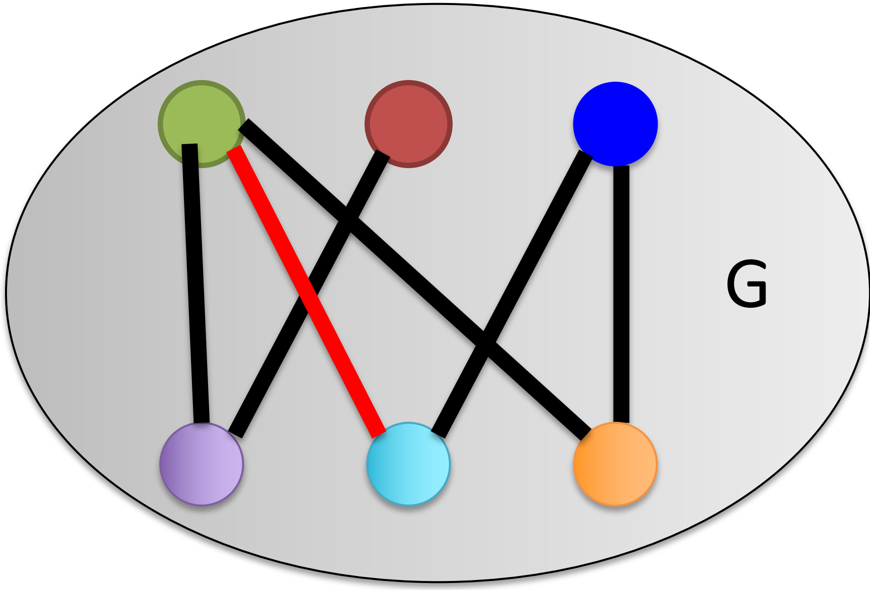




Direction 1

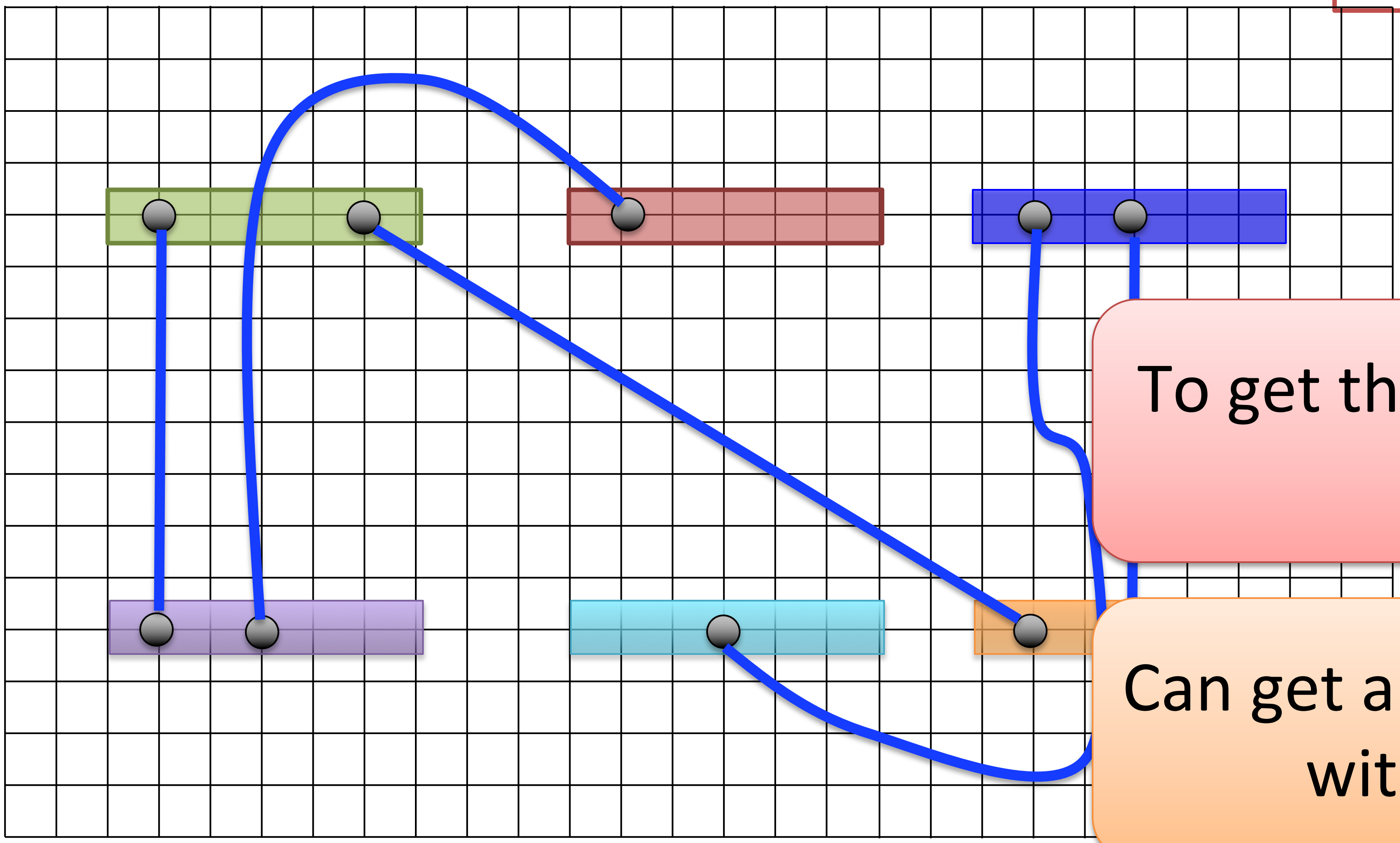
Claim: the reduction preserves solution value to within polylog(n) factor!

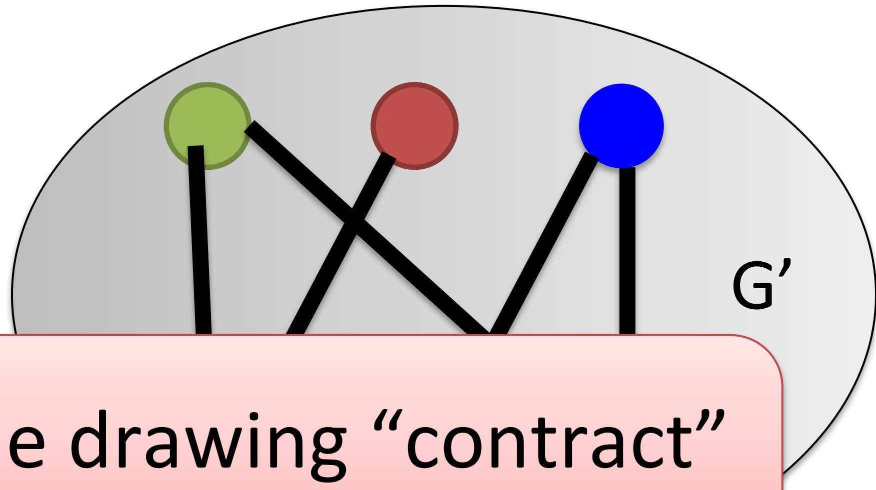




Direction 1

Claim: the reduction preserves solution value to within polylog(n) factor!



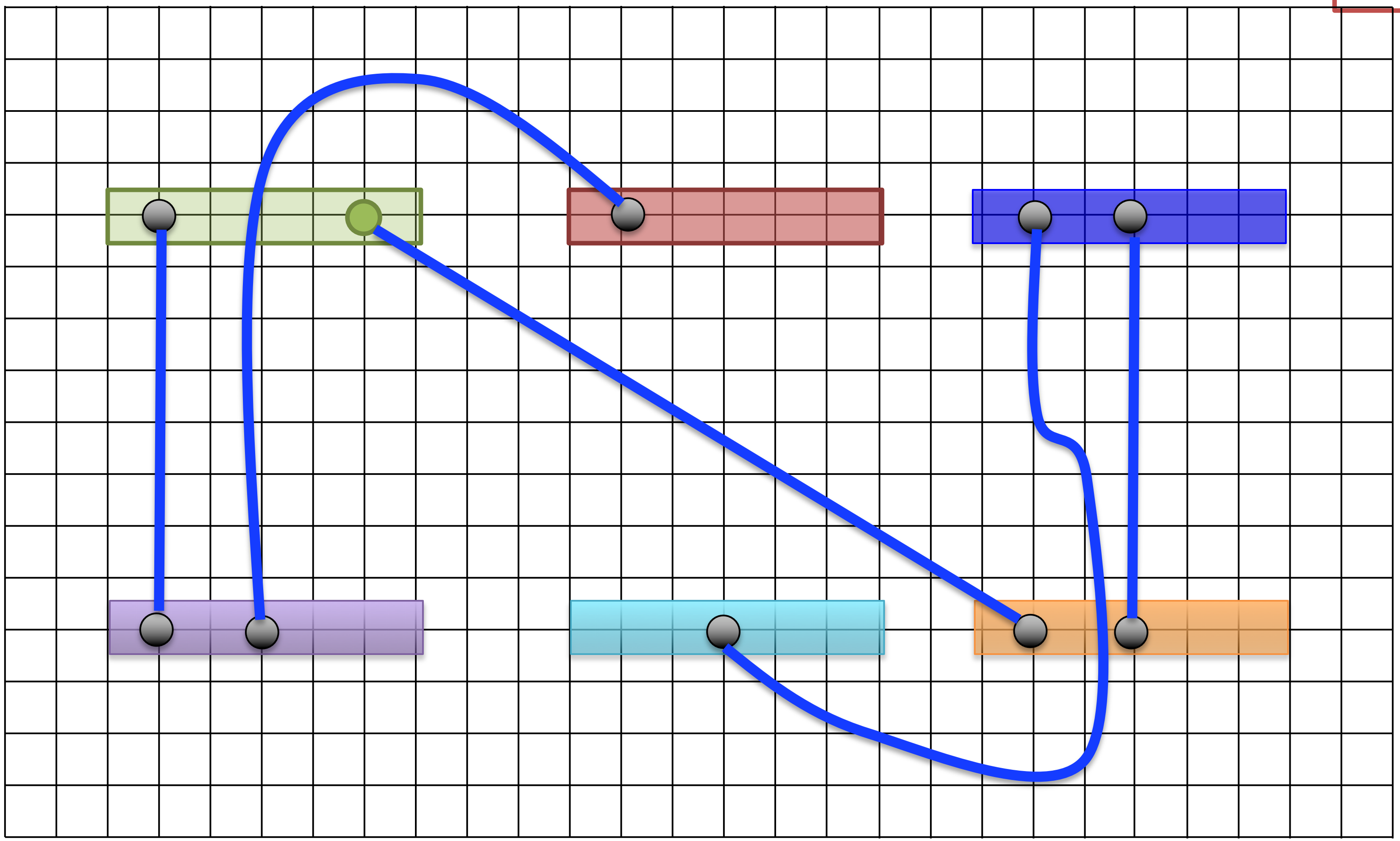


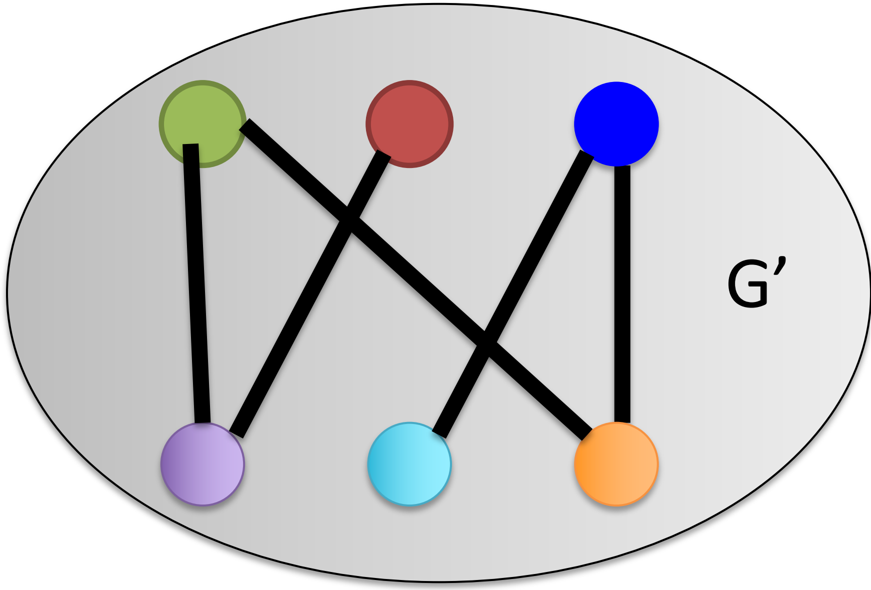
To get the drawing “contract” each block

Can get a drawing of the graph with few crossings

Direction 1

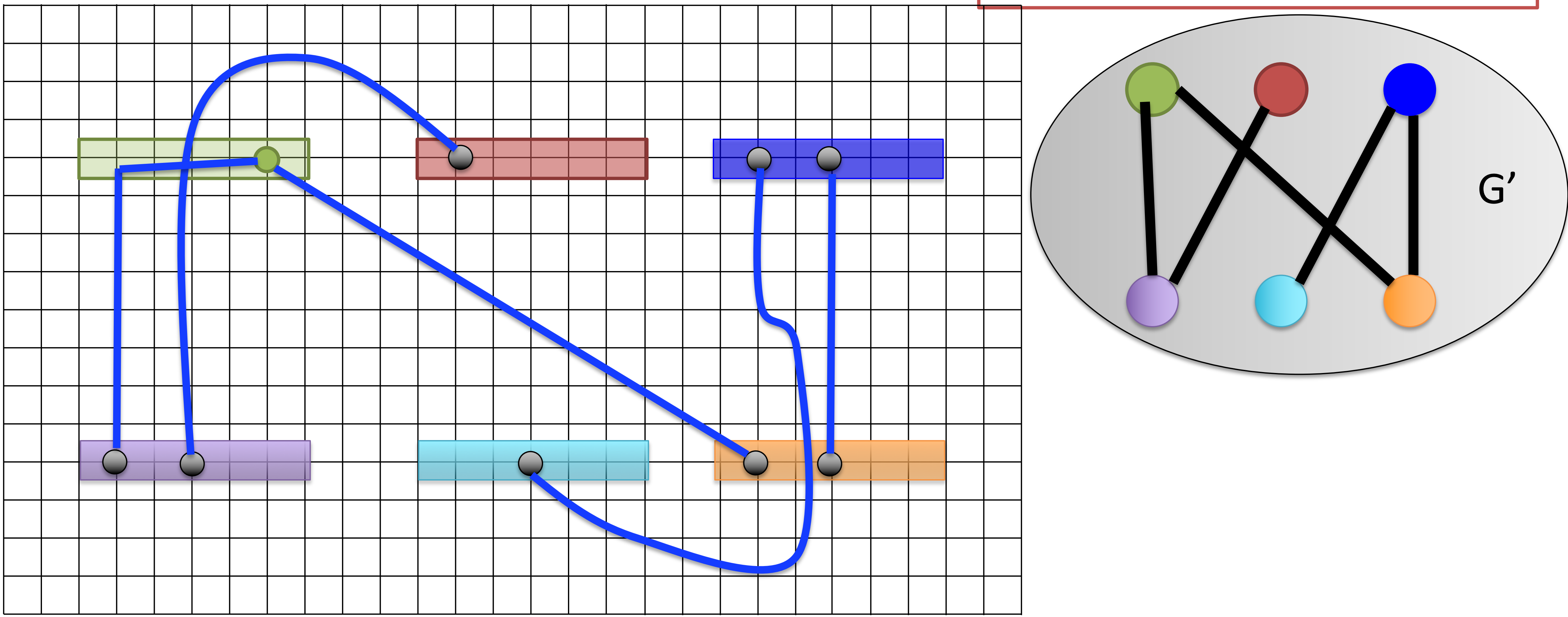
Claim: the reduction preserves solution value to within polylog(n) factor!





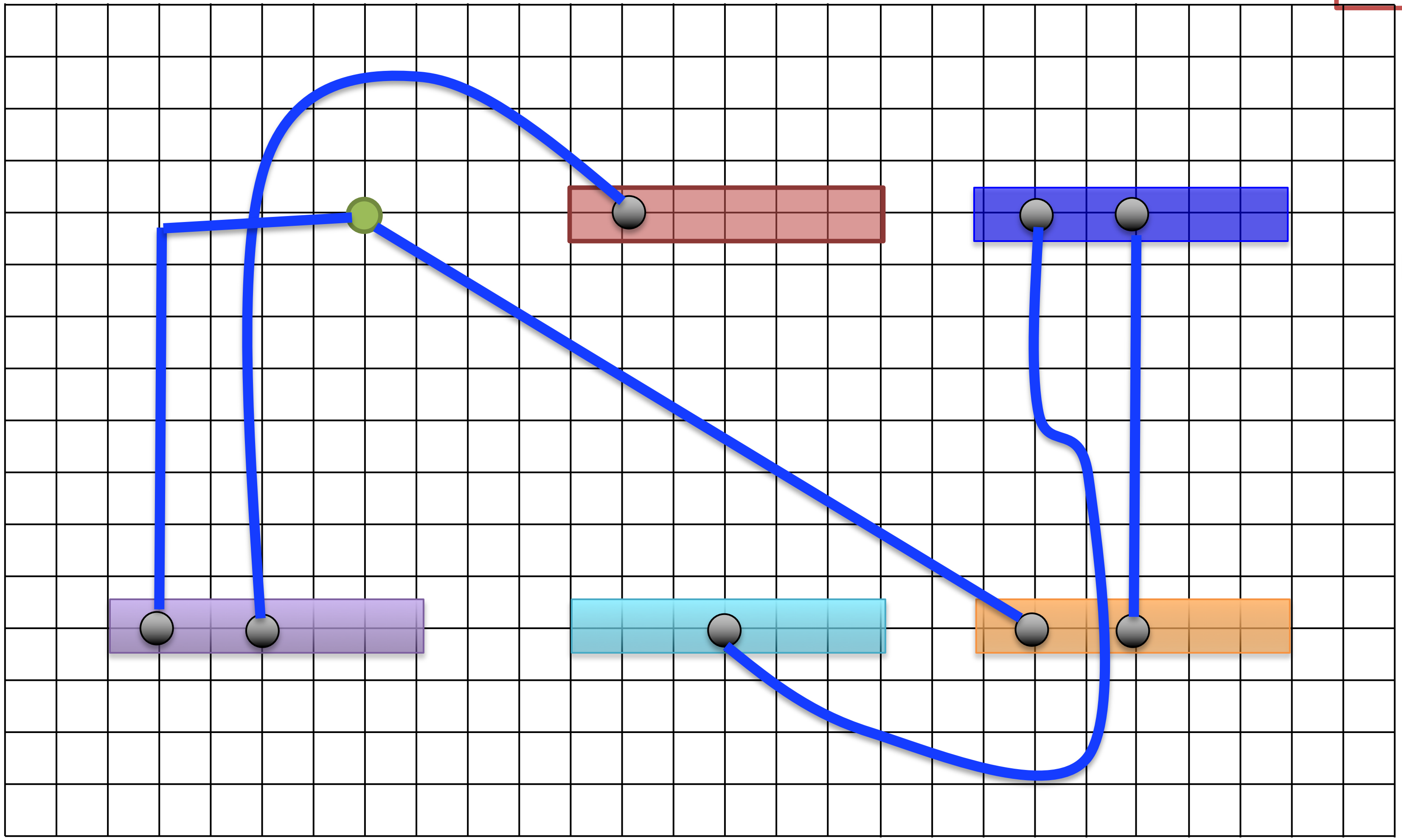
Direction 1

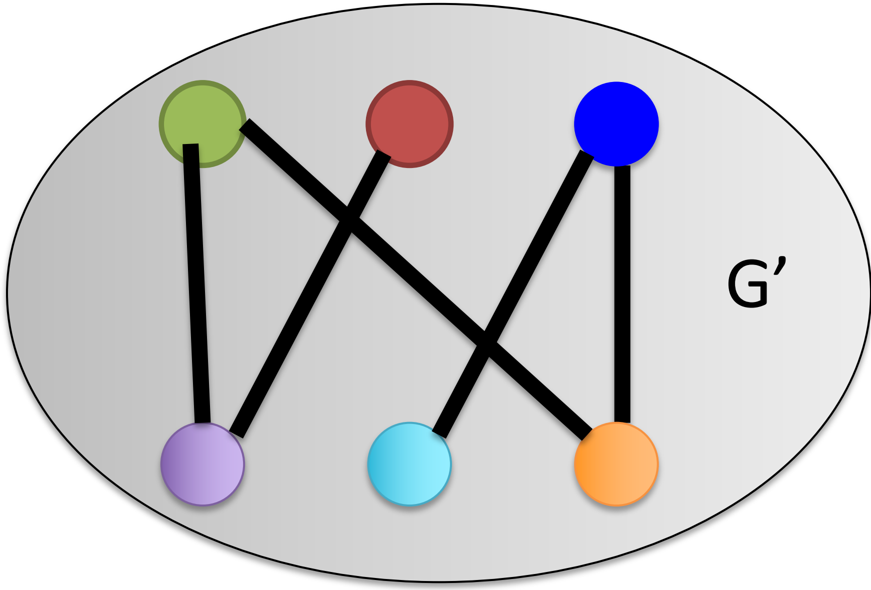
Claim: the reduction preserves solution value to within polylog(n) factor!



Direction 1

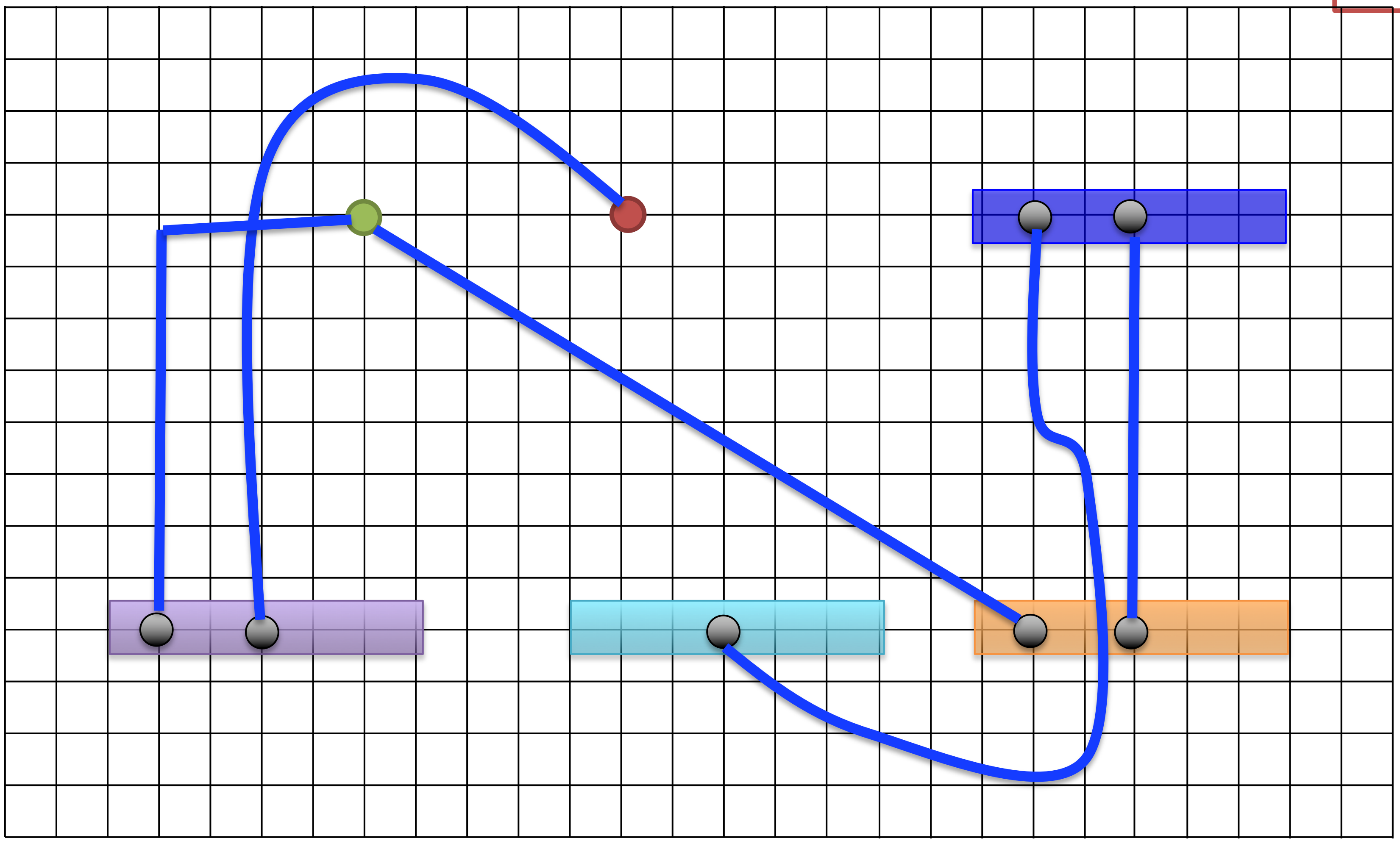
Claim: the reduction preserves solution value to within polylog(n) factor!

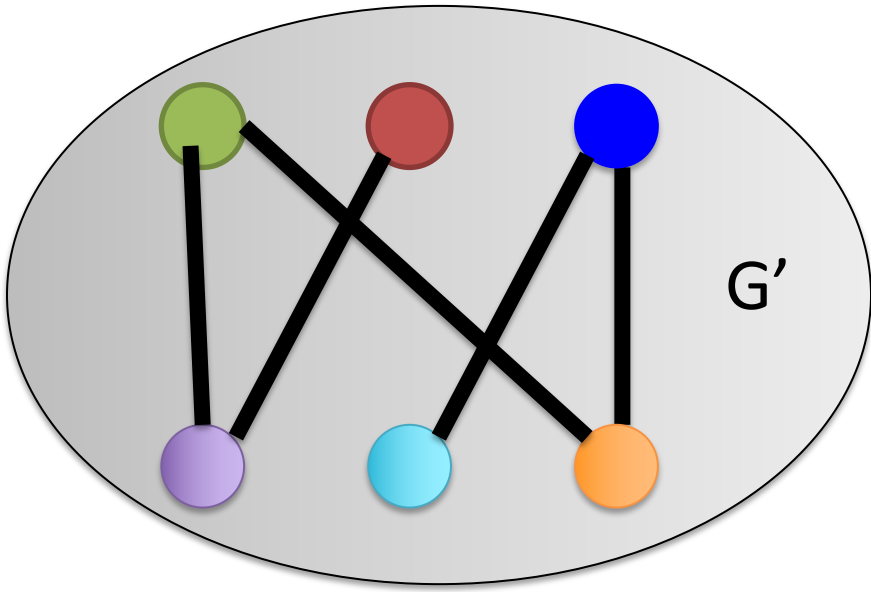




Direction 1

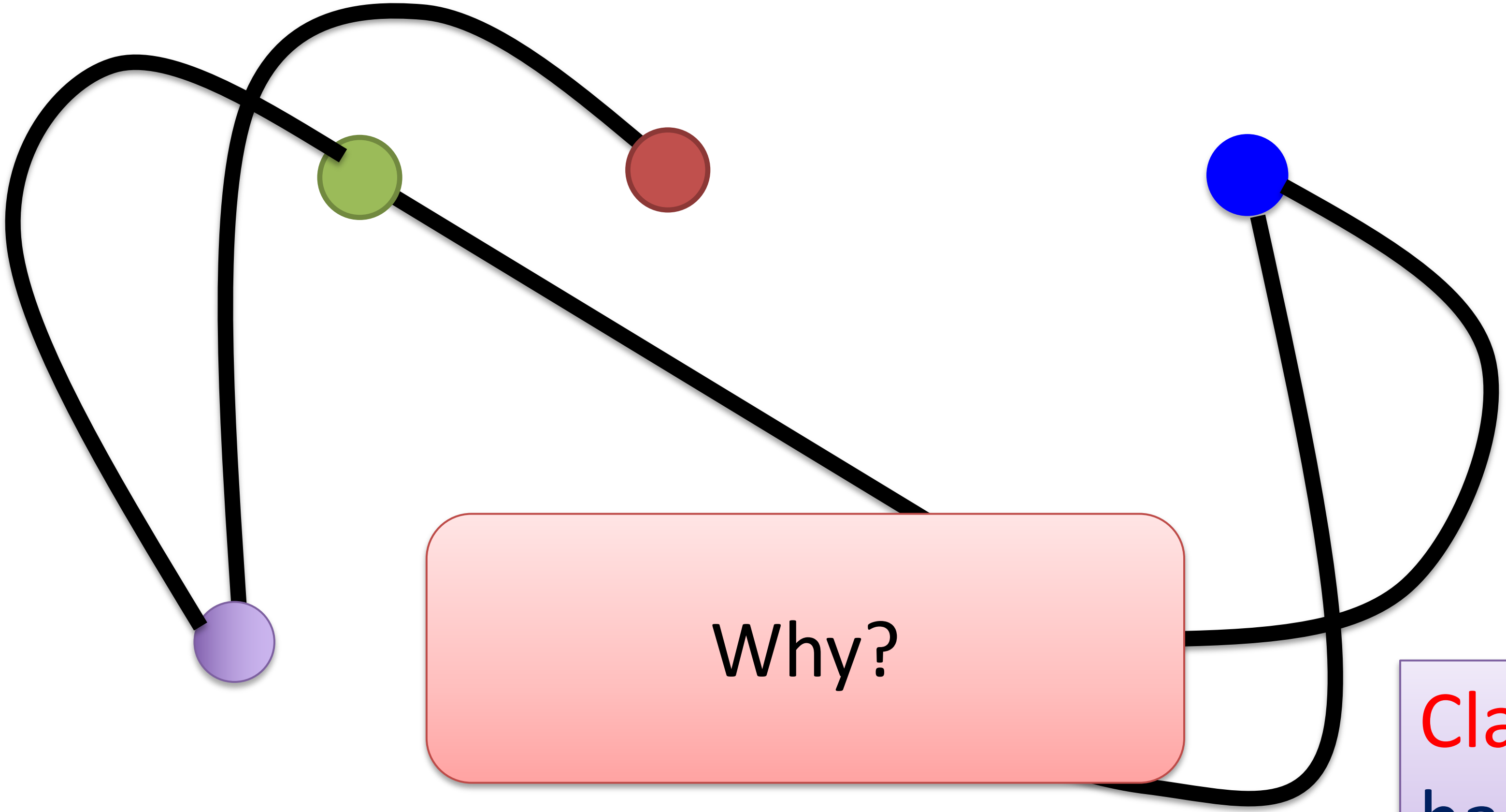
Claim: the reduction preserves solution value to within polylog(n) factor!

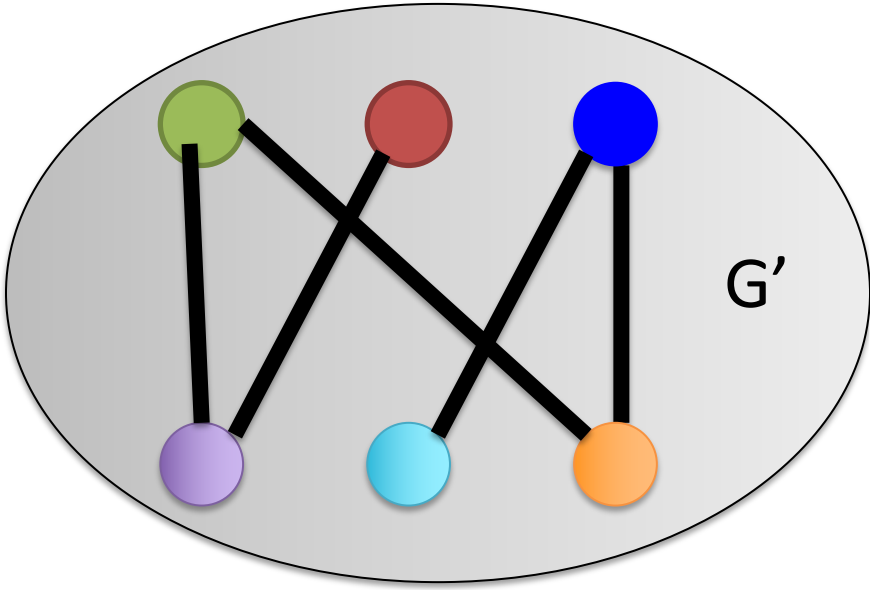




Direction 1

Claim: the reduction preserves solution value to within polylog(n) factor!

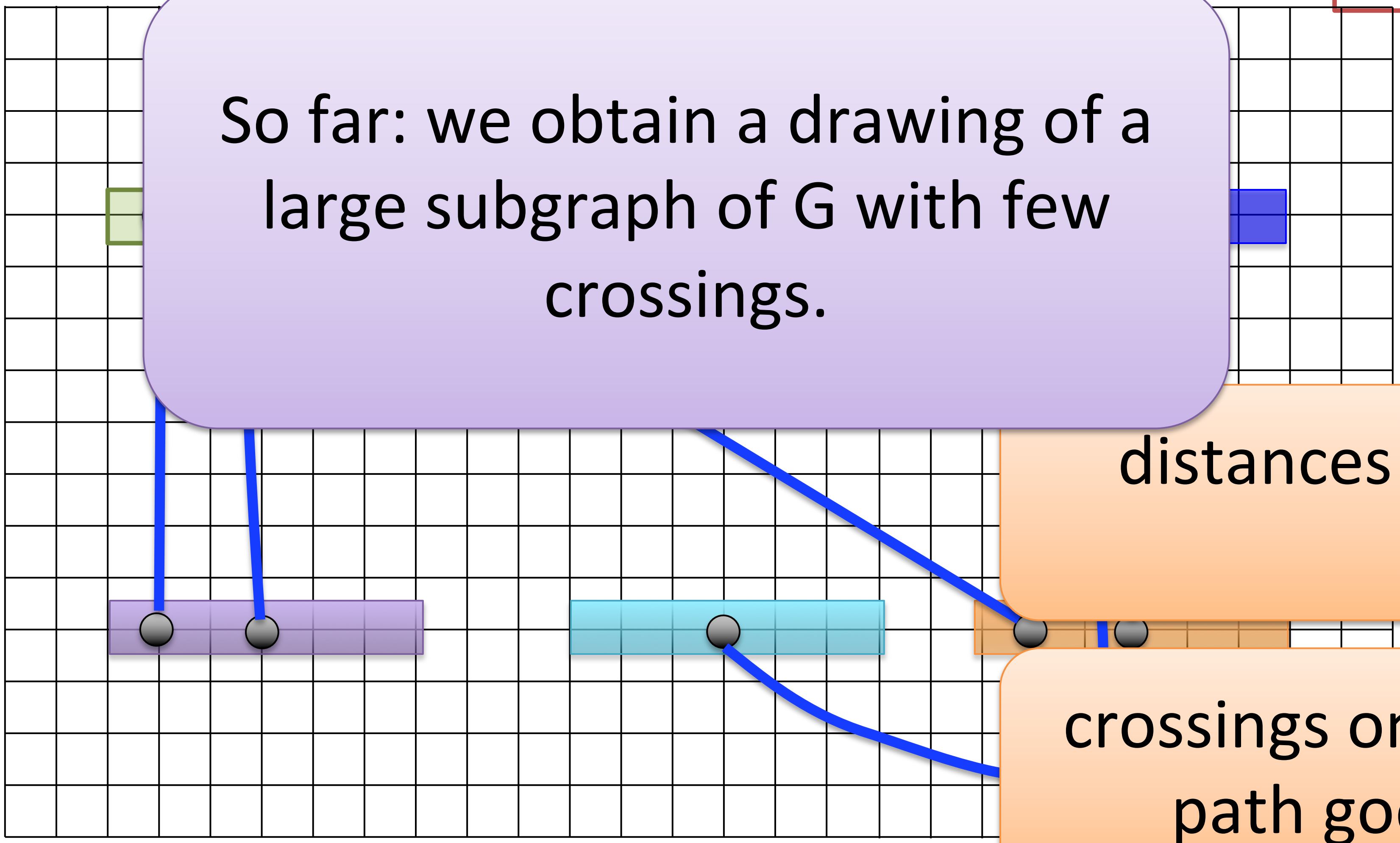




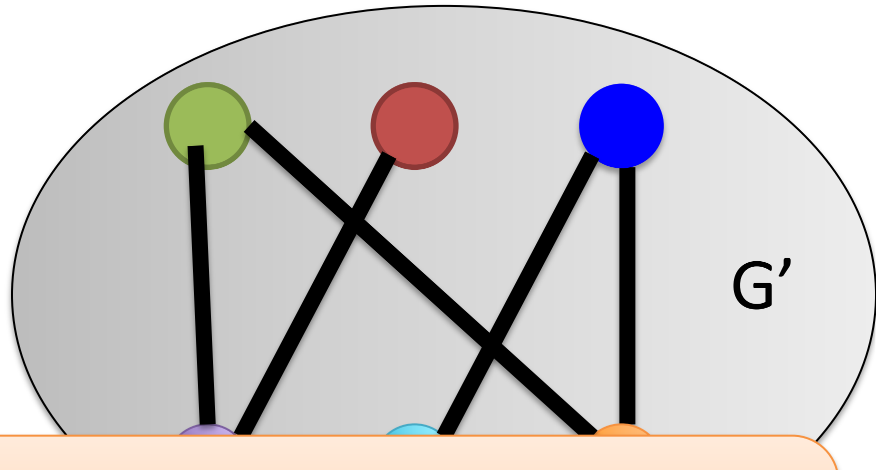
Claim: this drawing of G’ has few crossings.

Direction 1

Claim: the reduction preserves solution value to within polylog(n) factor!



So far: we obtain a drawing of a large subgraph of G with few crossings.



distances within a block small enough

crossings only introduced when a path goes through a block

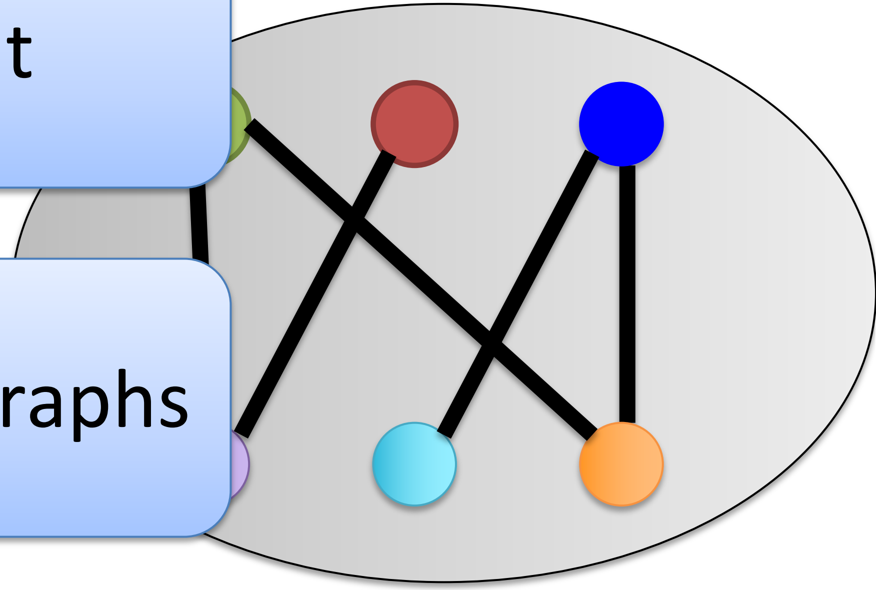
Direction 1

Claim: the reduction preserves solution value to within polylog(n) factor!

Planar graphs have very small balanced cut

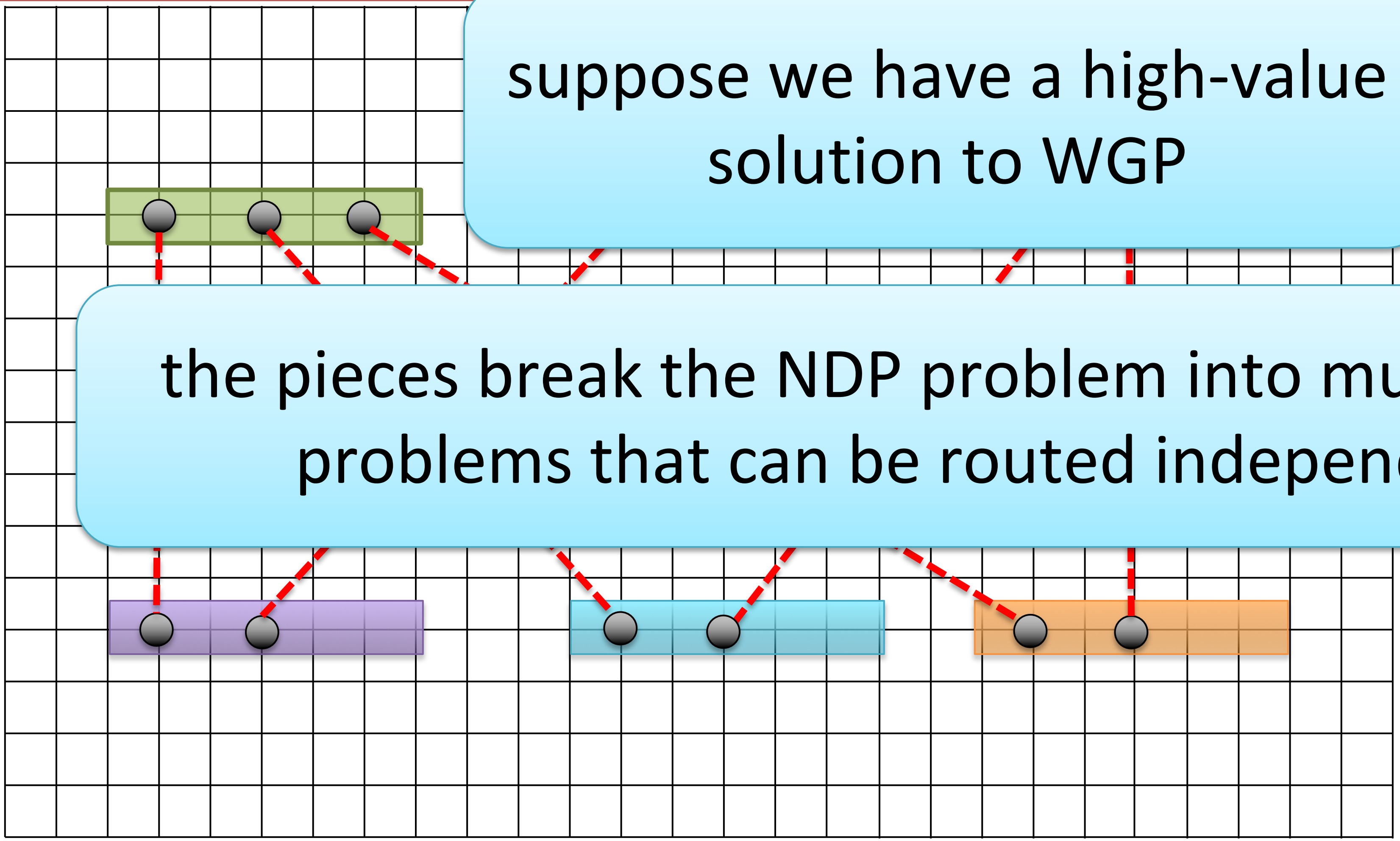
graphs with few crossings behave like planar graphs

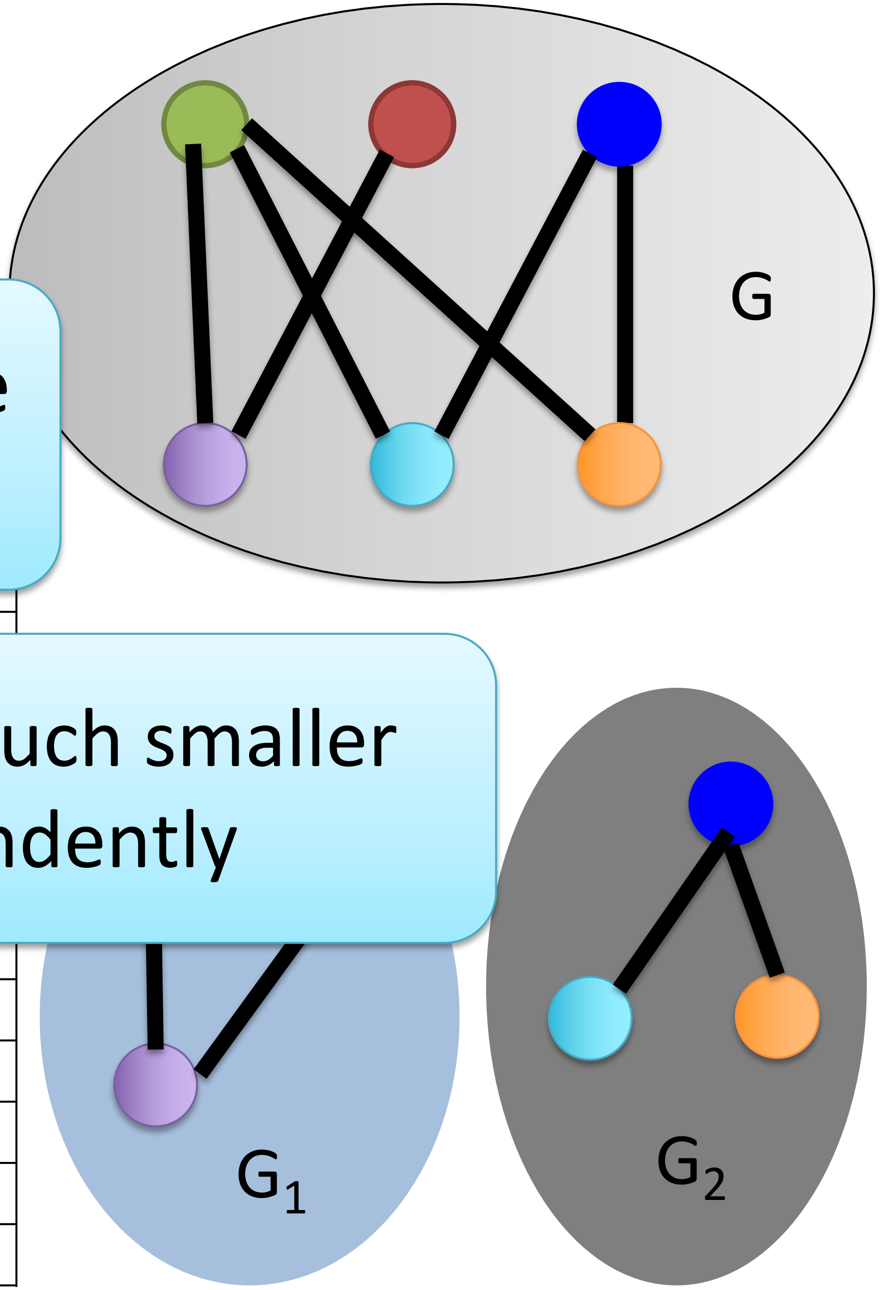
can cut into p balanced pieces by cutting few edges



Claim: the reduction preserves solution value to within polylog(n) factor!

Direction 2





suppose we have a high-value solution to WGP

the pieces break the NDP problem into much smaller problems that can be routed independently

Plan:

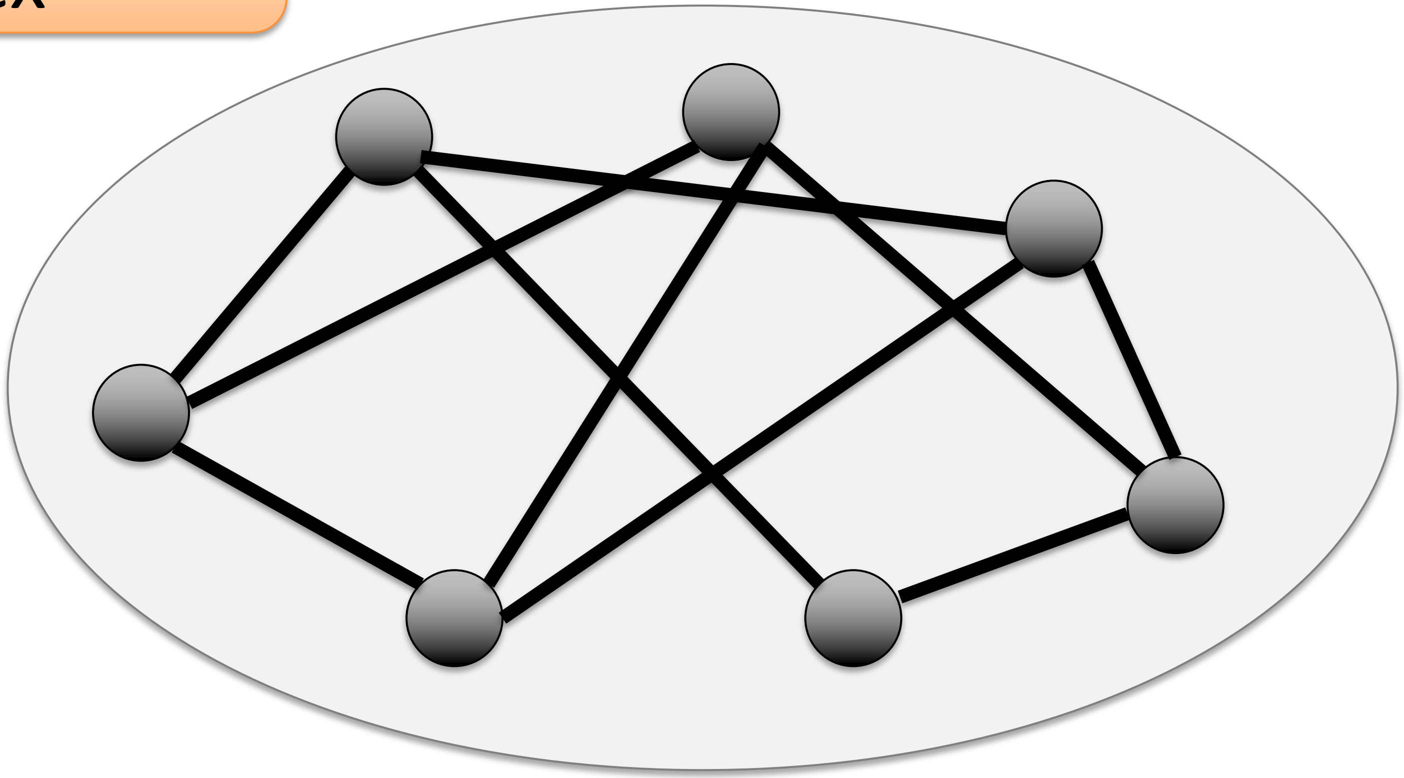
1. NDP in grids is at least as hard as WGP ✓
2. Prove hardness of WGP

Hardness of WGP

Starting Point: 3COL(5)

Input: 5-regular graph G.

3-coloring: assigning Red, Blue or Green color to each vertex

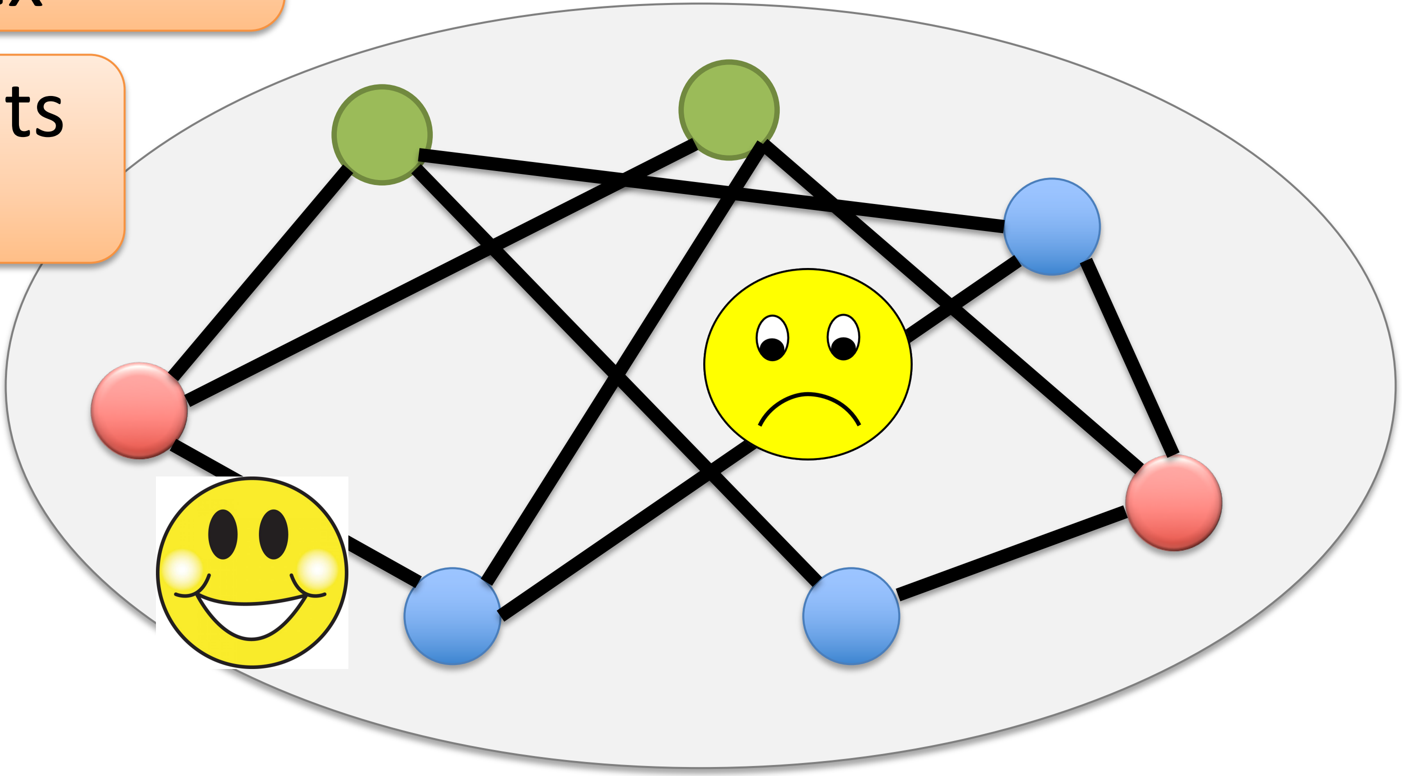


Starting Point: 3COL(5)

Input: 5-regular graph G.

3-coloring: assigning Red, Blue or Green color to each vertex

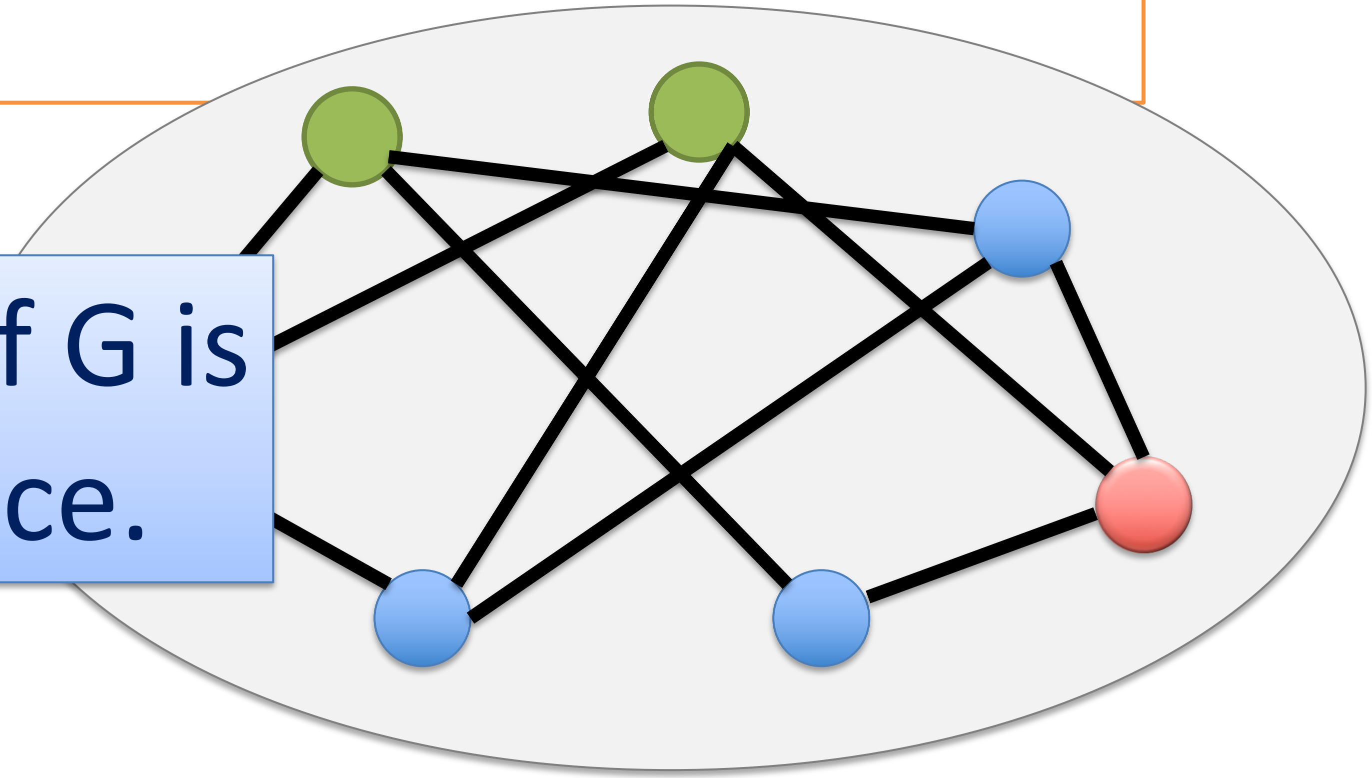
Edge is happy iff both endpoints have different colors



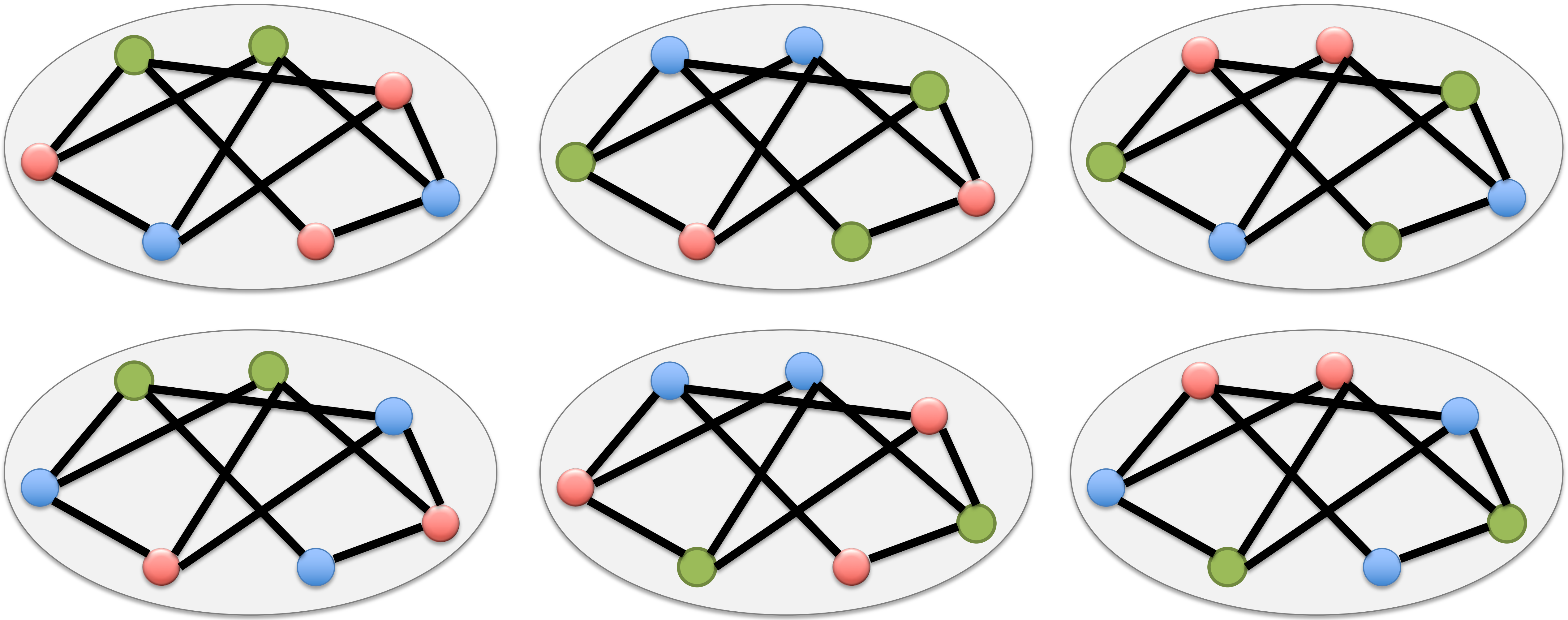
Starting Point: 3COL(5)

Input: 5-regular graph G.

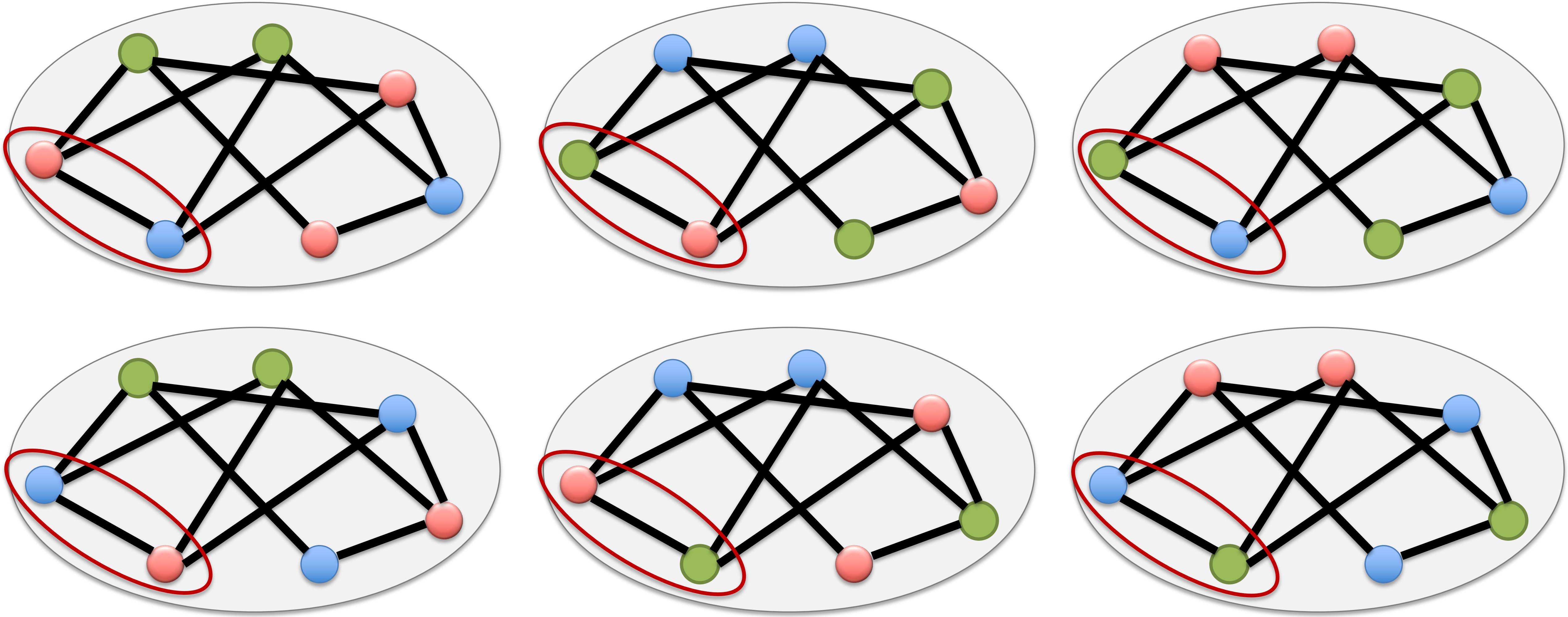
- G is Yes-Instance if there is a coloring where every edge is happy
- G is No-Instance if in every coloring at least 0.01-fraction of edges are unhappy



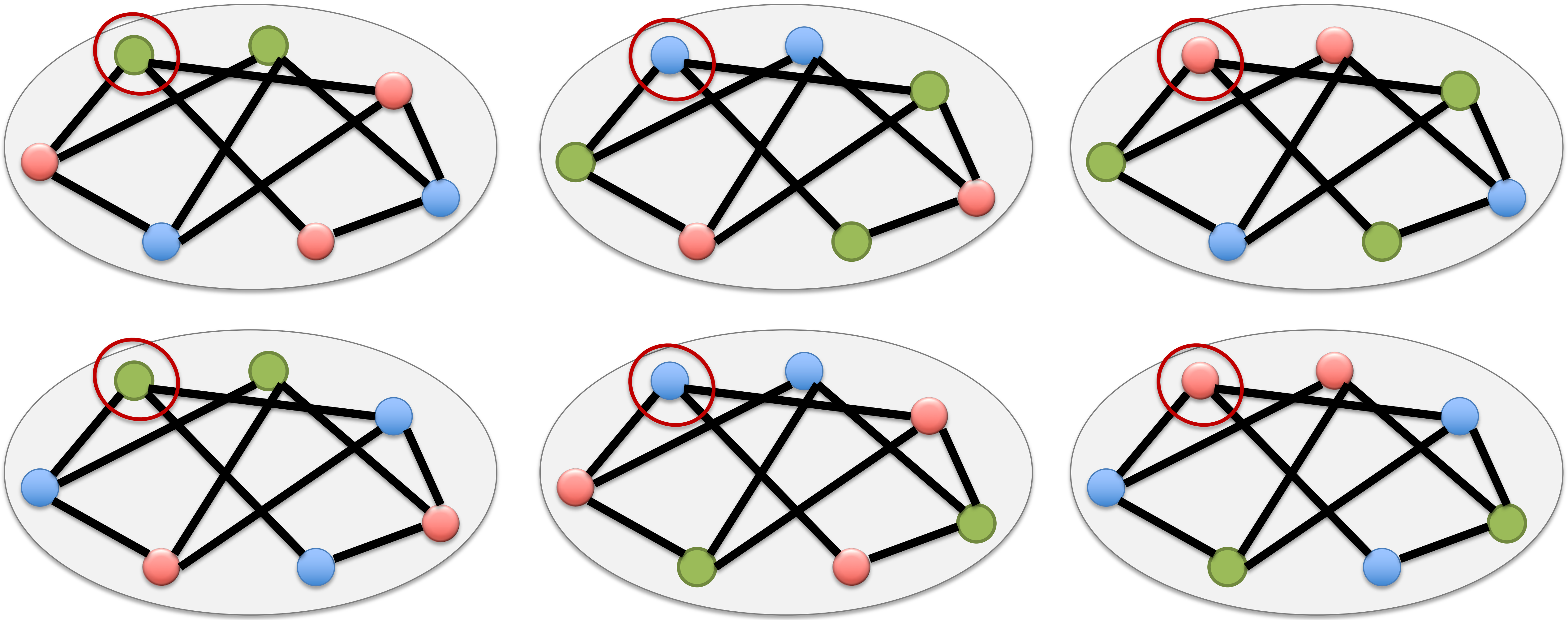
Thm: NP-hard to tell if G is a Yes or a No instance.



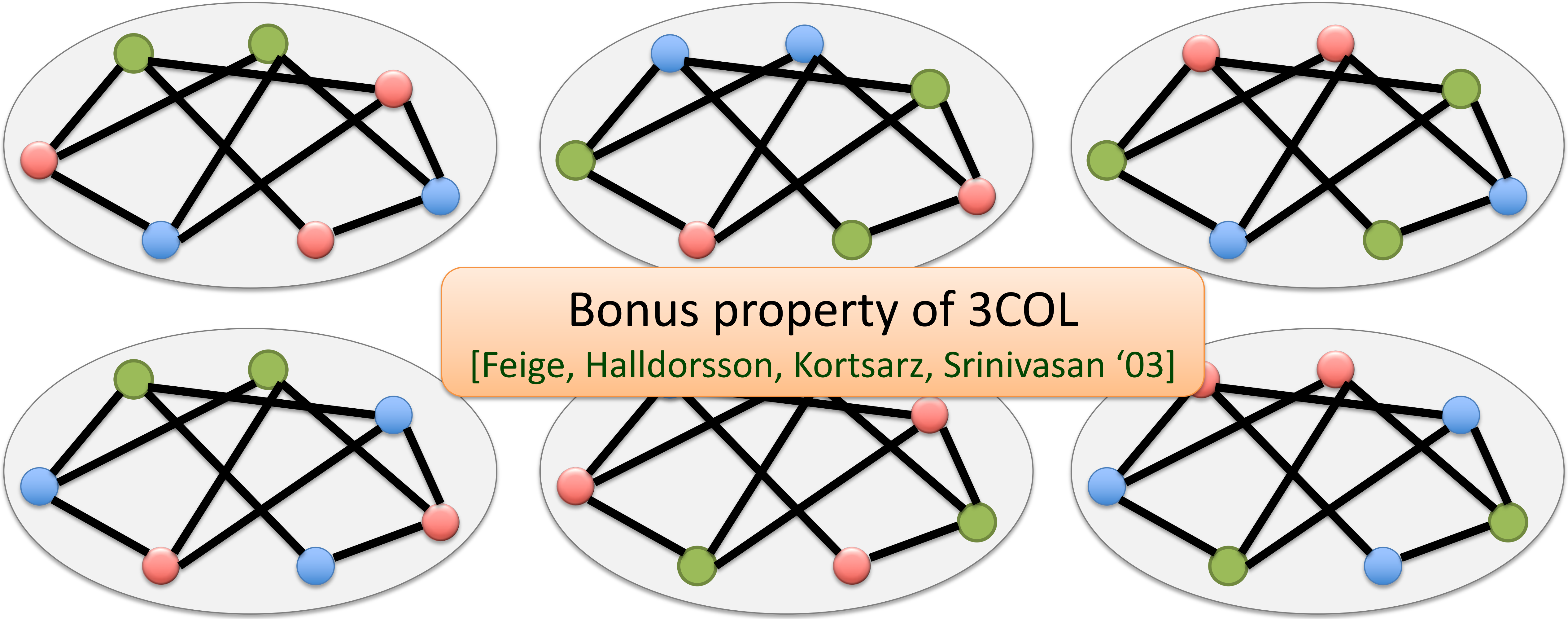
If there is a coloring that makes all edges happy, then there are 6 such colorings!



- For every edge, each legal coloring appears exactly once



- For every edge, each legal coloring appears exactly once
- For each vertex, every coloring appears exactly twice

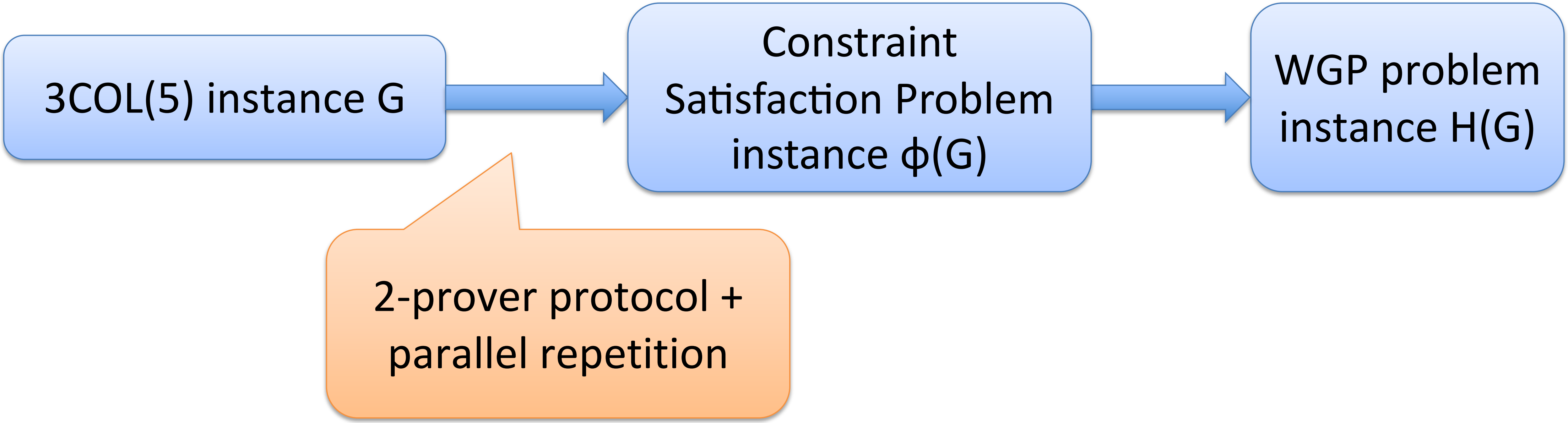


Bonus property of 3COL

[Feige, Halldorsson, Kortsarz, Srinivasan ‘03]

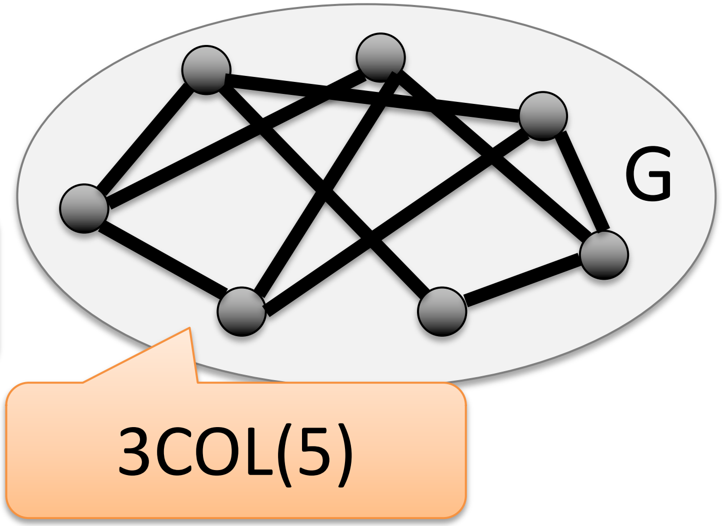
- For every edge, each legal coloring appears exactly once
- For each vertex, every coloring appears exactly twice

Next Steps


3COL(5) instance G
Constraint Satisfaction Problem instance ϕ(G)
WGP problem instance H(G)
2-prover protocol + parallel repetition

r – number of repetitions

A CSP Instance $\phi(G)$



"edge"-variables

x_1

x_2

x_3

x_4

⋮

x_N

"vertex"-variables

y_1

y_2

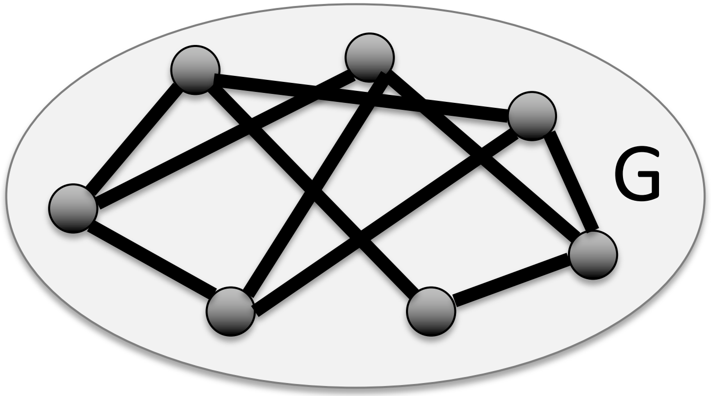
y_3

⋮

y_M

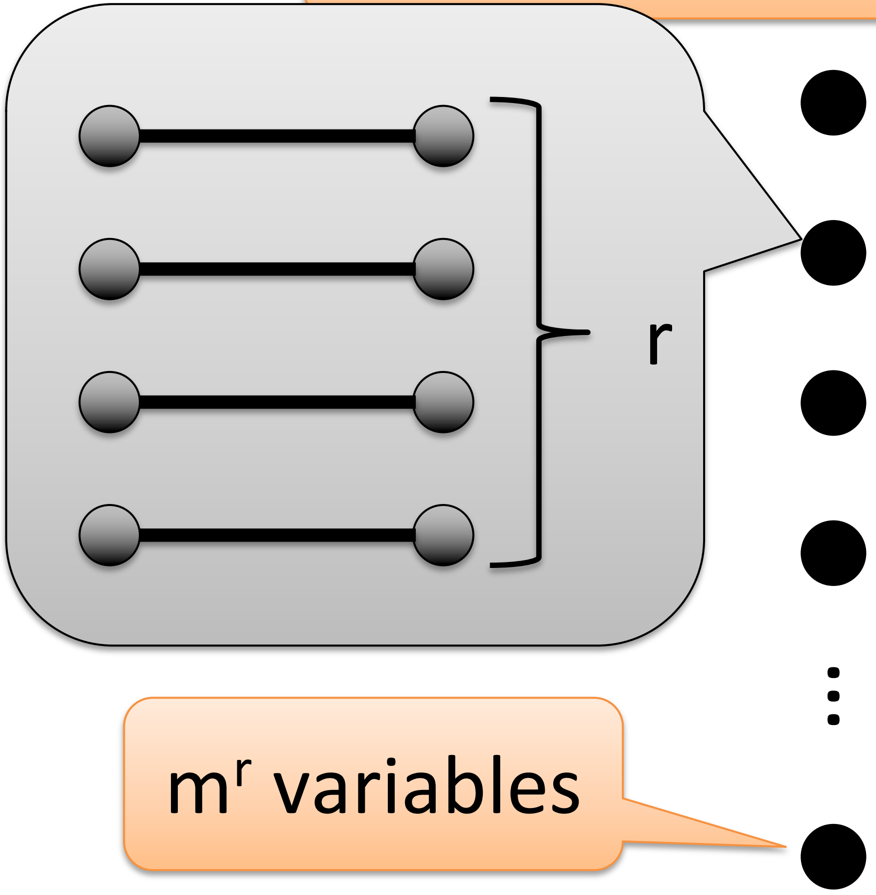
A CSP Instance $\phi(G)$

r – number of repetitions



“edge”-variables

“vertex”-variables



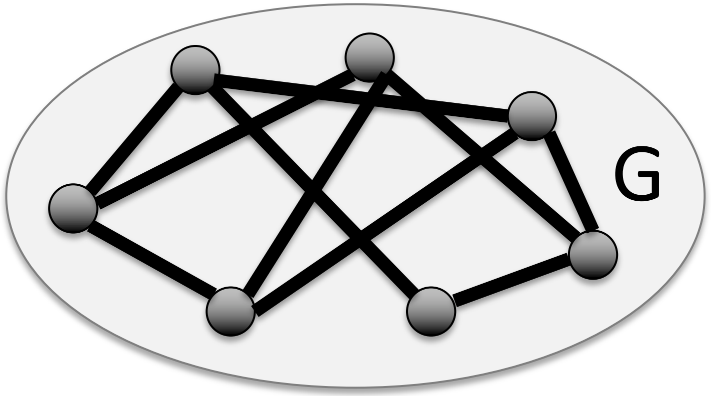
m^r variables

For every sequence of r edges of G, there is a variable on the left

An assignment to the variable is a legal coloring of the edges

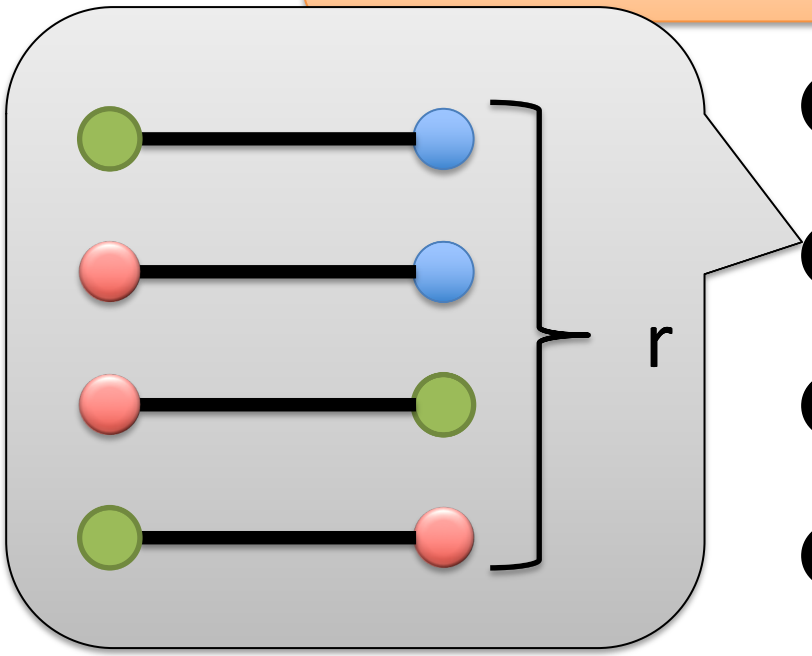
r – number of repetitions

A CSP Instance $\phi(G)$



“edge”-variables

“vertex”-variables

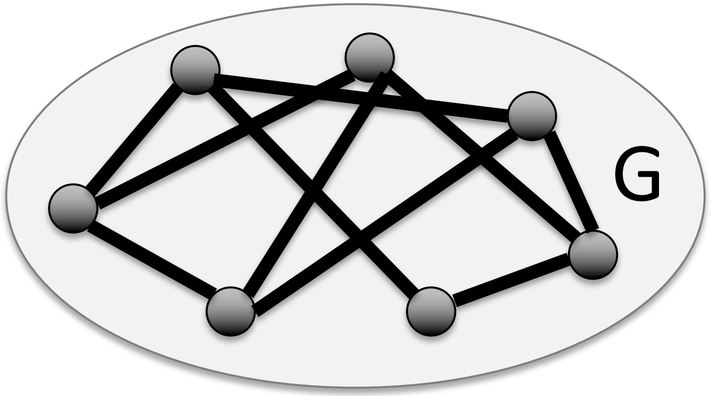


For every sequence of r edges of G, there is a variable on the left

An assignment to the variable is a legal coloring of the edges

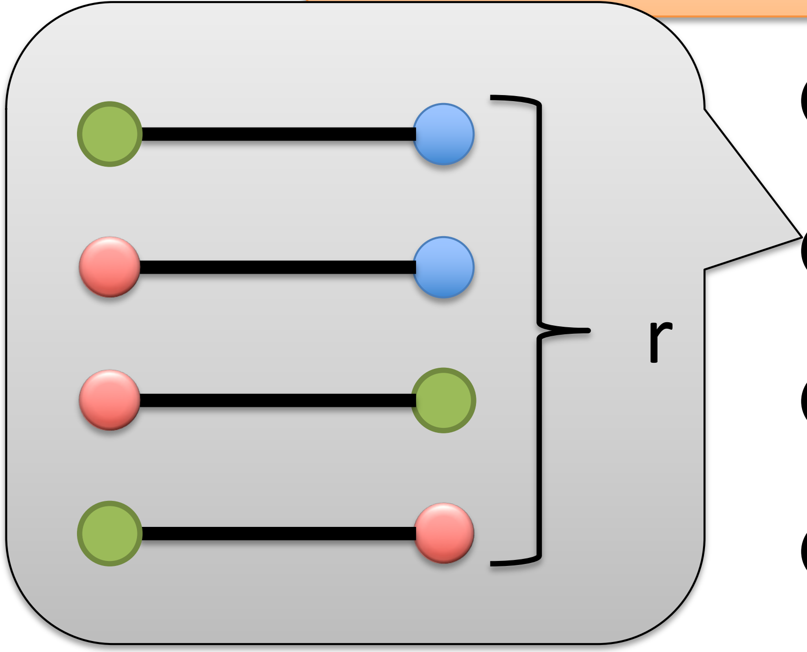
r – number of repetitions

A CSP Instance $\phi(G)$



"edge"-variables

"vertex"-variables



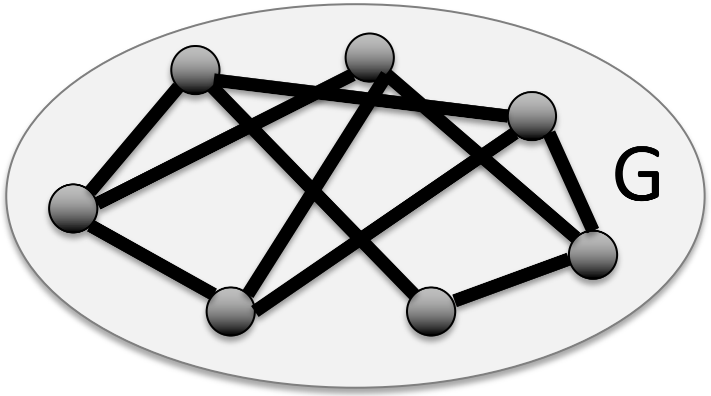
For every sequence of r edges of G, there is a variable on the left

need not be consistent across edges

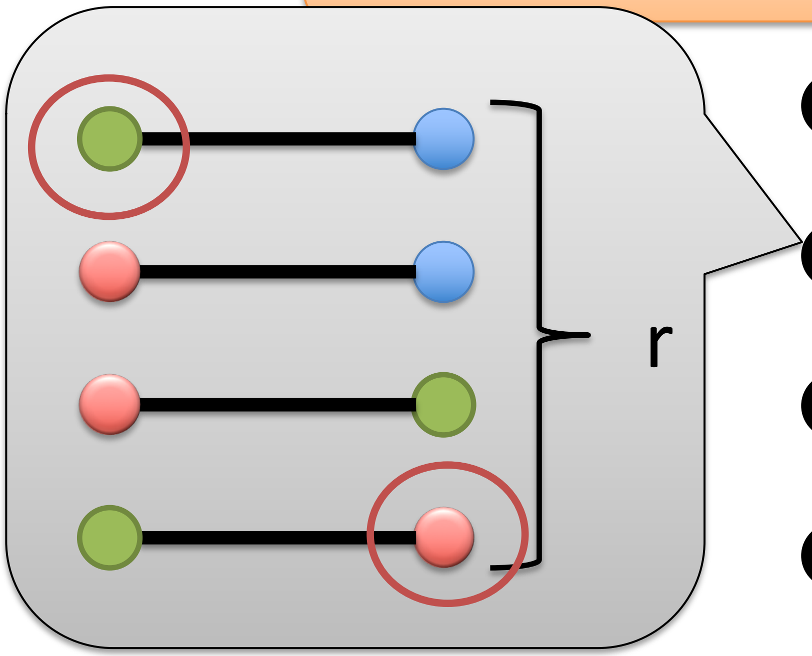
An assignment to the variable is a legal coloring of the edges

A CSP Instance ϕ(G)

r – number of repetitions



“edge”-variables



“vertex”-variables

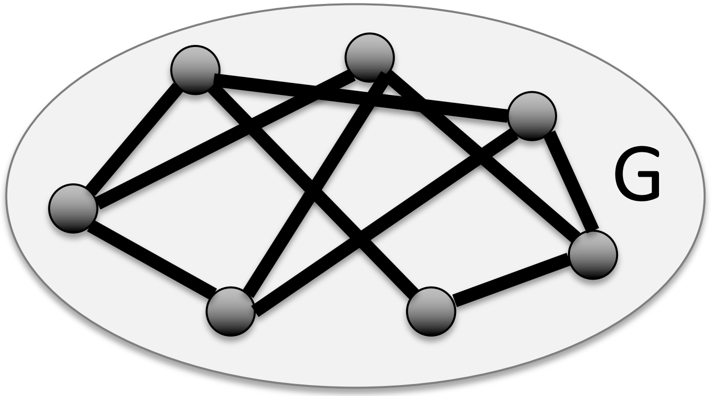
need not be consistent across edges

For every sequence of r edges of G, there is a variable on the left

An assignment to the variable is a legal coloring of the edges

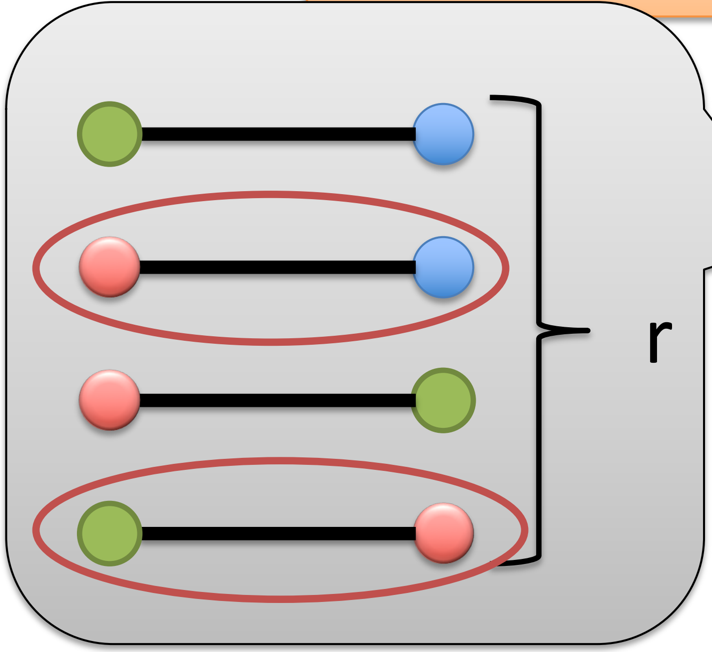
r – number of repetitions

A CSP Instance $\phi(G)$



"edge"-variables

"vertex"-variables



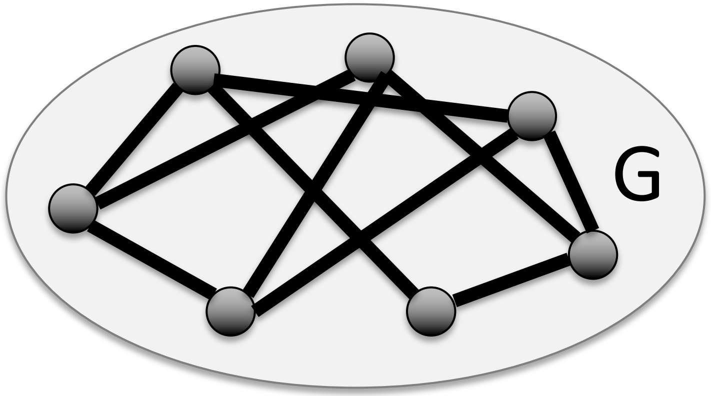
need not be consistent across edges

For every sequence of r edges of G, there is a variable on the left

An assignment to the variable is a legal coloring of the edges

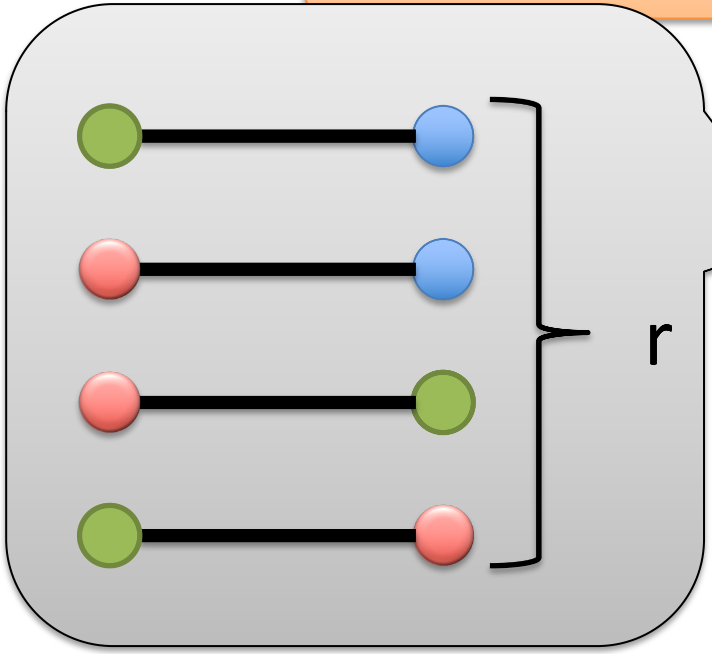
r – number of repetitions

A CSP Instance $\phi(G)$



“edge”-variables

“vertex”-variables



For every sequence of r edges of G, there is a variable on the left

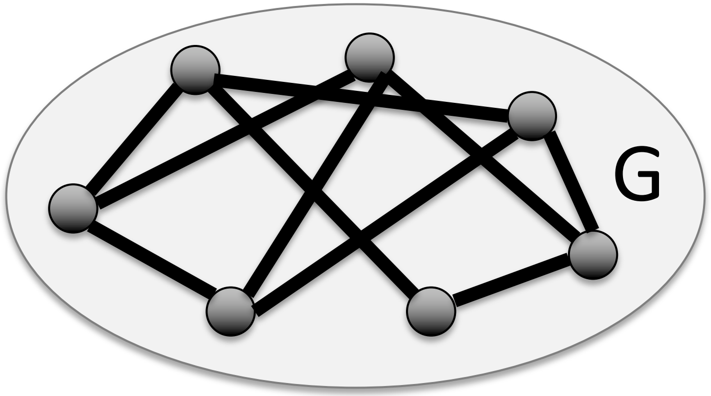
need not be consistent across edges

An assignment to the variable is a legal coloring of the edges

6^r possible assignments per variable

r – number of repetitions

A CSP Instance ϕ(G)

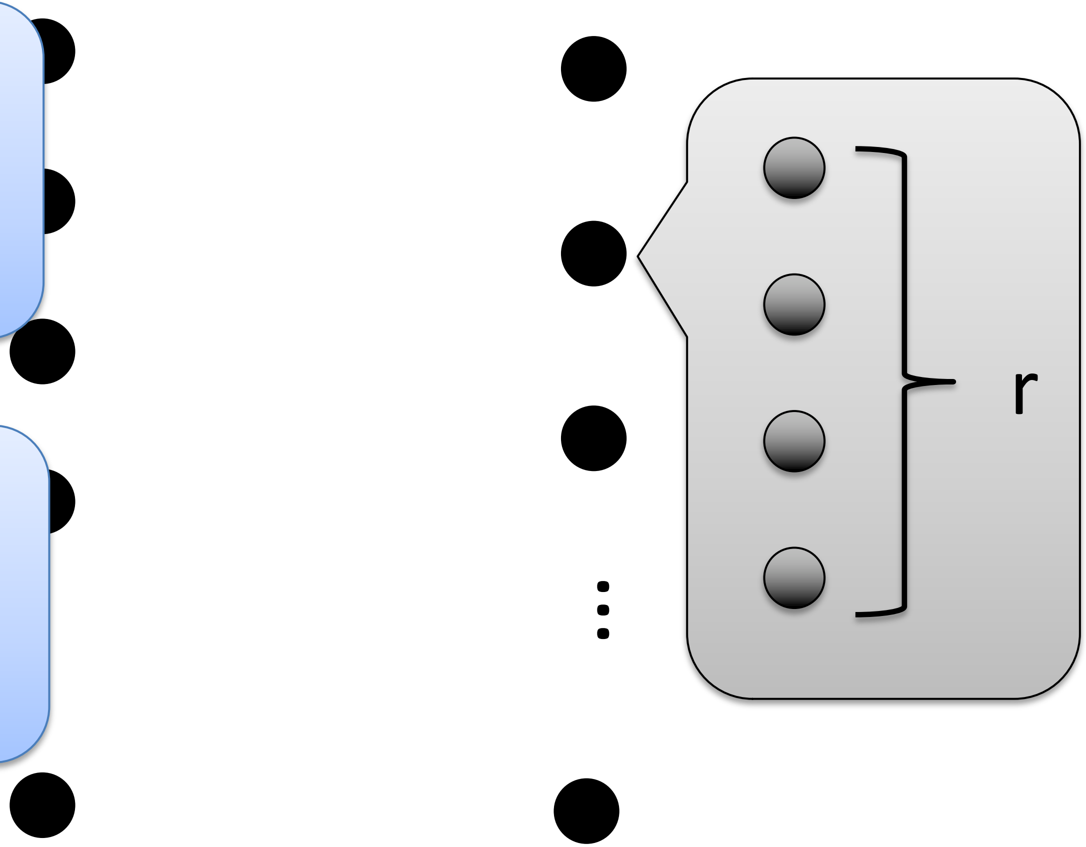


“edge”-variables

“vertex”-variables

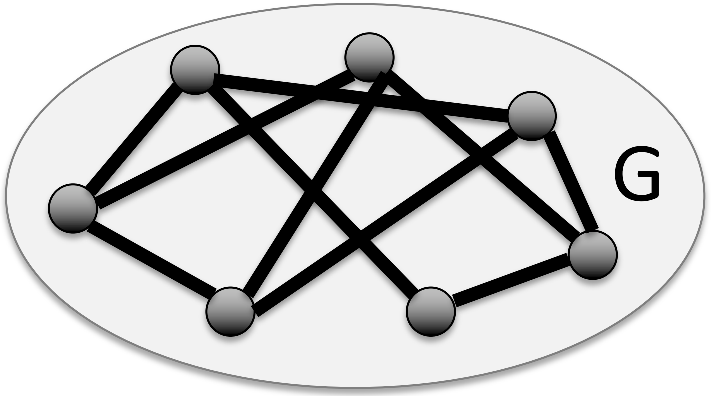
For every sequence of r vertices of G, there is a variable on the right

An assignment to the variable is a coloring of the vertices



A CSP Instance ϕ(G)

r – number of repetitions



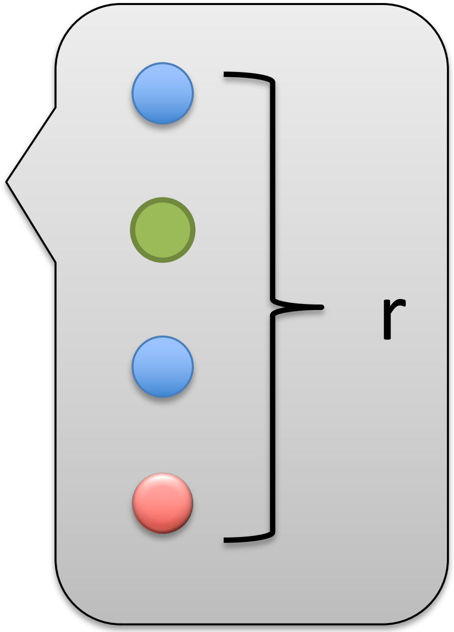
“edge”-variables

“vertex”-variables

For every sequence of r vertices of G, there is a variable on the right

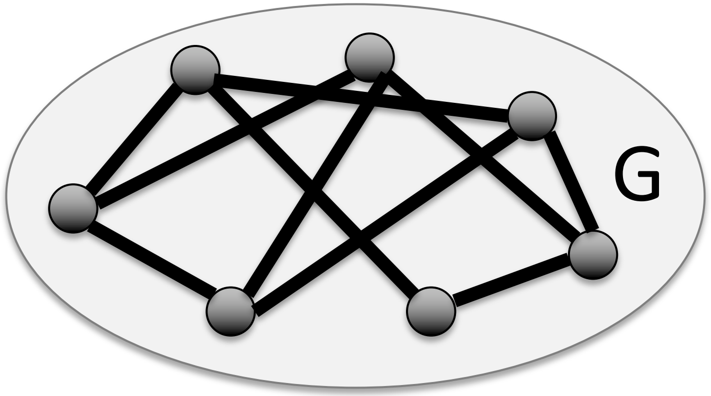
need not be consistent across vertices

An assignment to the variable is a coloring of the vertices



r – number of repetitions

A CSP Instance $\phi(G)$

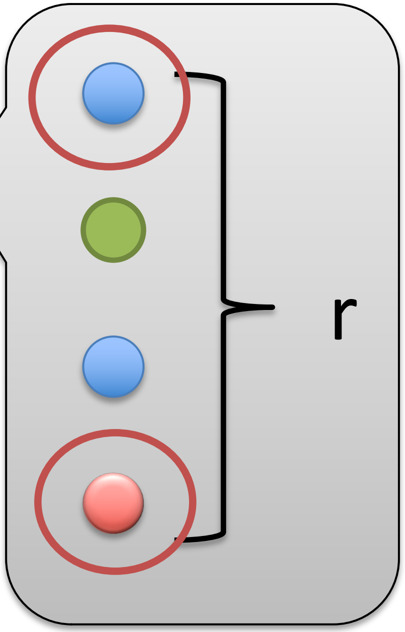


"edge"-variables

"vertex"-variables

For every sequence of r vertices of G, there is a variable on the right

need not be consistent across vertices

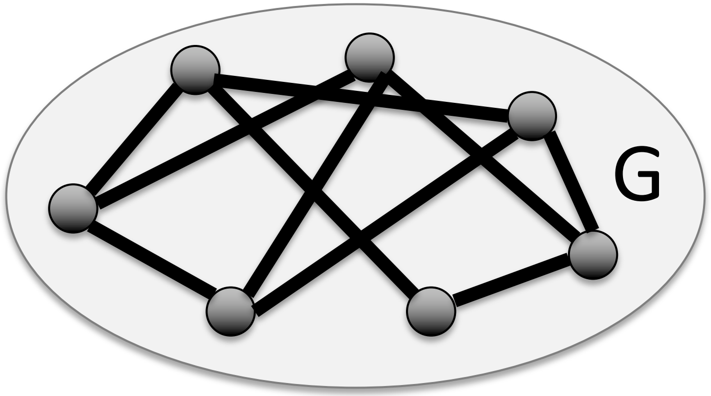


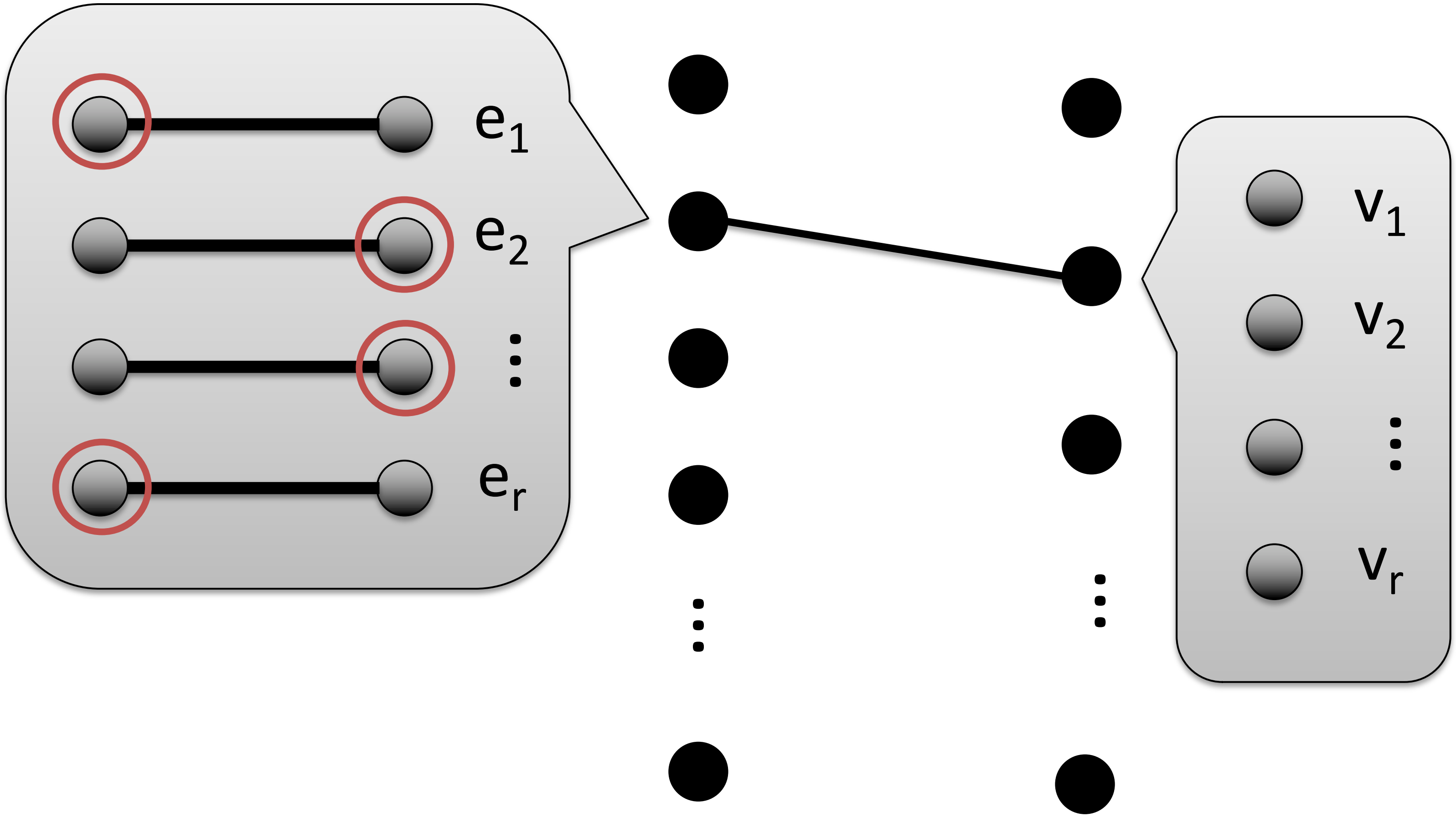
An assignment to the variable is a coloring of the vertices

3^r possible assignments per variable

r – number of repetitions

A CSP Instance $\phi(G)$

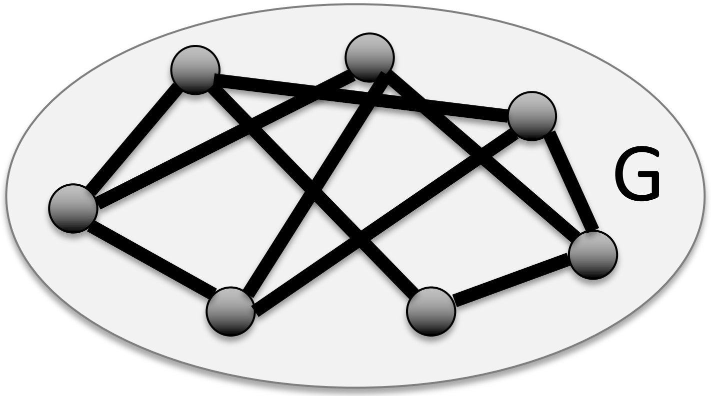


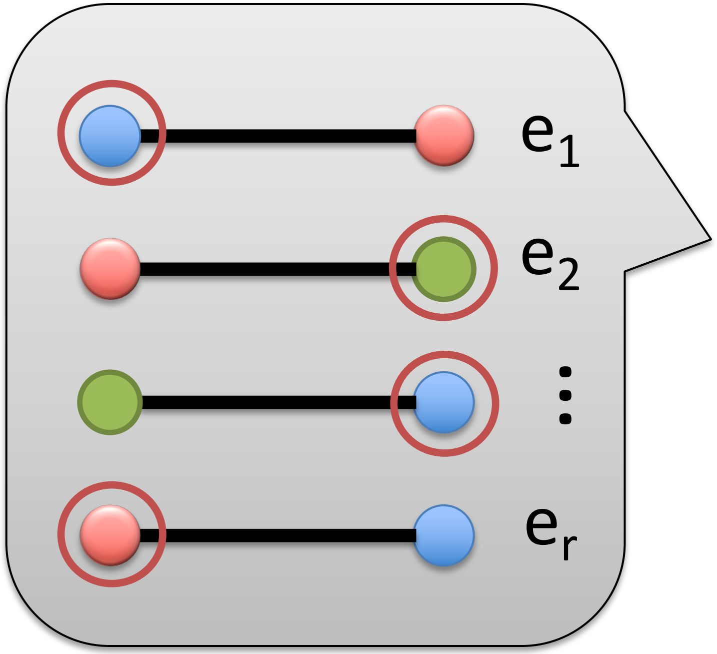


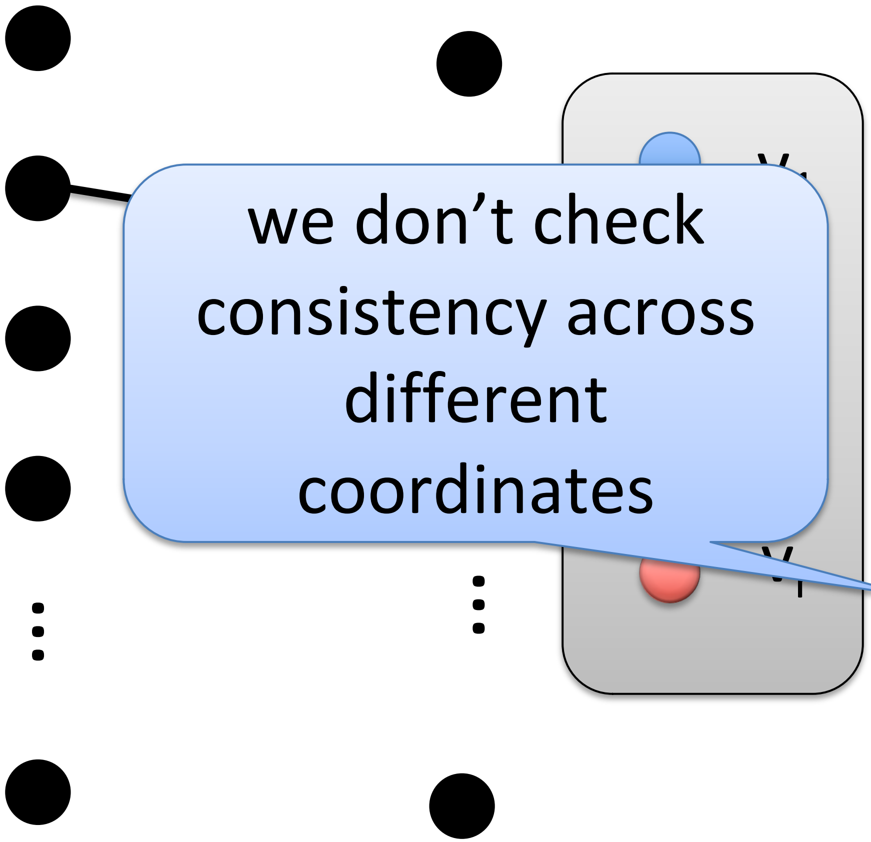
put a constraint iff $\forall i$, v_i is an endpoint of e_i.

A CSP Instance $\phi(G)$

r – number of repetitions







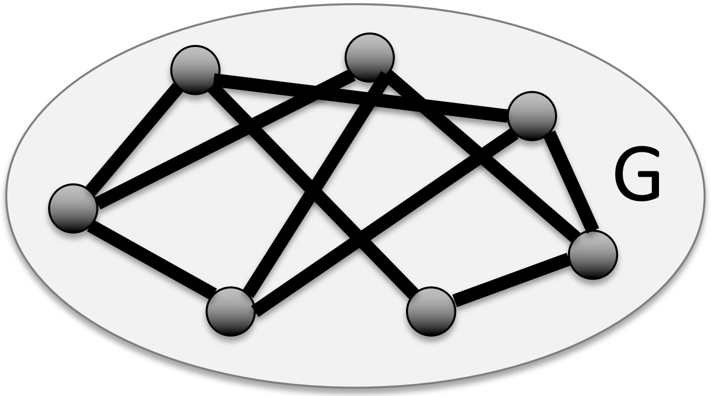
we don't check consistency across different coordinates

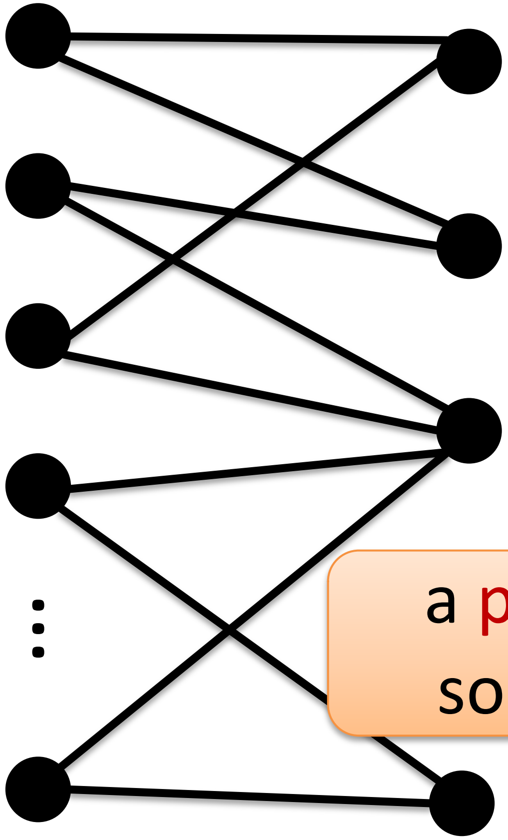
put a constraint iff $\forall i$, v_i is an endpoint of e_i.

constraint is satisfied iff $\forall i$, both assignments to v_i are the same

r – number of repetitions

A CSP Instance $\phi(G)$



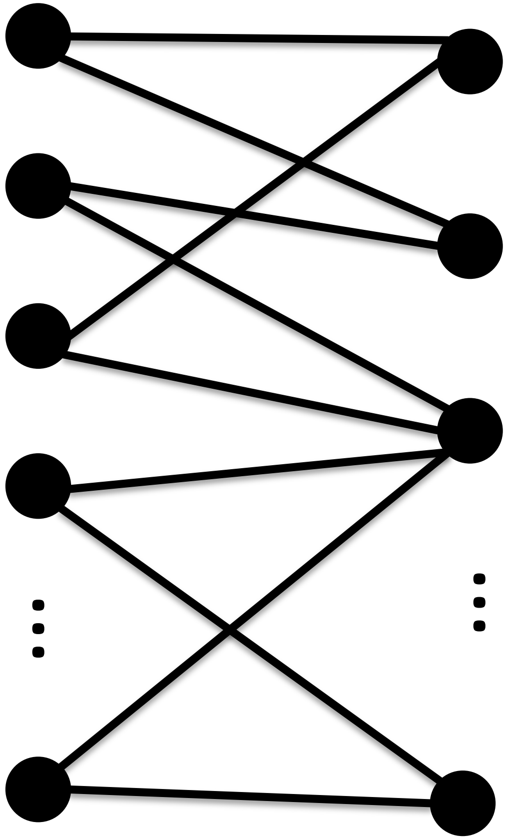


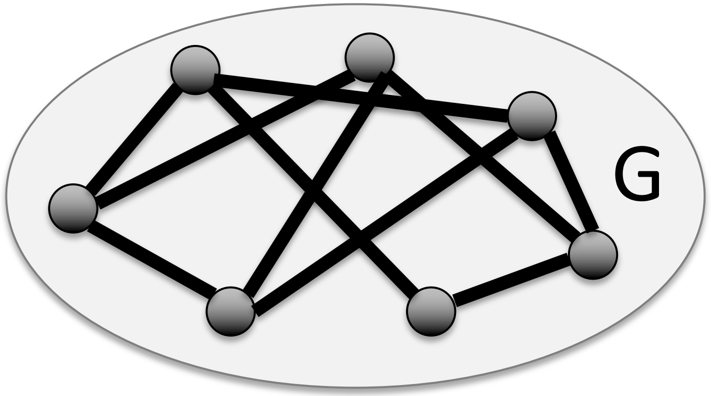
If G is a Yes-Instance, there is an assignment to variables satisfying all constraints

a perfect solution

r – number of repetitions

A CSP Instance ϕ(G)



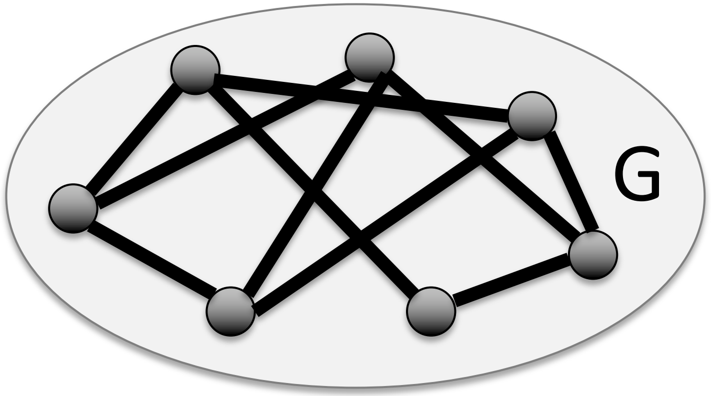


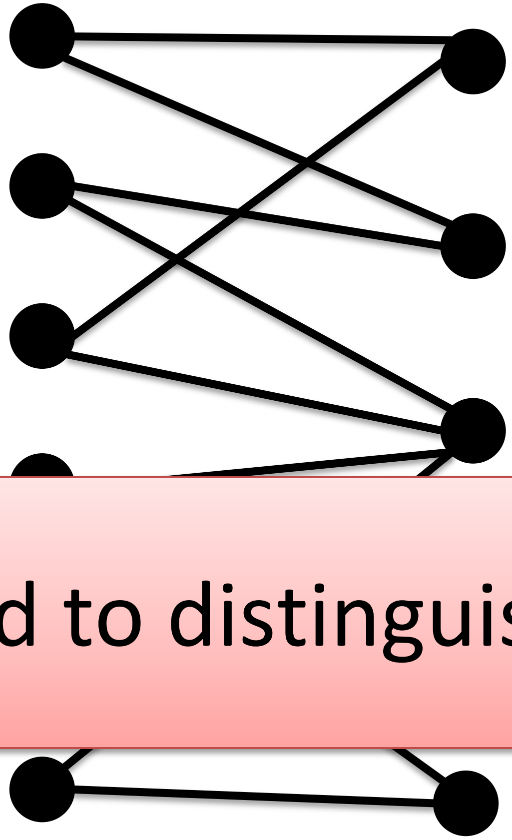
If G is a Yes-Instance, there is an assignment to variables satisfying all constraints

If G is a No-Instance, any assignment satisfies $\leq 1/2^{\Omega(r)}$-fraction of constraints

r – number of repetitions

A CSP Instance $\phi(G)$





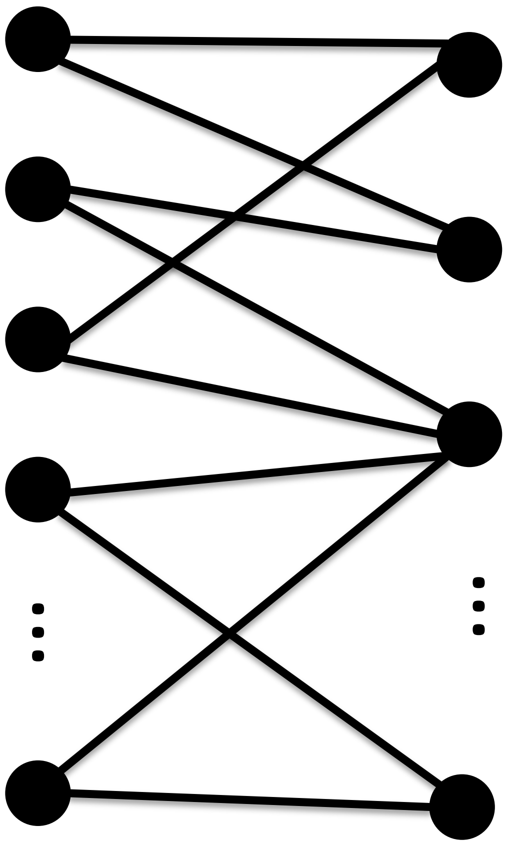
NP-hard to distinguish

If G is a Yes-Instance, there is an assignment to variables satisfying all constraints

If G is a No-Instance, any assignment satisfies $\leq 1/2^{\Omega(r)}$-fraction of constraints

Bonus Property for Yes Instance!

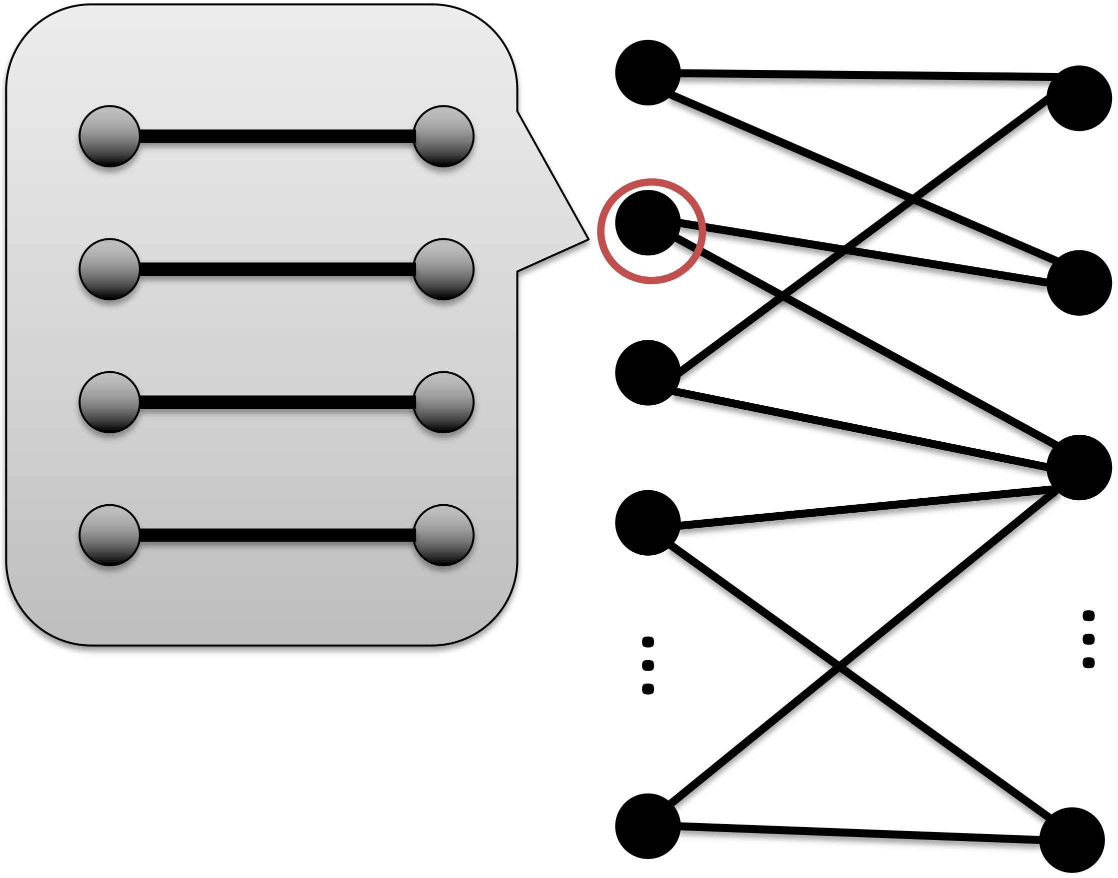
r – number of repetitions

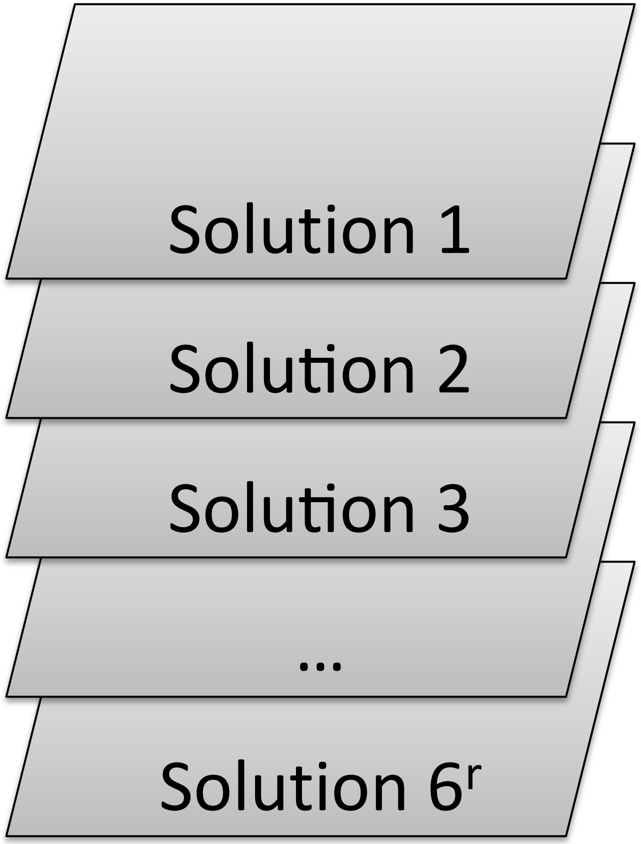


Perfect Solution

r – number of repetitions

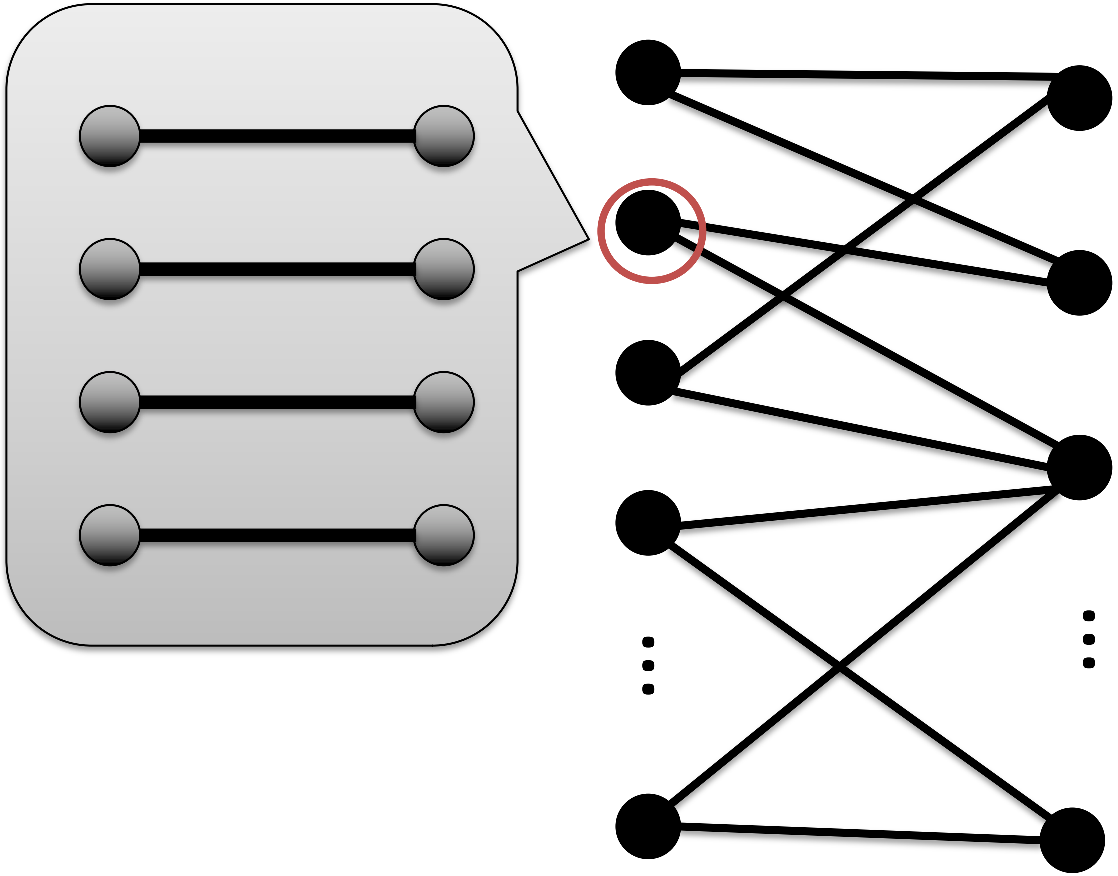
Bonus Property for Yes Instance!

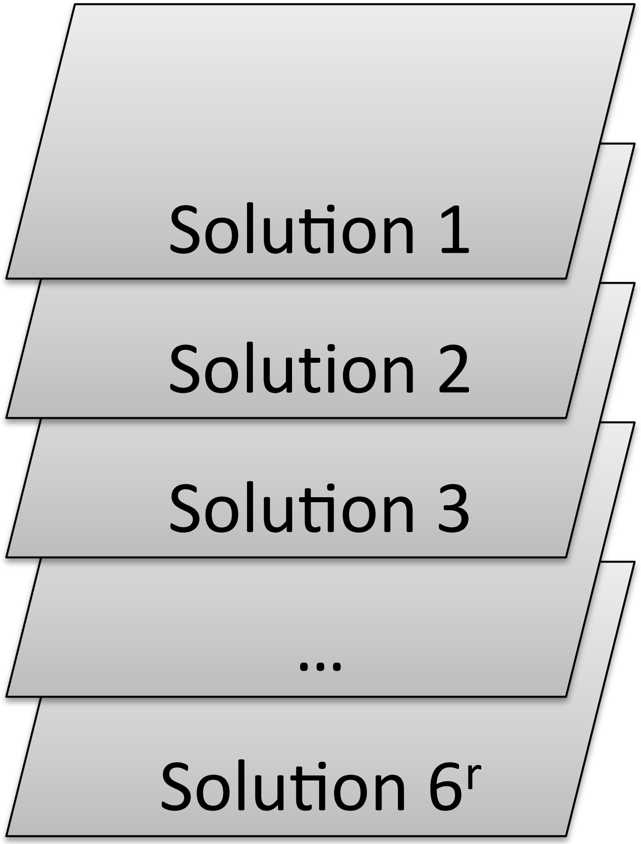




r – number of repetitions

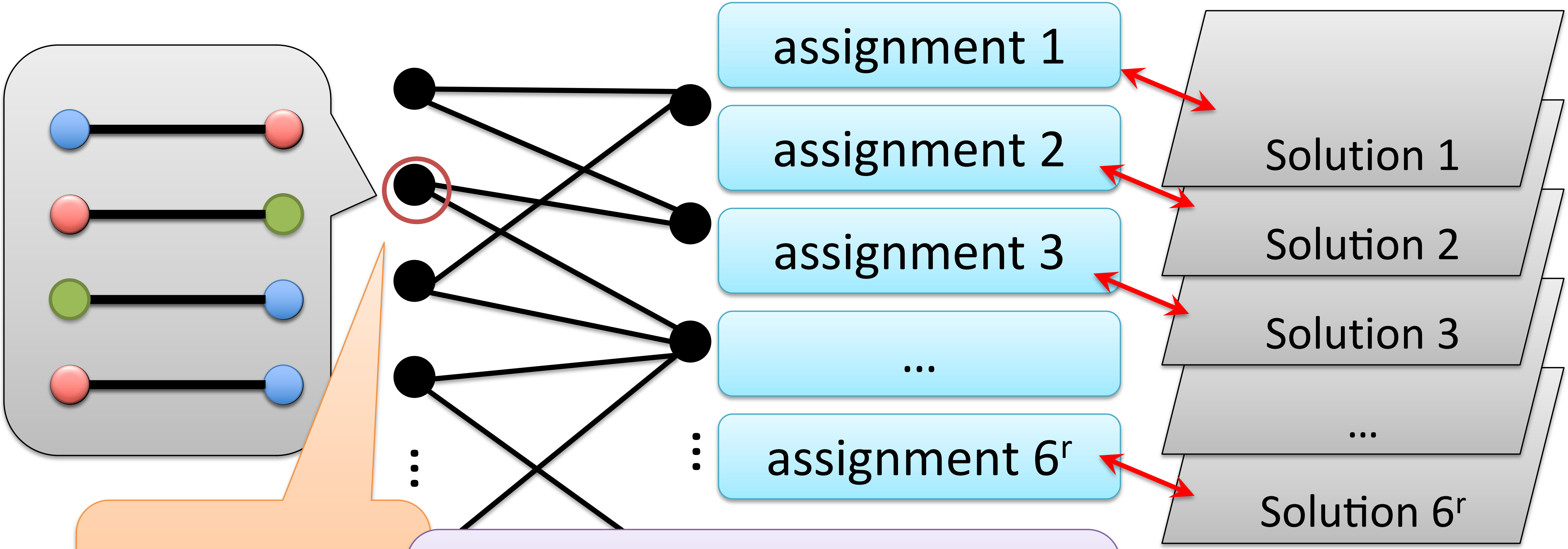
Bonus Property for Yes Instance!





r – number of repetitions

Bonus Property for Yes Instance!

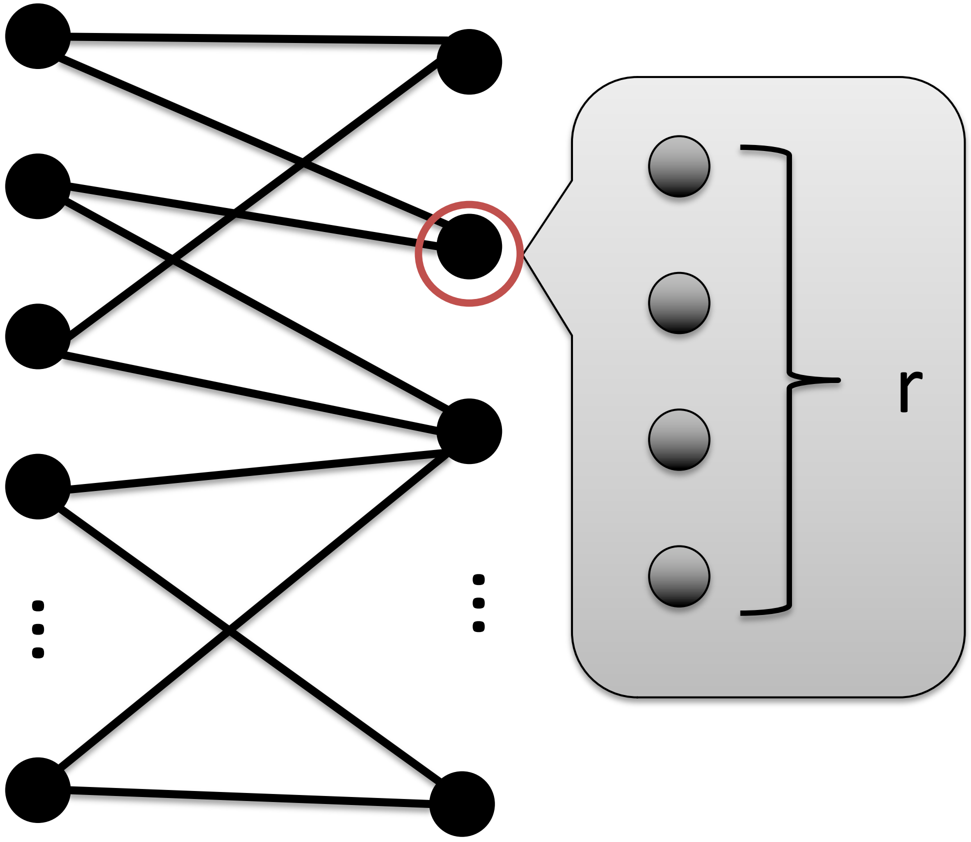


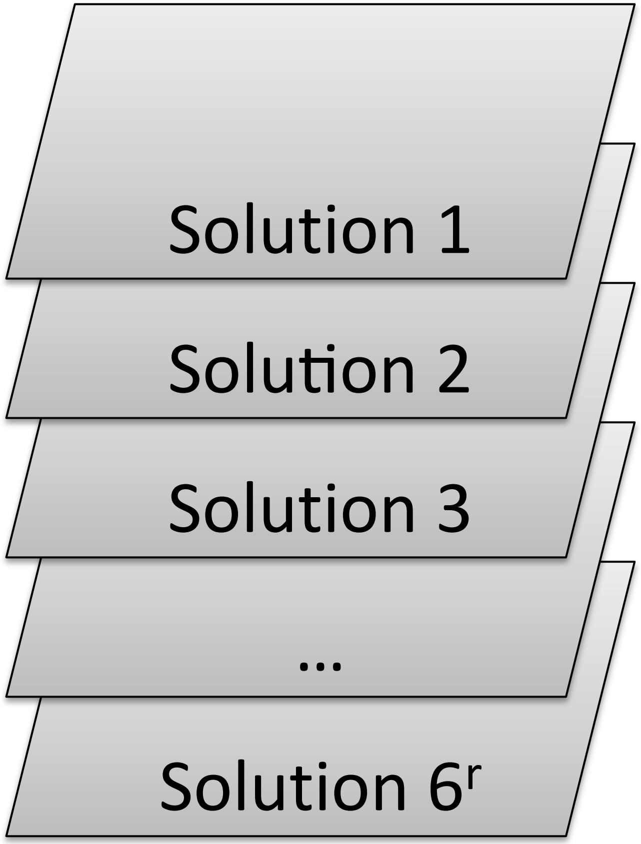
6^r possible assignments

Each assignment appears in exactly 1 solution!

r – number of repetitions

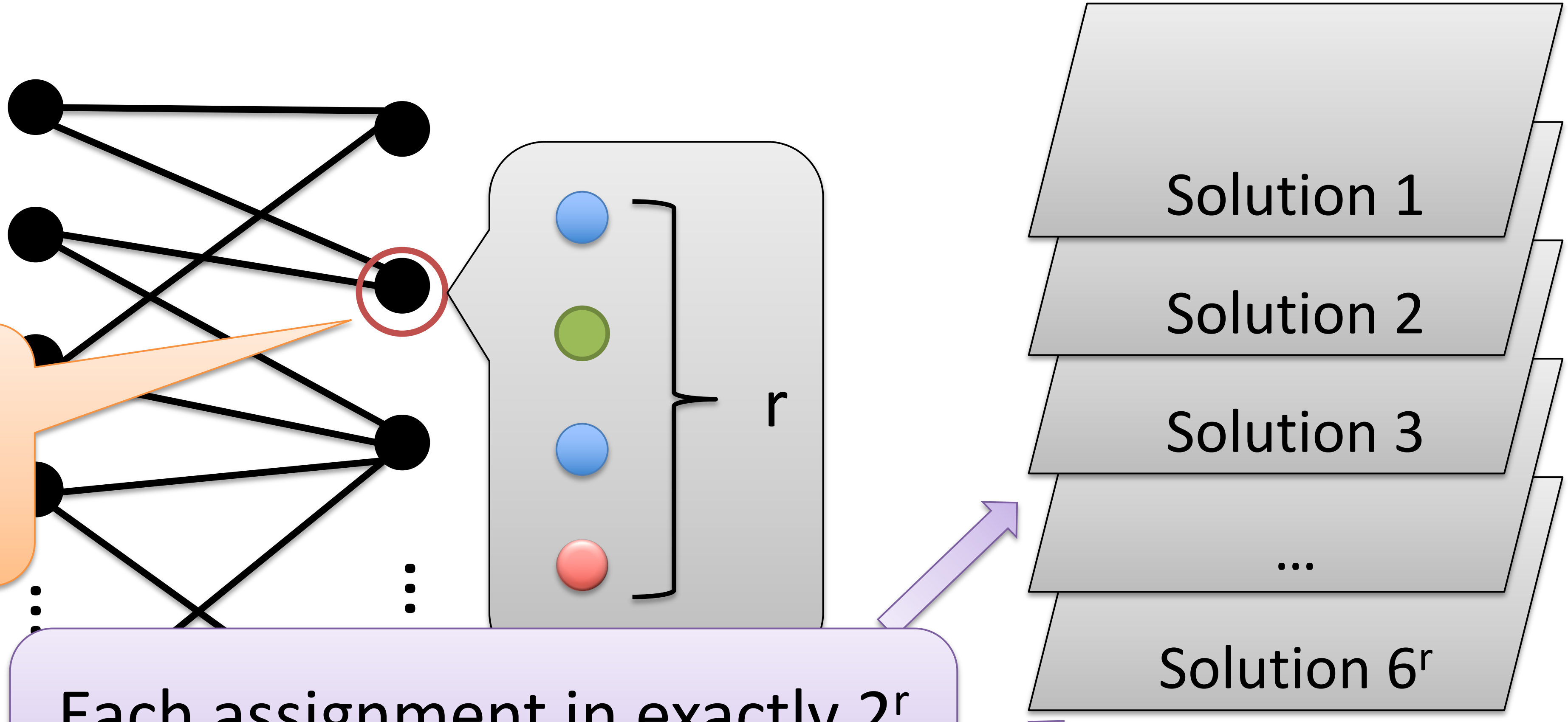
Bonus Property for Yes Instance!





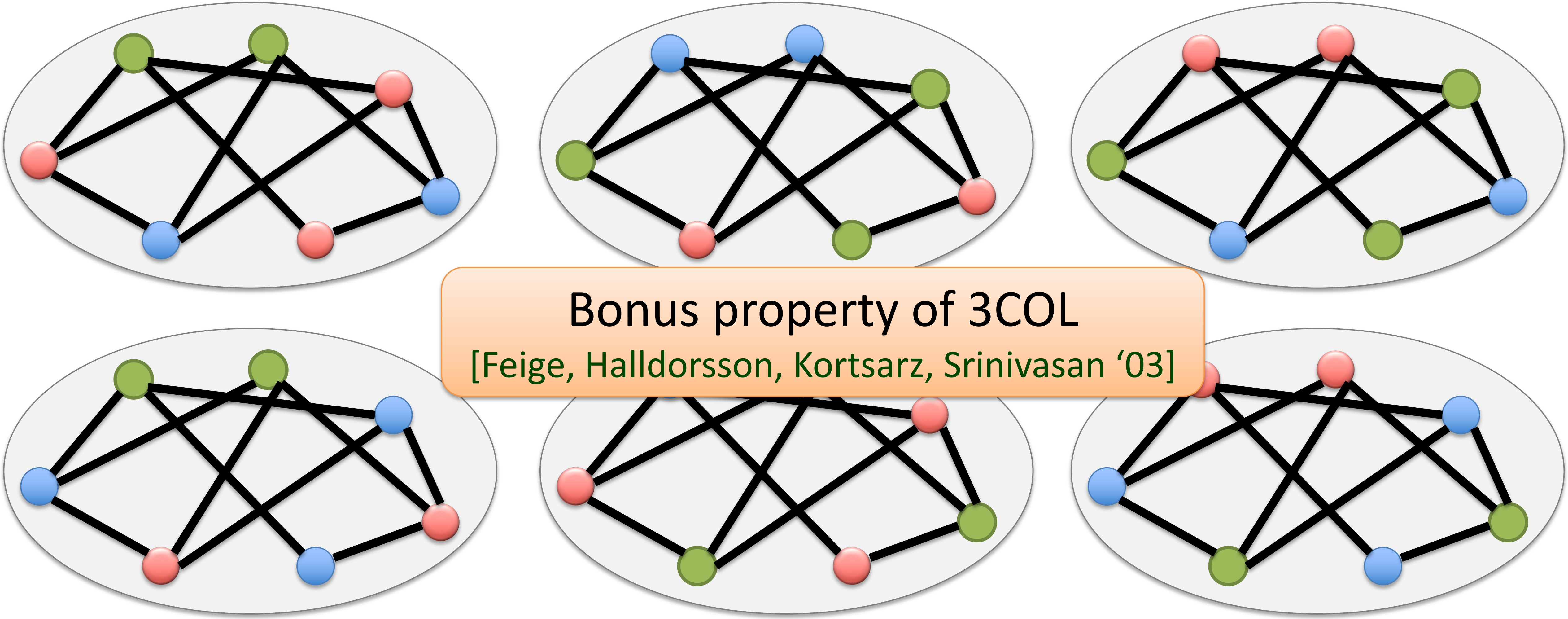
Bonus Property for Yes Instance!

r – number of repetitions



3^r possible assignments

Each assignment in exactly 2^r solutions!

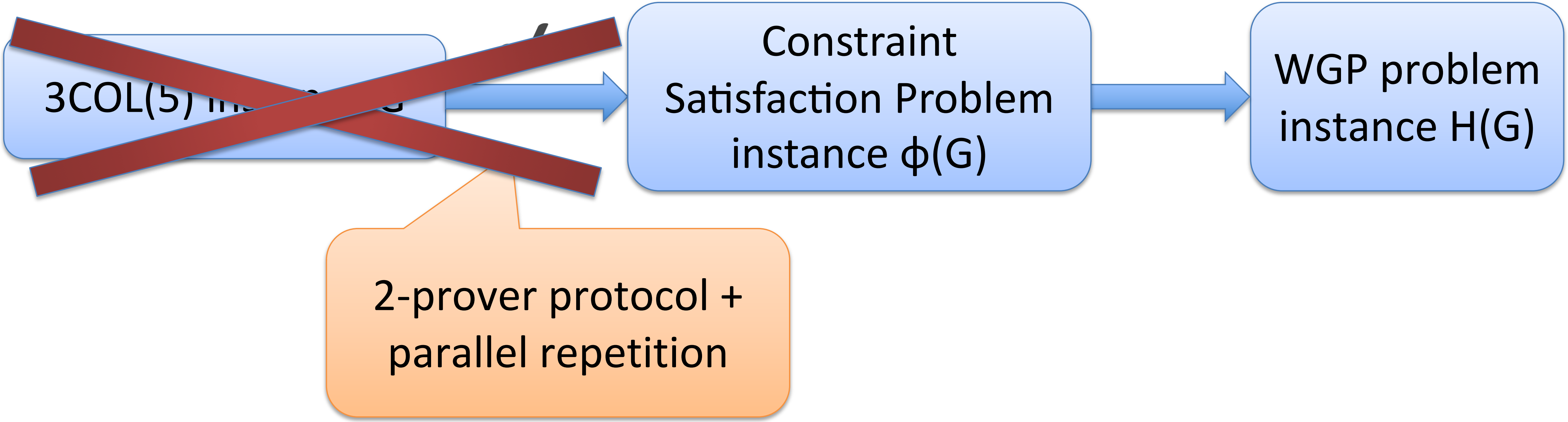


Bonus property of 3COL

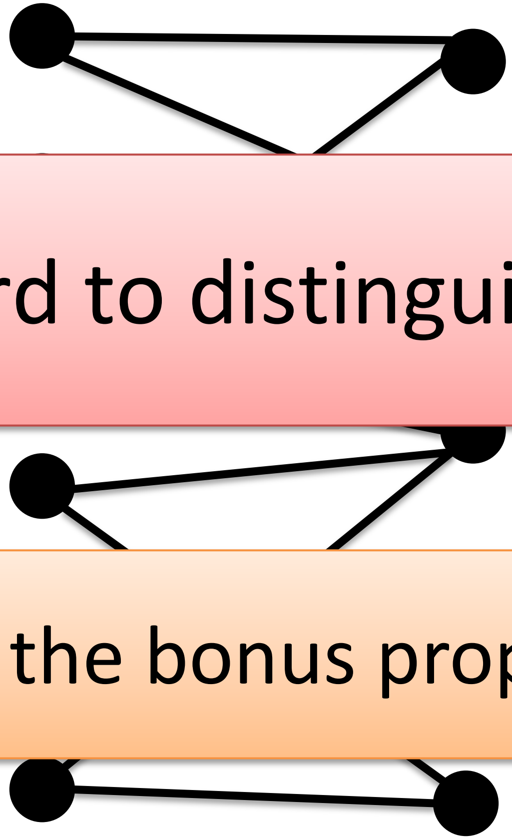
[Feige, Halldorsson, Kortsarz, Srinivasan ‘03]

- For every edge, each legal coloring appears exactly once
- For each vertex, every coloring appears exactly twice

Next Steps


~~3COL(5)~~
Constraint Satisfaction Problem instance ϕ(G)
WGP problem instance H(G)
2-prover protocol + parallel repetition

A CSP Problem Instance ϕ



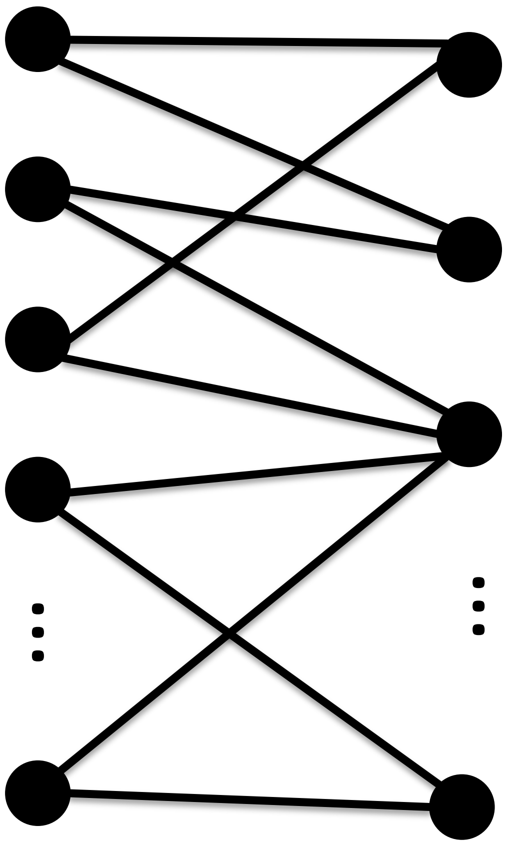
NP-hard to distinguish

ϕ is a Yes-Instance, if there is an assignment to variables satisfying all constraints

ϕ is a No-Instance, if any assignment satisfies $\leq 1/2^{\Omega(r)}$-fraction of constraints

+ the bonus property

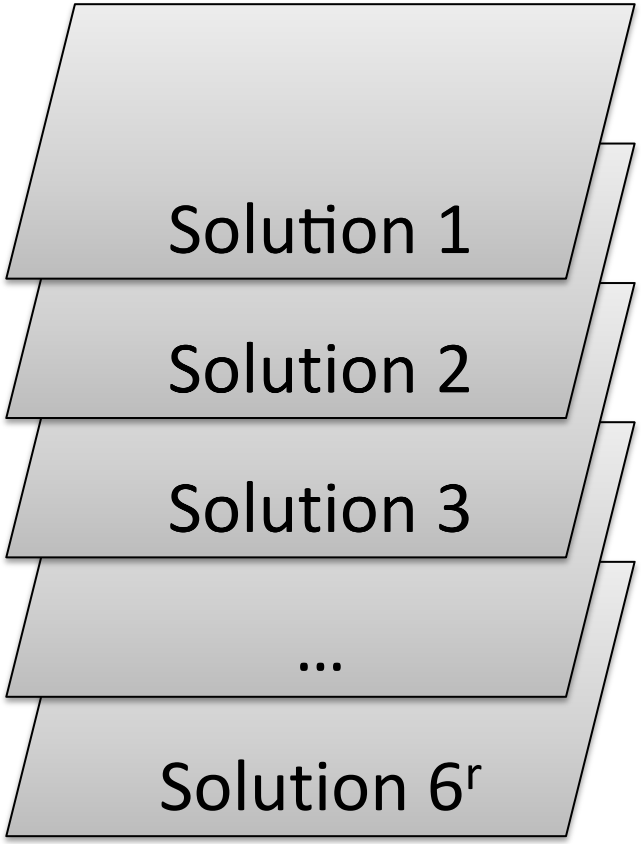
Bonus Property



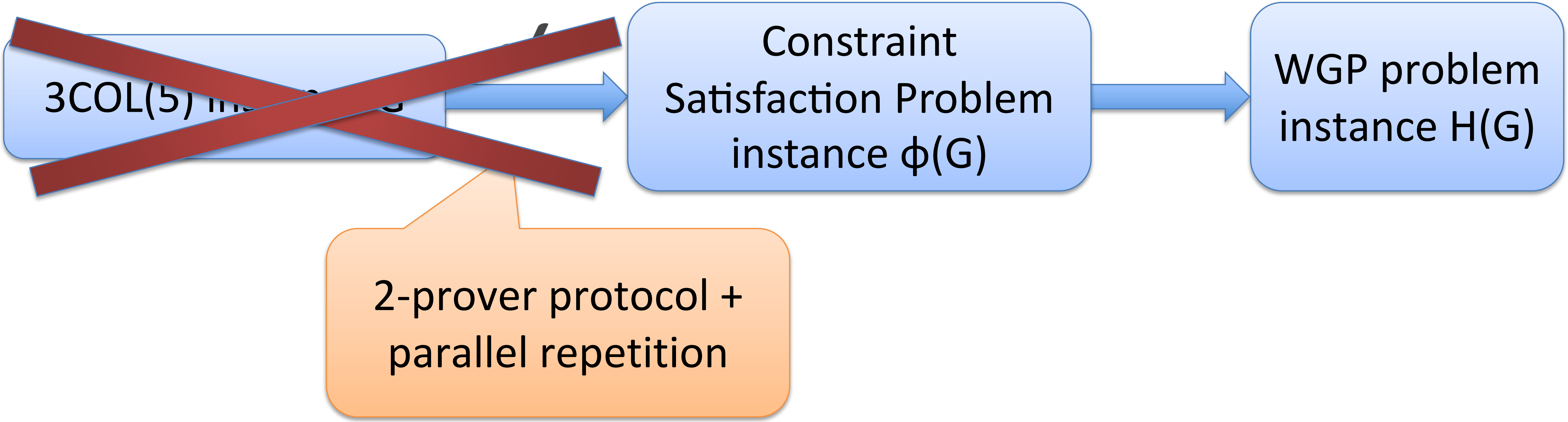
ϕ is a Yes-Instance, there are 6^r perfect solutions

for each var on left each assignment appears in 1 solution

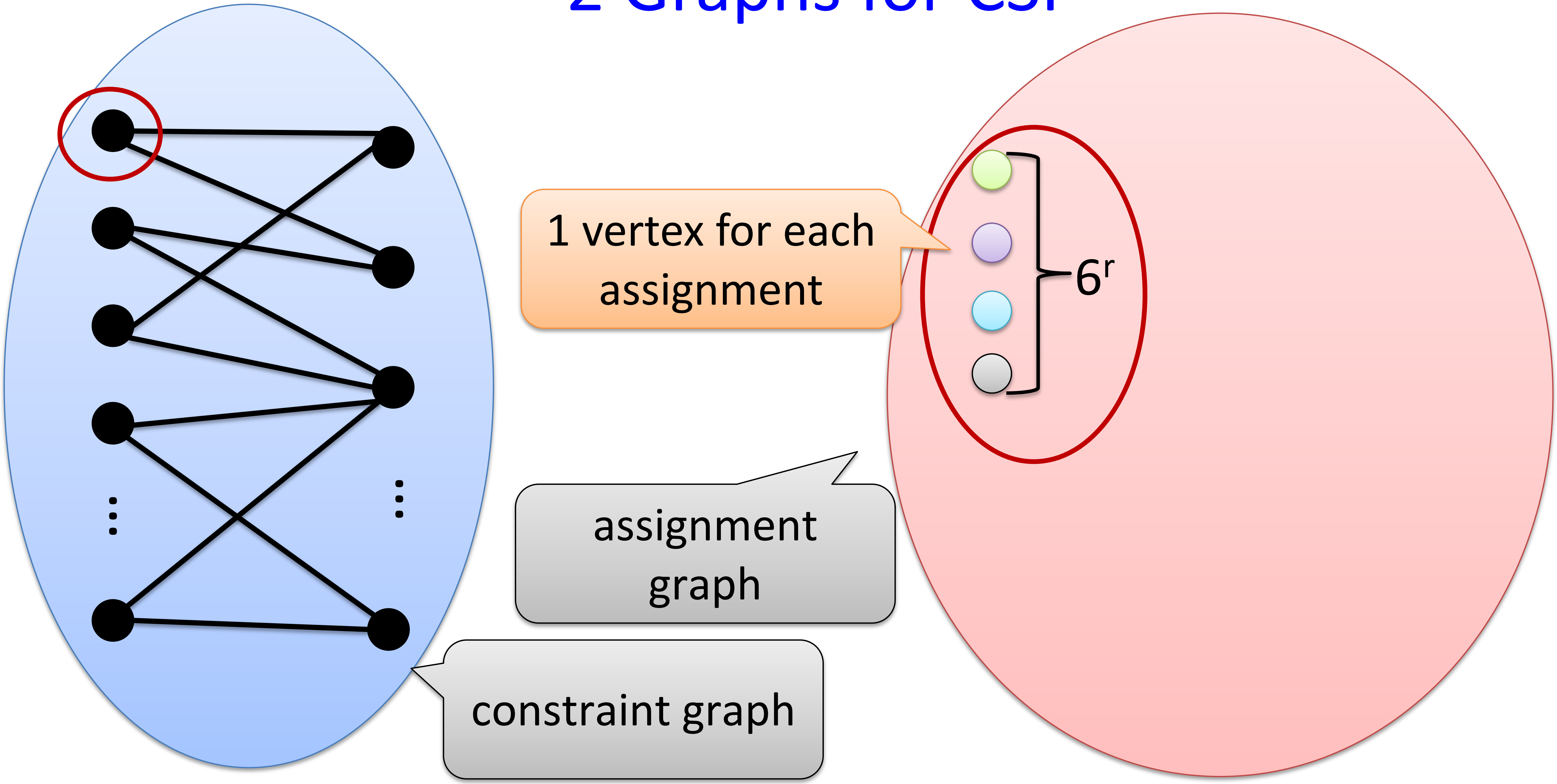
for each var on right each assignment appears in 2^r solution



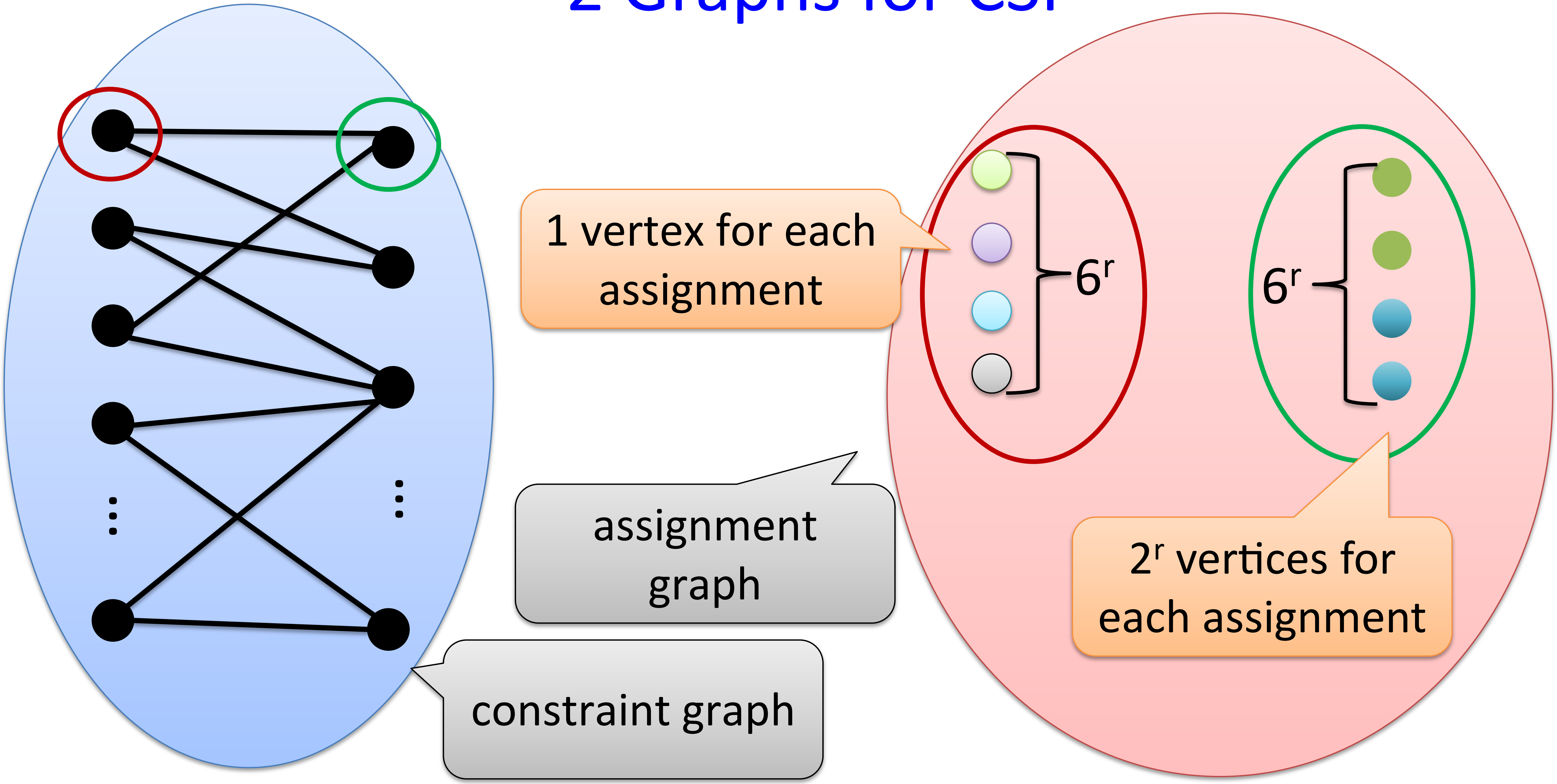
Next Steps


3COL(5)
Constraint Satisfaction Problem instance ϕ(G)
WGP problem instance H(G)
2-prover protocol + parallel repetition

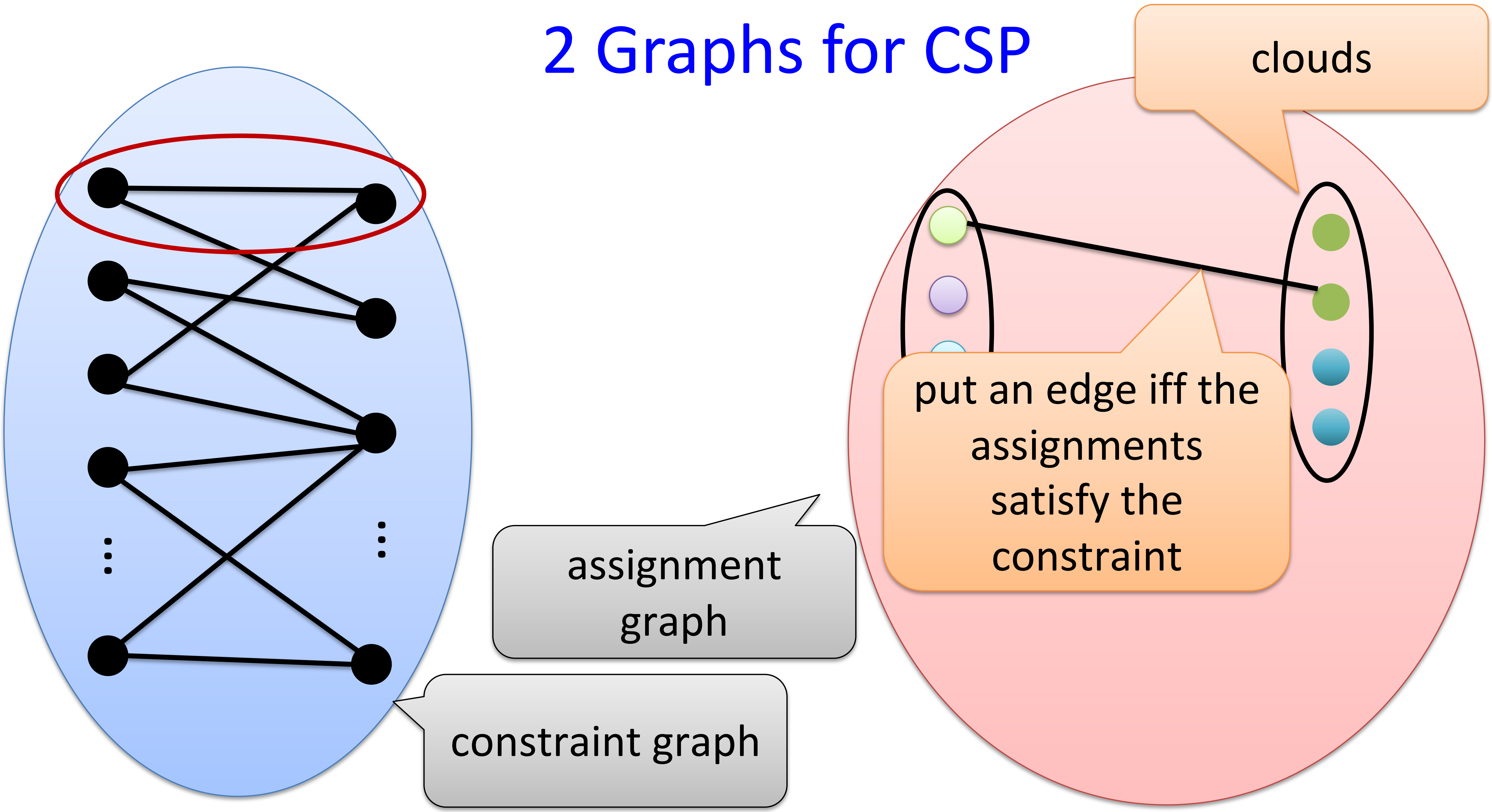
2 Graphs for CSP



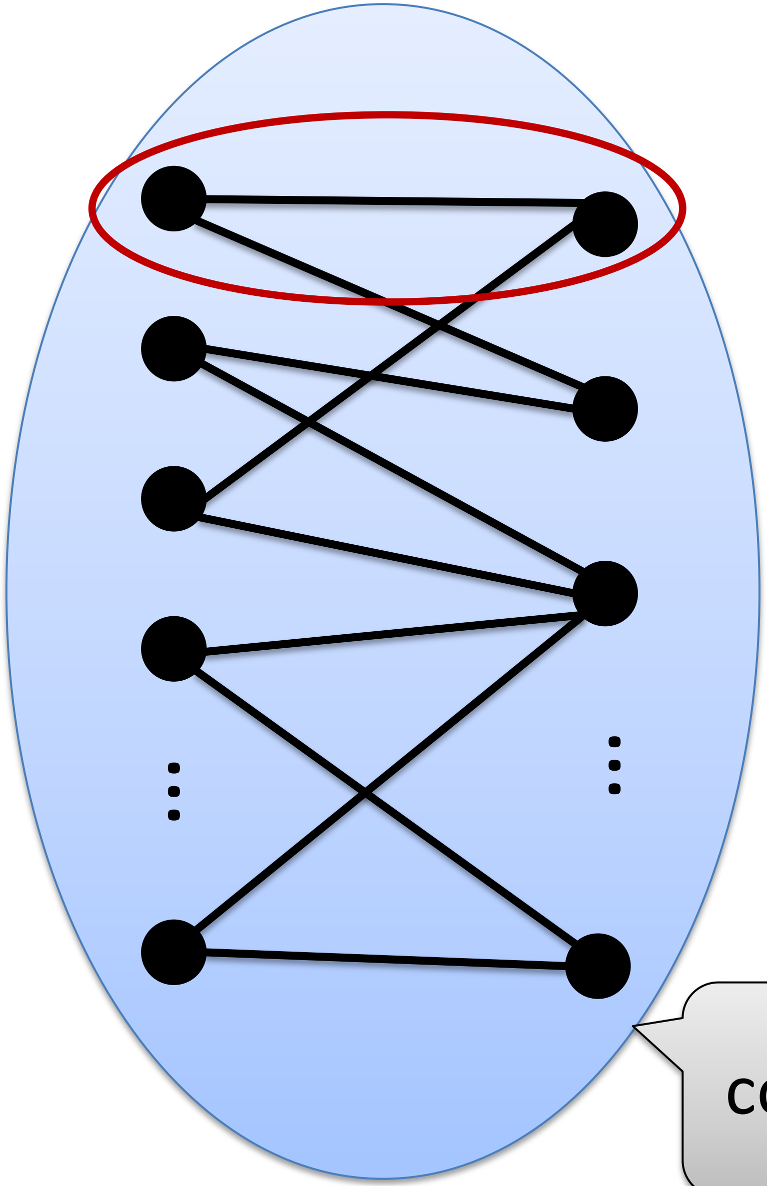
2 Graphs for CSP

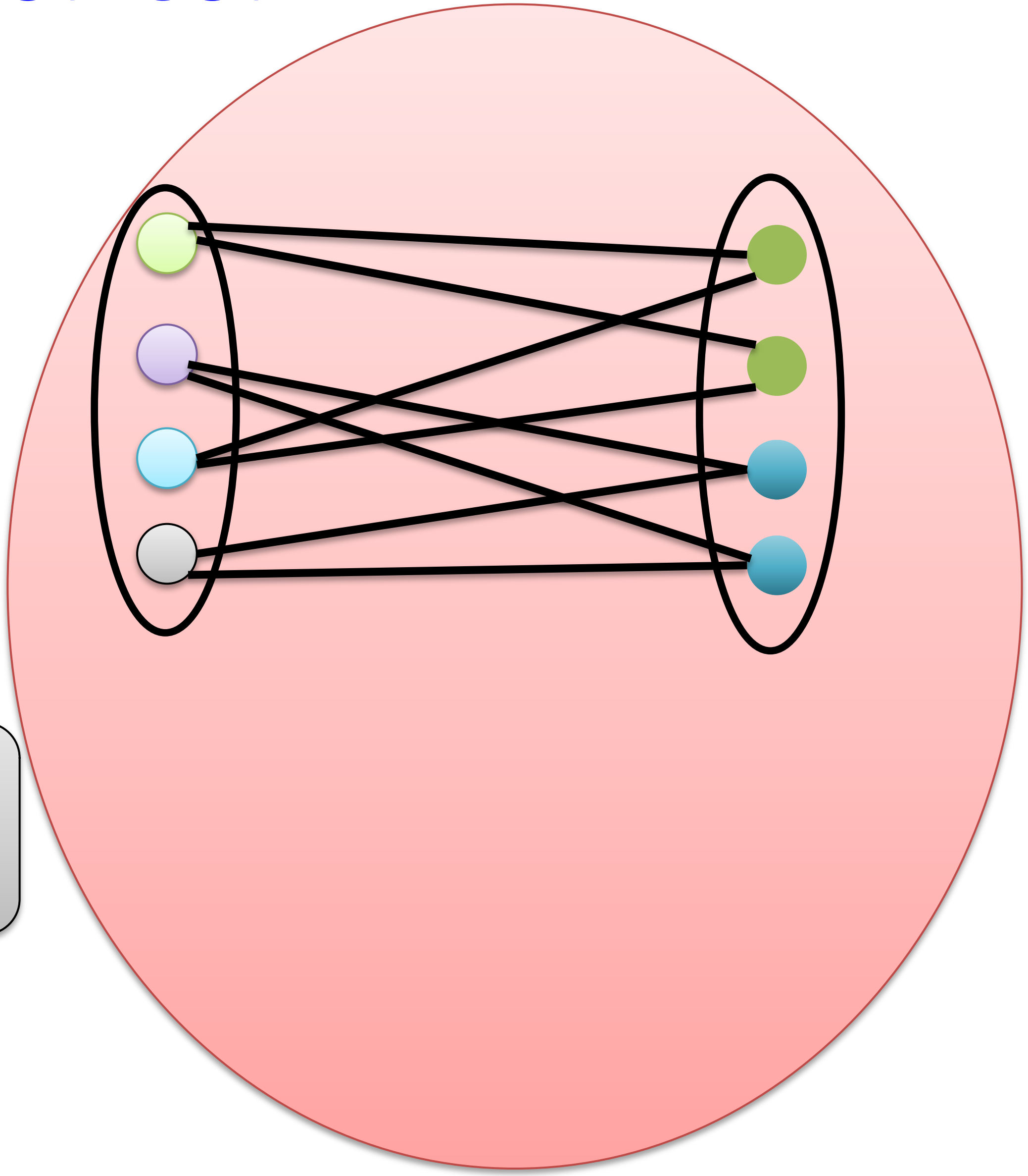


2 Graphs for CSP



2 Graphs for CSP

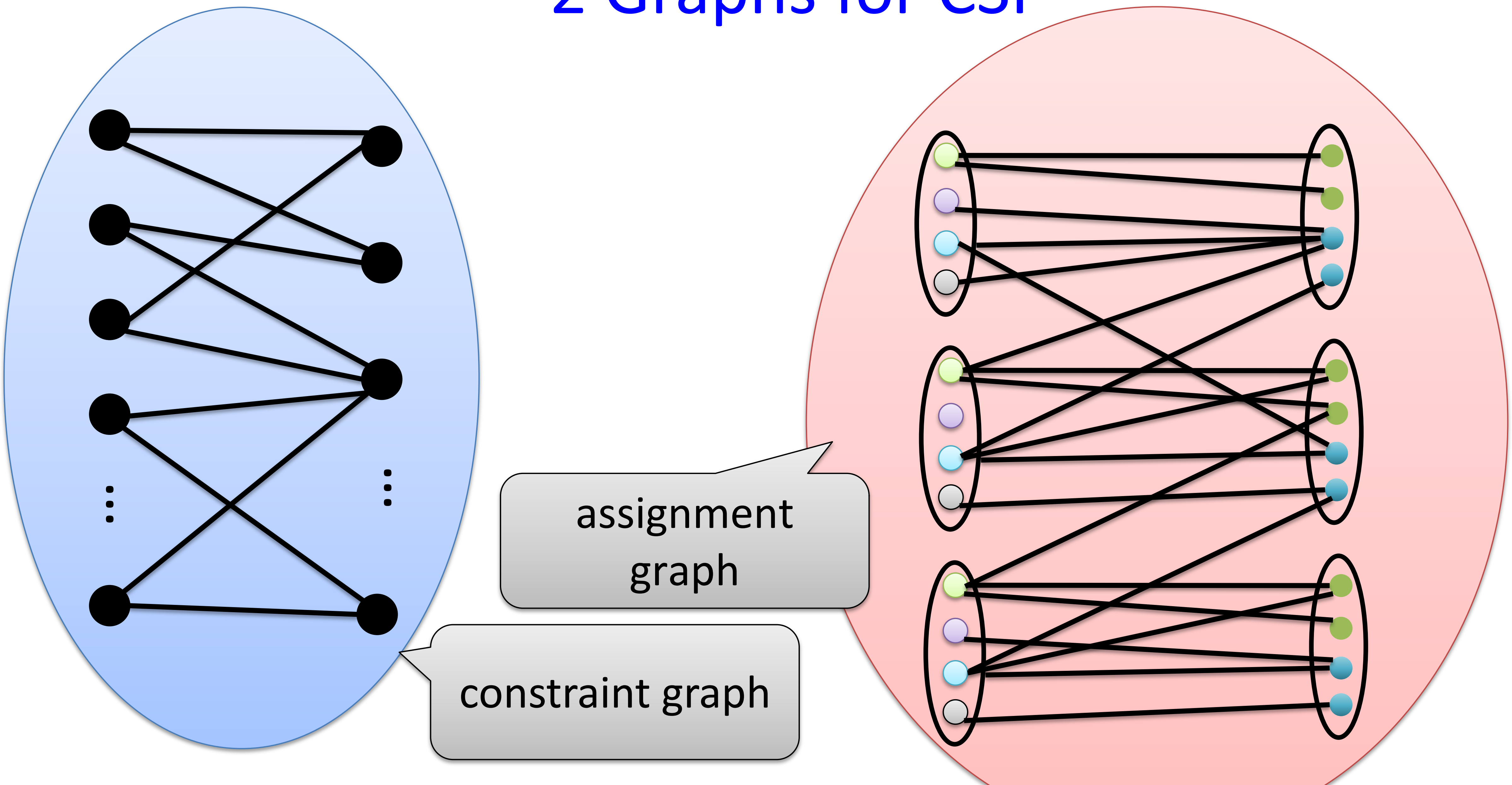




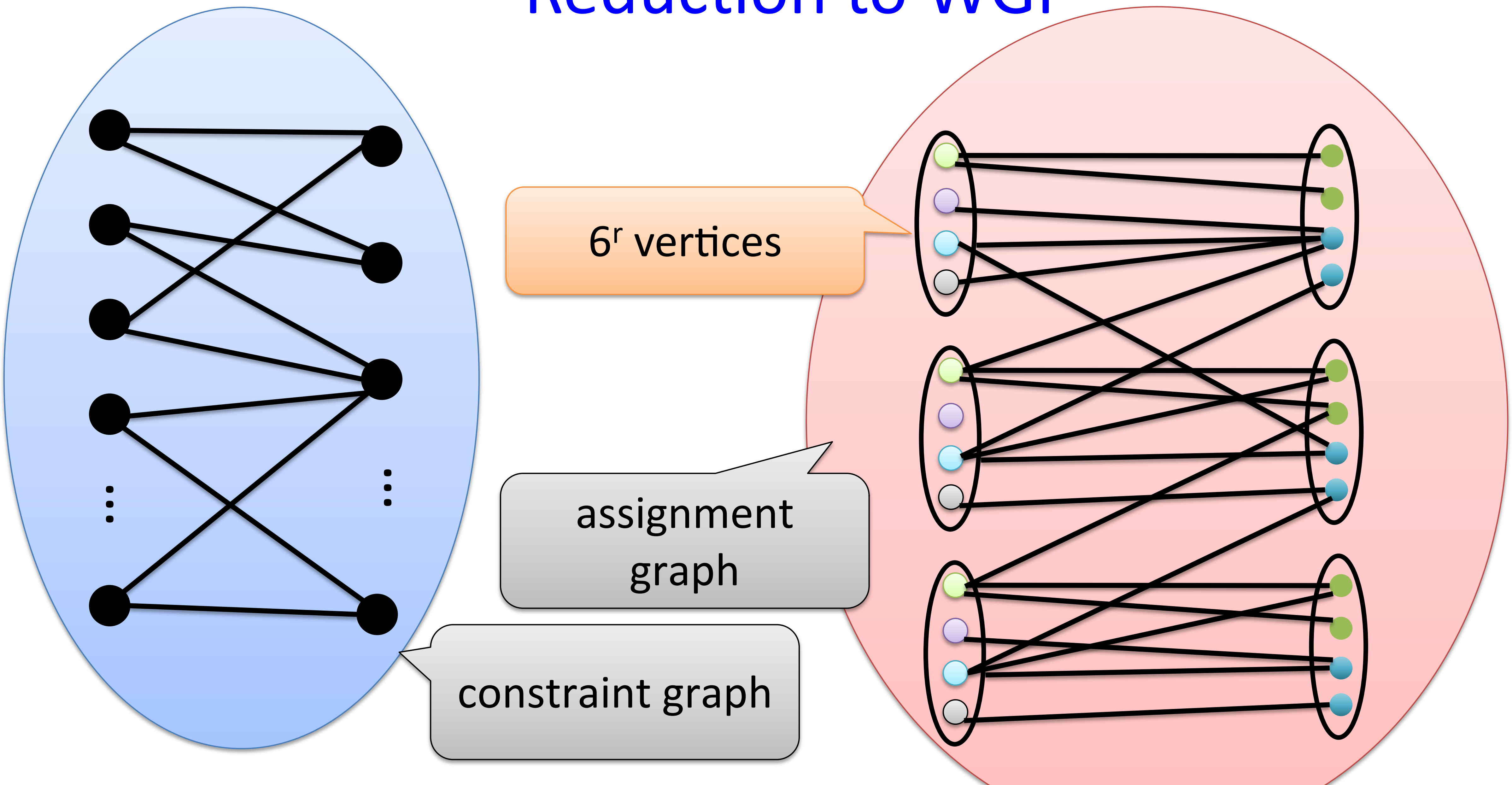
assignment graph

constraint graph

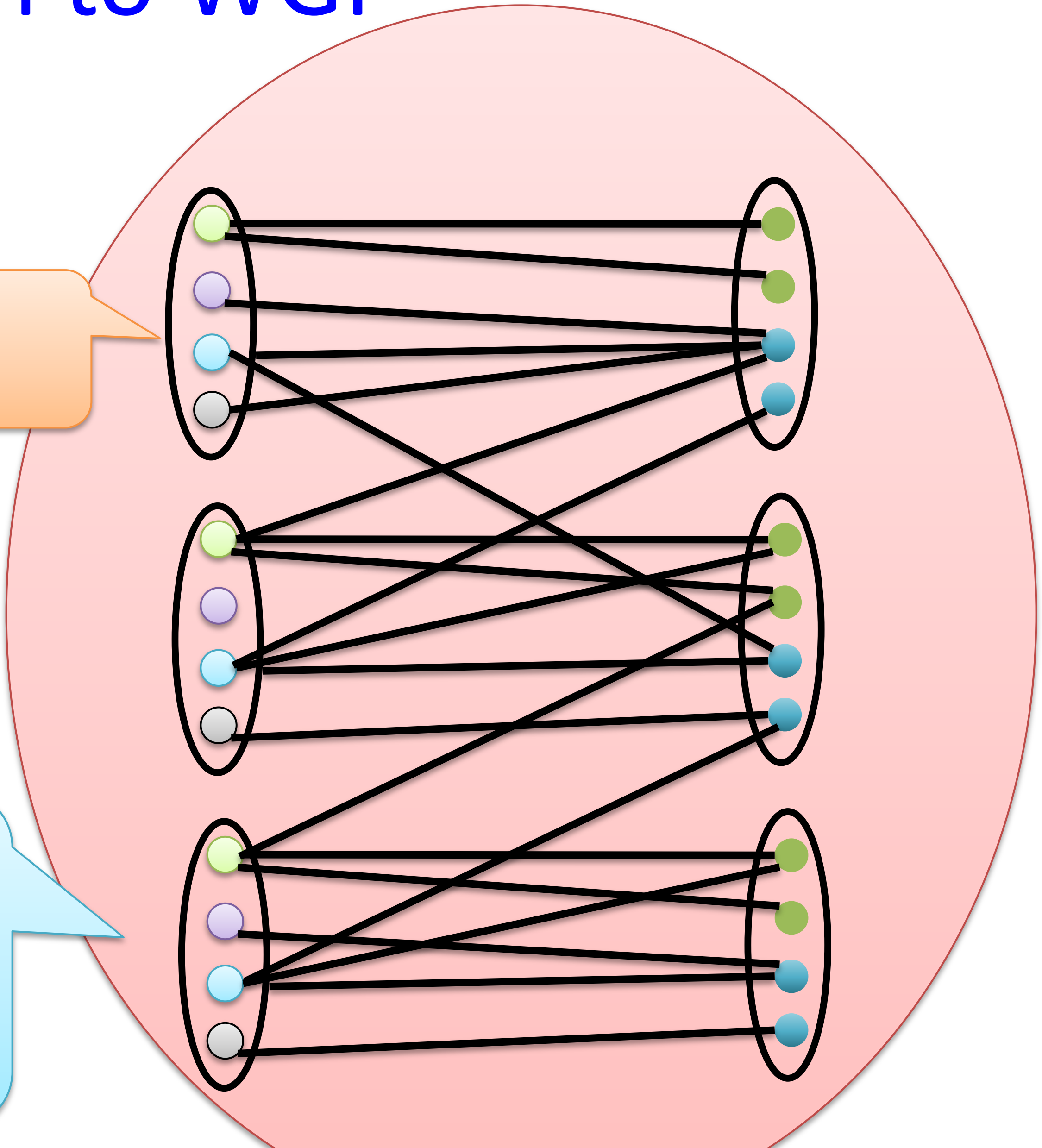
2 Graphs for CSP



Reduction to WGP



Reduction to WGP

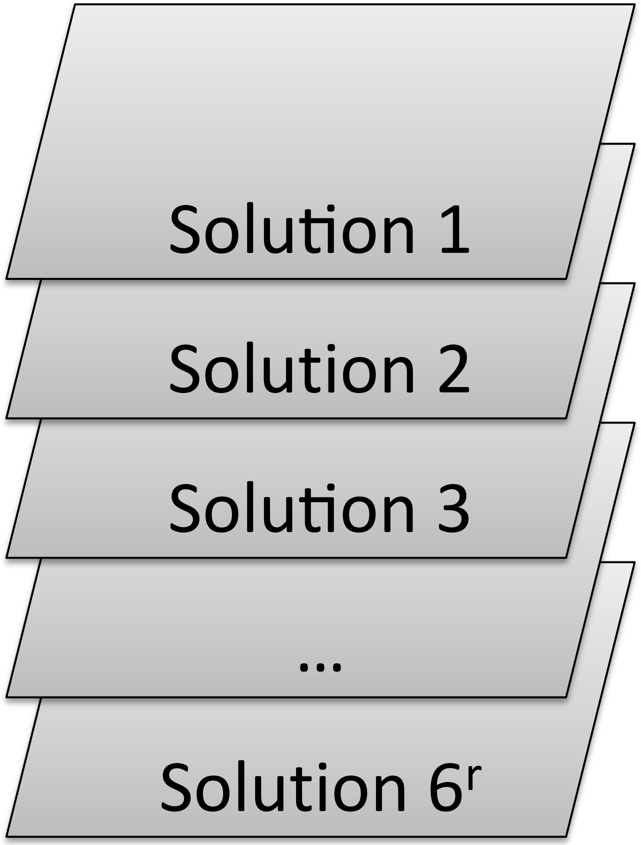


6^r vertices

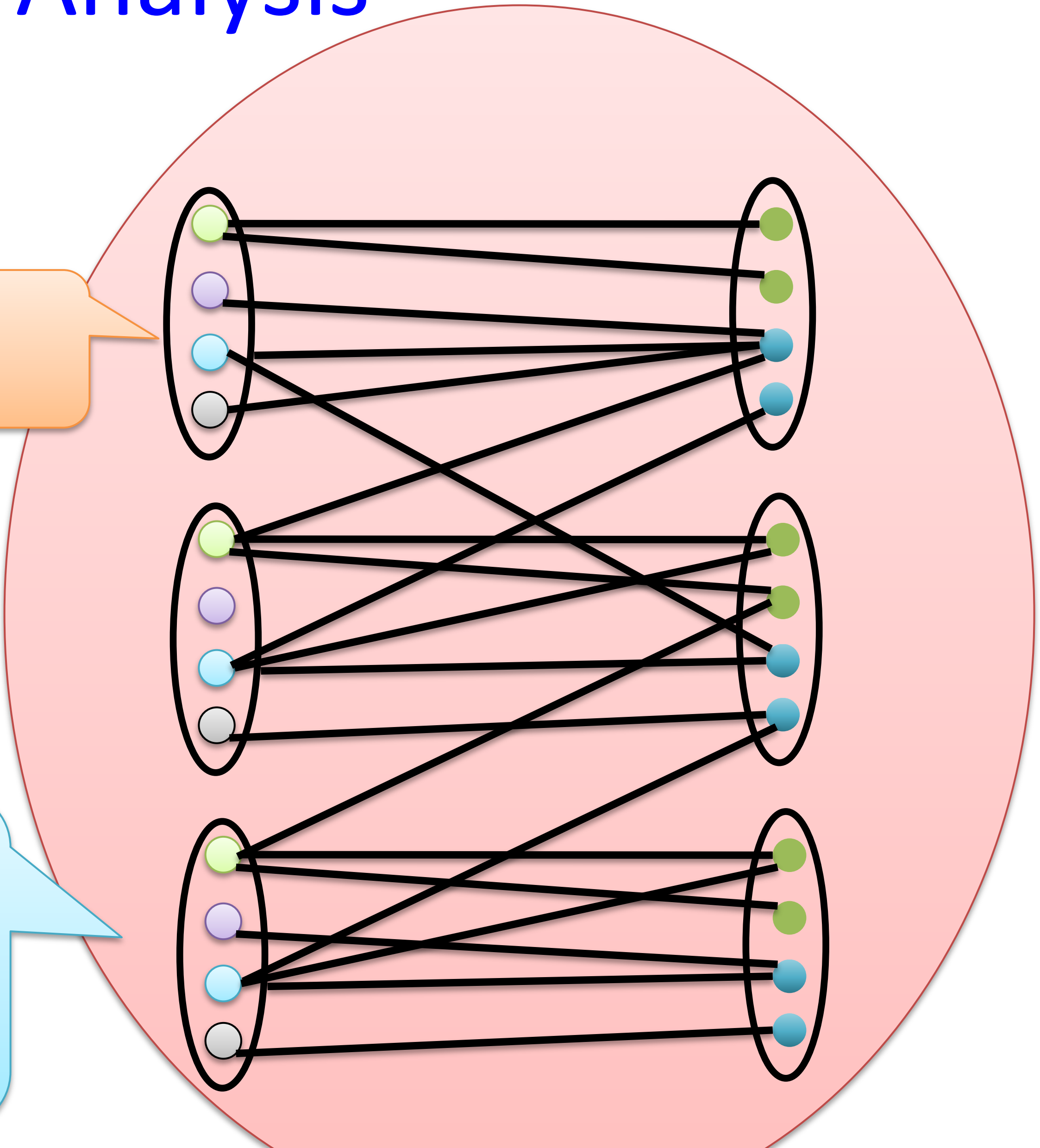
Input to WGP problem

- $p=6^r$
- L=#constraints

Yes Case Analysis



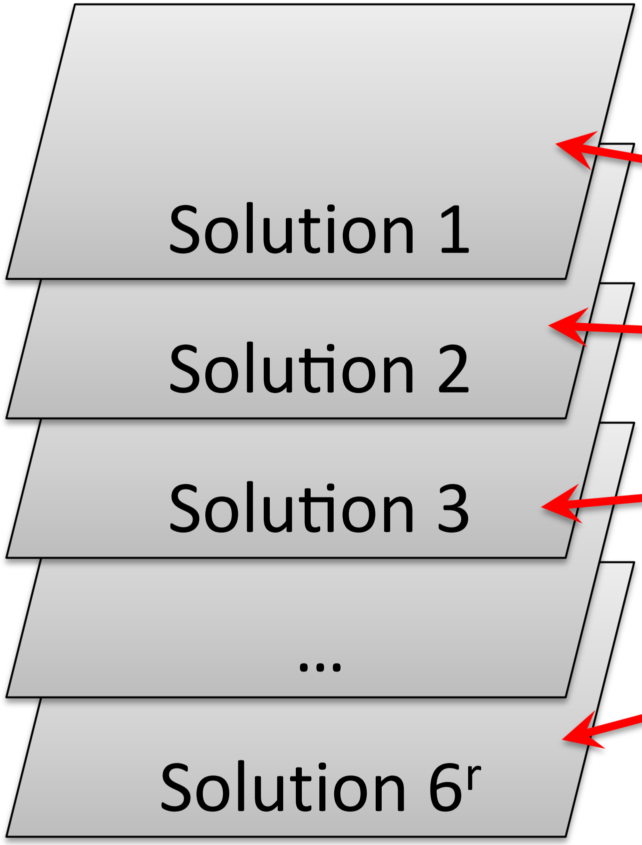
6^r vertices

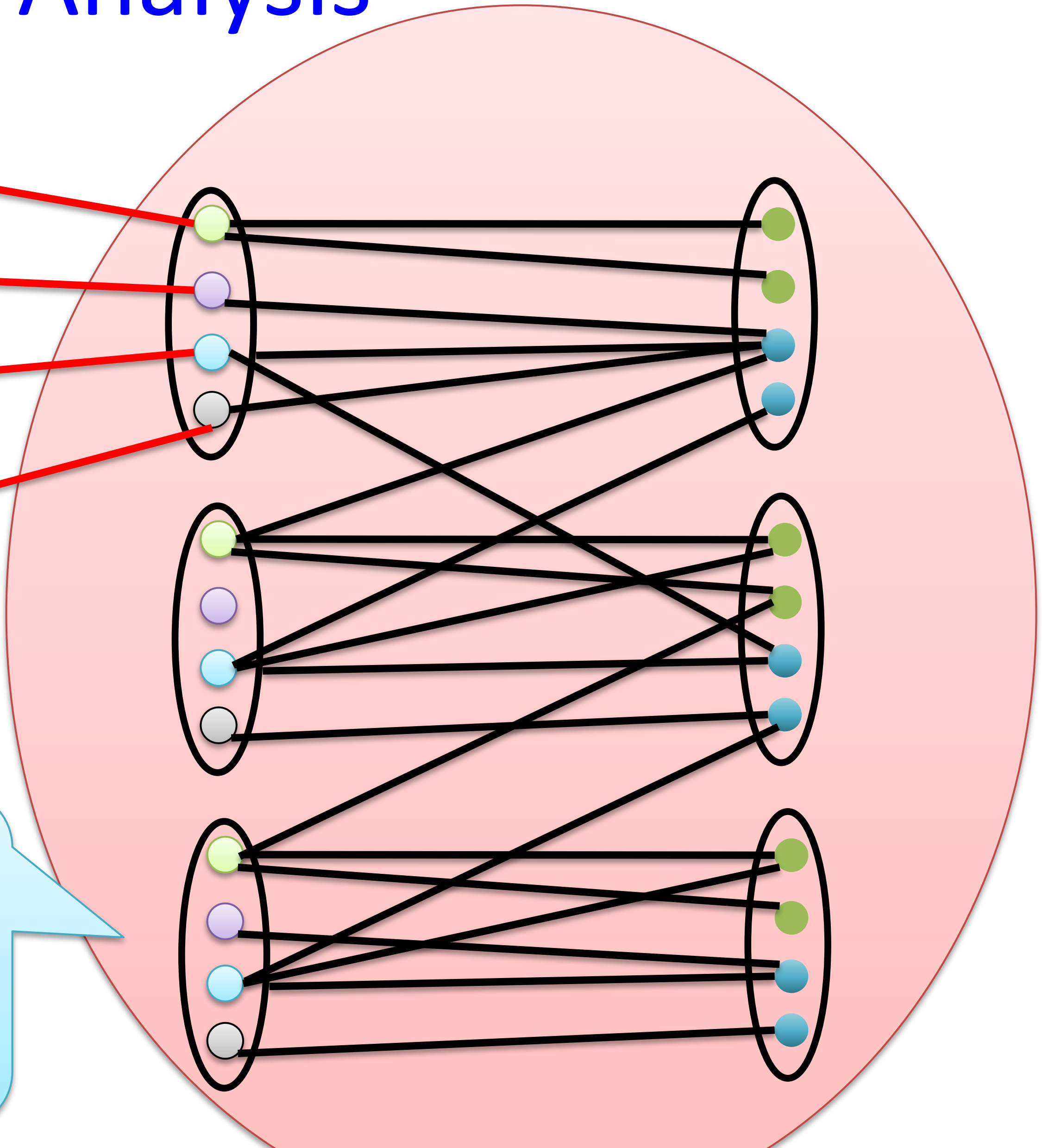


Input to WGP problem

- $p=6^r$
- L=#constraints

Yes Case Analysis

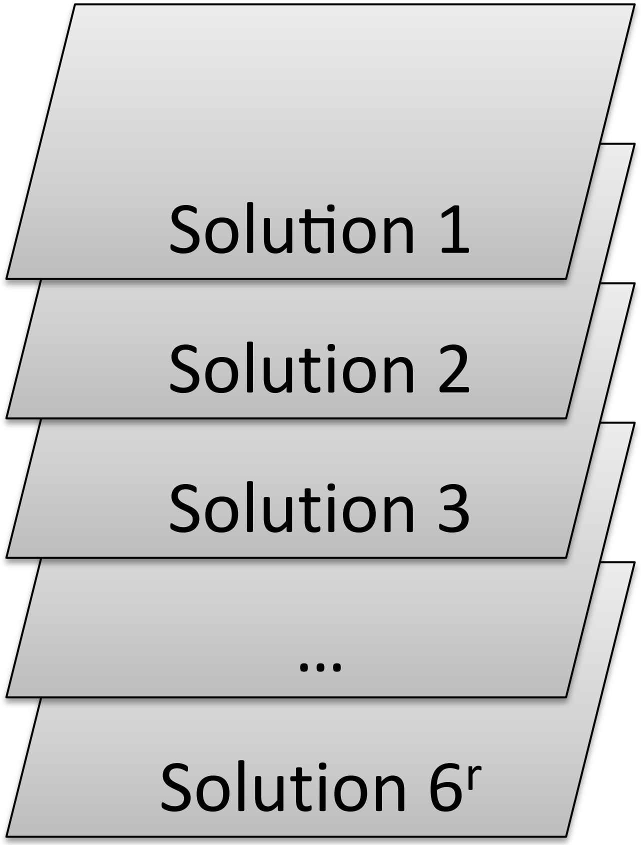


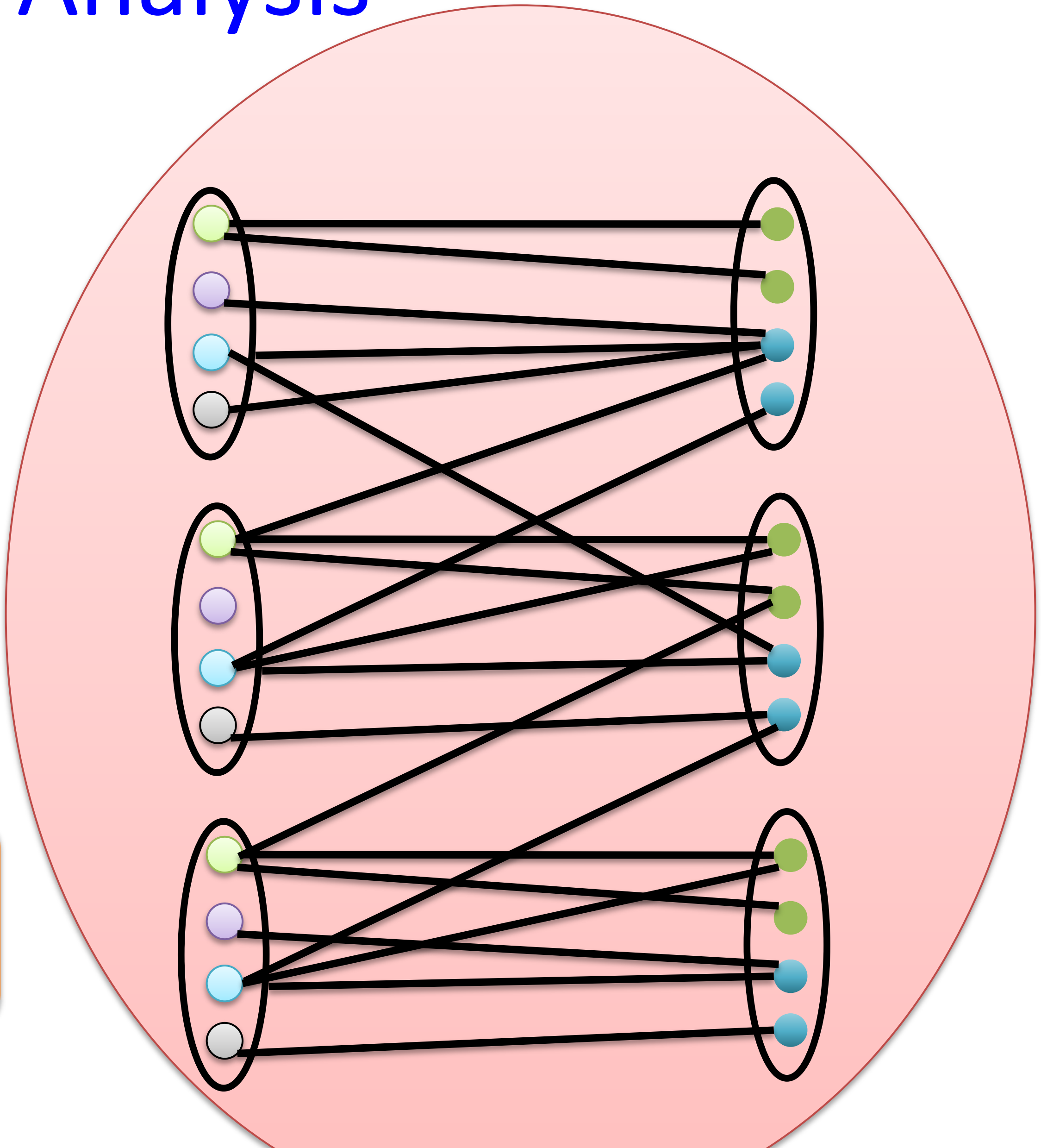


Input to WGP problem

- $p=6^r$
- L=#constraints

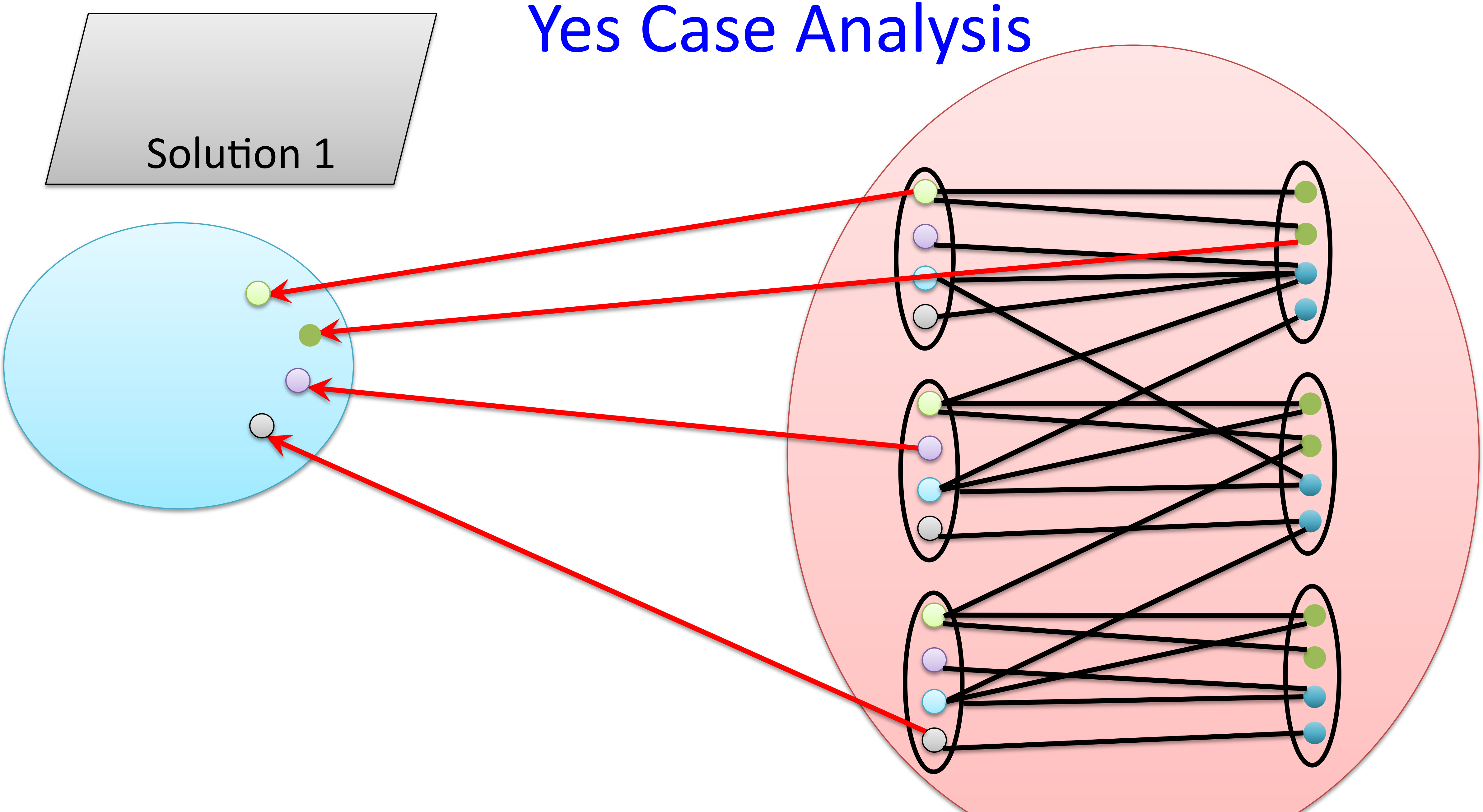
Yes Case Analysis





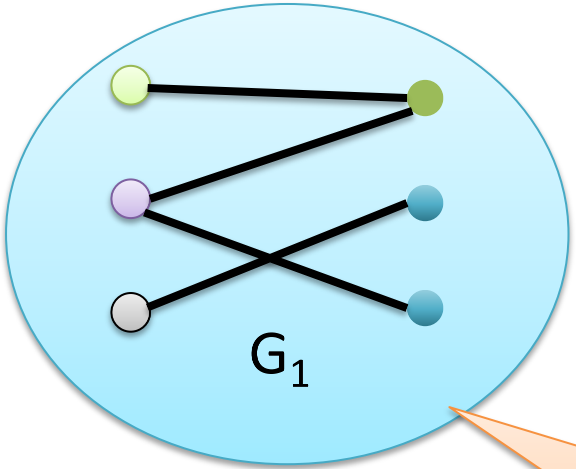
Each solution defines a piece in the partition

Yes Case Analysis

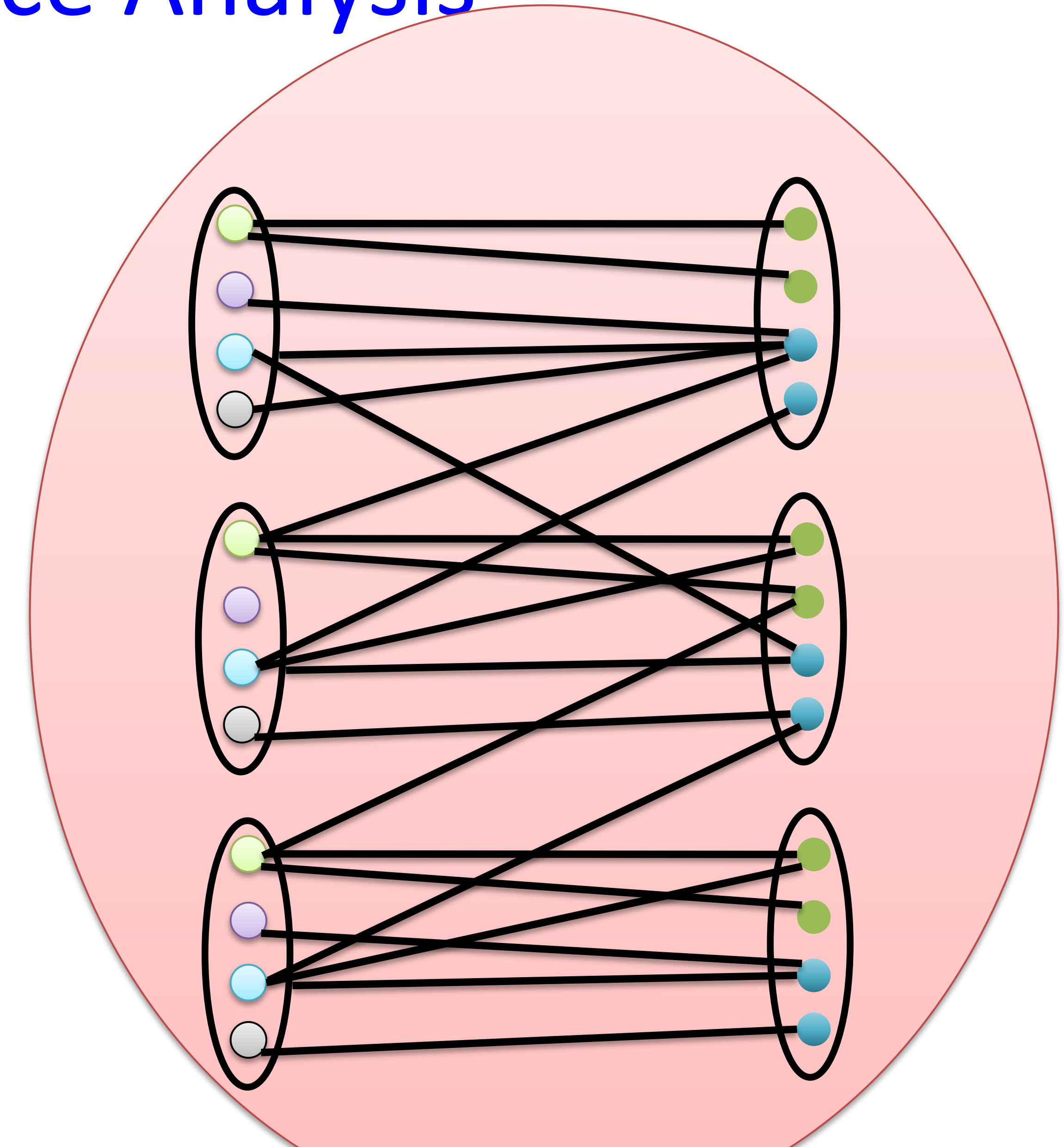


Yes Instance Analysis

Solution 1



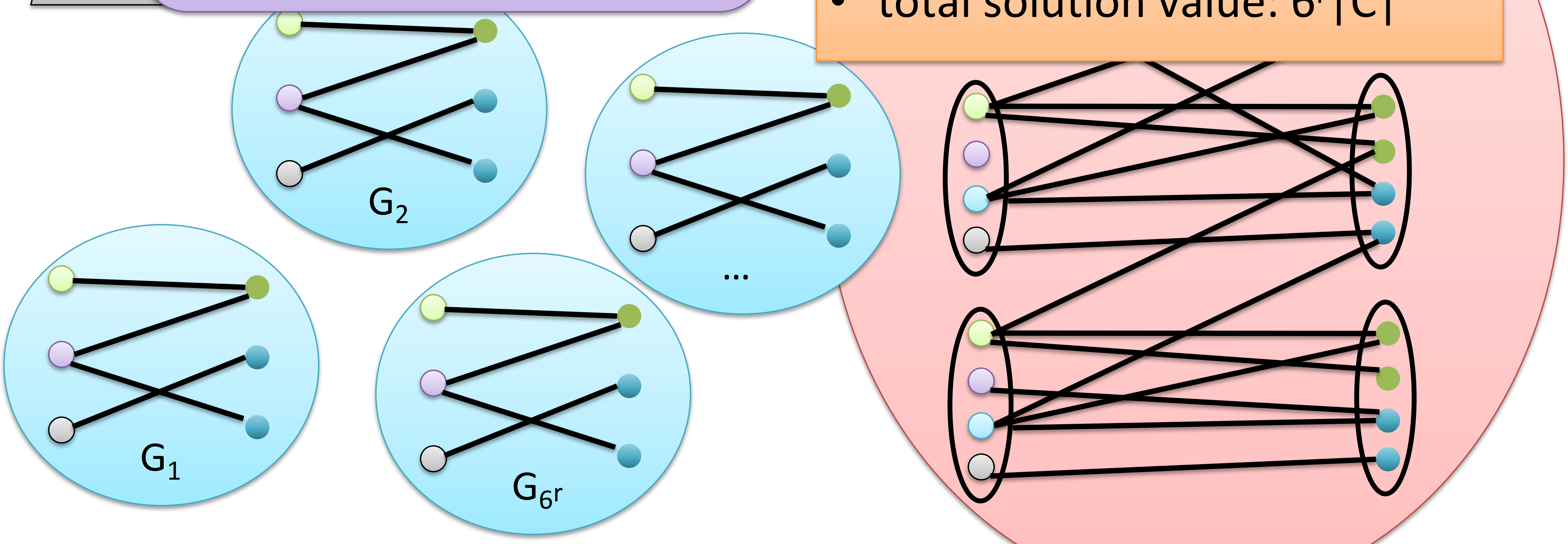
will collect 1 edge per constraint



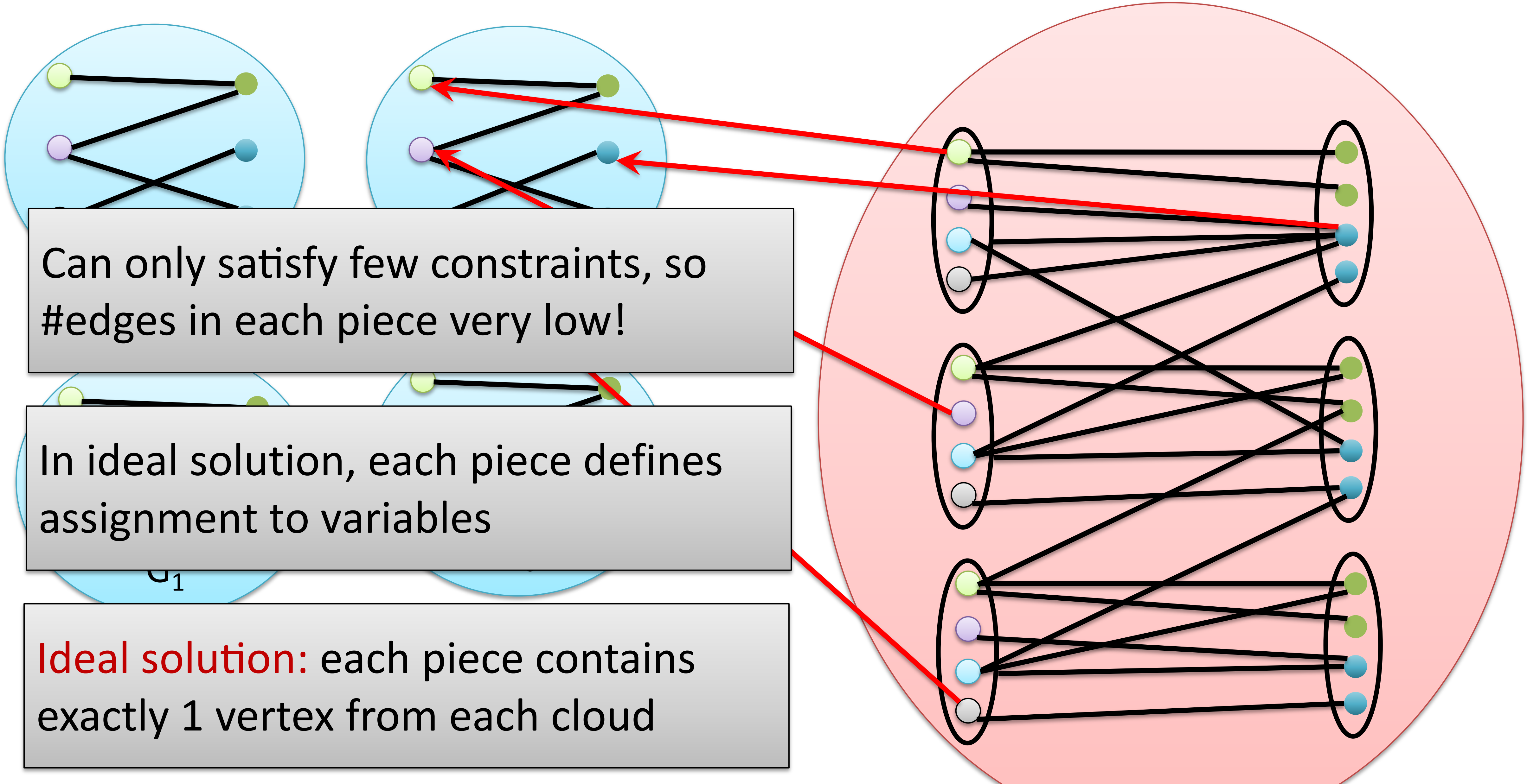
Yes Instance Analysis

In No-Instance p and L stay the same.
Want to show: solution value is low

- $p=6^r$ pieces
- each piece contributes $L=|C|$ edges
- total solution value: $6^r|C|$



No Instance Analysis

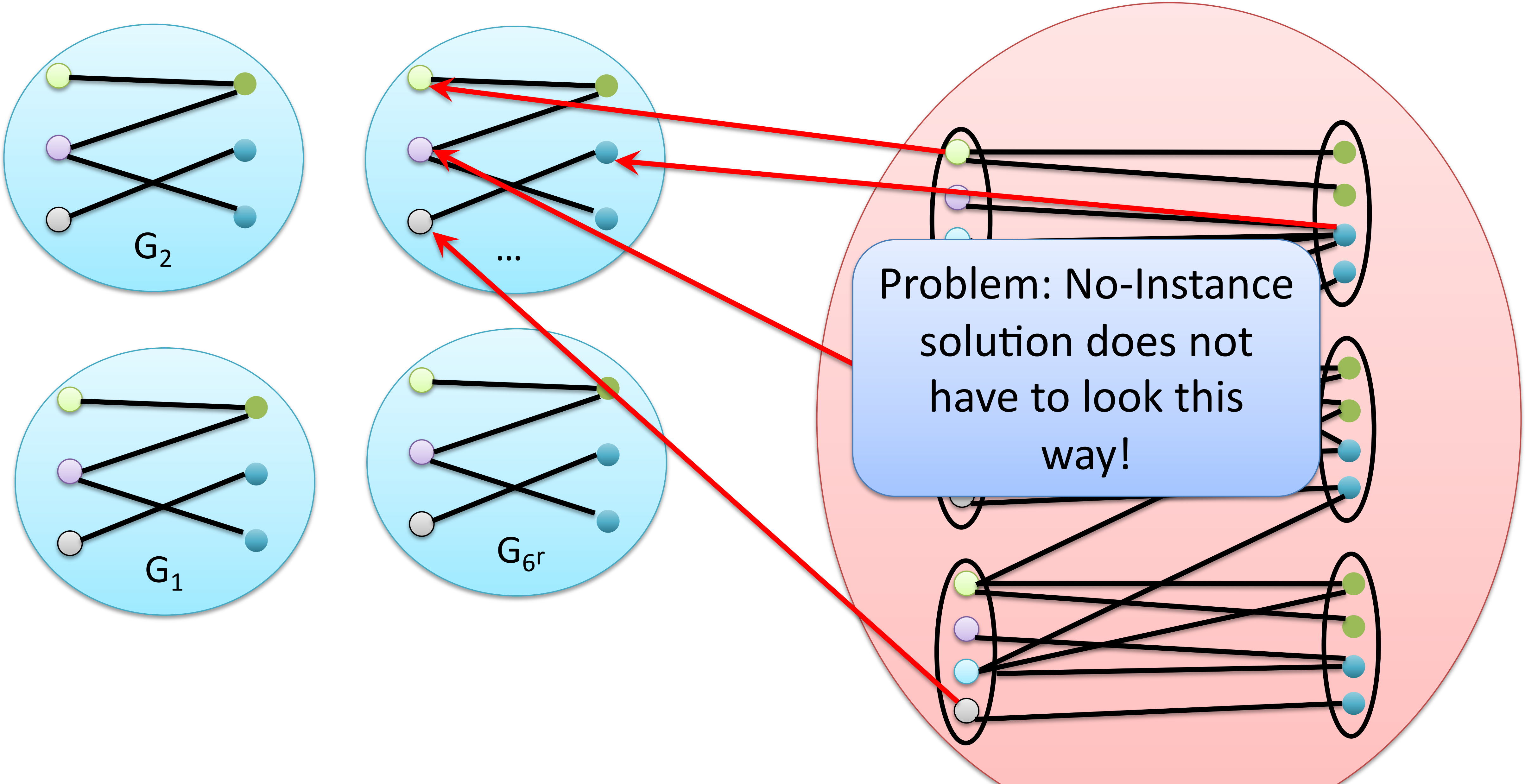


Can only satisfy few constraints, so #edges in each piece very low!

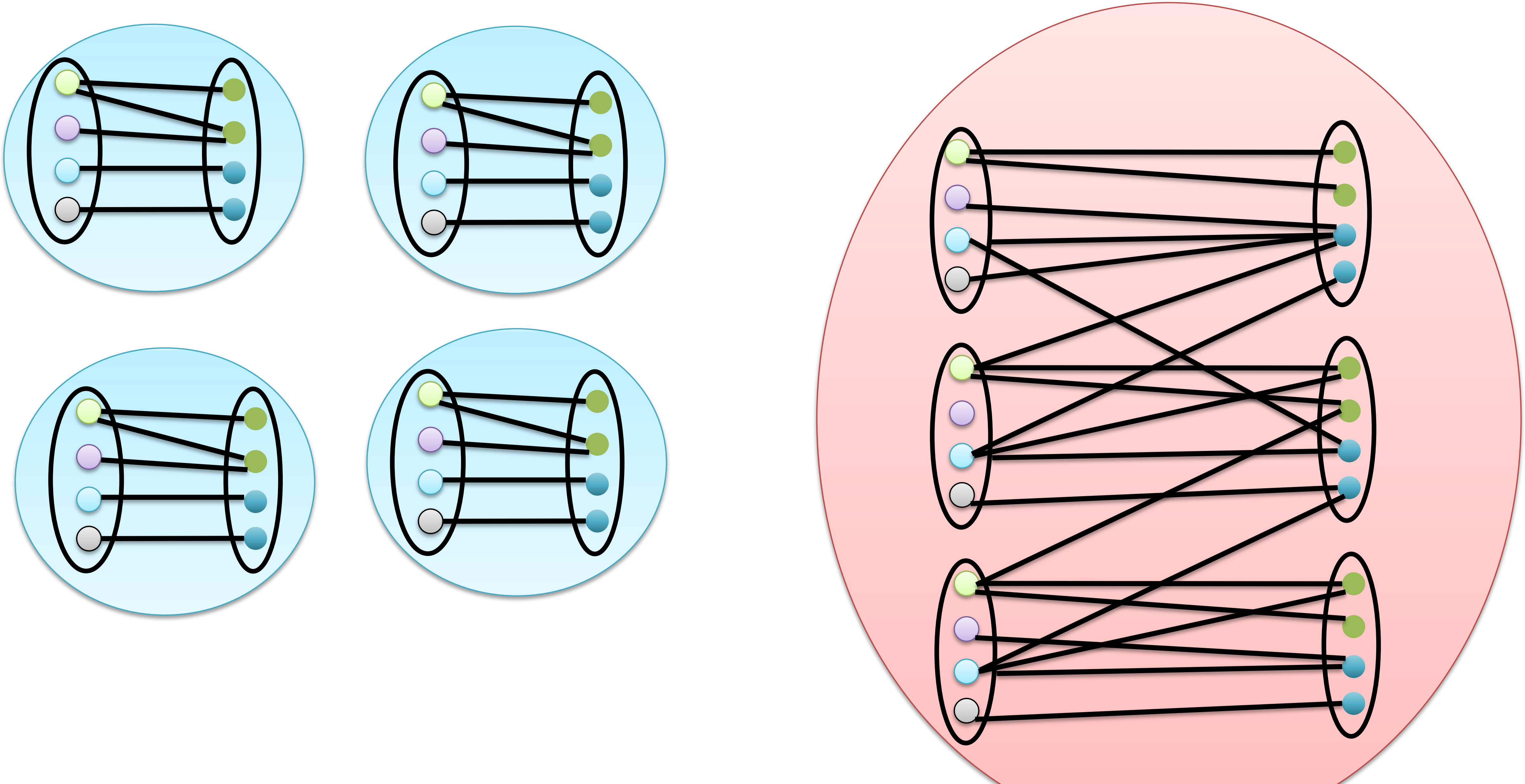
In ideal solution, each piece defines assignment to variables

Ideal solution: each piece contains exactly 1 vertex from each cloud

No Instance Analysis

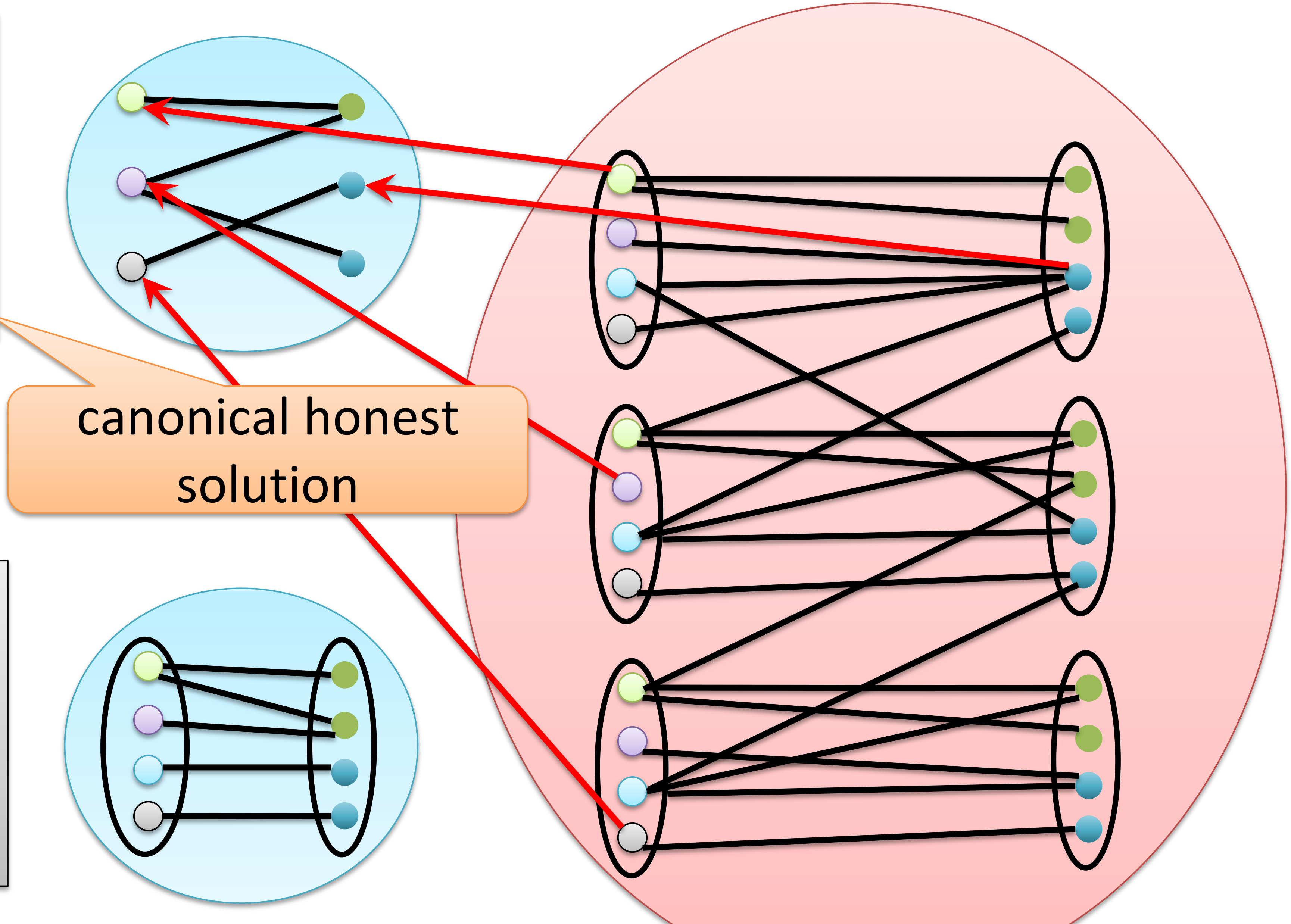


No Instance Analysis



Two Extreme Solutions

Ideal solution: each piece contains exactly 1 vertex from each cloud

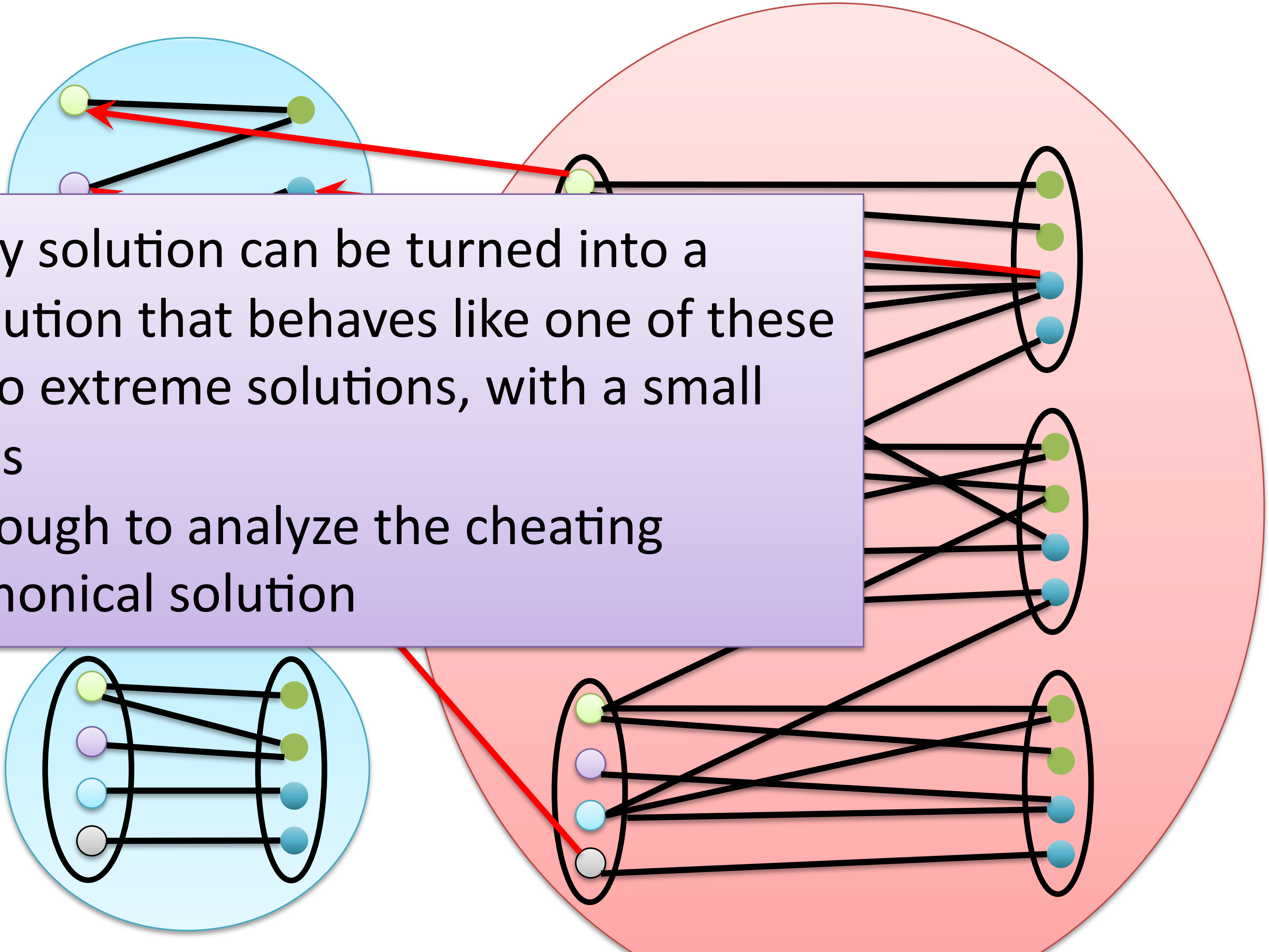


canonical cheating solution: each cloud is contained in some piece

Two Extreme Solutions

Ideal solution: each piece contains exactly 1 vertex from each cloud

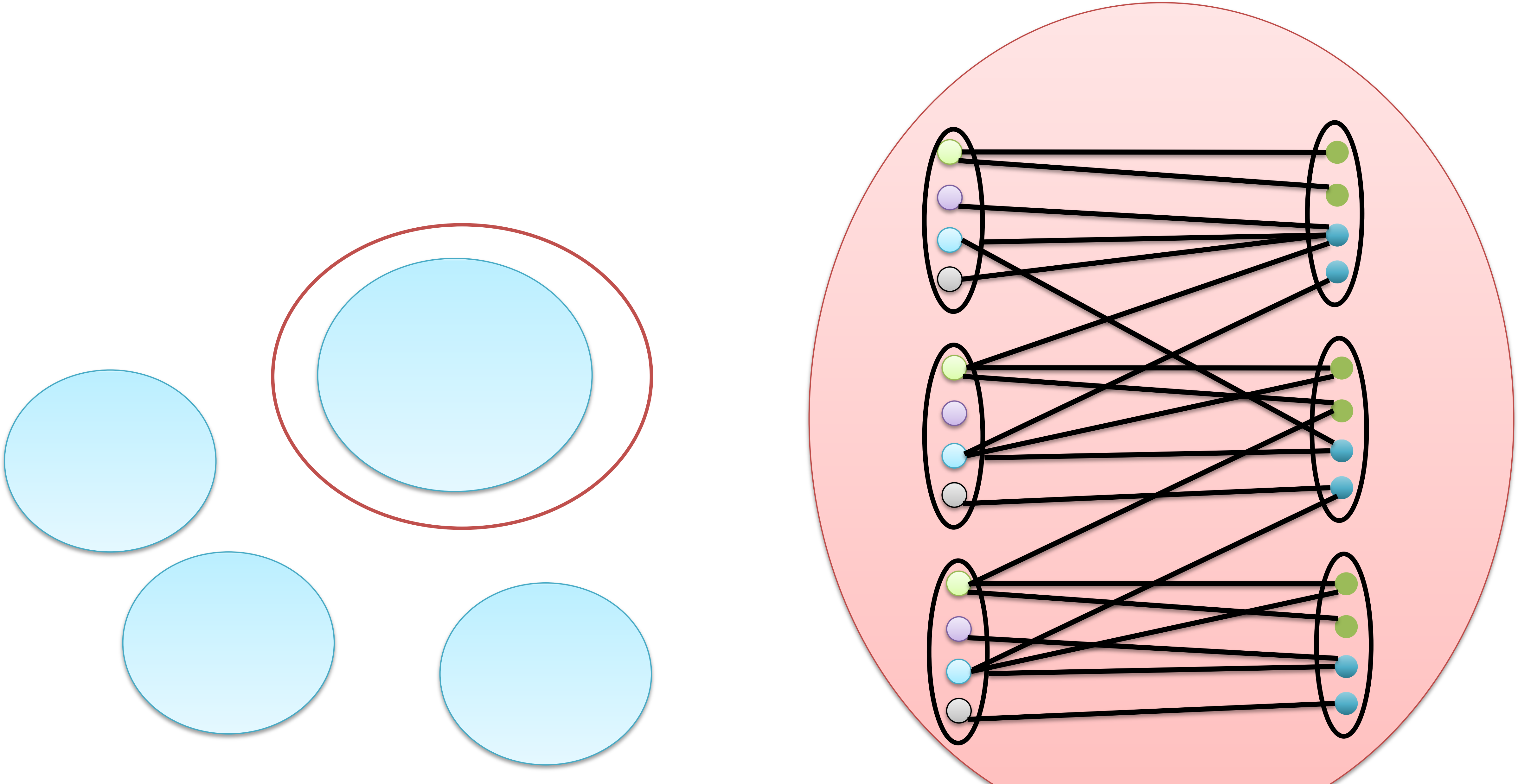
canonical cheating solution: each cloud is contained in some piece



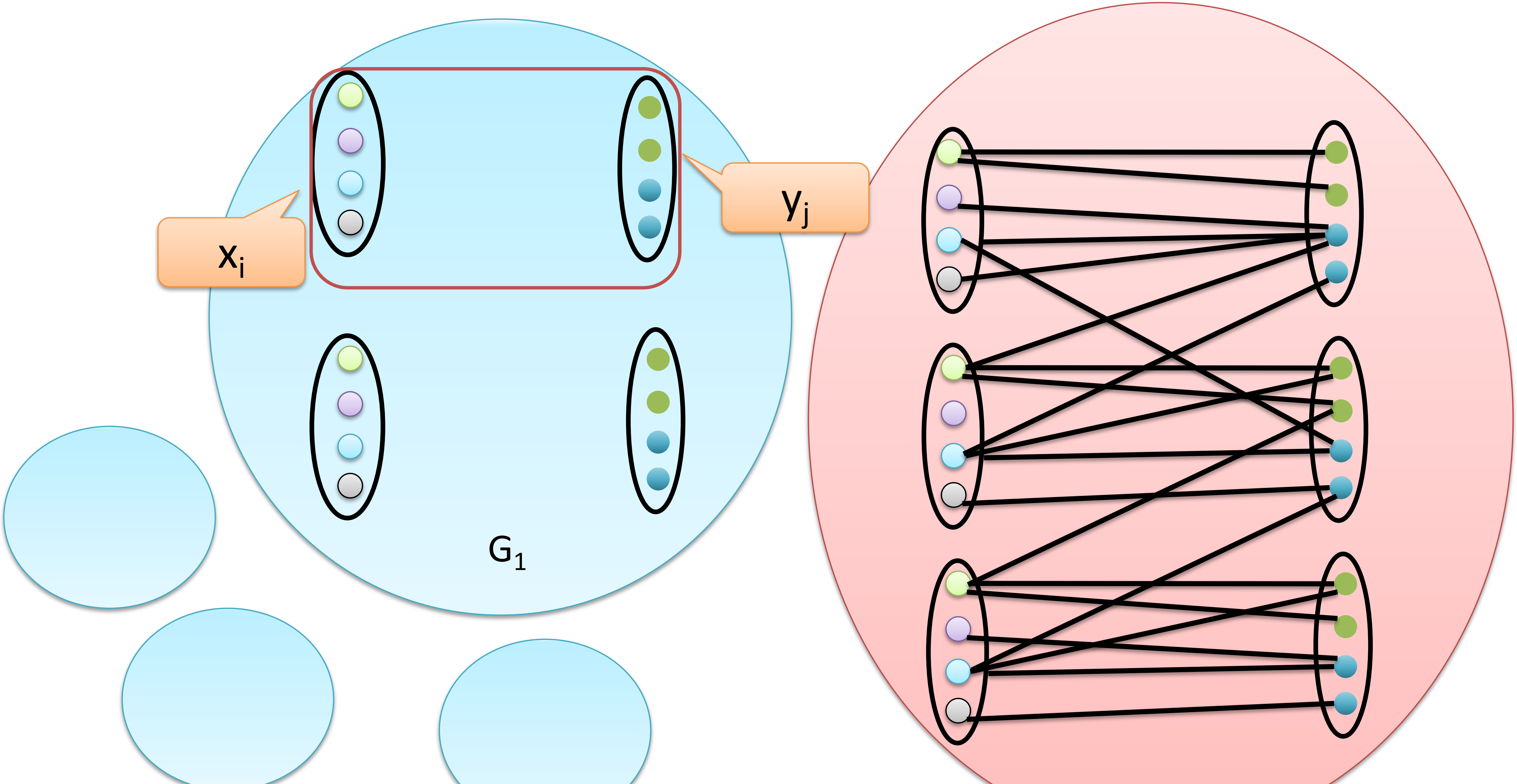
- Any solution can be turned into a solution that behaves like one of these two extreme solutions, with a small loss
- Enough to analyze the cheating canonical solution

A Technical Issue

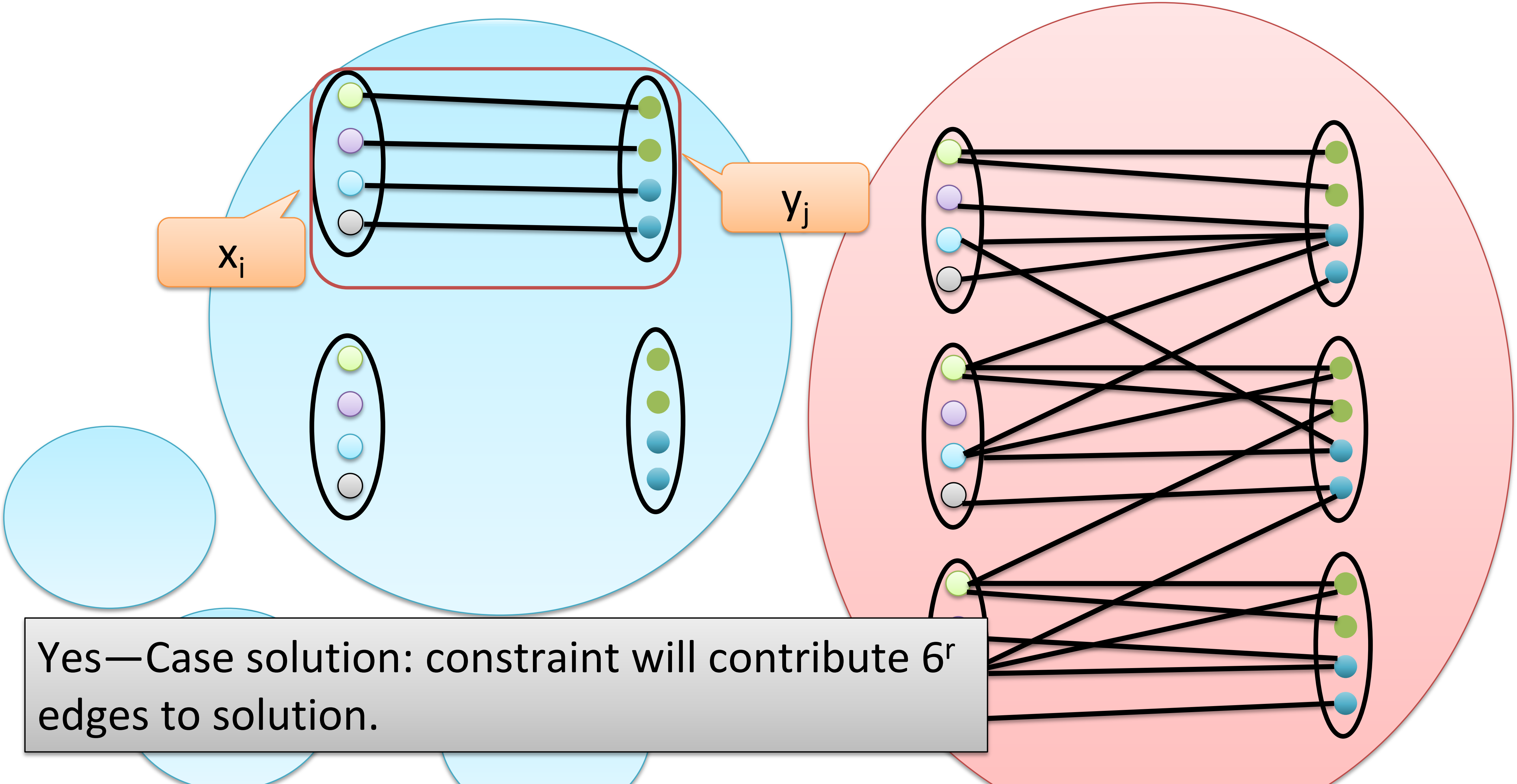
Canonical Cheating Solution



Canonical Cheating Solution

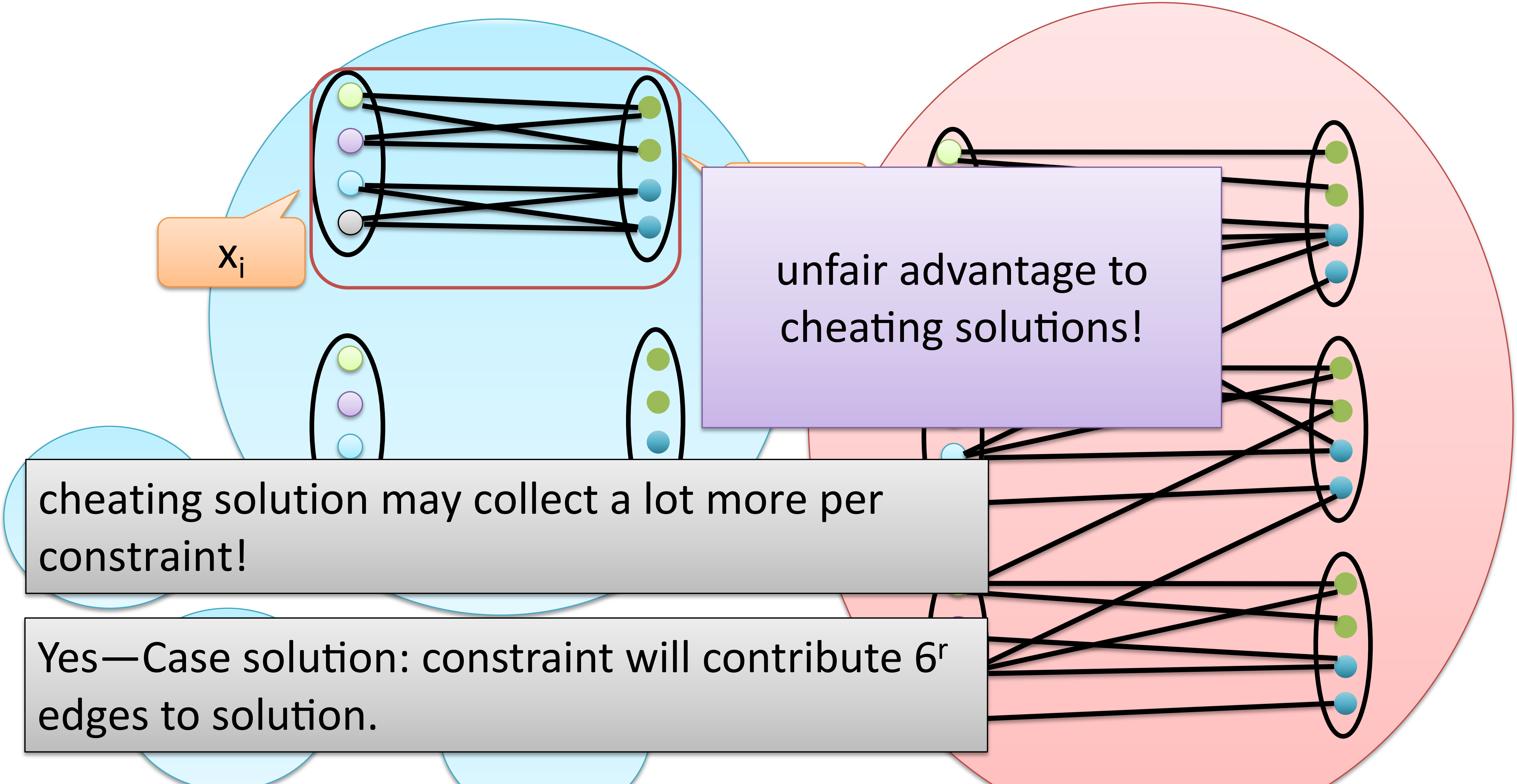


Canonical Cheating Solution



Yes—Case solution: constraint will contribute 6^r edges to solution.

Canonical Cheating Solution

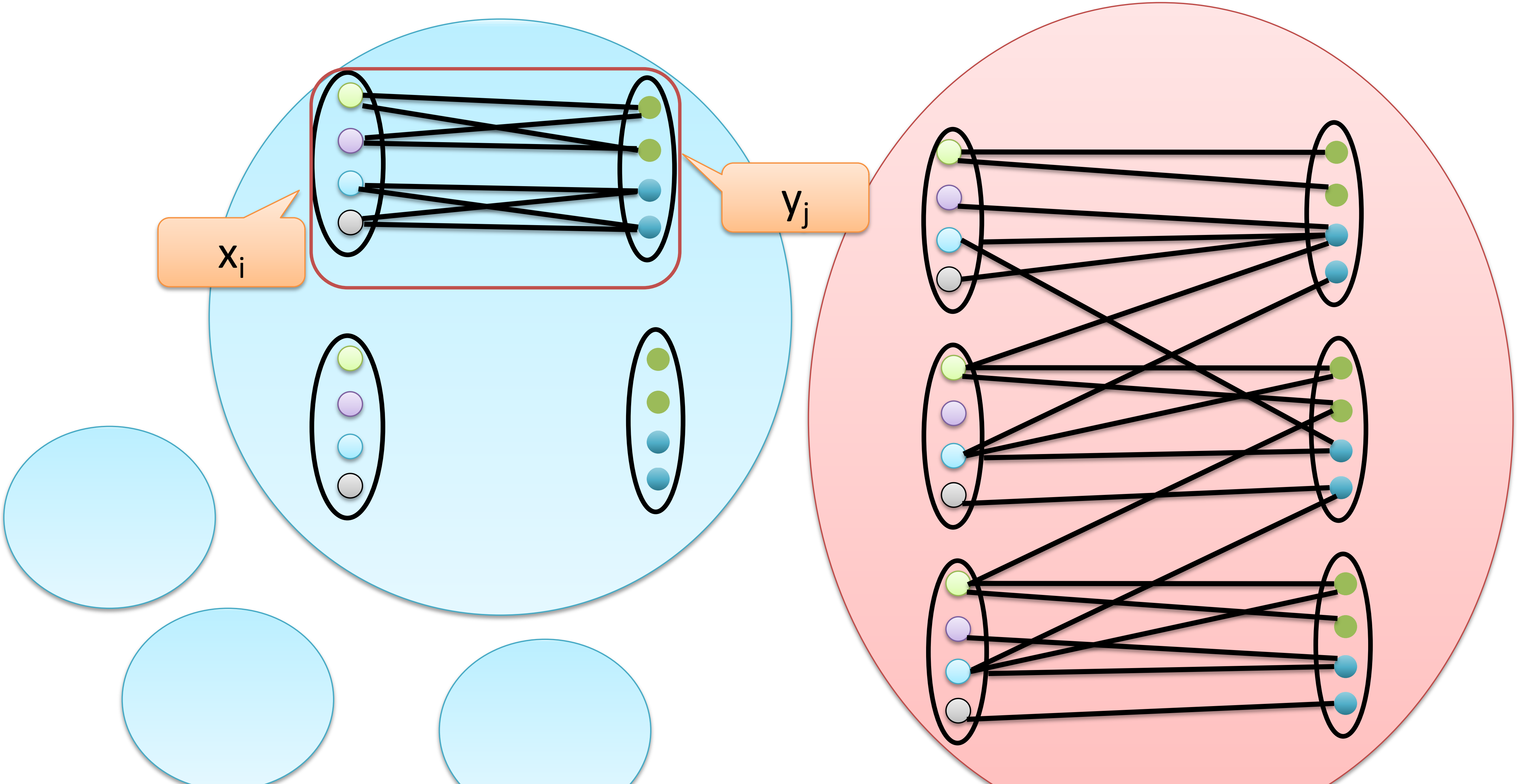


unfair advantage to cheating solutions!

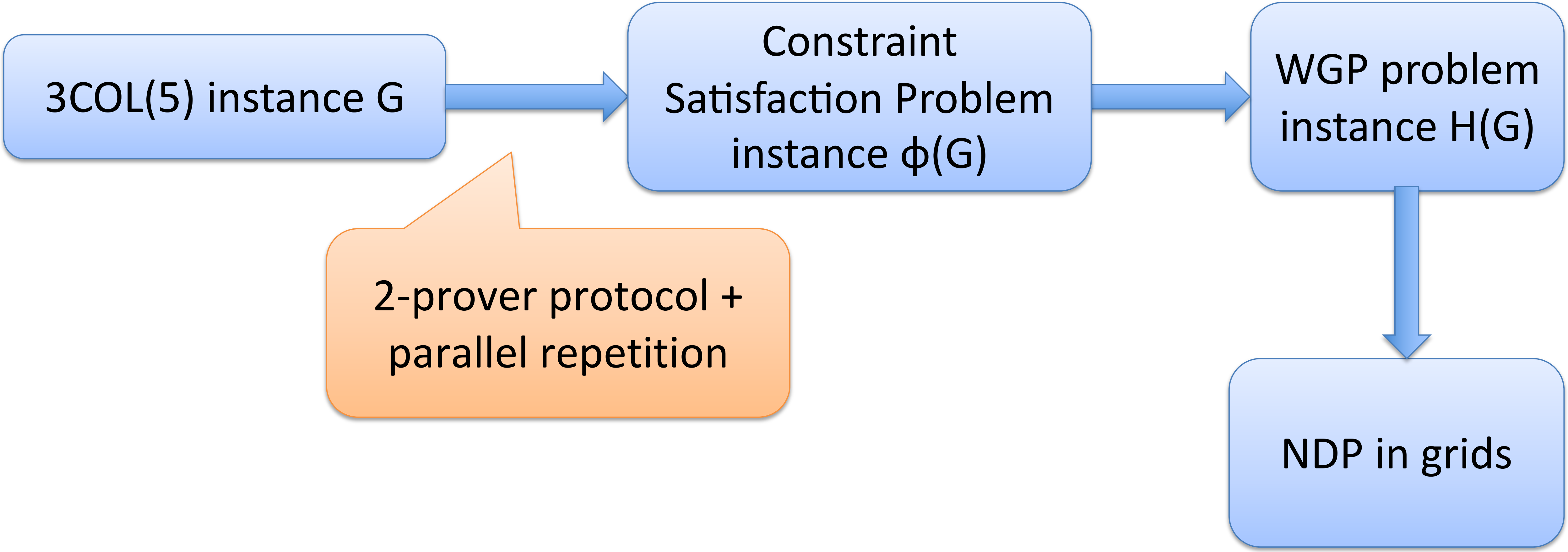
cheating solution may collect a lot more per constraint!

Yes—Case solution: constraint will contribute 6^r edges to solution.

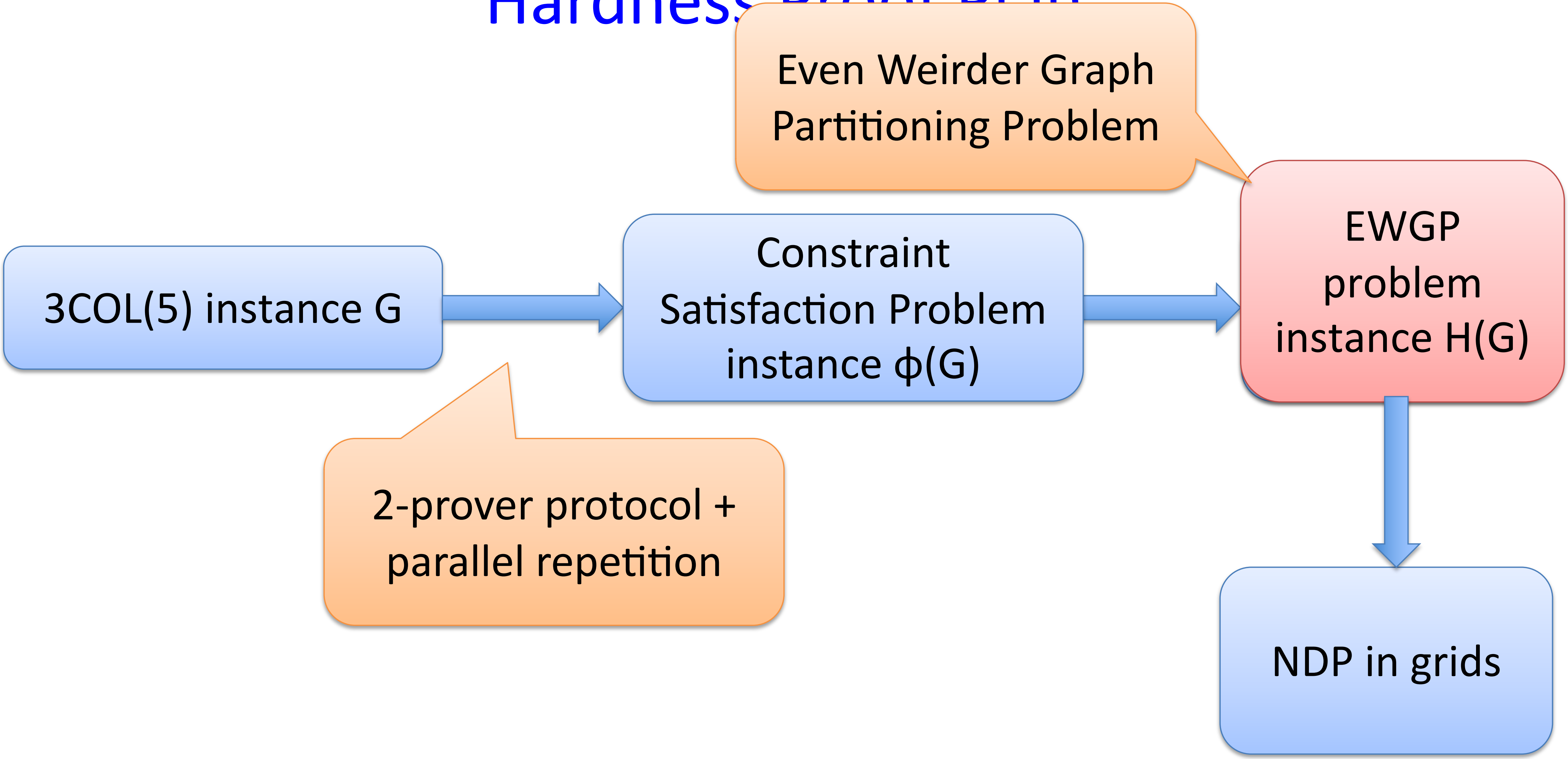
Solution: Cheat



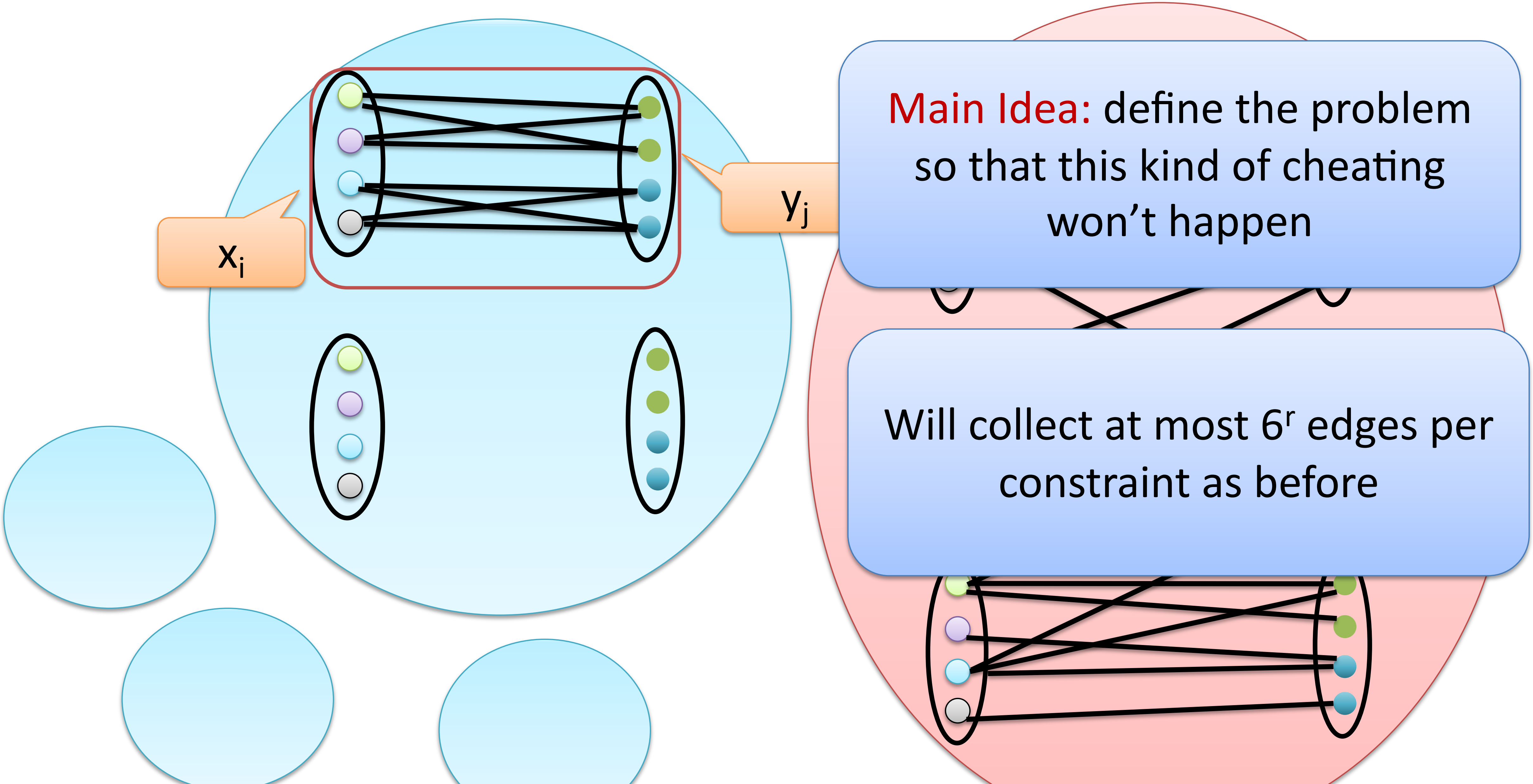
Hardness Proof Plan



Hardness Proof Plan



EWGP

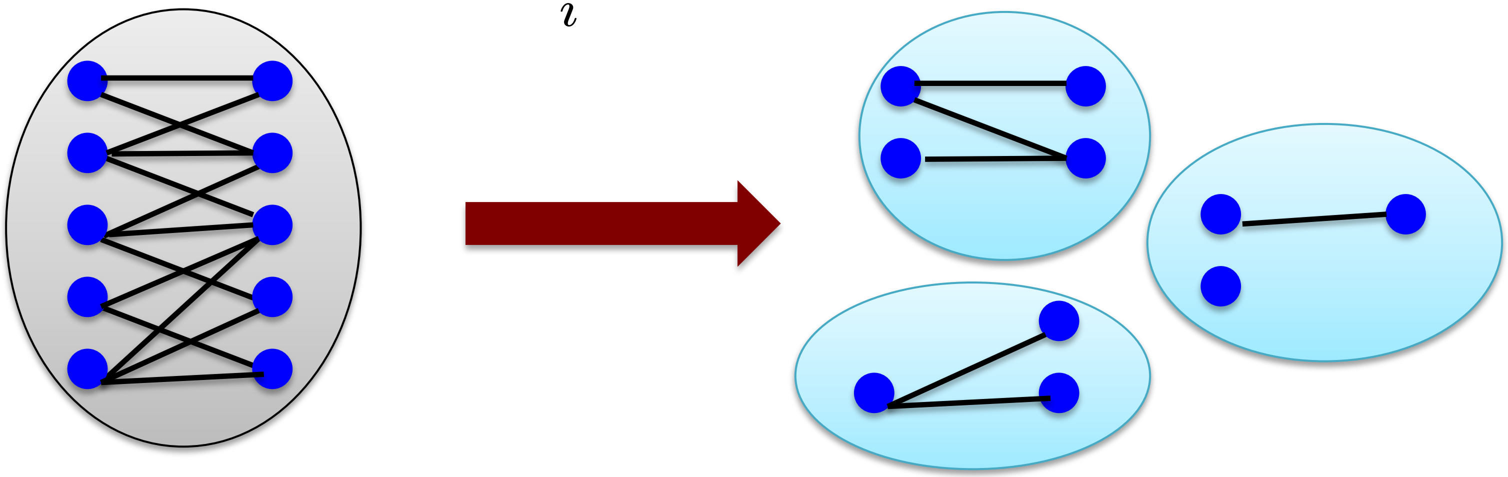


Main Idea: define the problem so that this kind of cheating won't happen

Will collect at most 6^r edges per constraint as before

Weird Graph Partitioning Problem (WGP)

- Input: bipartite graph G=(V,E), integers p, L.
- Output:
  - partition G into p vertex-induced subgraphs.
  - for each subgraph G_i, select a subset E_i of at most L edges
  - Goal: maximize $\sum_i |E_i|$

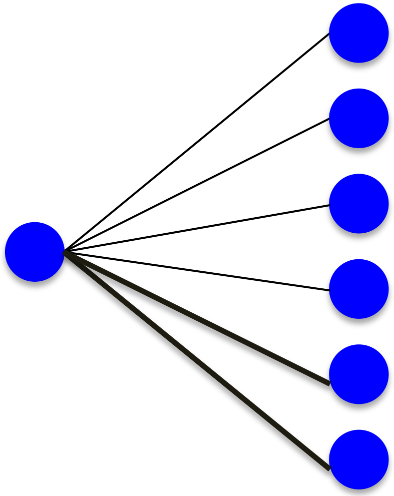


Weird Graph Partitioning Problem (WGP)

- Input: bipartite graph G=(V,E), integers p, L.
- Output:
  - partition G into p vertex-induced subgraphs.
  - for each subgraph G_i, select a subset E_i of at most L edges
  - Goal: maximize $\sum_i |E_i|$

Extra:

- For every vertex, incident edges are partitioned into bundles

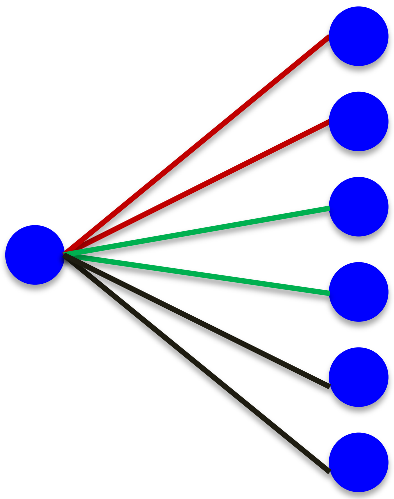


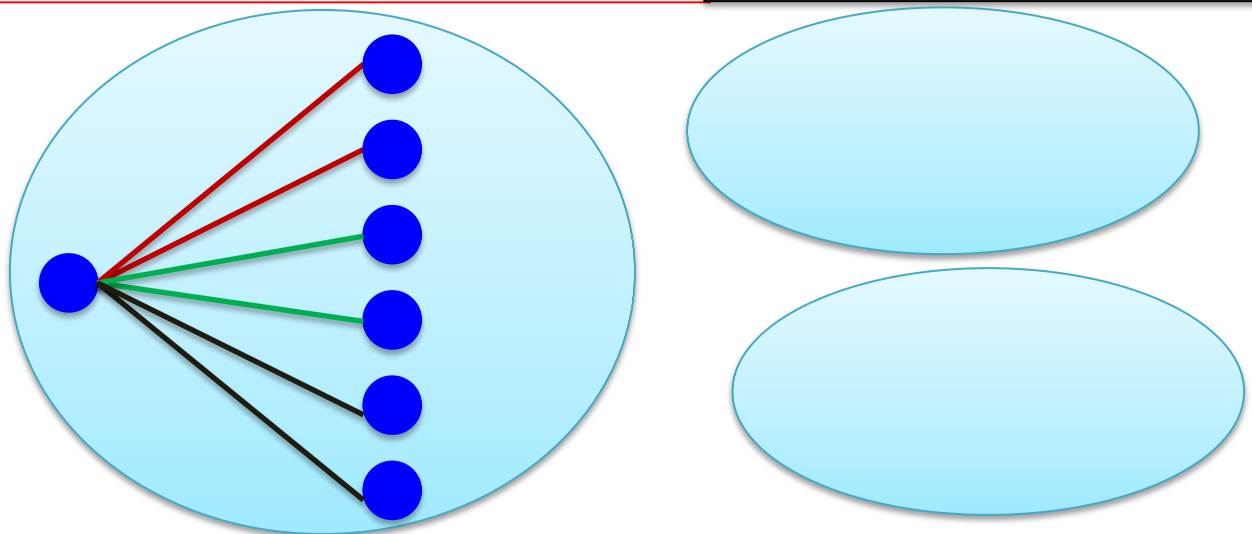
Weird Graph Partitioning Problem (WGP)

- Input: bipartite graph G=(V,E), integers p, L.
- Output:
  - partition G into p vertex-induced subgraphs.
  - for each subgraph G_i, select a subset E_i of at most L edges
  - Goal: maximize $\sum_i |E_i|$

Extra:

- For every vertex, incident edges are partitioned into bundles
- may only take 1 edge per bundle



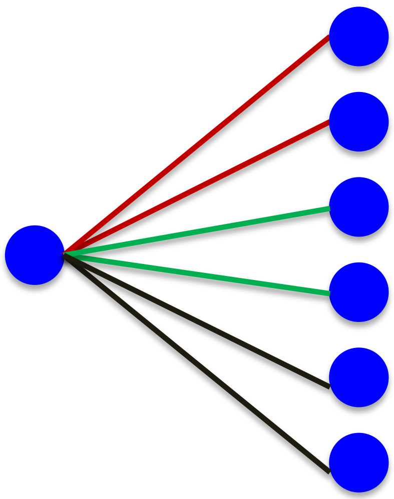


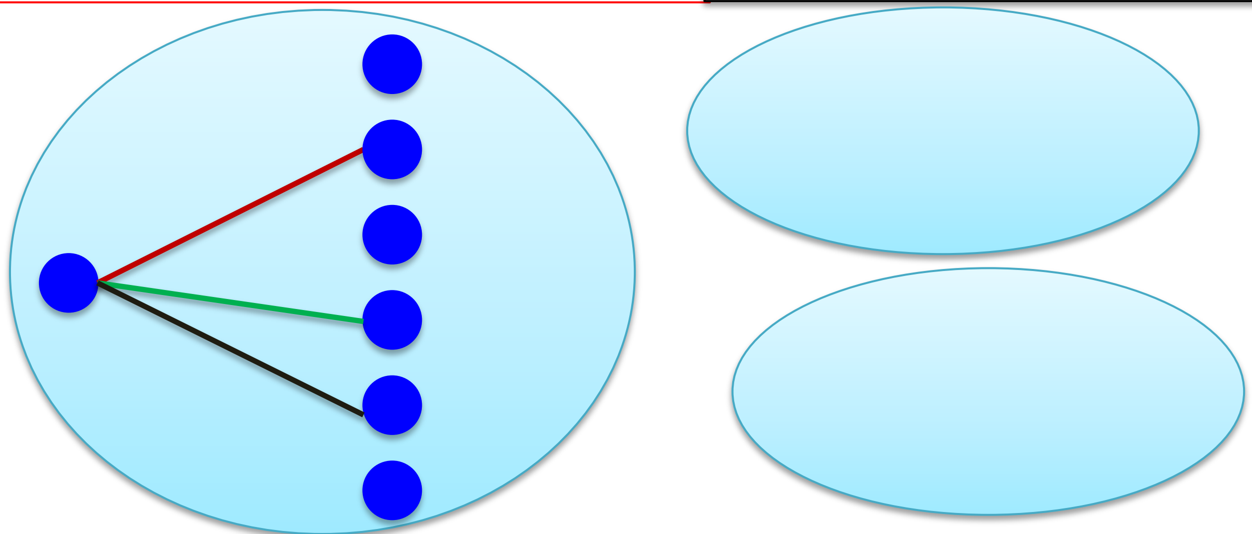
Weird Graph Partitioning Problem (WGP)

- Input: bipartite graph G=(V,E), integers p, L.
- Output:
  - partition G into p vertex-induced subgraphs.
  - for each subgraph G_i, select a subset E_i of at most L edges
  - Goal: maximize $\sum_i |E_i|$

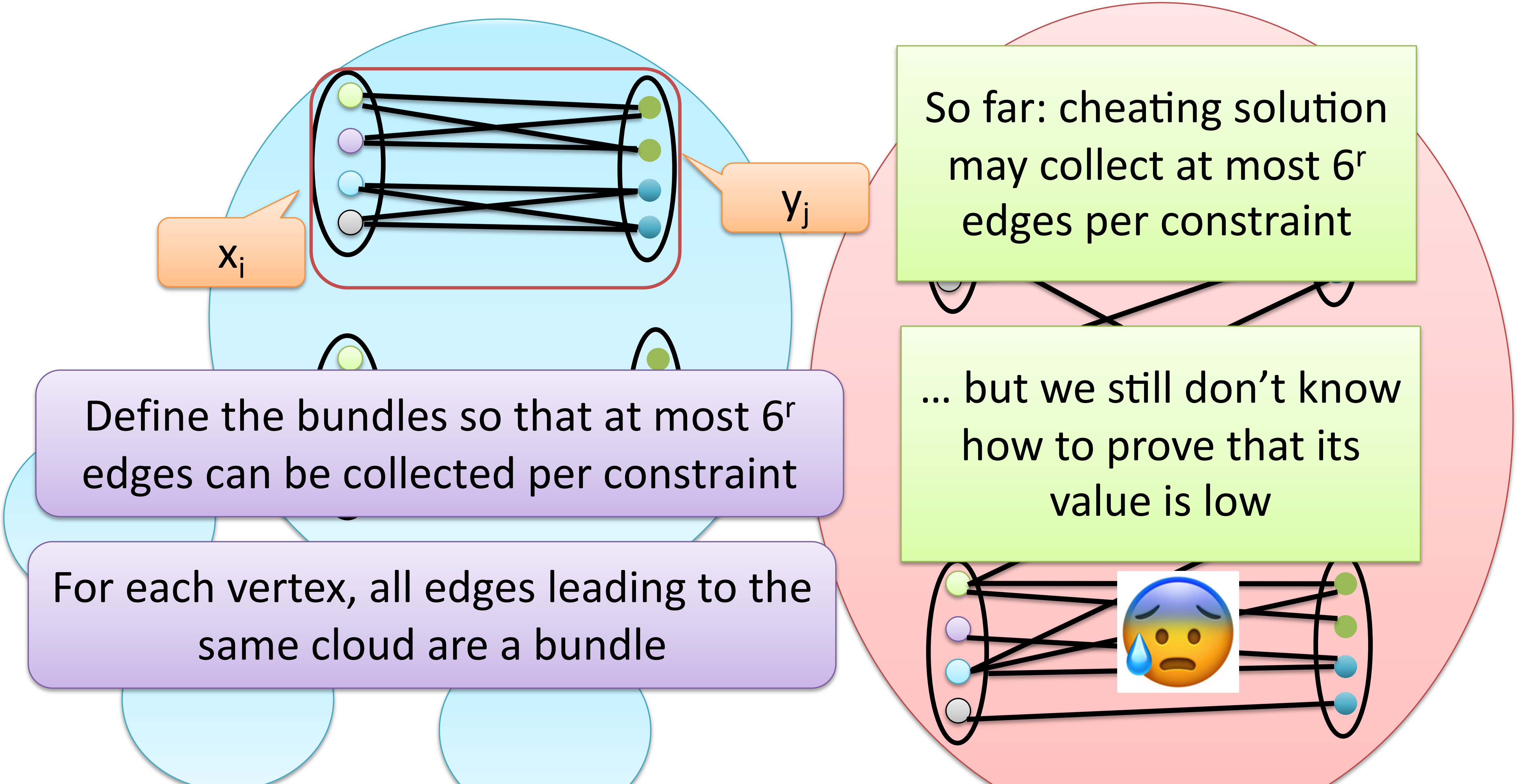
Extra:

- For every vertex, incident edges are partitioned into bundles
- may only take 1 edge per bundle





Canonical Cheating Solution



So far: cheating solution may collect at most 6^r edges per constraint

Define the bundles so that at most 6^r edges can be collected per constraint

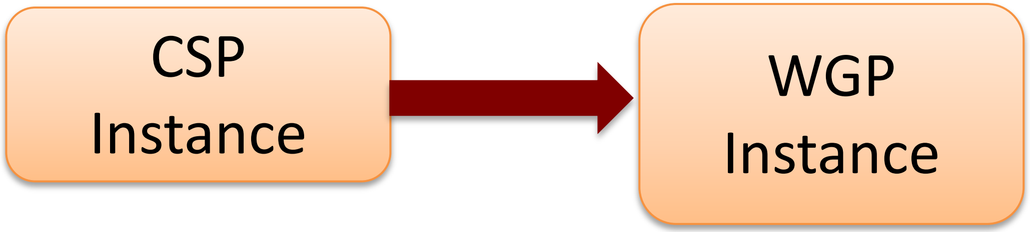
… but we still don't know how to prove that its value is low

For each vertex, all edges leading to the same cloud are a bundle

End of the Technical Issue

Main Idea 2: Cook not Karp

Standard Karp Reduction

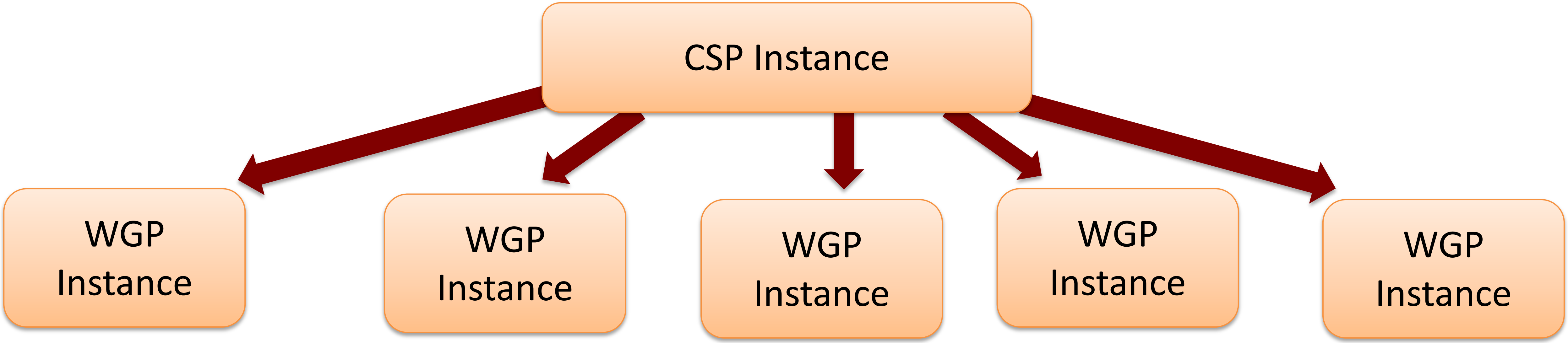


- If CSP is a Yes-Instance, WGP has a solution of large value
- If CSP is a No-Instance, every solution to WGP has low value

we don't know how to prove this…

Our Reduction (Cook)

Assume for contradiction that there is an α-approximation algorithm A for WGP.

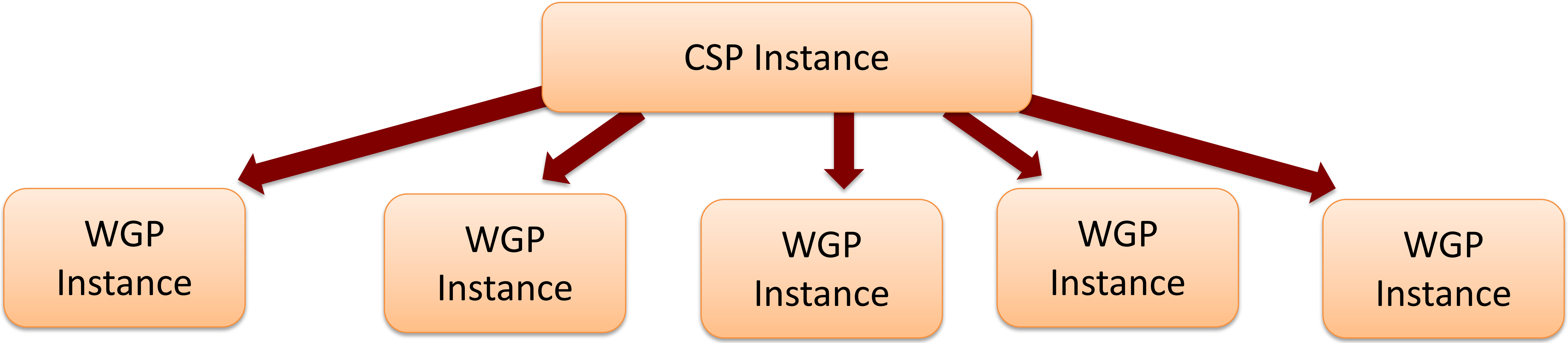


If CSP is a Yes-Instance, each WGP instance has a high-value solution

prescribed value

Our Reduction (Cook)

Assume for contradiction that there is an α-approximation algorithm A for WGP.

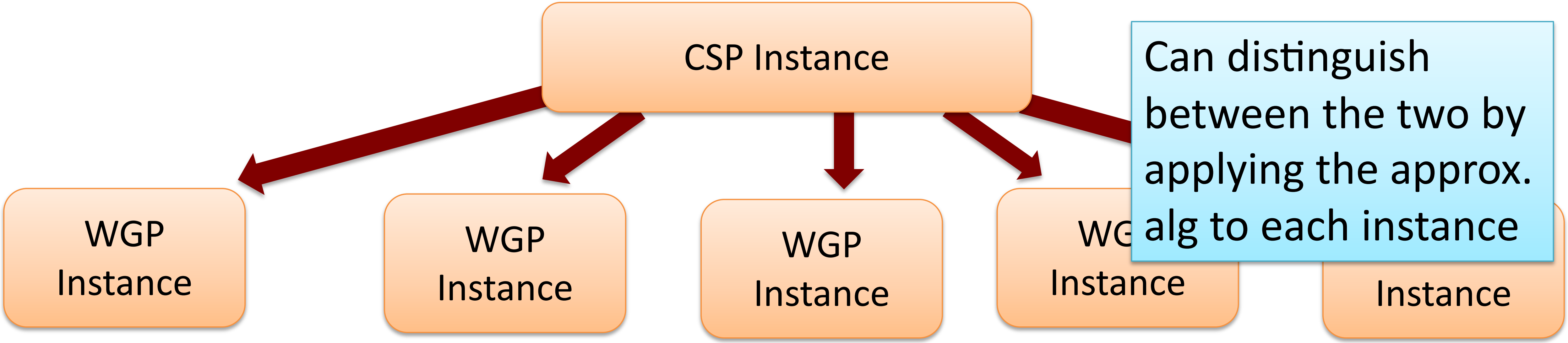


If CSP is a Yes-Instance, each WGP instance has a high-value solution

If CSP is a No-Instance, some WGP instance only has low-value solutions

Our Reduction (Cook)

Assume for contradiction that there is an α-approximation algorithm A for WGP.



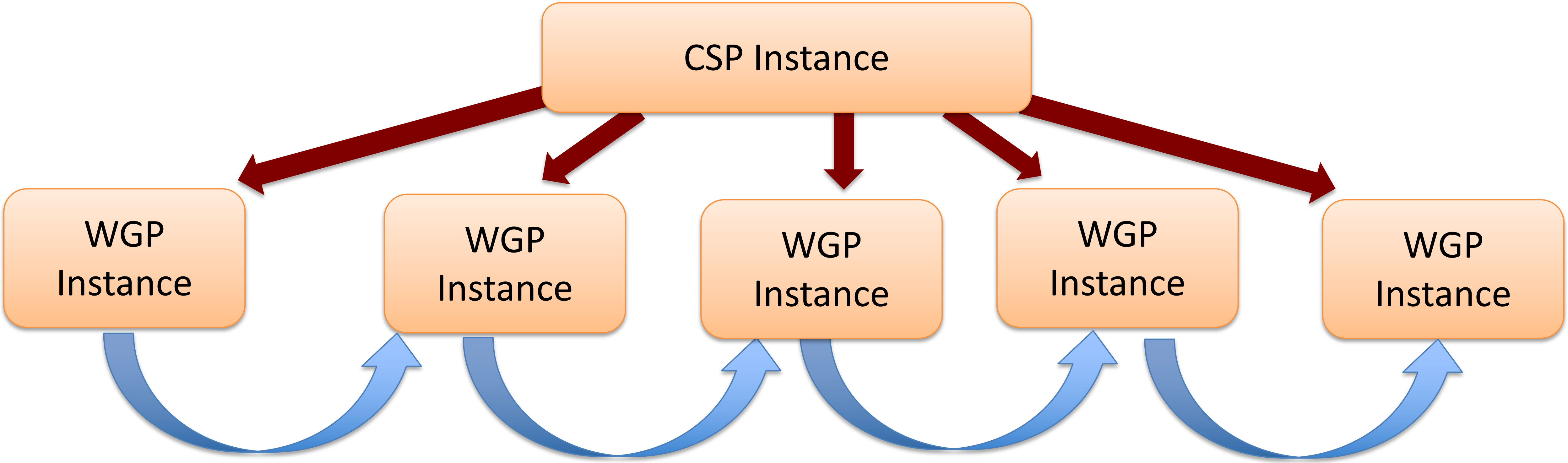
Can distinguish between the two by applying the approx. alg to each instance

If CSP is a Yes-Instance, each WGP instance has a high-value solution

If CSP is a No-Instance, some WGP instance only has low-value solutions

Our Reduction (Cook)

Assume for contradiction that there is an α-approximation algorithm A for WGP.

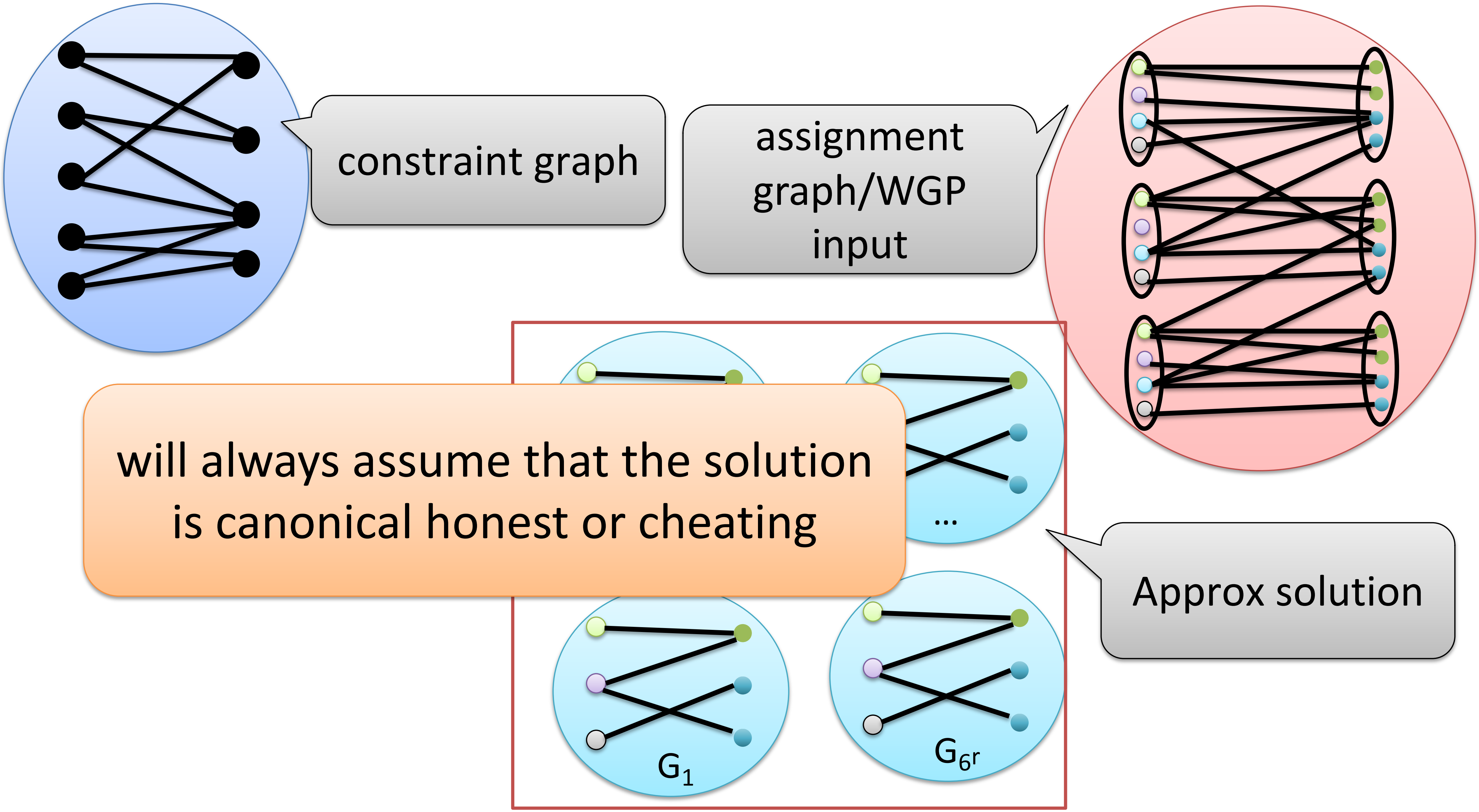


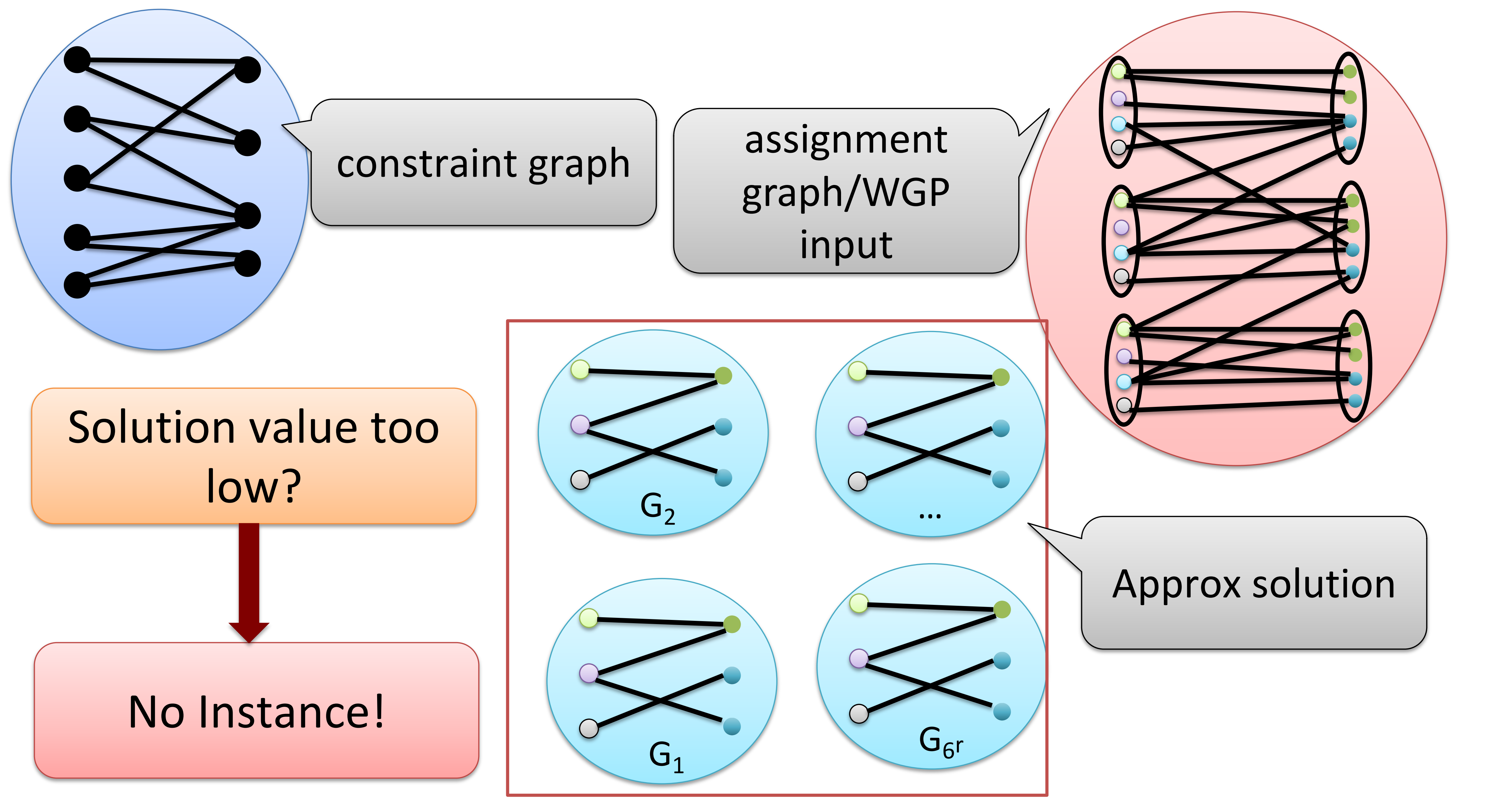
Construction of each instance depends on solution produced by A to previous instances!

Reduction Overview

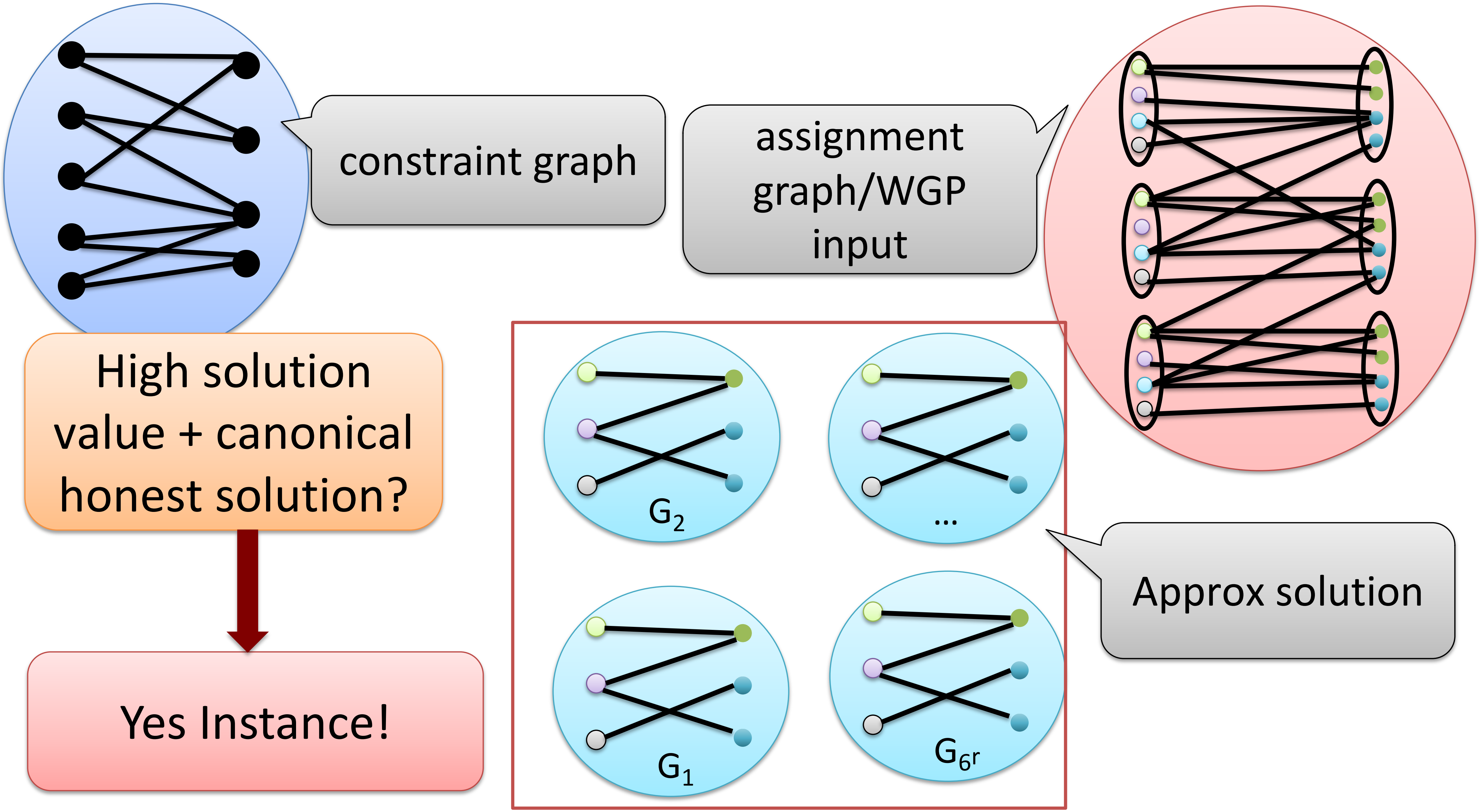
Assume for contradiction that there is an α-approximation algorithm A for WGP.

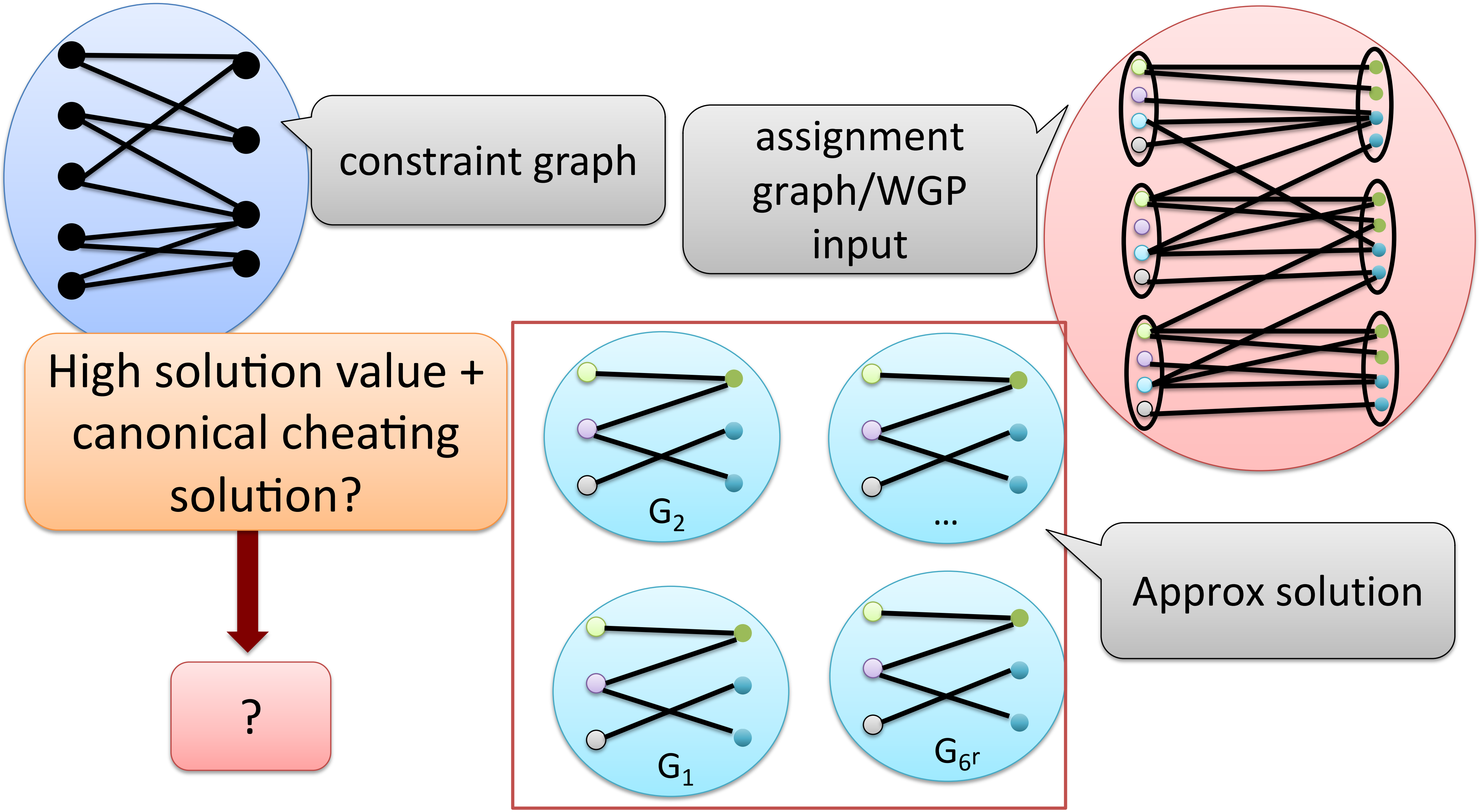
will use the algorithm to distinguish yes and no instances of CSP

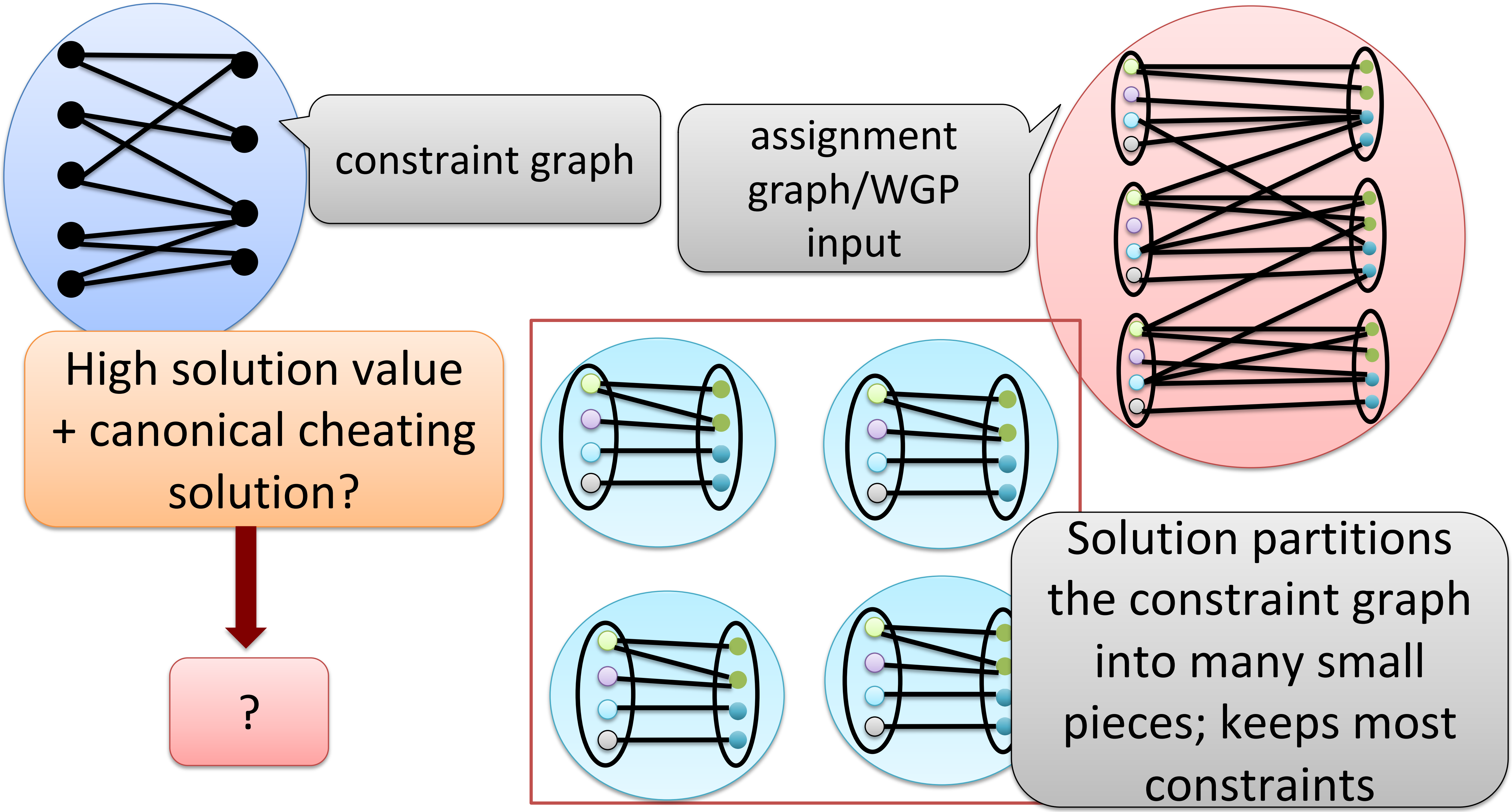

constraint graph
assignment graph/WGP input
will always assume that the solution is canonical honest or cheating
…
G_1
G_{6r}
Approx solution

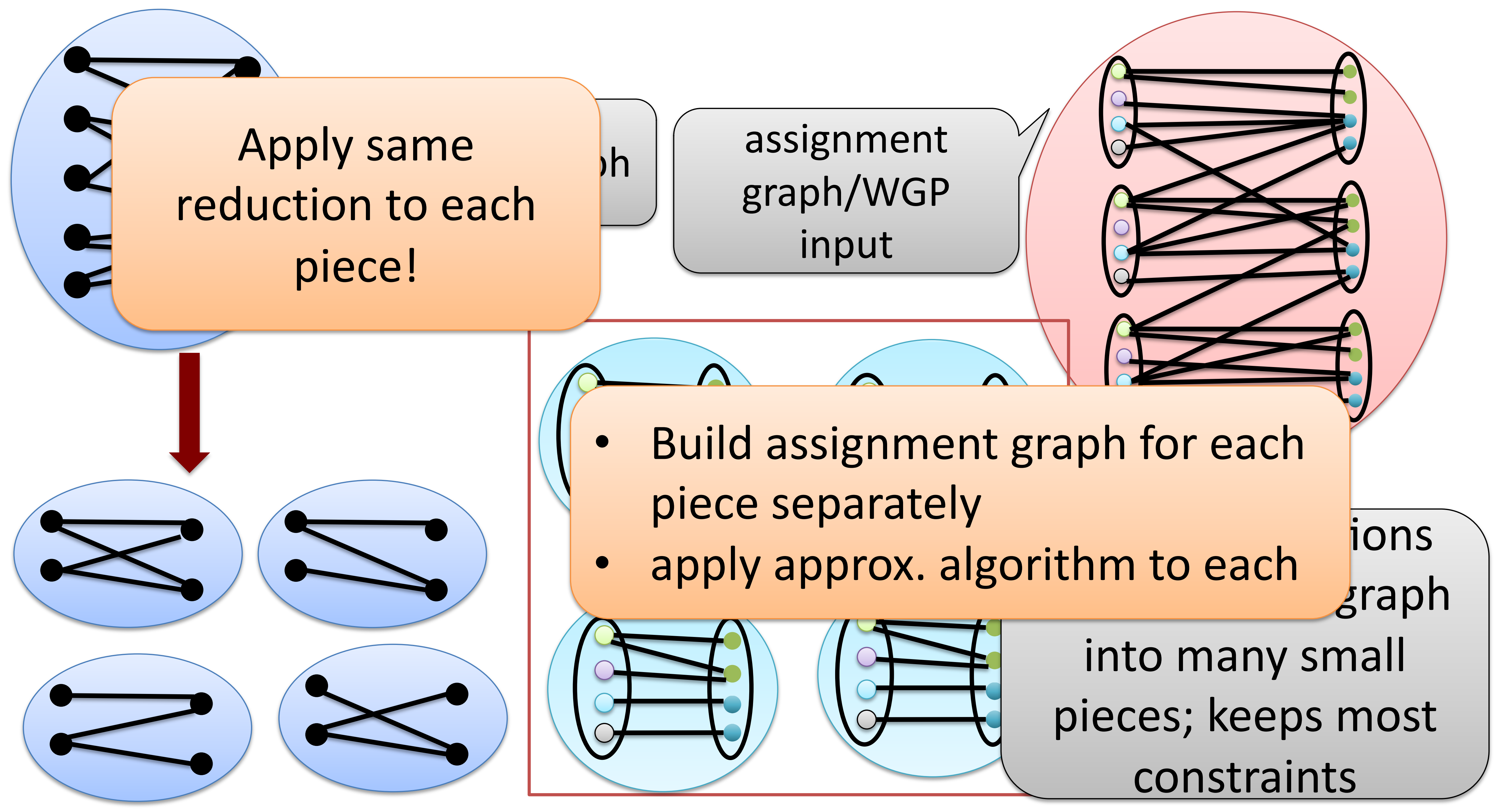


constraint graph
assignment graph/WGP input
Solution value too low?
No Instance!
G2
...
G1
G6r
Approx solution


constraint graph
assignment graph/WGP input
High solution value + canonical honest solution?
Yes Instance!
G_2
…
G_1
G_{6r}
Approx solution


constraint graph
assignment graph/WGP input
High solution value + canonical cheating solution?
?
G_2
...
G_1
G_{6^r}
Approx solution


constraint graph
assignment graph/WGP input
High solution value + canonical cheating solution?
?
Solution partitions the constraint graph into many small pieces; keeps most constraints


Apply same reduction to each piece!
assignment graph/WGP input
Build assignment graph for each piece separately
apply approx. algorithm to each
into many small pieces; keeps most constraints

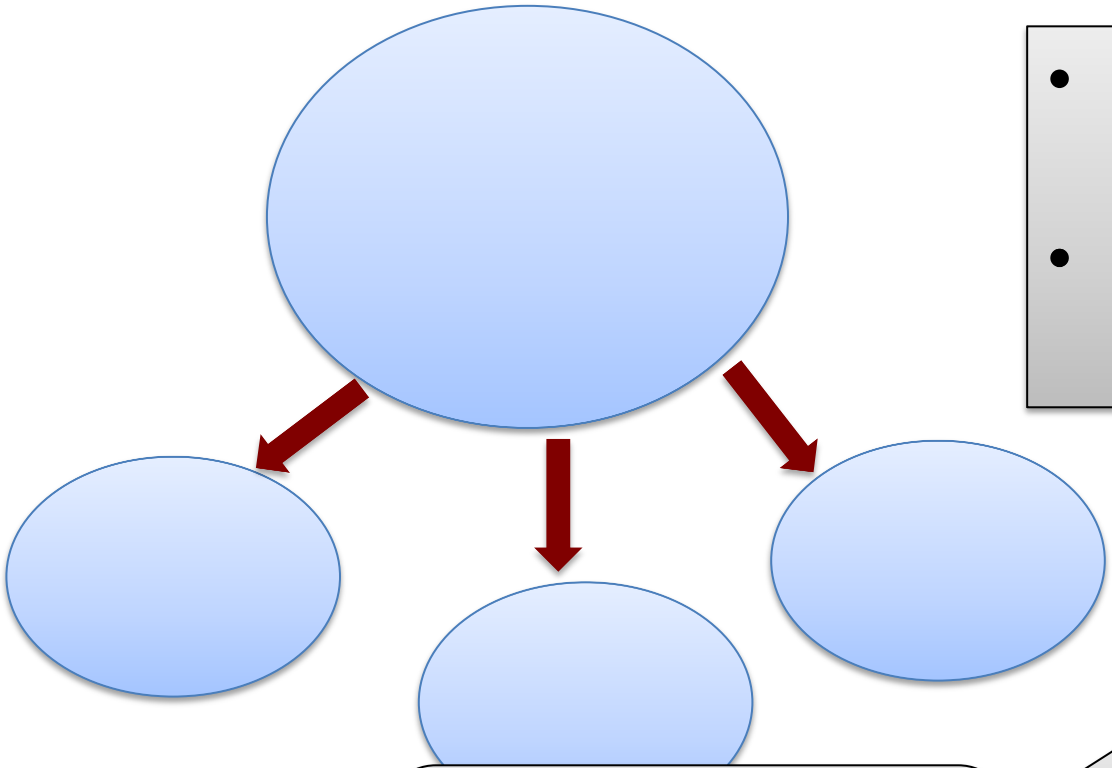
The Big Picture

constraint graph

Will either:

- correctly determine that it’s a Yes or a No Instance
- or cut into much smaller pieces, preserving many constraints

The Big Picture



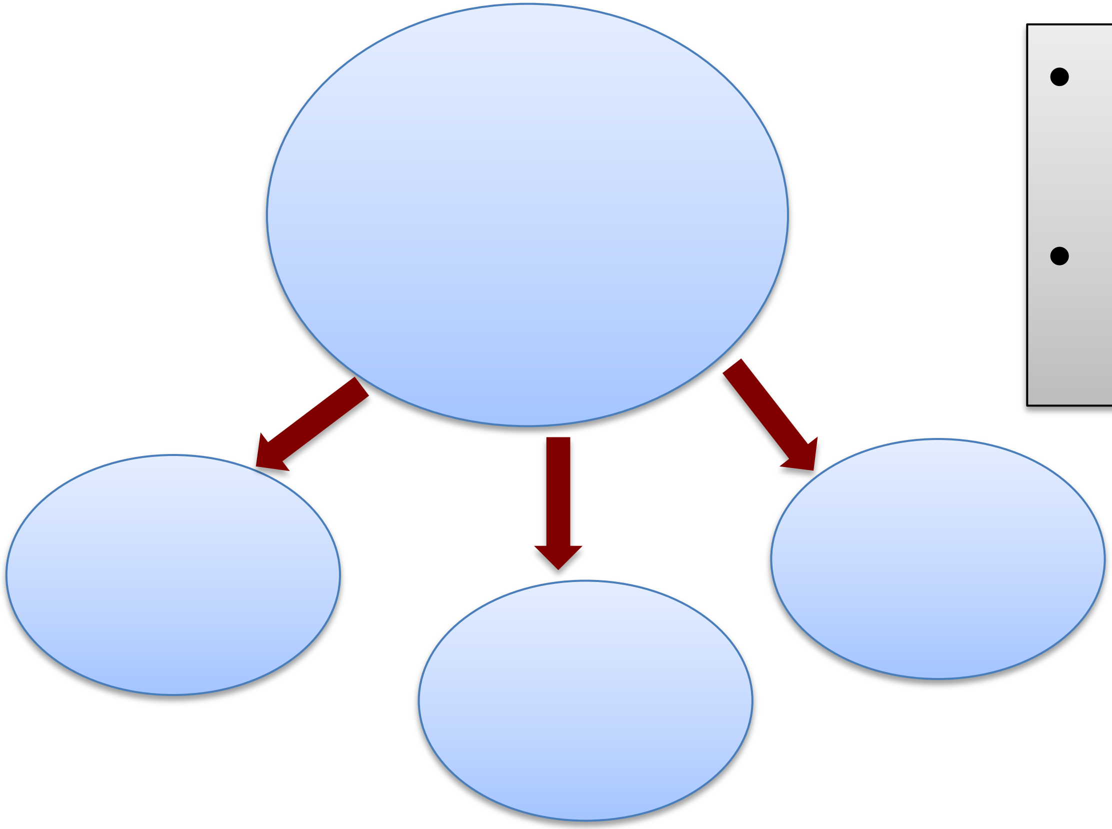
- Reduce each piece to WGP instance separately
- Apply approx. algorithm to each WGP instance

Solution value in any piece too low?

relatively to #constraints in piece

No Instance!

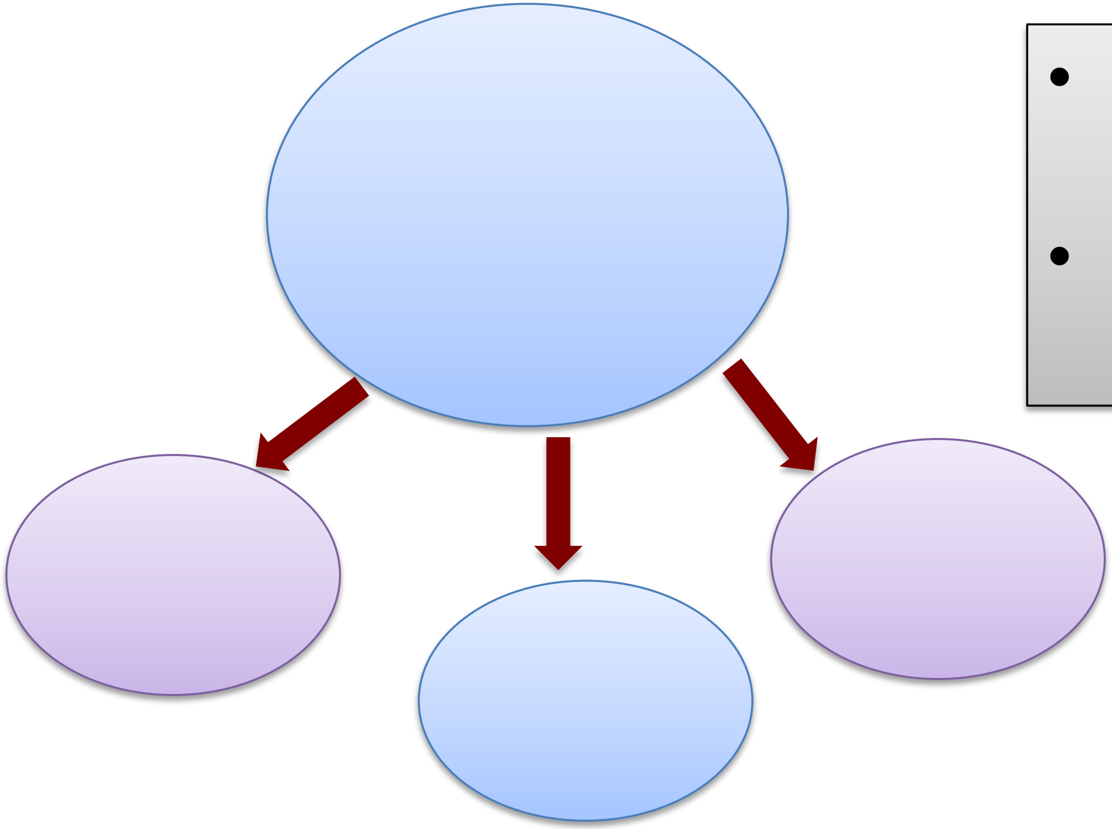
The Big Picture



- Reduce each piece to WGP instance separately
- Apply approx. algorithm to each WGP instance

a piece w high solution value and honest solution becomes inactive

The Big Picture

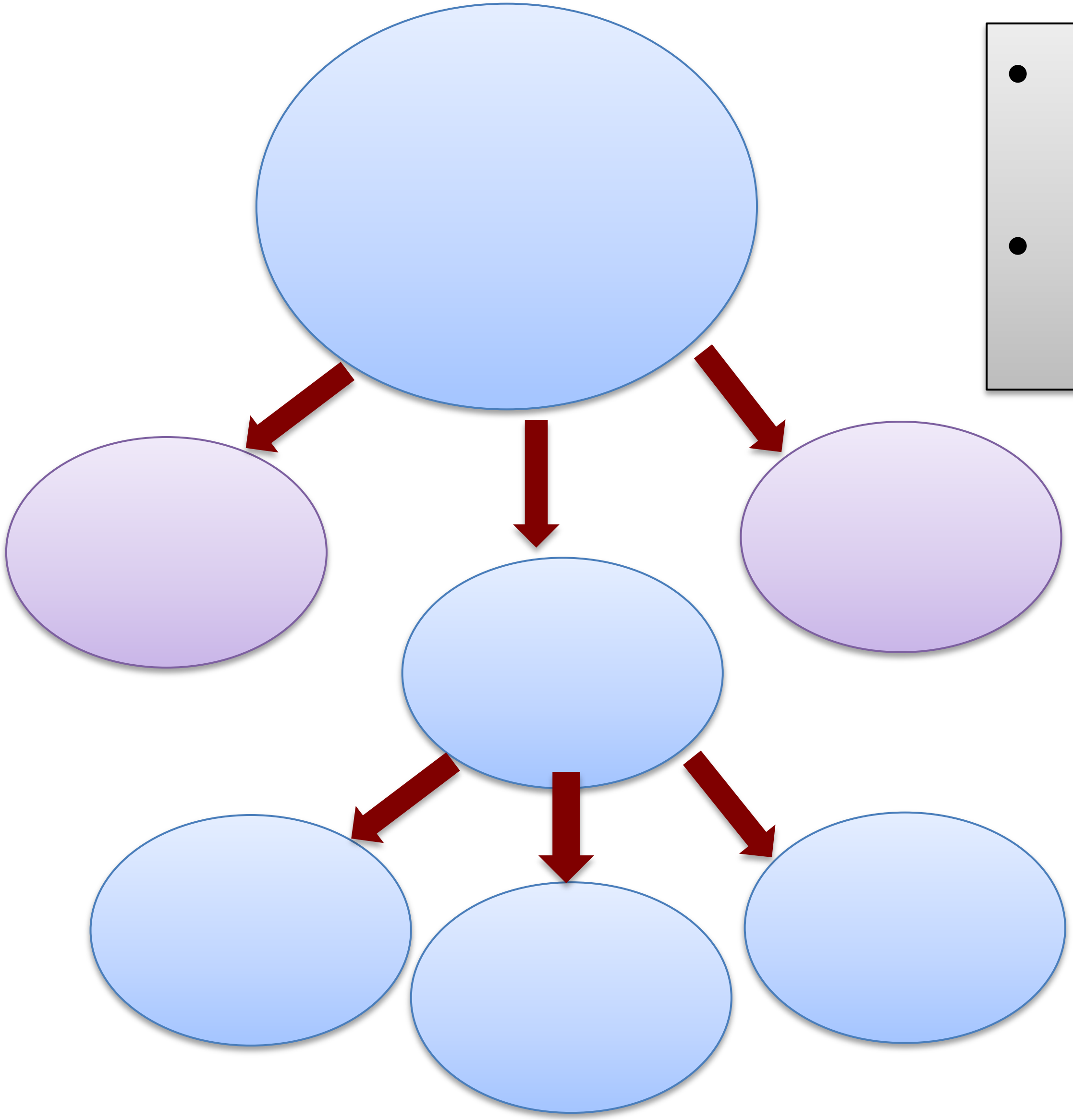


- Reduce each piece to WGP instance separately
- Apply approx. algorithm to each WGP instance

a piece w high solution value and honest solution becomes inactive

each piece w high solution value and cheating solution is cut again

The Big Picture

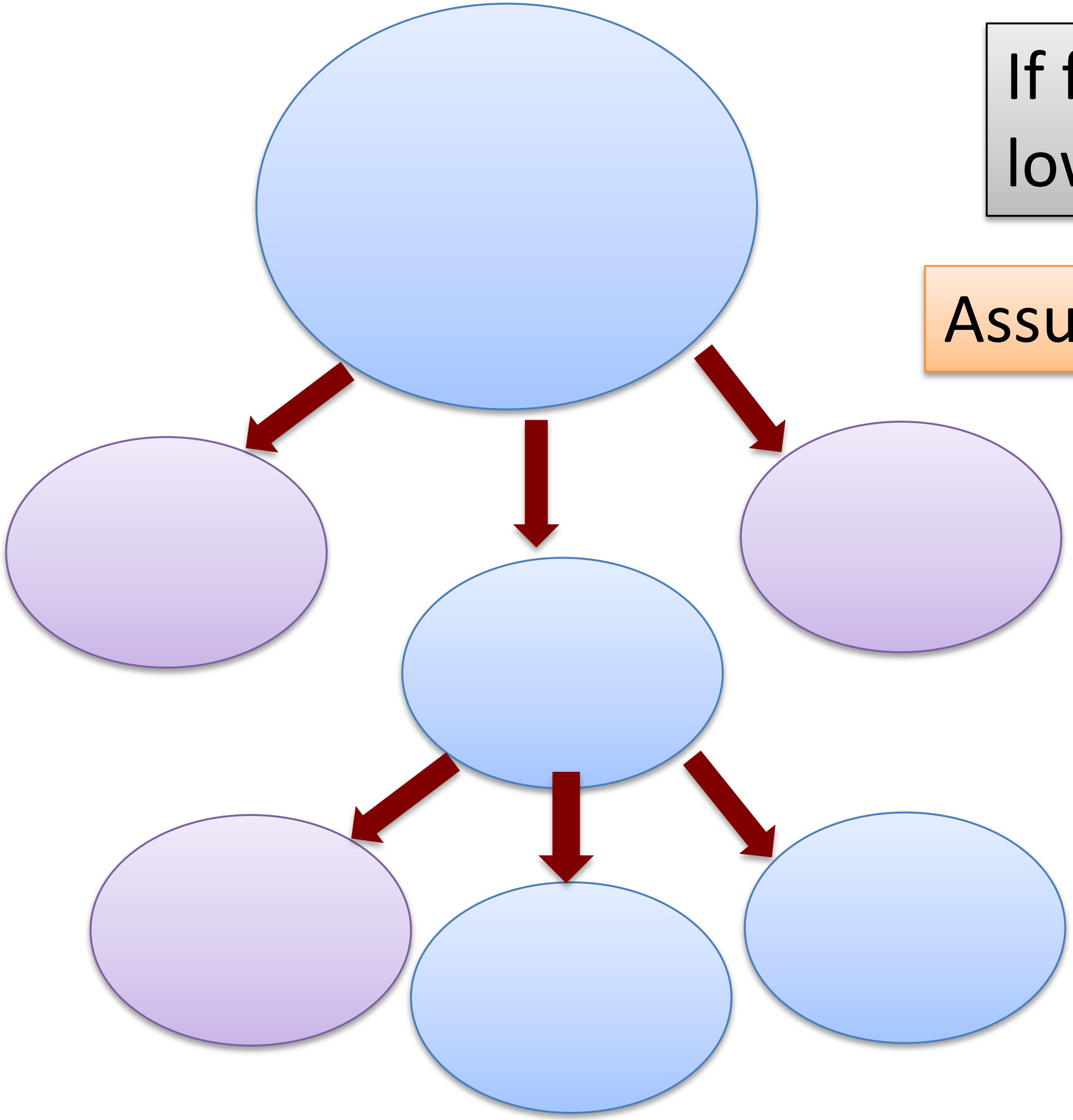


- Reduce each piece to WGP instance separately
- Apply approx. algorithm to each WGP instance

a piece w high solution value and honest solution becomes inactive

each piece w high solution value and cheating solution is cut again

The Big Picture



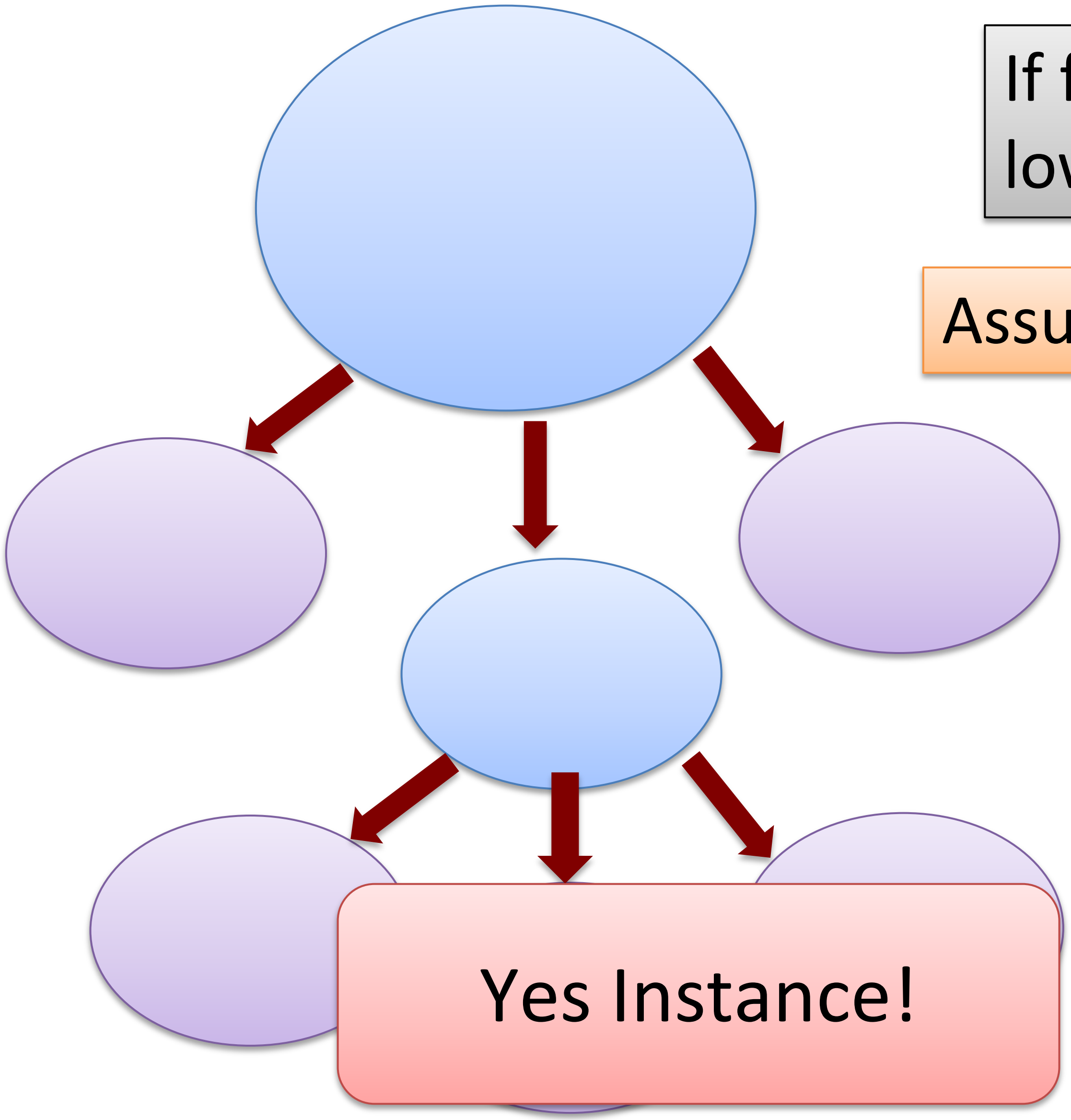
If for any resulting cluster we get a solution of low cost, we know it's a No-Instance.

Assume this never happens

Can't cut forever

when we stop cutting, every current cluster is inactive, so we can satisfy many of its constraints

The Big Picture



If for any resulting cluster we get a solution of low cost, we know it’s a No-Instance.

Assume this never happens

Can’t cut forever

when we stop cutting, every current cluster is inactive, so we can satisfy many of its constraints

many constraints are preserved, so we can satisfy many constraints overall

Summary: Main Ideas

- Introduce intermediate problem WGP
- Can modify it to suit our reduction
- Cook not Karp reduction.

Single-Shot vs Multi-shot Reductions

- Intuitively, it feels like multi-shot reductions should be more powerful
- But in almost all cases, single-shot reductions are sufficient

Exception: NP-hardness of embedding metrics into L_1 [Karzanov]

Single-Shot vs Multi-shot Reductions

- Intuitively, it feels like multi-shot reductions should be more powerful
- But in almost all cases, single-shot reductions are sufficient
- It is possible that one can construct a single-shot reduction from 3-Coloring to NDP

a bug, not a feature?

Conclusions

- We showed: almost polynomial hardness of NDP in grids
  - tradeoffs between hardness factor and complexity assumption.
- Congestion minimization:
  - $O(\log n / \log \log n)$-approximation algorithm
  - $\Omega(\log \log n)$-hardness of approximation

Thank you!