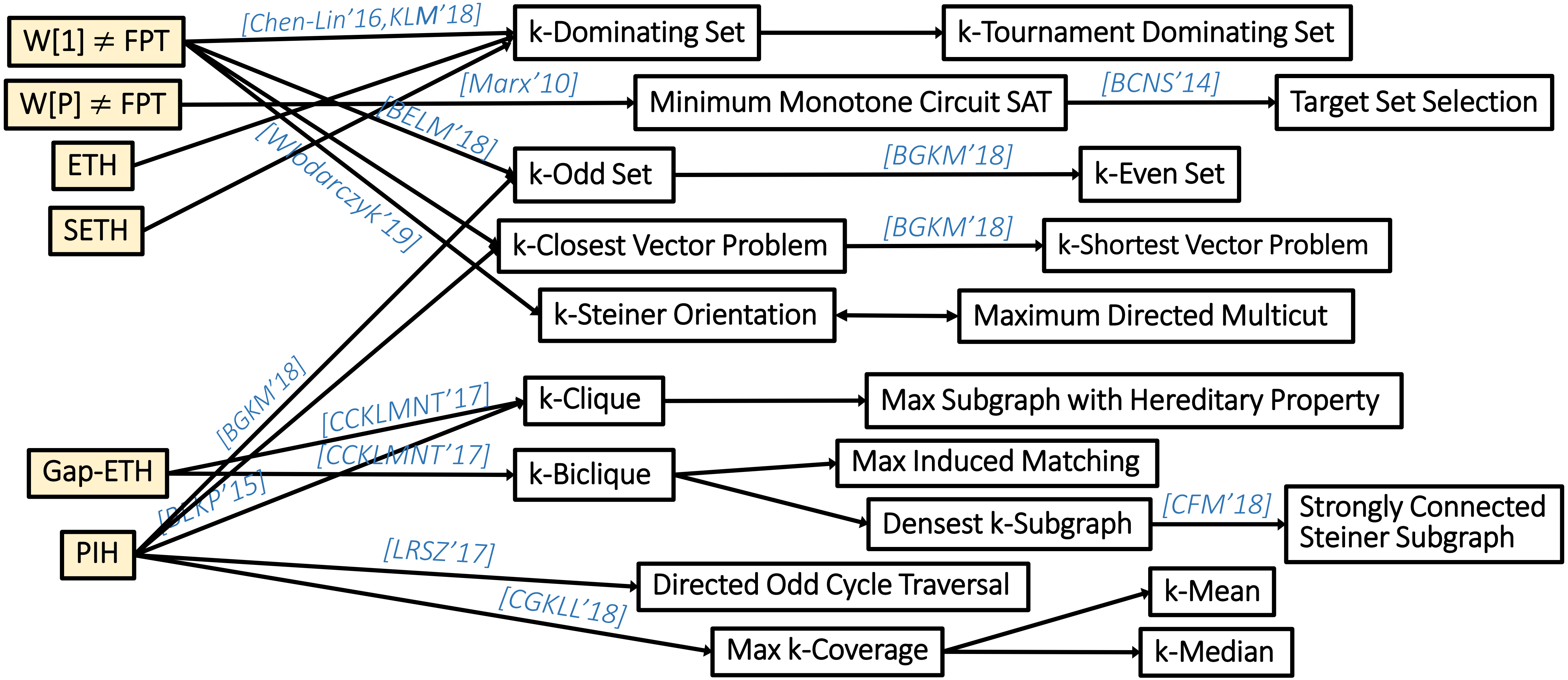
Parameterized Inapproximability

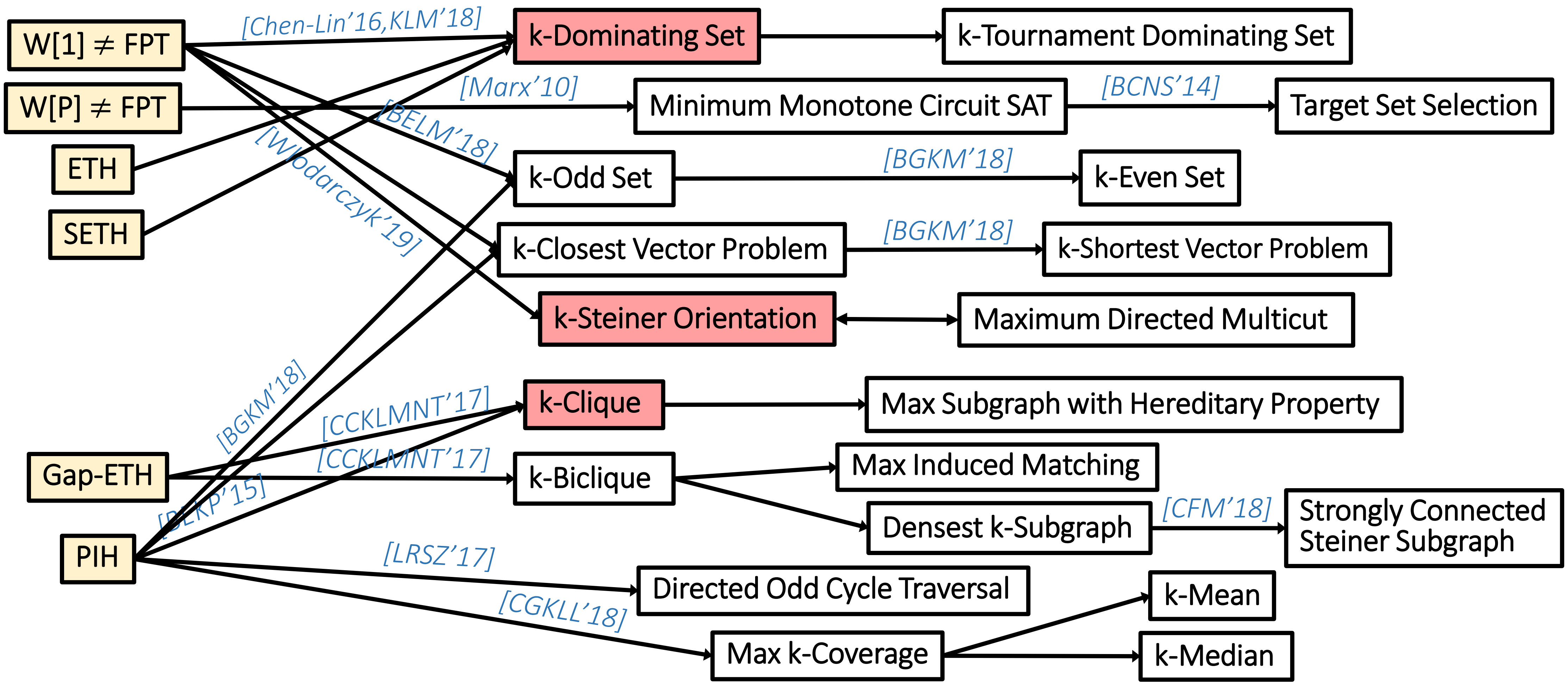
Pasin Manurangsi

Google Research

Parameterized Inapproximability: *Recent Developments*



Parameterized Inapproximability: *Recent Developments*



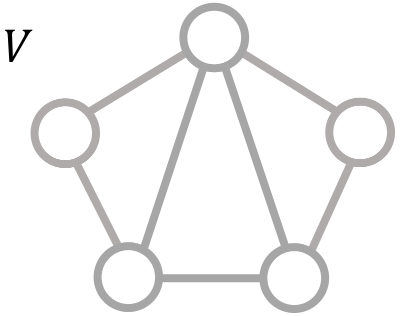
Parameterized Complexity: *A Brief Introduction*

k-Clique

Input: Graph $G = (V, E)$, integer k

Output: A subset $S \subseteq V$ of size k that induces a clique

V

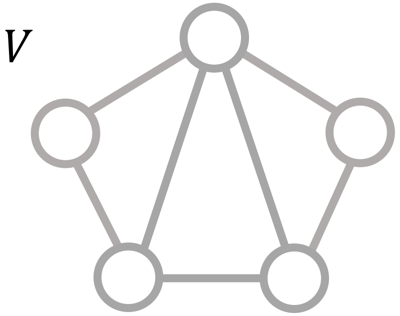


$k = 3$

k-Dominating Set

Input: Graph $G = (V, E)$, integer k

V

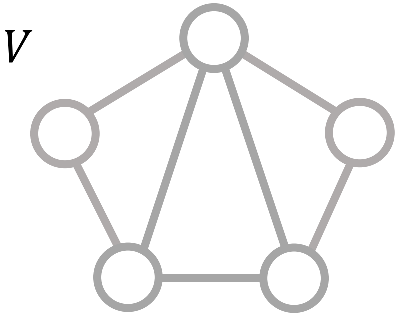


$k = 2$

k-Vertex Cover

Input: Graph $G = (V, E)$, integer k

V



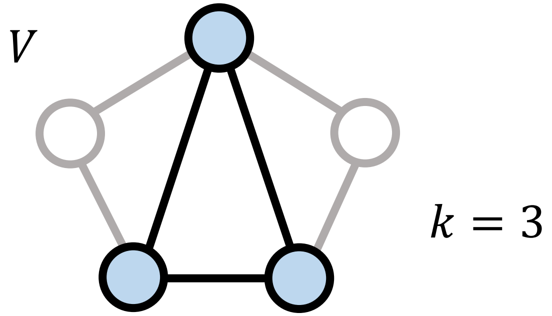
$k = 3$

Parameterized Complexity: *A Brief Introduction*

k-Clique

Input: Graph $G = (V, E)$, integer k

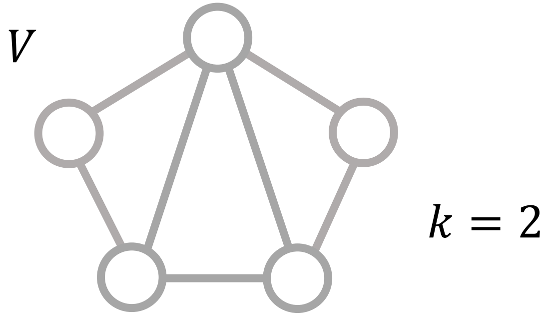
Output: A subset $S \subseteq V$ of size k that induces a clique



k-Dominating Set

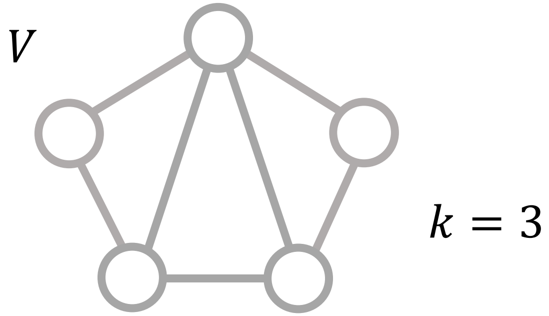
Input: Graph $G = (V, E)$, integer k

Output: A subset $S \subseteq V$ of size k such that $S \cup N(S) = V$



k-Vertex Cover

Input: Graph $G = (V, E)$, integer k

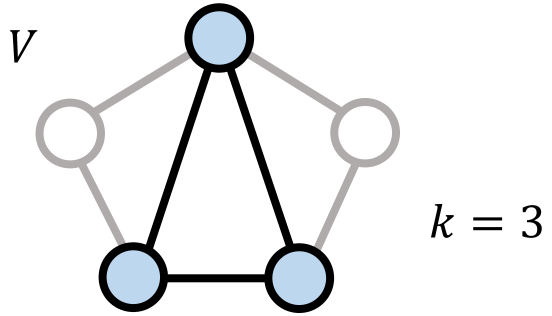


Parameterized Complexity: *A Brief Introduction*

k-Clique

Input: Graph $G = (V, E)$, integer k

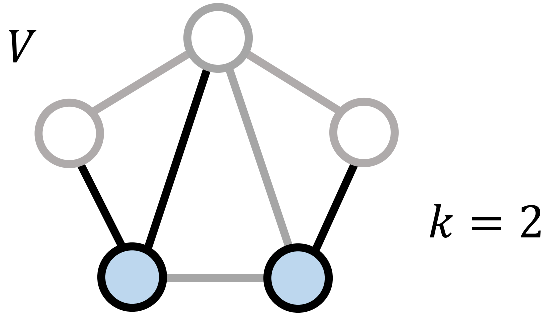
Output: A subset $S \subseteq V$ of size k that induces a clique



k-Dominating Set

Input: Graph $G = (V, E)$, integer k

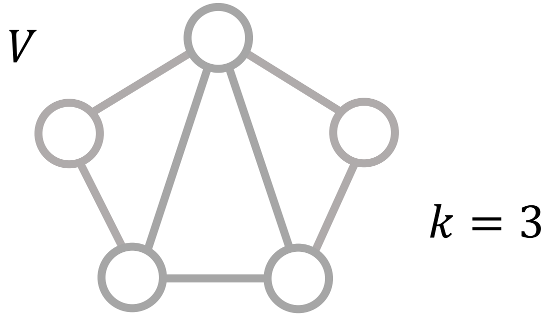
Output: A subset $S \subseteq V$ of size k such that $S \cup N(S) = V$



k-Vertex Cover

Input: Graph $G = (V, E)$, integer k

Output: A subset $S \subseteq V$ of size k that covers all edges

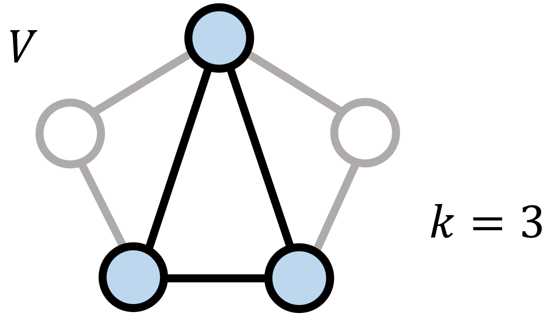


Parameterized Complexity: *A Brief Introduction*

k-Clique

Input: Graph $G = (V, E)$, integer k

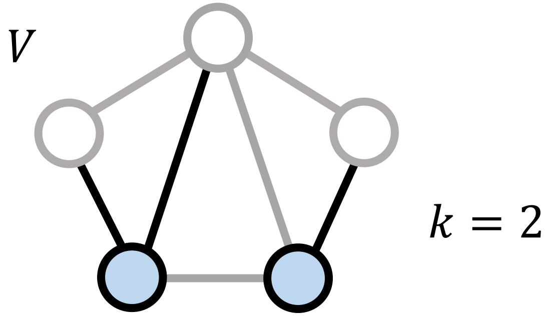
Output: A subset $S \subseteq V$ of size k that induces a clique



k-Dominating Set

Input: Graph $G = (V, E)$, integer k

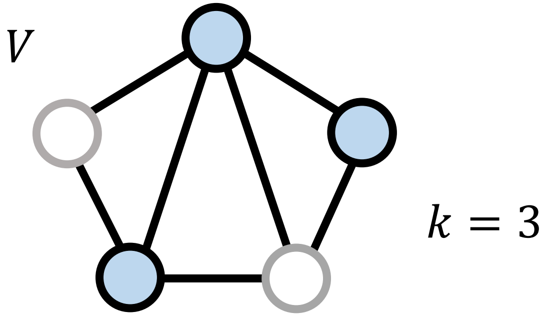
Output: A subset $S \subseteq V$ of size k such that $S \cup N(S) = V$



k-Vertex Cover

Input: Graph $G = (V, E)$, integer k

Output: A subset $S \subseteq V$ of size k that covers all edges



Is there any poly time algo for the problems?

Unlikely: all are *NP*-complete *[Karp'72]*

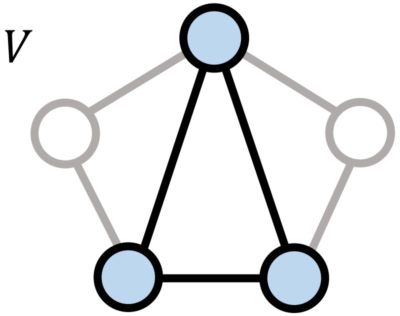
Parameterized Complexity: *A Brief Introduction*

k-Clique

Input: Graph $G = (V, E)$, integer k

Output: A subset $S \subseteq V$ of size k that induces a clique

V



$k = 3$

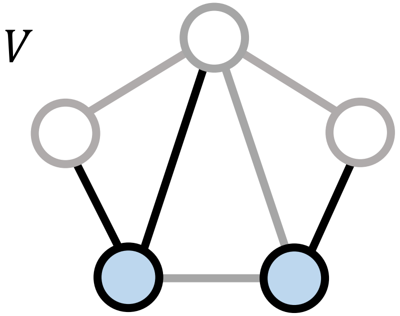
?

k-Dominating Set

Input: Graph $G = (V, E)$, integer k

Output: A subset $S \subseteq V$ of size k such that $S \cup N(S) = V$

V



$k = 2$

?

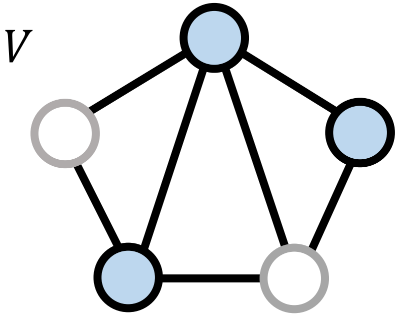
k-Vertex Cover

Parameter

Input: Graph $G = (V, E)$, integer k

Output: A subset $S \subseteq V$ of size k that covers all edges

V



$k = 3$

$2^k \cdot n^{O(1)}$ time algo

Enumeration Algorithm: $n^{O(k)}$ time

Parameterized Complexity: *A Brief Introduction*

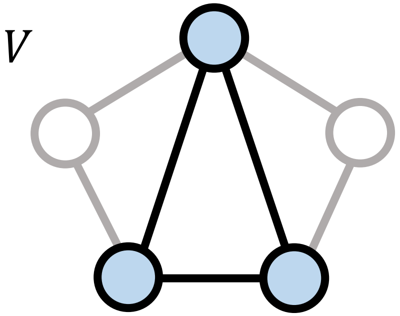
Parameter

k-Clique

Input: Graph $G = (V, E)$, integer k

Output: A subset $S \subseteq V$ of size k that induces a clique

V



$k = 3$

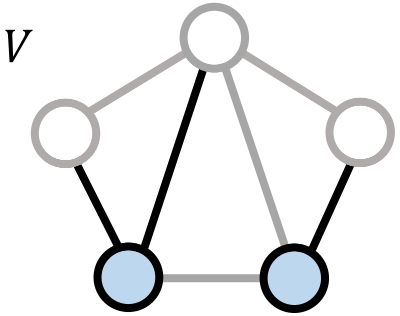
?

k-Dominating Set

Input: Graph $G = (V, E)$, integer k

Output: A subset $S \subseteq V$ of size k such that $S \cup N(S) = V$

V



$k = 2$

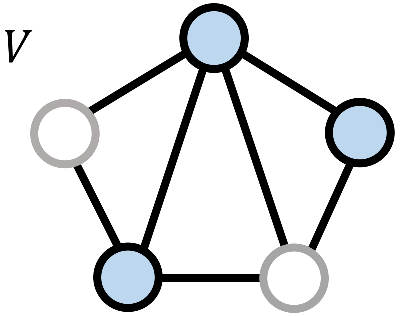
?

k-Vertex Cover

Input: Graph $G = (V, E)$, integer k

Output: A subset $S \subseteq V$ of size k that covers all edges

V



$k = 3$

$2^k \cdot n^{O(1)}$ time algo

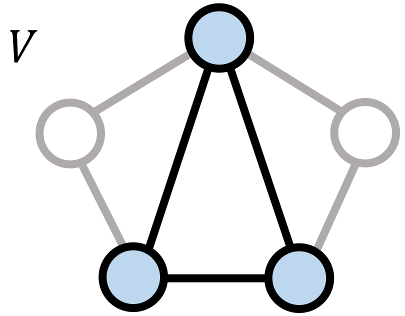
FPT Algo: $f(k)n^{O(1)}$-time for some function f

Parameterized Complexity: *A Brief Introduction*

k-Clique

Input: Graph $G = (V, E)$, integer k

Output: A subset $S \subseteq V$ of size k that induces a clique



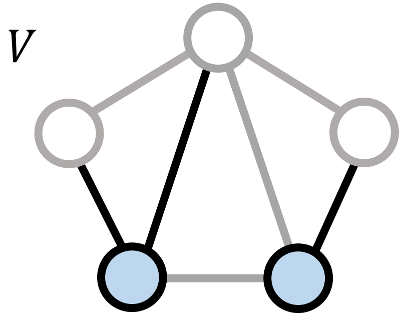
$k = 3$

[Downey-Fellows'92] W[1]-complete

k-Dominating Set

Input: Graph $G = (V, E)$, integer k

Output: A subset $S \subseteq V$ of size k such that $S \cup N(S) = V$



$k = 2$

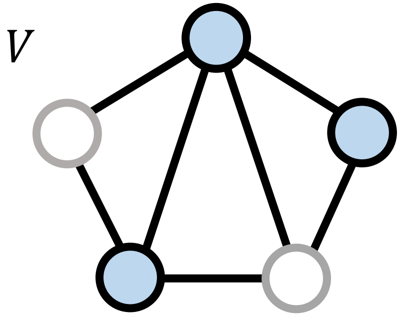
[Downey-Fellows'92] W[2]-complete

k-Vertex Cover

Parameter

Input: Graph $G = (V, E)$, integer k

Output: A subset $S \subseteq V$ of size k that covers all edges



$k = 3$

$2^k \cdot n^{O(1)}$ time algo

FPT Algo: $f(k)n^{O(1)}$-time for some function f

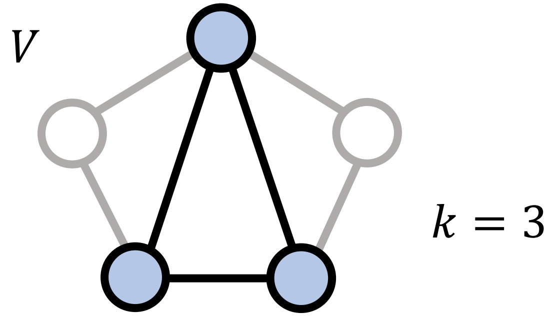
Parameterized Complexity: *A Brief Introduction*

k-Clique

Parameter

Input: A graph $G = (V, E)$, integer k

Output: A subset $S \subseteq V$ of size k that induces a clique



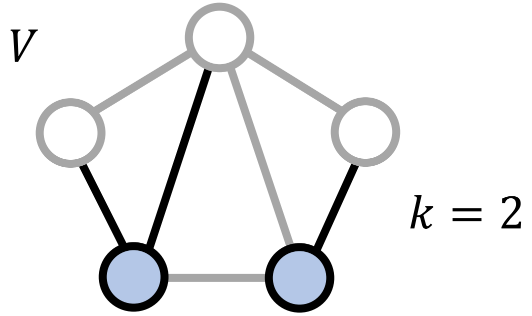
[Downey-Fellows'92] k-Clique is W[1]-complete

k-Dominating Set

Parameter

Input: A graph $G = (V, E)$, integer k

Output: A subset $S \subseteq V$ of size k such that $S \cup N(S) = V$



[Downey-Fellows'92] k-DomSet is W[2]-complete

FPT Algo: $f(k)n^{O(1)}$-time for some function f

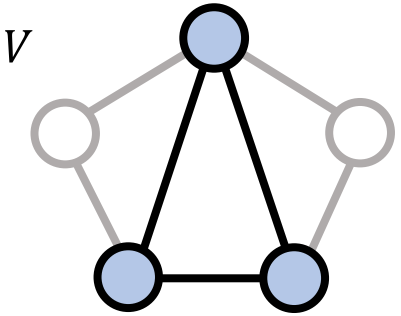
Parameterized *Approximation*

k-Clique

Parameter

Input: A graph $G = (V, E)$, integer k

Output: A subset $S \subseteq V$ of size k that induces a clique

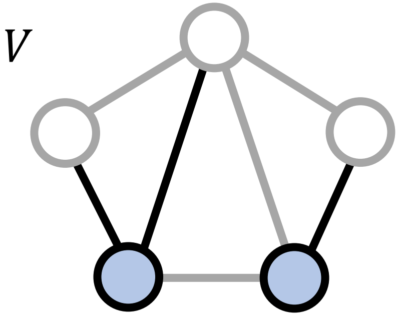


k-Dominating Set

Parameter

Input: A graph $G = (V, E)$, integer k

Output: A subset $S \subseteq V$ of size k such that $S \cup N(S) = V$



FPT Approx Algo: $g(k)$-approx $f(k)n^{O(1)}$-time for some function f, g

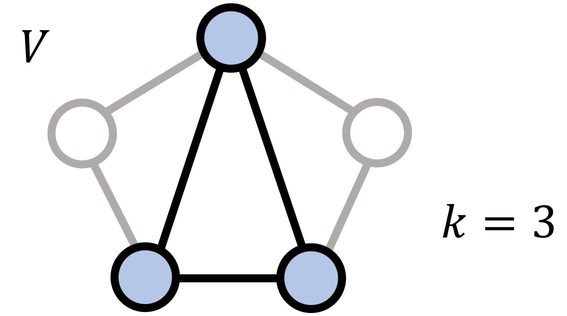
Parameterized *Approximation*

k-Clique

Parameter

Input: A graph $G = (V, E)$, integer k

Output: A subset $S \subseteq V$ of size $k/g(k)$ that induces a clique

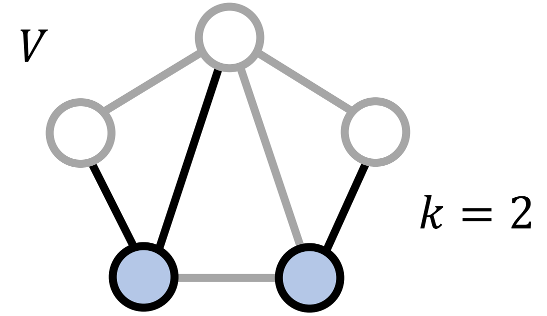


k-Dominating Set

Parameter

Input: A graph $G = (V, E)$, integer k

Output: A subset $S \subseteq V$ of size k such that $S \cup N(S) = V$



FPT Approx Algo: $g(k)$-approx $f(k)n^{O(1)}$-time for some function f, g

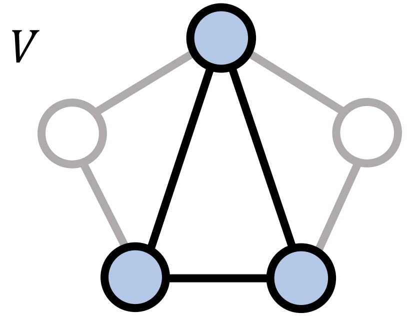
Parameterized *Approximation*

k-Clique

Parameter

Input: A graph $G = (V, E)$, integer k

Output: A subset $S \subseteq V$ of size $k/g(k)$ that induces a clique



$k = 3$

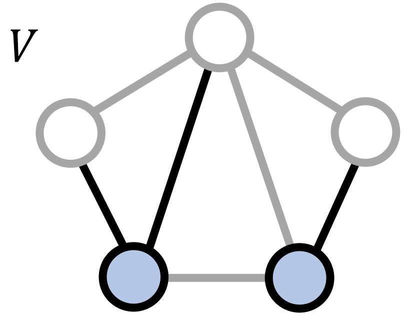
Is there 1.1-FPT-approx. algo?

k-Dominating Set

Parameter

Input: A graph $G = (V, E)$, integer k

Output: A subset $S \subseteq V$ of size $k \cdot g(k)$ such that $S \cup N(S) = V$



$k = 2$

Is there 1.1-FPT-approx. algo?

FPT Approx Algo: $g(k)$-approx $f(k)n^{O(1)}$-time for some function f, g

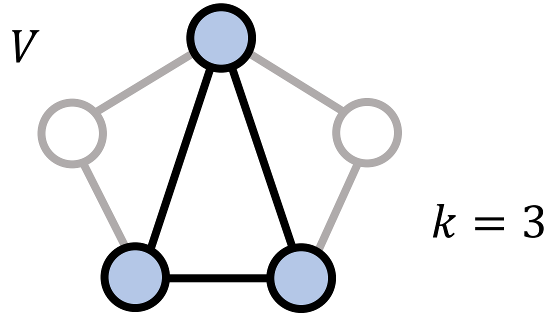
Parameterized *Approximation*

k-Clique

Parameter

Input: A graph $G = (V, E)$, integer k

Output: A subset $S \subseteq V$ of size $k/g(k)$ that induces a clique



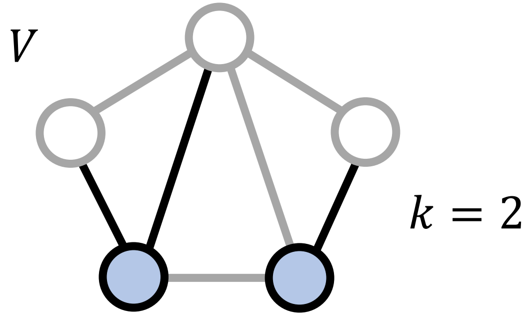
Is there $o(k)$-FPT-approx. algo?

k-Dominating Set

Parameter

Input: A graph $G = (V, E)$, integer k

Output: A subset $S \subseteq V$ of size $k \cdot g(k)$ such that $S \cup N(S) = V$



Is there 1.1-FPT-approx. algo?

FPT Approx Algo: $g(k)$-approx $f(k)n^{O(1)}$-time for some function f, g

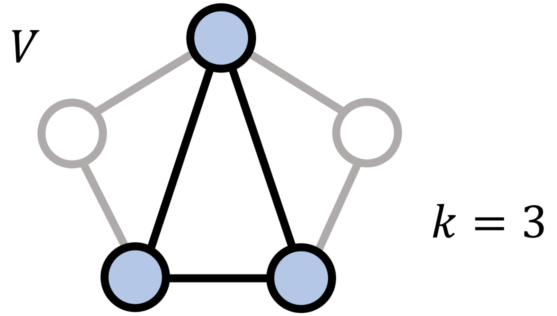
Parameterized *Approximation*

k-Clique

Parameter

Input: A graph $G = (V, E)$, integer k

Output: A subset $S \subseteq V$ of size $k/g(k)$ that induces a clique



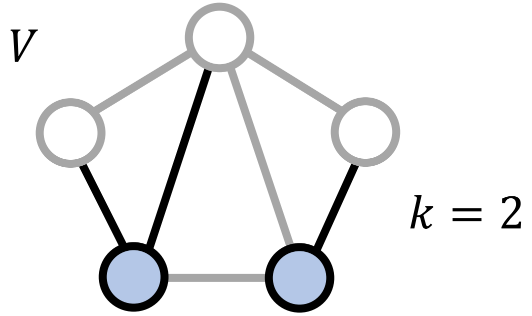
Is there $o(k)$-FPT-approx. algo?

k-Dominating Set

Parameter

Input: A graph $G = (V, E)$, integer k

Output: A subset $S \subseteq V$ of size $k \cdot g(k)$ such that $S \cup N(S) = V$



Is there $g(k)$-FPT-approx. algo for any g?

FPT Approx Algo: $g(k)$-approx $f(k)n^{O(1)}$-time for some function f, g

Complexity Assumptions

Exponential Time Hypotheses

No Gap

Exponential time Hypothesis (ETH) *[Impagliazzo, Paturi'01]*
No $2^{o(n)}$-time algo can decide whether a 3CNF formula is satisfiable

Strong ETH (SETH) *[Impagliazzo, Paturi'01]*
For any constant $\varepsilon > 0$, no $O(2^{(1-\varepsilon)n})$-time algorithm can decide whether a CNF formula is satisfiable

Gap

Gap ETH (Gap-ETH) *[Dinur'16] [M, Raghavendra'16]*
For some constant $\delta > 0$, no $2^{o(n)}$-time algorithm can distinguish between a satisfiable CNF formula and one which is not even $(1-\delta)$-satisfiable.

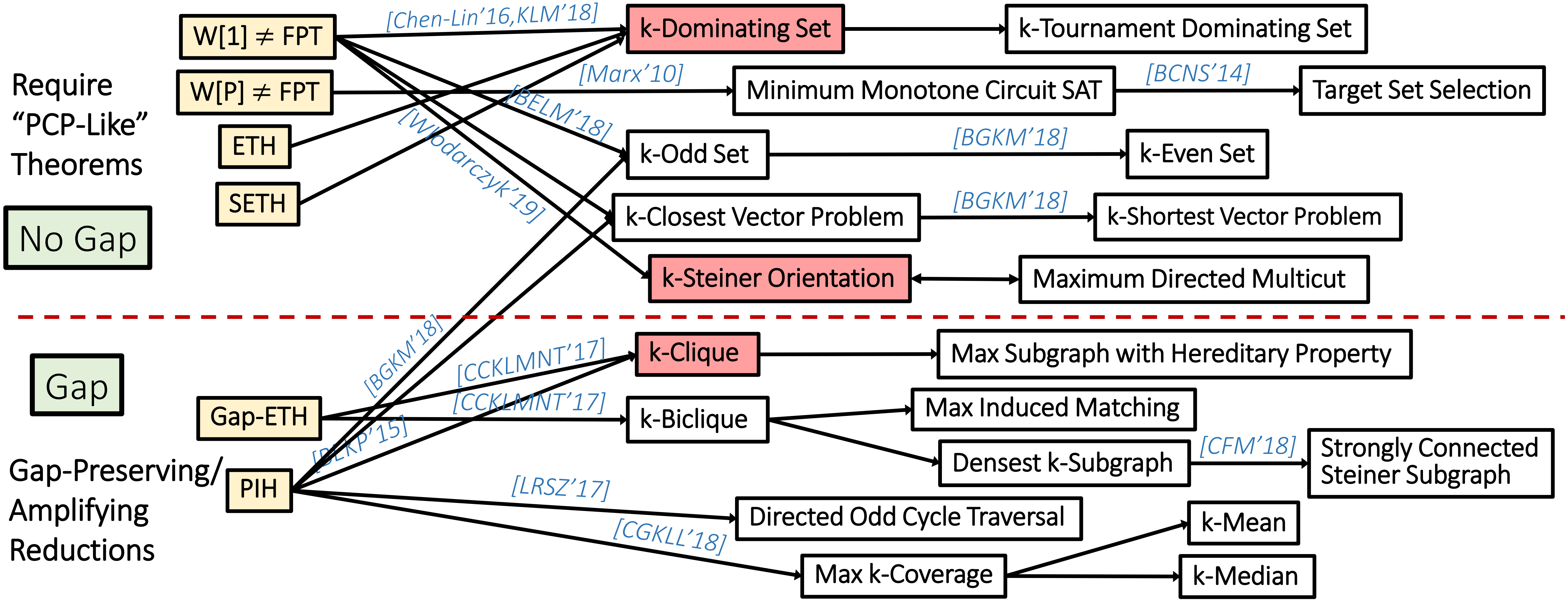
Parameterized Complexity

W[1] $\neq$ FPT Assumption *[Downey, Fellows'92]*
For any function f, no $f(k)n^{o(1)}$-time algo can decide whether a given graph contains a k-clique

W[2] $\neq$ FPT Assumption *[Downey, Fellows'92]*
For any function f, no $f(k)n^{o(1)}$-time algo can decide whether a given graph contains a dominating set of size k

Parameterized Inapproximability Hypothesis (PIH) *[Lokshtanov, Ramanujan, Saurabh, Zehavi'17]*
For some $\delta > 0$ and any function f, no $f(k)n^{O(1)}$-time algo can distinguish between a satisfiable 2CSP instance with k variables and alphabet size n and one which is not even $(1 - \delta)$-satisfiable.

Parameterized Inapproximability: *Recent Developments*



Part I:
Clique

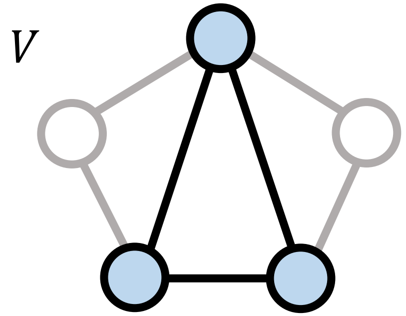
Parameterized Inapproximability

k-Clique

Input: A graph $G = (V, E)$, integer k

Parameter: k

Output: A subset $S \subseteq V$ of size k that induces a clique



FPT Algo: $f(k)n^{O(1)}$-time for some function f

[Downey-Fellows'92] k-Clique is W[1]-complete

[Chen-Chor-Fellows-Huang-Juedes-Kanj-Xia'05]

ETH $\Rightarrow$ no $f(k)n^{o(k)}$-time algo for k-Clique

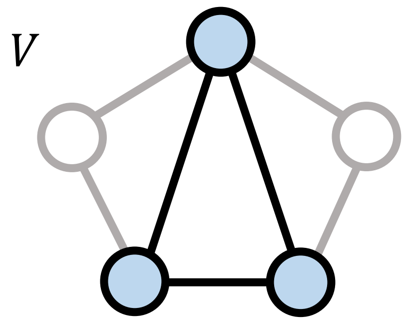
Parameterized Inapproximability

k-Clique

Input: A graph $G = (V, E)$, integer k

Parameter: k

Output: A subset $S \subseteq V$ of size $k/g(k)$ that induces a clique



FPT Approx Algo: $g(k)$-approx $f(k)n^{O(1)}$-time for some function f, g

[Downey-Fellows'92] k-Clique is W[1]-complete

[Chen-Chor-Fellows-Huang-Juedes-Kanj-Xia'05]

ETH $\Rightarrow$ no $f(k)n^{o(k)}$-time algo for k-Clique

[Bonnet-Escoffier-Kim-Paschos'15]

Gap-ETH $\Rightarrow$ no constant factor FPT approx algo for k-Clique

[Chalermsook-Cygan-Kortsarz-Laekhanukit-**M**-Nanongkai-Trevisan'17]

Gap-ETH $\Rightarrow$ no $g(k)$-approx $f(k)n^{o(k/g(k))}$-time algo for k-Clique for any $g = o(k)$

Inherently Enumerative

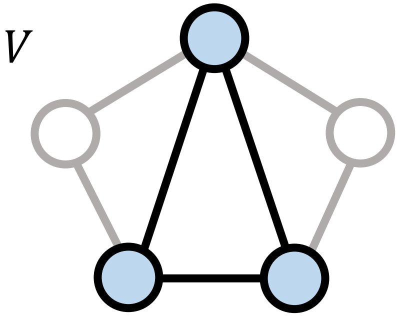
There exists $\delta > 0$ such that for any sufficiently large $k < r$, no $O(n^{\delta k})$-time algo can distinguish between

- $Clique(G) \leq k$
- $Clique(G) \geq r$

Parameterized Inapproximability

k-Clique

Input: A graph $G = (V, E)$, integer k
Parameter: k
Output: A subset $S \subseteq V$ of size $k/g(k)$ that induces a clique



FPT Approx Algo: $g(k)$-approx $f(k)n^{O(1)}$-time for some function f, g

[Downey-Fellows'92] k-Clique is W[1]-complete

[Chen-Chor-Fellows-Huang-Juedes-Kanj-Xia'05]
ETH ⇒ no $f(k)n^{o(k)}$-time algo for k-Clique

[Bonnet-Escoffier-Kim-Paschos'15]
Gap-ETH ⇒ no constant factor FPT approx algo for k-Clique

[Chalermsook-Cygan-Kortsarz-Laekhanukit-**M**-Nanongkai-Trevisan'17]
Gap-ETH ⇒ no $g(k)$-approx $f(k)n^{o(k/g(k))}$-time algo for k-Clique for any $g = o(k)$

Inherently Enumerative

There exists $\delta > 0$ such that for any sufficiently large $k < r$, no $O(n^{\delta k})$-time algo can distinguish between
- $Clique(G) \leq k$
- $Clique(G) \geq r$

Proof Sketch: *Inherently Enumerability of k-Clique*

3SAT Instance $\boldsymbol{\Phi}$
Size: n
YES: $val(\Phi) = 1$
NO: $val(\Phi) < 0.99$

$\longrightarrow$

Clique Instance $\boldsymbol{G}$
Size: $N = r \cdot 2^{O(n/k)}$
YES: $Clique(\mathrm{G}) = r$
NO: $Clique(\mathrm{G}) \leq k$

Lower Bound: $2^{\Omega(n)}$

Lower Bound: $N^{\Omega(k)}$

GOAL
Assuming Gap-ETH, there exists $\delta > 0$ such that for any sufficiently large $k < r$, no $\mathrm{O}(N^{\delta k})$-time algo can distinguish between
- $Clique(G) \leq k$
- $Clique(G) \geq r$

Proof Sketch: *Inherently Enumerability of k-Clique*

3SAT Instance Φ
Size: n
YES: $val(\Phi) = 1$
NO: $val(\Phi) < 0.99$

Clique Instance G
Size: $N = r \cdot 2^{O(n/k)}$ ✓
YES: $Clique(\mathrm{G}) = r$ ✓
NO: $Clique(\mathrm{G}) \leq k$

[Bellare-Goldreich-Sudan'98]
...96]
...wasser-Lovasz-Safra-Szegedy'96]

At most $r \cdot 2^{1000m/k}$ vertices

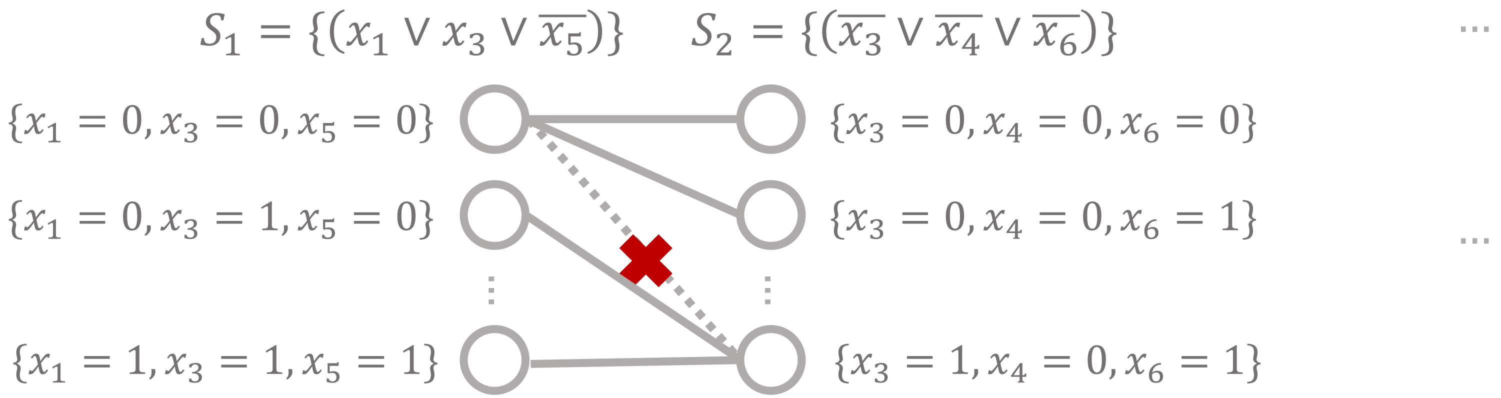
Pick all partial assignments of a satisfying assignment!

$(x_1 \vee x_3 \vee \overline{x_5})$
$\wedge (\overline{x_3} \vee \overline{x_4} \vee \overline{x_6})$
$\wedge (x_2 \vee \overline{x_4} \vee x_6) \wedge \cdots$

m = # of clauses

Assume w.l.o.g. $m = O(n)$

- Random subsets $S_1, S_2, \ldots, S_r$ of $1000m/k$ clauses
- For each S_i, create one vertex for a partial assignment to S_i
- Join two vertices by an edge if they are consistent



Proof Sketch: *Inherently Enumerability of k-Clique*

GOAL: $val(\Phi) < 0.99 \Rightarrow Clique(G) < k$

Suppose that $Clique(G) \geq k$

$S_1 = \{(x_1 \vee x_3 \vee \overline{x_5})\}$ $\quad S_2 = \{(\overline{x_3} \vee \overline{x_4} \vee \overline{x_6})\}$ $\quad \cdots \quad S_r = \{(\overline{x_5} \vee \overline{x_n} \vee \overline{x_{n-1}})\}$

$\{x_1 = 0, x_3 = 0, x_5 = 0\}$ $\quad \{x_3 = 0, x_4 = 0, x_6 = 0\}$ $\quad \{x_3 = 0, x_4 = 0, x_6 = 0\}$

$\{x_1 = 0, x_3 = 1, x_5 = 0\}$ $\quad \{x_3 = 0, x_4 = 0, x_6 = 1\}$ $\quad \cdots \quad \{x_3 = 0, x_4 = 0, x_6 = 1\}$

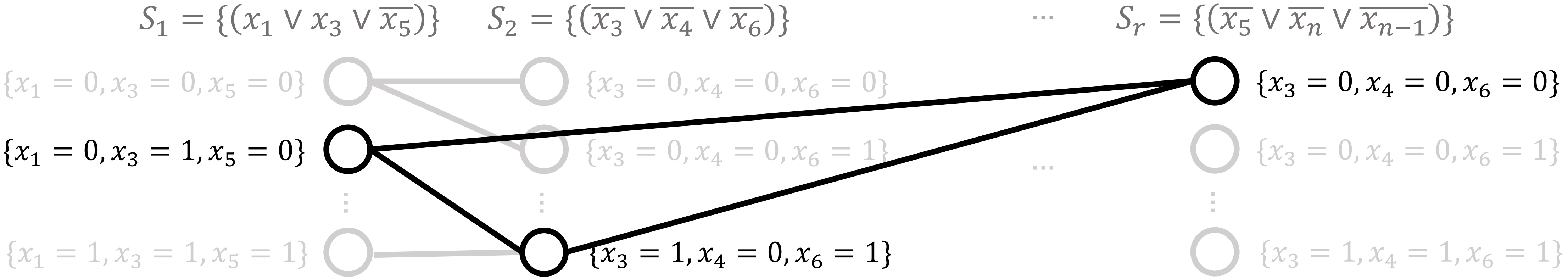
$\vdots \quad \vdots \quad \vdots$

$\{x_1 = 1, x_3 = 1, x_5 = 1\}$ $\quad \{x_3 = 1, x_4 = 0, x_6 = 1\}$ $\quad \{x_3 = 1, x_4 = 1, x_6 = 1\}$

Proof Sketch: *Inherently Enumerability of k-Clique*

GOAL: $val(\Phi) < 0.99 \Rightarrow Clique(G) < k$

Suppose that $Clique(G) \geq k$



Let these vertices be from $S_{i_1}, S_{i_2}, \ldots, S_{i_k}$ → An assignment that satisfies all clauses in $S_{i_1} \cup S_{i_2} \cup \cdots \cup S_{i_k}$ → $val(\Phi) \geq 0.99$

Recall $S_{i_1}, S_{i_2}, \cdots, S_{i_k}$ are random subsets of clauses of size $1000m/k$ → With high probability, $\left|S_{i_1} \cup S_{i_2} \cup \cdots \cup S_{i_k}\right| \geq 0.99m$

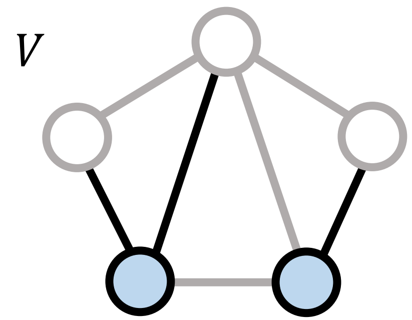
QED!

Part II:
Dominating Set

Parameterized Inapproximability

k-Dominating Set

Input: A graph $G = (V, E)$, integer k
Parameter: k
Output: A subset $S \subseteq V$ of size k such that $S \cup N(S) = V$



FPT Algo: $f(k)n^{O(1)}$-time for some function f

[Downey-Fellows'92] k-Dominating Set is W[2]-complete

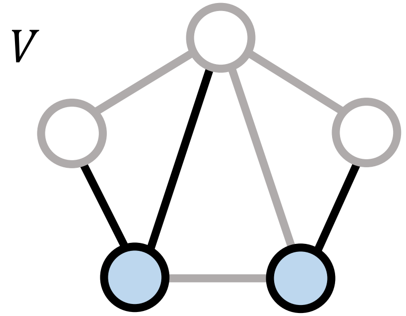
[Chen-Chor-Fellows-Huang-Juedes-Kanj-Xia'05]
ETH $\Rightarrow$ no $f(k)n^{o(k)}$-time algo for k-Dom Set

[Patrascu-Williams'10]
SETH $\Rightarrow$ no $f(k)n^{k-\varepsilon}$-time algo for k-Dom Set

Parameterized Inapproximability

k-Dominating Set

Input: A graph $G = (V, E)$, integer k
Parameter: k
Output: A subset $S \subseteq V$ of size $g(k) \cdot k$ such that $S \cup N(S) = V$



FPT Approx Algo: $g(k)$-approx $f(k)n^{O(1)}$-time for some function f, g

[Downey-Fellows'92] k-Dominating Set is W[2]-complete

[Chen-Chor-Fellows-Huang-Juedes-Kanj-Xia'05]
ETH $\Rightarrow$ no $f(k)n^{o(k)}$-time algo for k-Dom Set

[Patrascu-Williams'10]
SETH $\Rightarrow$ no $f(k)n^{k-\varepsilon}$-time algo for k-Dom Set

---

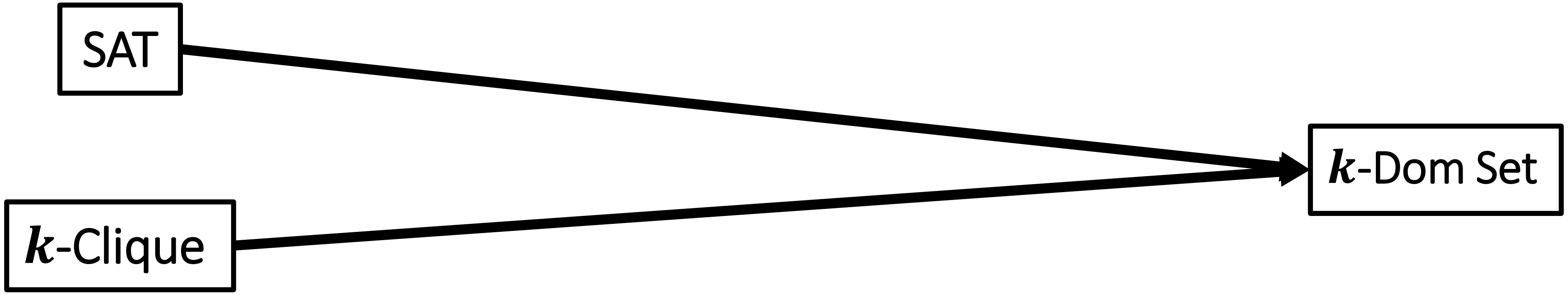
[Chen-Lin'16] W[1] $\neq$ FPT $\Rightarrow$ no constant factor FPT approx algo for k-Dom Set

*[Karthik-Laekhanukit-**M**'18]*
W[1] $\neq$ FPT $\Rightarrow$ no FPT $g(k)$-approx algo for k-Dom Set
ETH $\Rightarrow$ no $g(k)$-approx $f(k)n^{o(k)}$-time algo for k-Dom Set
SETH $\Rightarrow$ no $g(k)$-approx $f(k)n^{k-\varepsilon}$-time algo for k-Dom Set

[Lin'18] Alternative (beautiful) proof

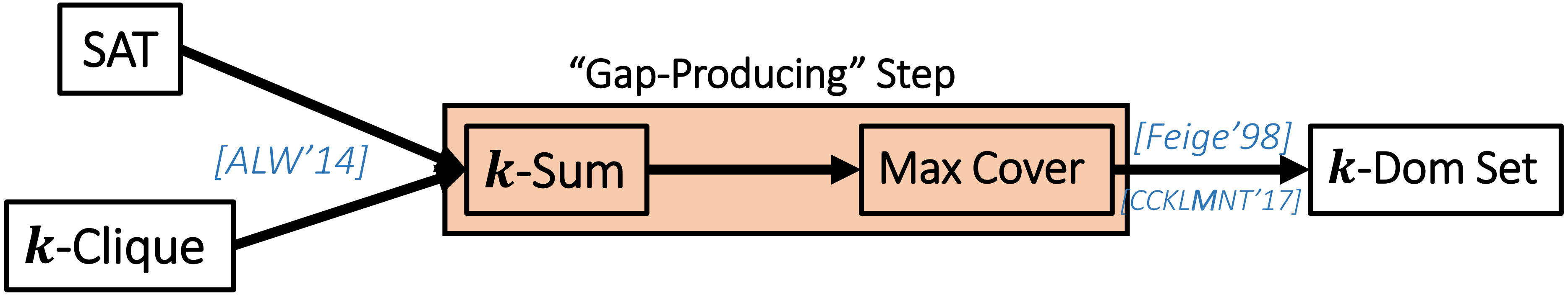
Proof Sketch: *Total Inapproximability of $\boldsymbol{k}$-Dom Set*

Overview



Proof Sketch: *Total Inapproximability of k-Dom Set*

Overview



Key: **Communication Protocol ⇒ hardness of approximation**

"Distributed PCP" framework of *[Abboud-Rubinstein-Williams'18]*

Proof Sketch: *Total Inapproximability of k-Dom Set*

k-Sum

Input: k sets of n integers $A_1, \ldots, A_k \subseteq [-n^{2k}, n^{2k}]$

Output: Whether there exists $a_1 \in A_1, \ldots, a_k \in A_k$ s.t. $a_1 + \cdots + a_k = 0$

[Abboud-Williams-Lewi'14]

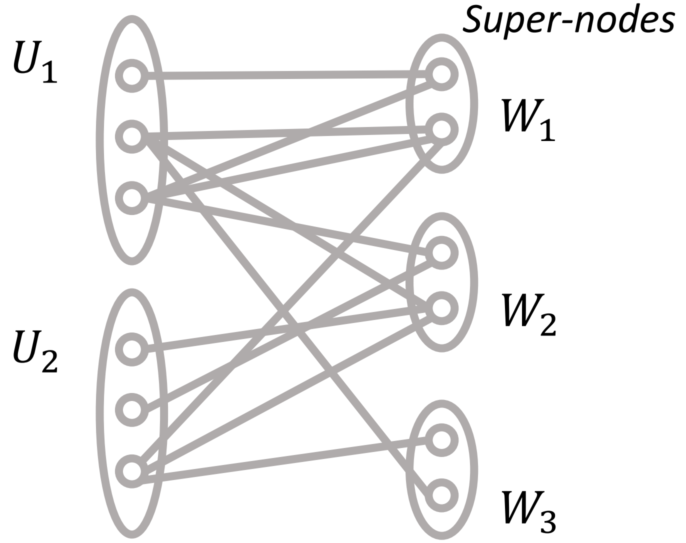
k-Sum is W[1]-complete

ETH ⇒ no $f(k)n^{o(k)}$-time algo for k-Sum

Max Cover

Input: A bipartite graph $(U_1 \cup \cdots \cup U_h, W_1 \cup \cdots \cup W_k, E)$

Parameter: k



Proof Sketch: *Total Inapproximability of k-Dom Set*

k-Sum

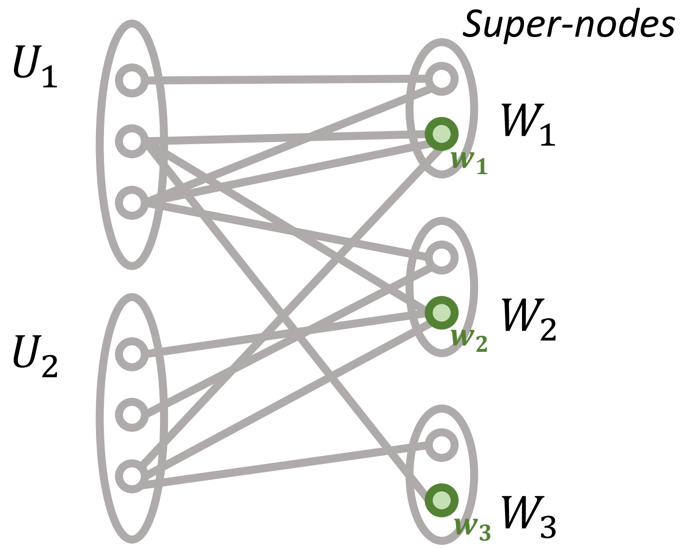
Input: k sets of n integers $A_1, \dots, A_k \subseteq [-n^{2k}, n^{2k}]$

Output: Whether there exists $a_1 \in A_1, \dots, a_k \in A_k$ s.t. $a_1 + \cdots + a_k = 0$

[Abboud-Williams-Lewi'14]
k-Sum is W[1]-complete
ETH ⇒ no $f(k)n^{o(k)}$-time algo for k-Sum

Max Cover

Input: A bipartite graph $(U_1 \cup \cdots \cup U_h, W_1 \cup \cdots \cup W_k, E)$
Parameter: k
Output: *$w_1 \in W_1, \dots, w_k \in W_k$ that "covers" maximum number of vertices in U*



Proof Sketch: *Total Inapproximability of k-Dom Set*

k-Sum

Input: k sets of n integers $A_1, \ldots, A_k \subseteq [-n^{2k}, n^{2k}]$

Output: Whether there exists $a_1 \in A_1, \ldots, a_k \in A_k$ s.t. $a_1 + \cdots + a_k = 0$

[Abboud-Williams-Lewi'14]

k-Sum is W[1]-complete

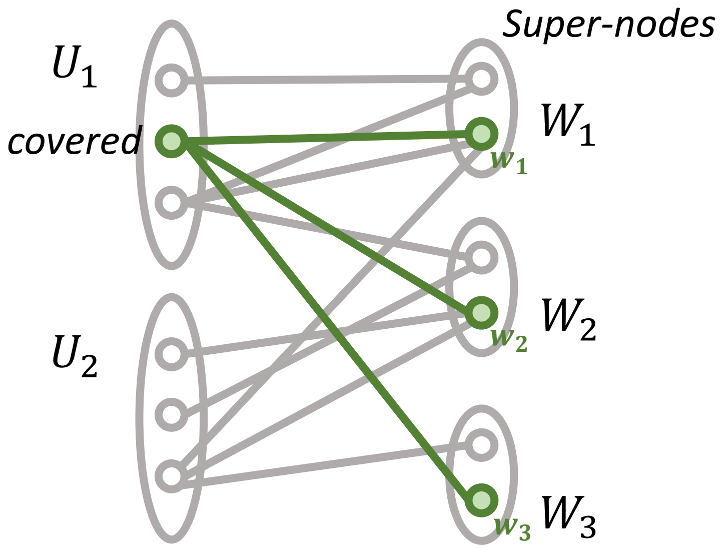
ETH $\Rightarrow$ no $f(k)n^{o(k)}$-time algo for k-Sum

Max Cover

Input: A bipartite graph $(U_1 \cup \cdots \cup U_h, W_1 \cup \cdots \cup W_k, E)$

Parameter: k

Output: *$w_1 \in W_1, \ldots, w_k \in W_k$ that "covers" maximum number of vertices in U*



Proof Sketch: *Total Inapproximability of k-Dom Set*

k-Sum

Input: k sets of n integers $A_1, \dots, A_k \subseteq [-n^{2k}, n^{2k}]$

Output: Whether there exists $a_1 \in A_1, \dots, a_k \in A_k$ s.t. $a_1 + \cdots + a_k = 0$

[Abboud-Williams-Lewi'14]

k-Sum is W[1]-complete

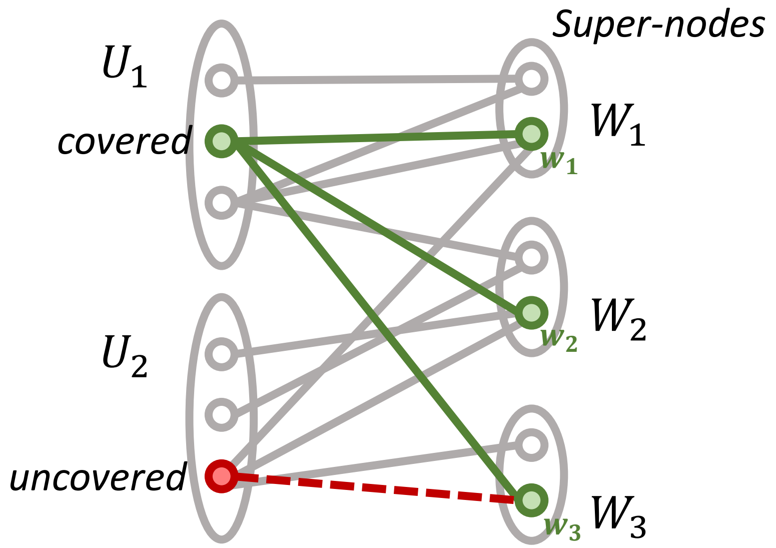
ETH $\Rightarrow$ no $f(k)n^{o(k)}$-time algo for k-Sum

Max Cover

Input: A bipartite graph $(U_1 \cup \cdots \cup U_h, W_1 \cup \cdots \cup W_k, E)$

Parameter: k

Output: $w_1 \in W_1, \dots, w_k \in W_k$ *that "covers" maximum number of vertices in U*



Proof Sketch: *Total Inapproximability of k-Dom Set*

$\boldsymbol{k}$-Sum Instance $(\boldsymbol{A_1}, \dots, \boldsymbol{A_k})$

Size: n

Parameter: k

YES: $\exists a_1 \in A_1, \dots, a_k \in A_k, a_1 + \cdots + a_k = 0$

NO: $\forall a_1 \in A_1, \dots, a_k \in A_k, a_1 + \cdots + a_k \neq 0$

ETH Lower Bound: $n^{\Omega(k)}$

W[1]-hard

$\longrightarrow$

Max Cover Instance $\boldsymbol{G}$

Size: $N = n^{1+o(1)}$

Parameter: k

YES: $MaxCover(\mathrm{G}) = h$

NO: $MaxCover(\mathrm{G}) \leq h/r$

ETH Lower Bound: $N^{\Omega(k)}$

W[1]-hard

Proof Sketch: *Total Inapproximability of k-Dom Set*

$\boldsymbol{k}$-Sum Instance $(\boldsymbol{A_1, \dots, A_k})$

Size: n

Parameter: k

YES: $\exists a_1 \in A_1, \dots, a_k \in A_k, a_1 + \cdots + a_k = 0$

NO: $\forall a_1 \in A_1, \dots, a_k \in A_k, a_1 + \cdots + a_k \neq 0$

$\rightarrow$

Max Cover Instance $\boldsymbol{G}$

Size: $N = n^{1+o(1)}$

Parameter: k

YES: $MaxCover(\mathrm{G}) = h$

NO: $MaxCover(\mathrm{G}) \leq h/r$

ETH Lower Bound: $n^{\Omega(k)}$

W[1]-hard

ETH Lower Bound: $N^{\Omega(k)}$

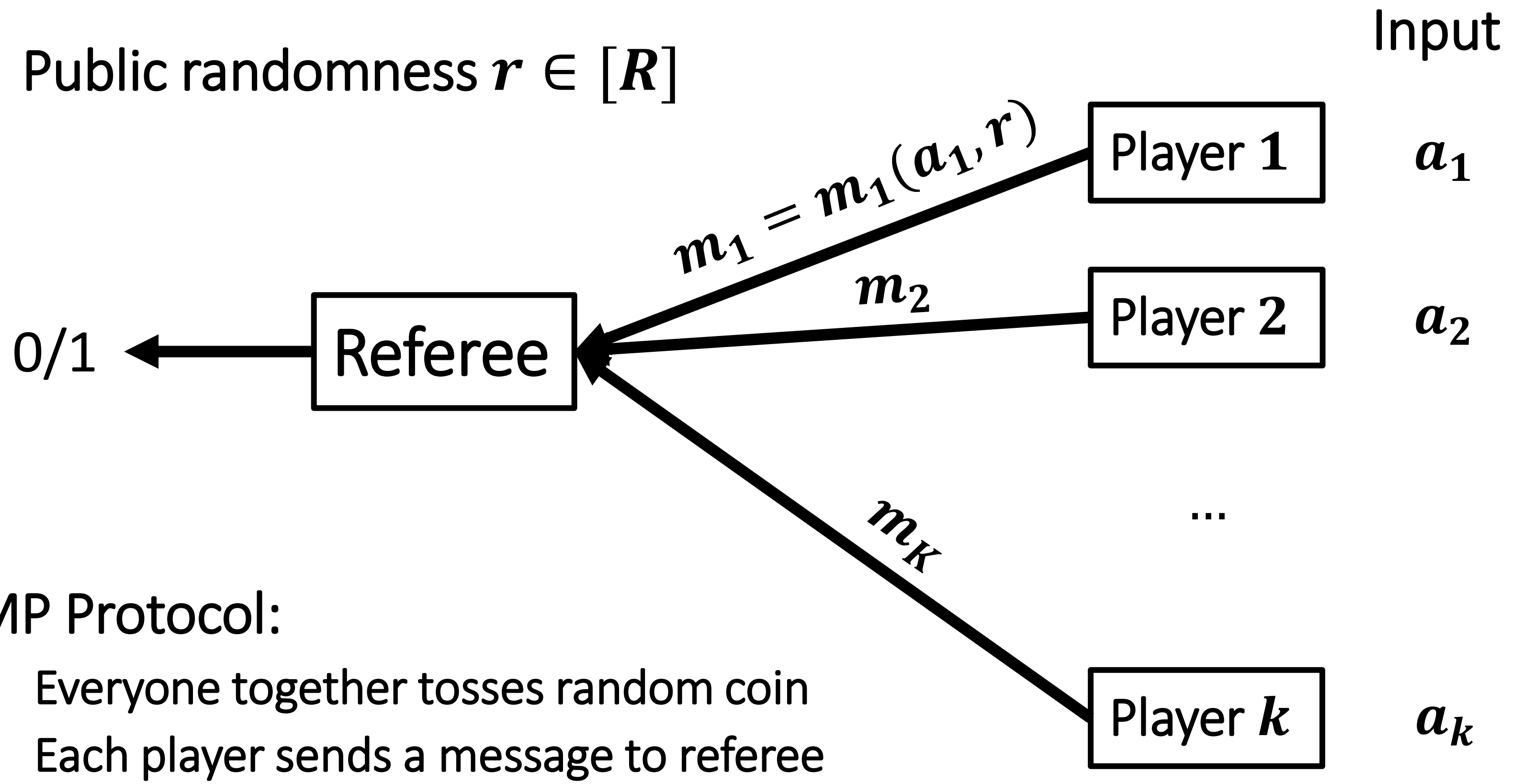
W[1]-hard

Key: **Communication Protocol ⇒ hardness of approximation**

"Distributed PCP" framework of *[Abboud-Rubinstein-Williams'18]*

Communication Model: Simultaneous Message Passing (SMP)

Public randomness $r \in [R]$



Goal: Compute $f(a_1, \dots, a_k) \in \{0, 1\}$

Guarantee

Completeness:

If $f(a_1, \dots, a_k) = 1$, then always output 1.

Soundness:

If $f(a_1, \dots, a_k) = 0$, output 0 w.p. 1/2.

SMP Protocol:

1. Everyone together tosses random coin
2. Each player sends a message to referee
3. Referee output 0 or 1

Zero-Sum Communication Problem

Public randomness $r \in [R]$

Input

$m_1 = m_1(a_1, r)$

Player 1 — a_1

m_2

Player 2 — a_2

...

m_k

Player k — a_k

$\in [-M, M]$

0/1 ← Referee

Goal: Compute

$\mathbf{1}[a_1 + \cdots + a_k = 0]$

<u>Guarantee</u>

Completeness:

If $f(a_1, \ldots, a_k) = 1$, then always output 1.

Soundness:

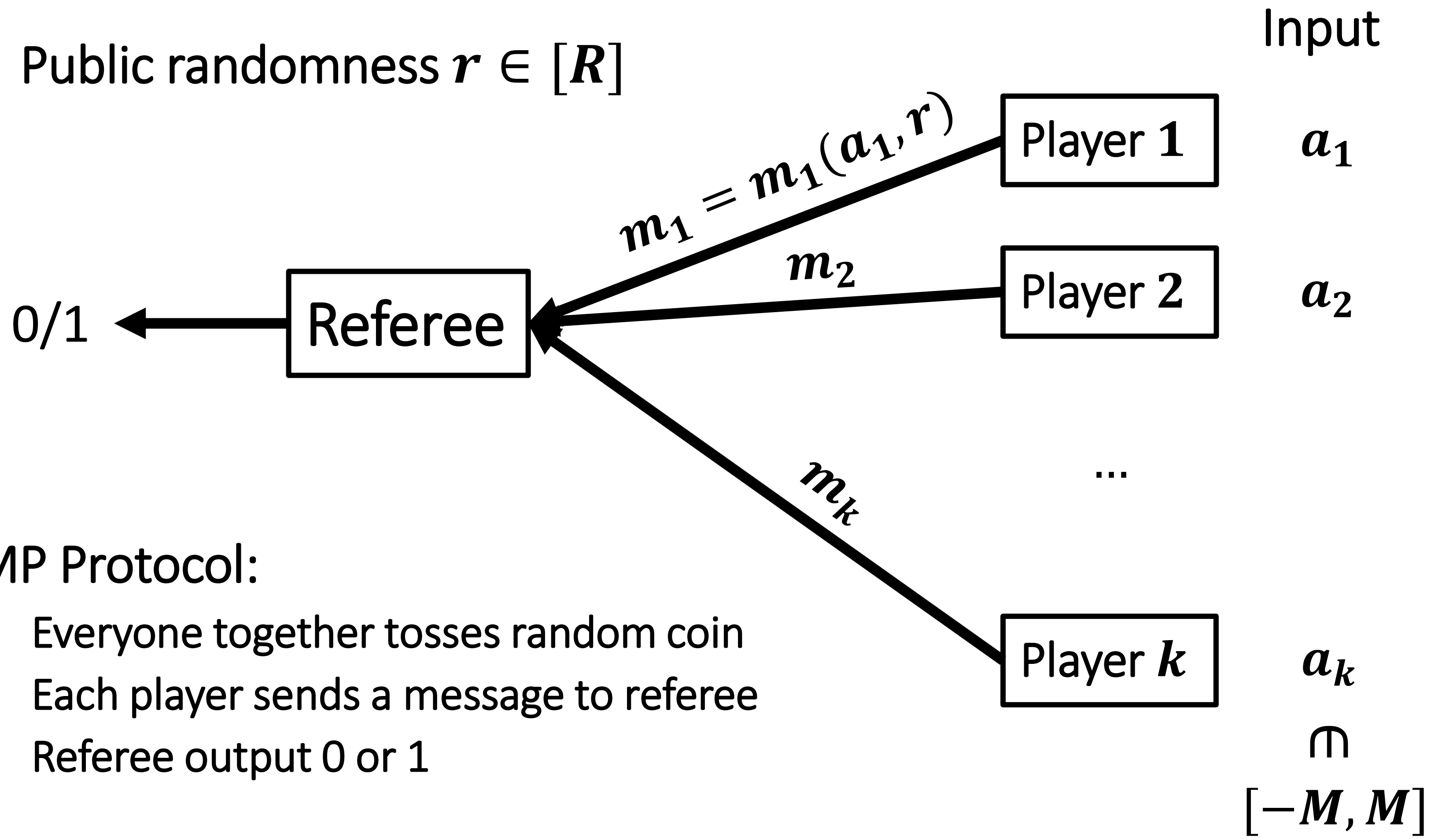
If $f(a_1, \ldots, a_k) = 0$, output 0 w.p. 1/2.

SMP Protocol:

1. Everyone together tosses random coin
2. Each player sends a message to referee
3. Referee output 0 or 1

Zero-Sum Communication Problem

Public randomness $r \in [R]$



Goal: Compute

$\mathbf{1}[a_1 + \cdots + a_k = 0]$

Guarantee

Completeness:

If $a_1 + \cdots + a_k = 0$, then always output 1.

Soundness:

If $a_1 + \cdots + a_k \neq 0$, output 0 w.p. 1/2.

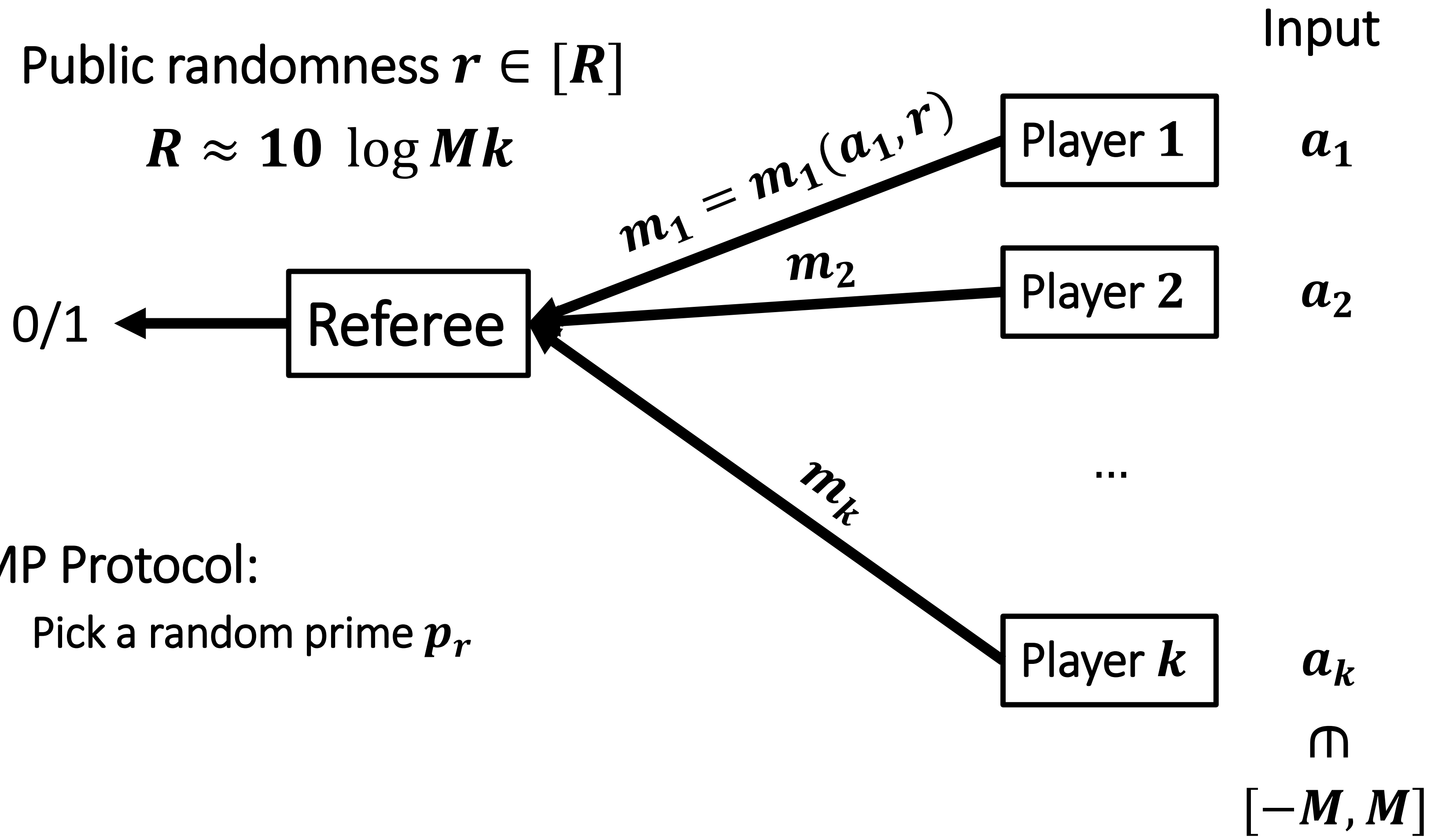
SMP Protocol:

1. Everyone together tosses random coin
2. Each player sends a message to referee
3. Referee output 0 or 1

Zero-Sum Communication Protocol

Public randomness $r \in [R]$

$R \approx 10\ \log Mk$



Goal: Compute $\mathbf{1}[a_1 + \cdots + a_k = 0]$

Guarantee

Completeness:

If $a_1 + \cdots + a_k = 0$, then always output 1.

Soundness:

If $a_1 + \cdots + a_k \neq 0$, output 0 w.p. 1/2.

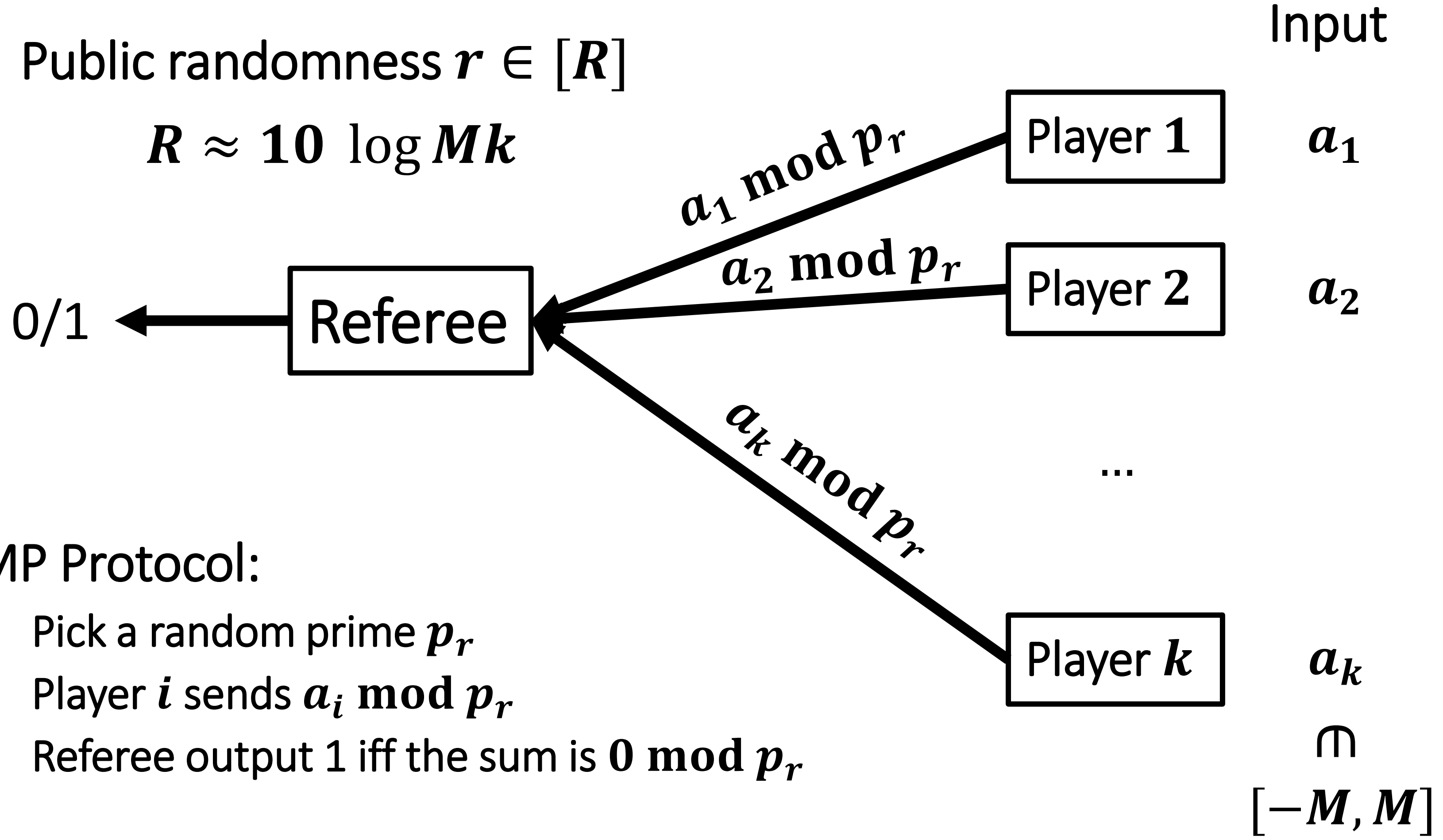
SMP Protocol:

1. Pick a random prime p_r

Zero-Sum Communication Protocol

Public randomness $r \in [R]$

$R \approx 10 \log Mk$



Goal: Compute
$\mathbf{1}[a_1 + \cdots + a_k = 0]$

Guarantee

Completeness:
If $a_1 + \cdots + a_k = 0$,
then always output 1.

Soundness:
If $a_1 + \cdots + a_k \neq 0$,
output 0 w.p. 1/2.

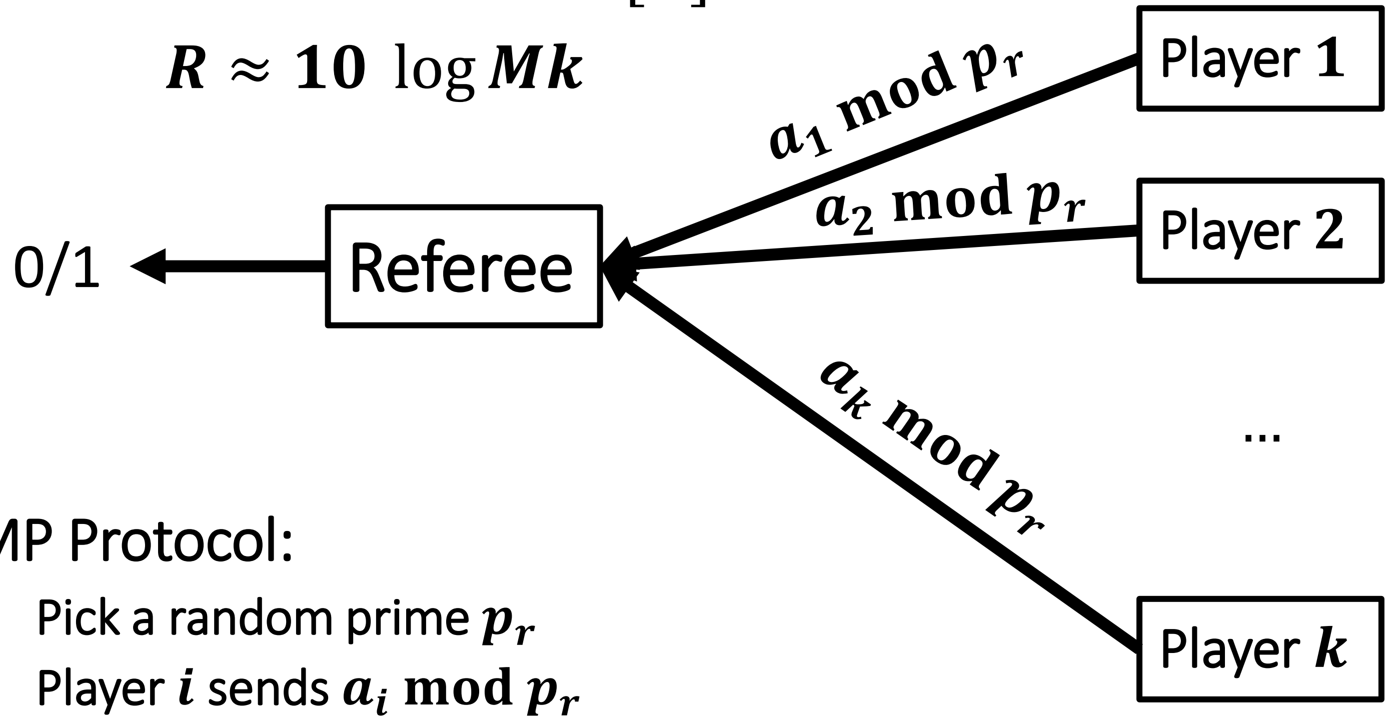
SMP Protocol:

1. Pick a random prime p_r
2. Player i sends $a_i \bmod p_r$
3. Referee output 1 iff the sum is $0 \bmod p_r$

Communication Protocol ≡ k-Sum ⇒Max Cover

Public randomness $r \in [R]$

$R \approx 10 \log Mk$



Input

a_1

a_2

a_k

$\in$

$[-M, M]$

Goal: Compute
$\mathbf{1}[a_1 + \cdots + a_k = 0]$

Guarantee

Completeness:

If $a_1 + \cdots + a_k = 0$,
then always output 1.

Soundness:

If $a_1 + \cdots + a_k \neq 0$,
output 0 w.p. 1/2.

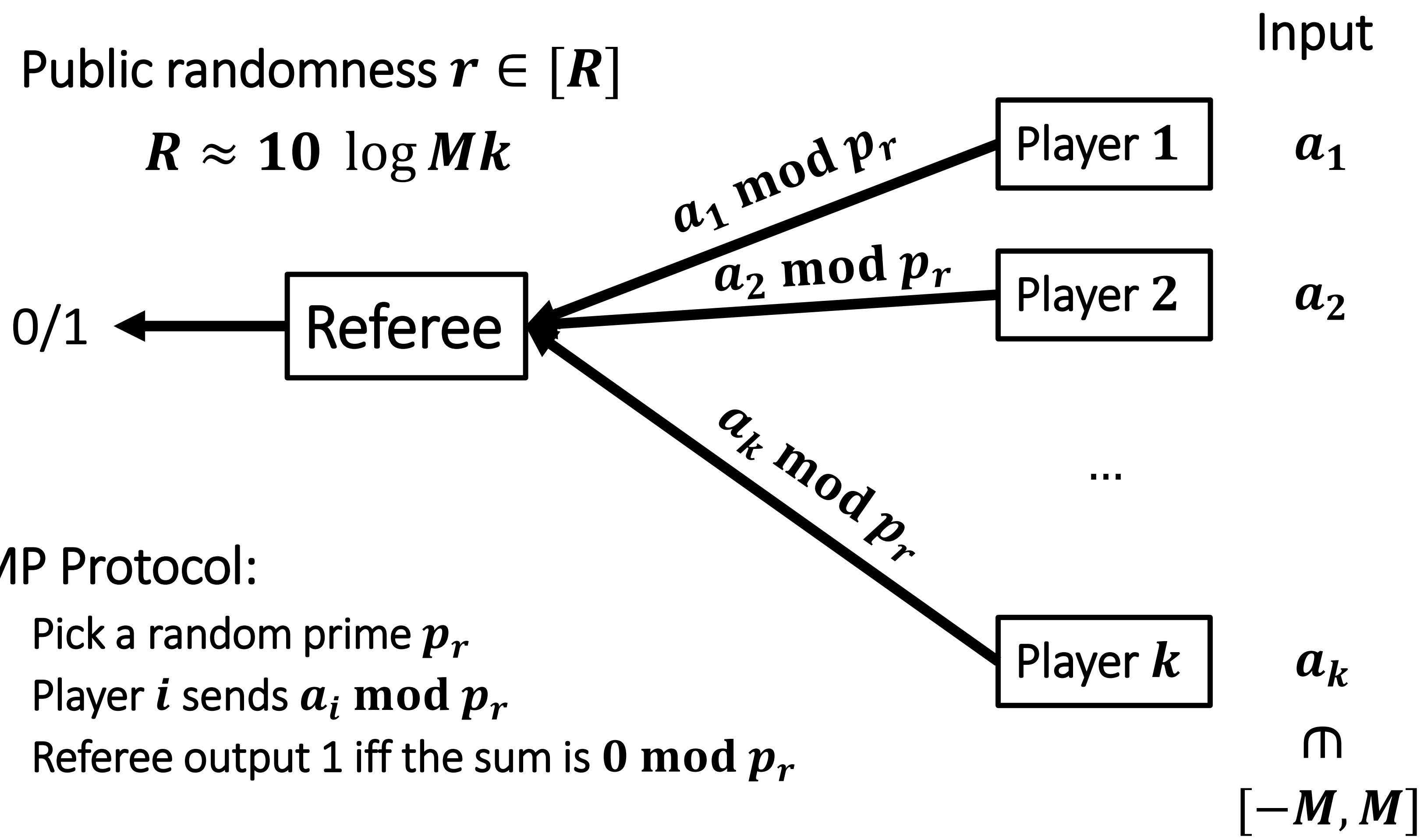
SMP Protocol:

1. Pick a random prime p_r
2. Player i sends $a_i \bmod p_r$
3. Referee output 1 iff the sum is $0 \bmod p_r$

Communication Protocol $\equiv$ k-Sum $\Rightarrow$Max Cover

Public randomness $\boldsymbol{r} \in [\boldsymbol{R}]$

$\boldsymbol{R} \approx \mathbf{10}\ \log \boldsymbol{Mk}$



<u>$\boldsymbol{k}$-Sum Problem</u>

Given $\boldsymbol{A_1}, \dots, \boldsymbol{A_k} \in [-\boldsymbol{M}, \boldsymbol{M}]$, determine whether there exist $\boldsymbol{a_1} \in \boldsymbol{A_1}, \dots, \boldsymbol{a_k} \in \boldsymbol{A_k}$ such that

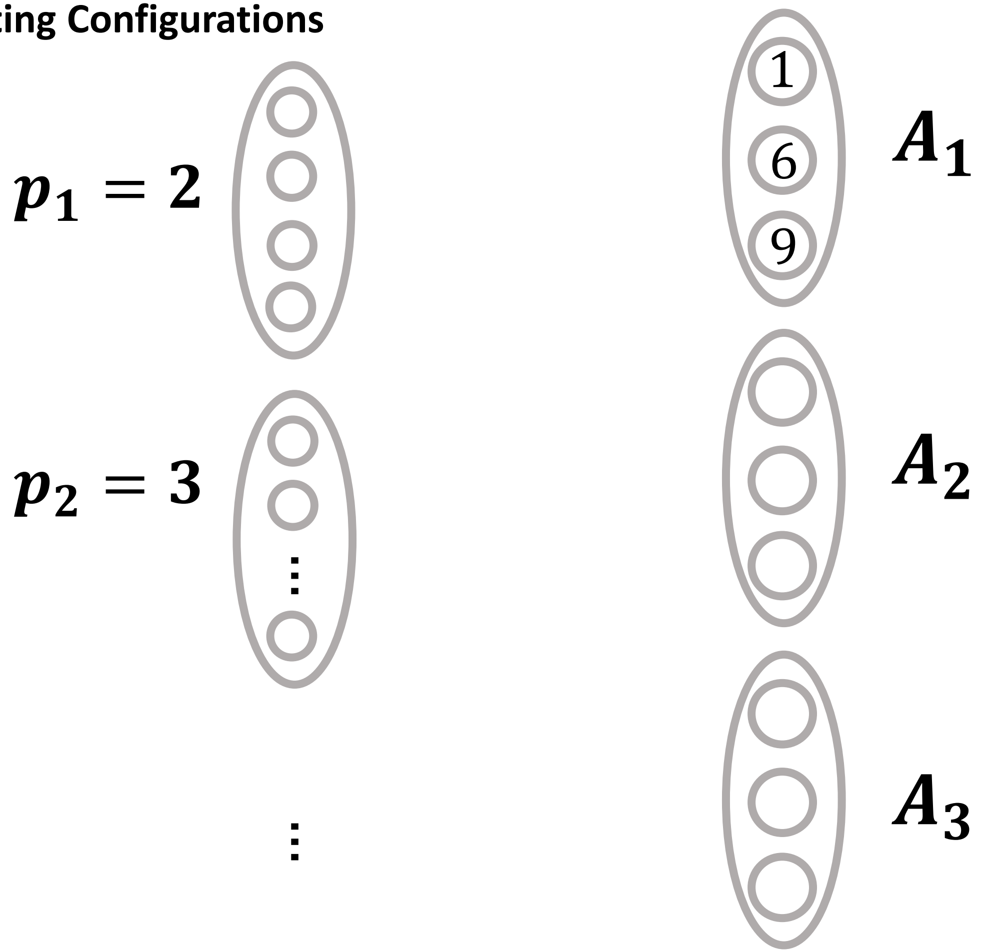
$$\boldsymbol{a_1} + \cdots + \boldsymbol{a_k} = \mathbf{0}$$

SMP Protocol:

1. Pick a random prime $\boldsymbol{p_r}$
2. Player $\boldsymbol{i}$ sends $\boldsymbol{a_i}$ **mod** $\boldsymbol{p_r}$
3. Referee output 1 iff the sum is $\mathbf{0}$ **mod** $\boldsymbol{p_r}$

Communication Protocol ≡ k-Sum ⇒Max Cover

Nodes ≡ Accepting Configurations

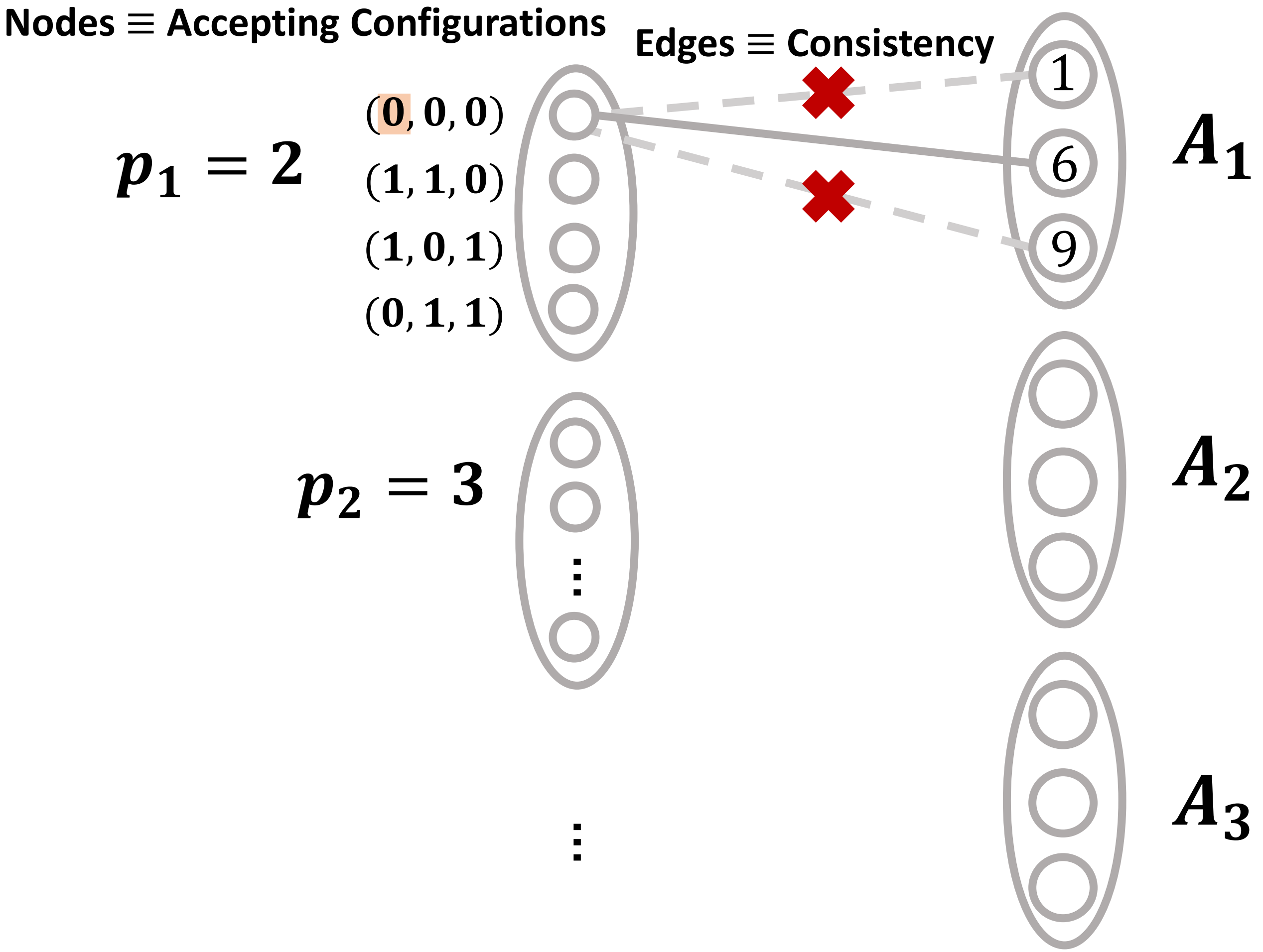


<u>$\boldsymbol{k}$-Sum Problem</u>

Given $\boldsymbol{A_1, \dots, A_k} \in [-\boldsymbol{M}, \boldsymbol{M}]$, determine whether there exist $\boldsymbol{a_1} \in \boldsymbol{A_1}, \dots, \boldsymbol{a_k} \in \boldsymbol{A_k}$ such that

$$\boldsymbol{a_1} + \cdots + \boldsymbol{a_k} = \boldsymbol{0}$$

Communication Protocol ≡ k-Sum ⇒ Max Cover

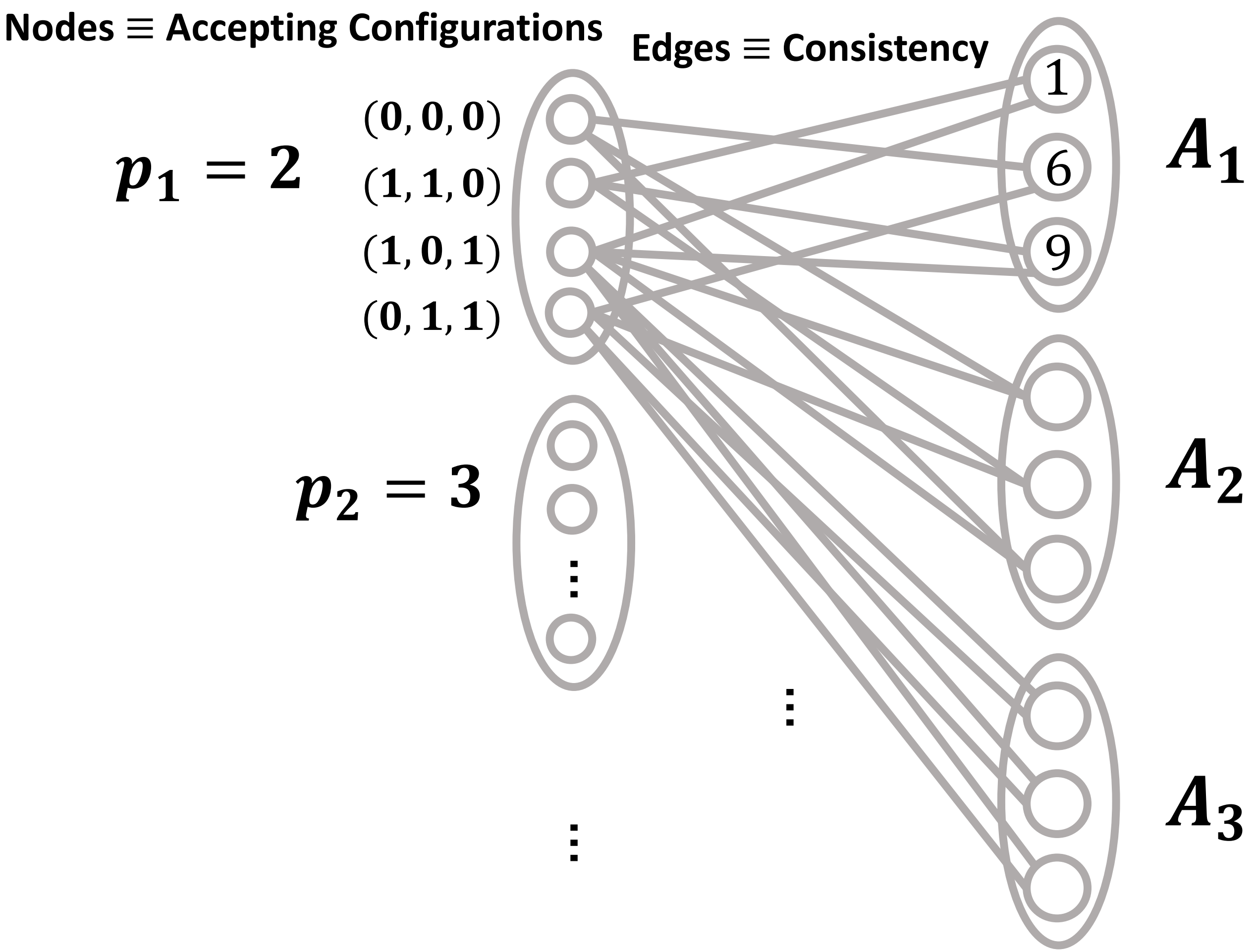


<u>k-Sum Problem</u>

Given $A_1, \dots, A_k \in [-M, M]$, determine whether there exist $a_1 \in A_1, \dots, a_k \in A_k$ such that

$$a_1 + \cdots + a_k = 0$$

Communication Protocol ≡ k-Sum ⇒Max Cover

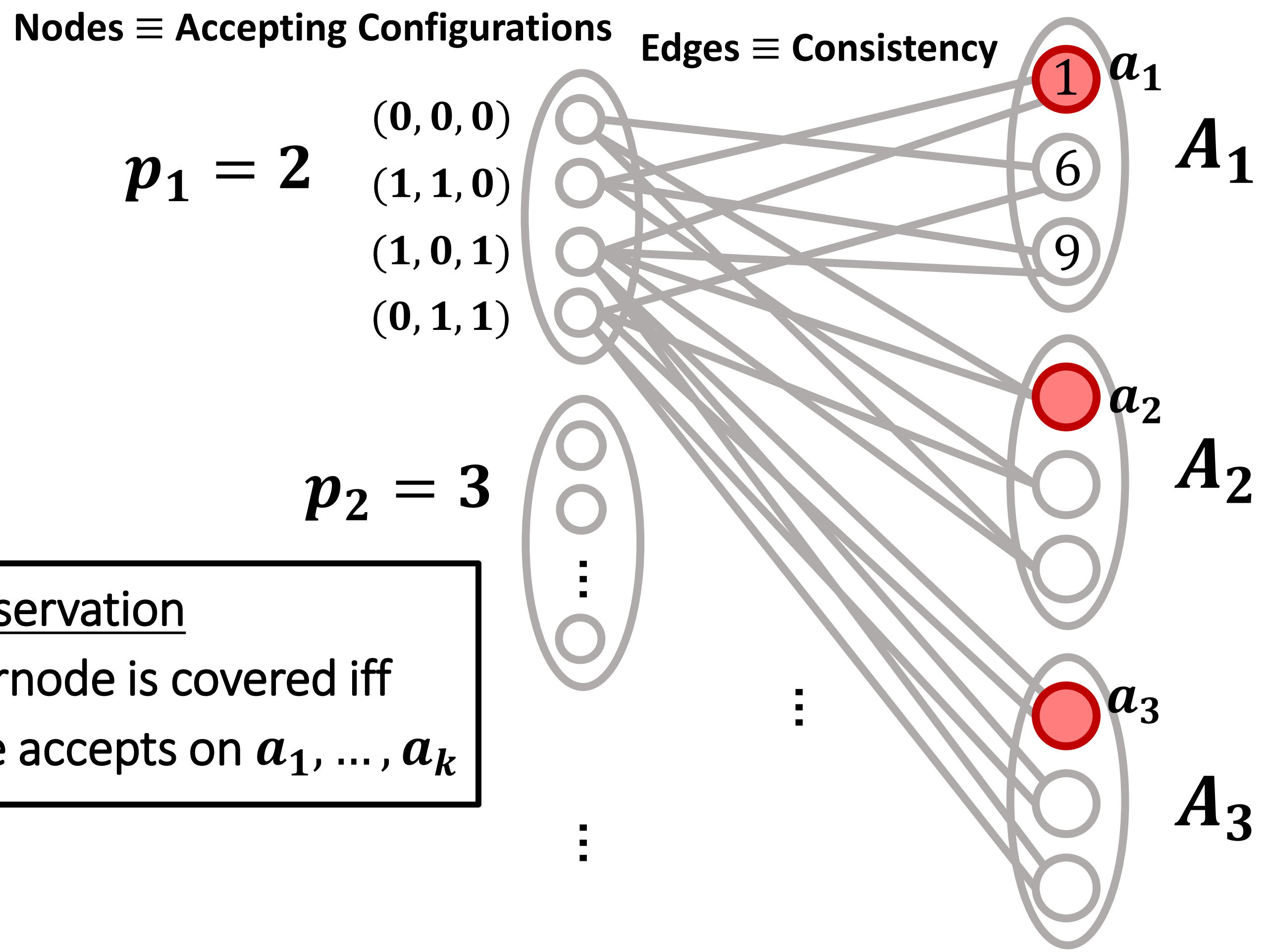


$\boldsymbol{k}$-Sum Problem

Given $A_1, \ldots, A_k \in [-M, M]$, determine whether there exist $a_1 \in A_1, \ldots, a_k \in A_k$ such that

$$a_1 + \cdots + a_k = 0$$

Communication Protocol $\equiv k$-Sum $\Rightarrow$ Max Cover



Key Observation

A Supernode is covered iff
Referee accepts on $a_1, \ldots, a_k$

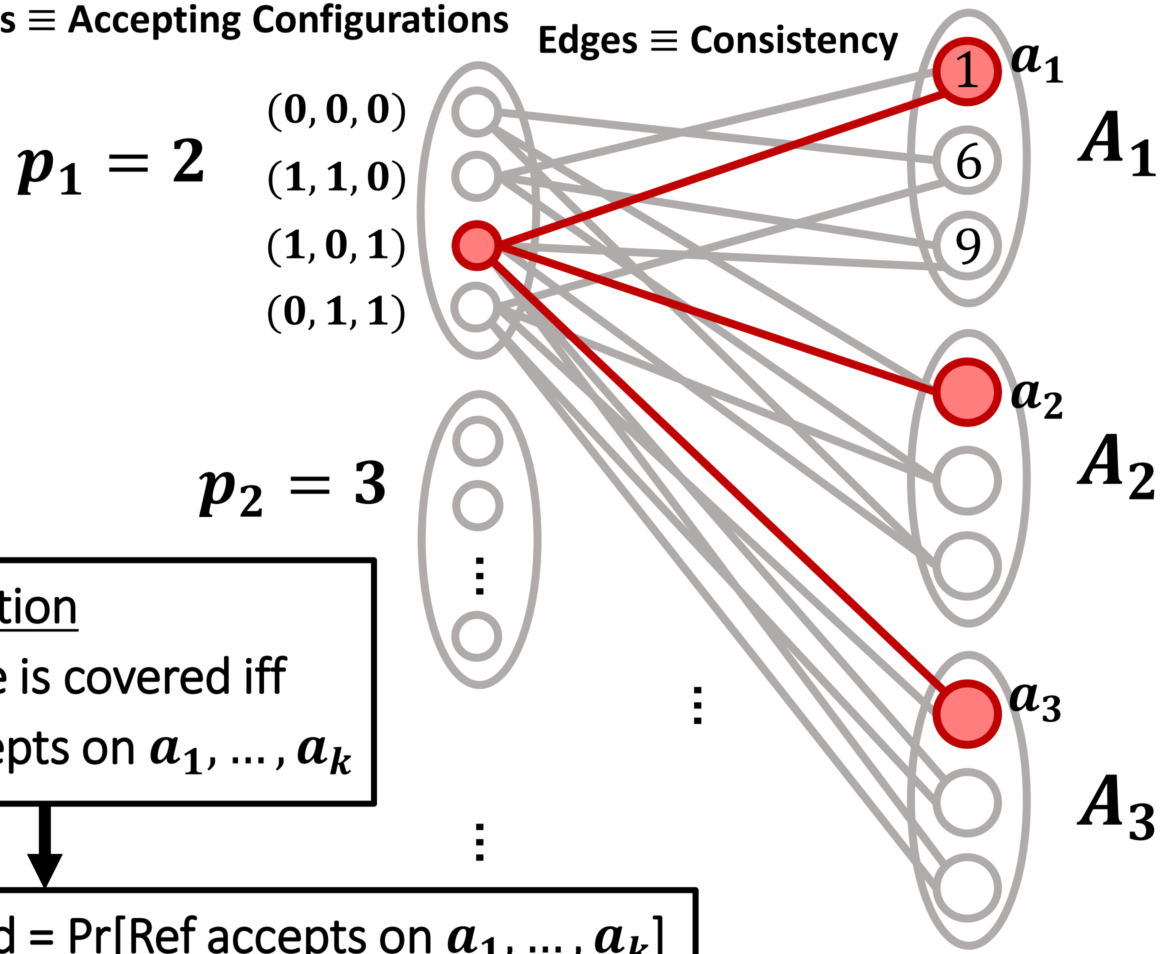
k-Sum Problem

Given $A_1, \ldots, A_k \in [-M, M]$, determine whether there exist $a_1 \in A_1, \ldots, a_k \in A_k$ such that

$$a_1 + \cdots + a_k = 0$$

Communication Protocol $\equiv k$-Sum $\Rightarrow$ Max Cover

Nodes $\equiv$ Accepting Configurations **Edges $\equiv$ Consistency**



<u>Key Observation</u>
A Supernode is covered iff Referee accepts on $a_1, \ldots, a_k$

Fraction covered = Pr[Ref accepts on $a_1, \ldots, a_k$]

<u>k-Sum Problem</u>
Given $A_1, \ldots, A_k \in [-M, M]$, determine whether there exist $a_1 \in A_1, \ldots, a_k \in A_k$ such that
$$a_1 + \cdots + a_k = 0$$

YES $\Rightarrow \mathbf{MaxCover}(G) = h$
NO $\Rightarrow \mathbf{MaxCover}(G) \leq h/2$

Remark: Size of G depends on parameters of the protocol

Recap

$\boldsymbol{k}$-Sum Instance $(\boldsymbol{A_1}, \dots, \boldsymbol{A_k})$

Size: $n = |A_1| + \cdots + |A_k|$

Parameter: k

YES: $\exists a_1 \in A_1, \dots, a_k \in A_k, a_1 + \cdots + a_k = 0$

NO: $\forall a_1 \in A_1, \dots, a_k \in A_k, a_1 + \cdots + a_k \neq 0$

SMP Communication Protocol for Zero-Sum

Max Cover Instance $\boldsymbol{G}$

Size: $N = n^{1+o(1)}$

Parameter: k

YES: $MaxCover(\mathrm{G}) = h$

NO: $MaxCover(\mathrm{G}) \leq h/2$

A General Framework

Product Space Problem $(\boldsymbol{A_1, \dots, A_k})$

Size: $n = |A_1| + \cdots + |A_k|$

Parameter: k

YES: $\exists a_1 \in A_1, \dots, a_k \in A_k, f(a_1, \cdots, a_k) = 1$

NO: $\forall a_1 \in A_1, \dots, a_k \in A_k, f(a_1, \cdots, a_k) = 0$

$\Big\downarrow$ SMP Communication Protocol for f

Max Cover Instance $\boldsymbol{G}$

Size: $N = n^{1+o(1)}$

Parameter: k

YES: $MaxCover(\mathrm{G}) = h$

NO: $MaxCover(\mathrm{G}) \leq h/2$

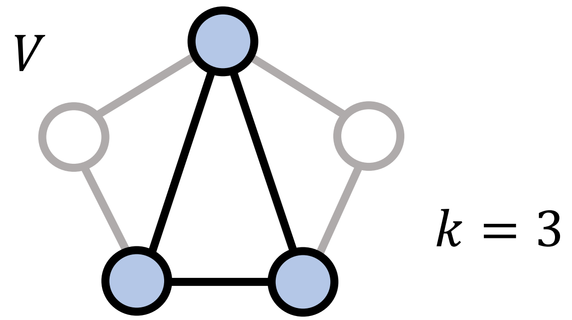
Clique as Product Space Problem

k-Clique

Input: A graph $G = (V, E)$, integer k

Parameter: k

Output: A subset $S \subseteq V$ of size k that induces a clique



Product Space Problem

Input: $\boldsymbol{A_1} = \cdots = \boldsymbol{A}_{\binom{k}{2}} = \vec{\boldsymbol{E}}$

Parameter: $\binom{k}{2}$

Predicate:

f "checks that the edges form a clique"

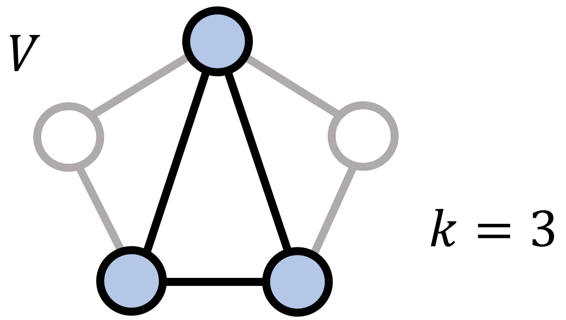
Clique as Product Space Problem

k-Clique

Input: A graph $G = (V, E)$, integer k

Parameter: k

Output: A subset $S \subseteq V$ of size k that induces a clique



Product Space Problem

Input: $\boldsymbol{A}_{(\mathbf{1},\mathbf{2})} = \cdots = \boldsymbol{A}_{(\boldsymbol{k}-\mathbf{1},\boldsymbol{k})} = \vec{\boldsymbol{E}}$

Parameter: $\binom{k}{2}$

Predicate:

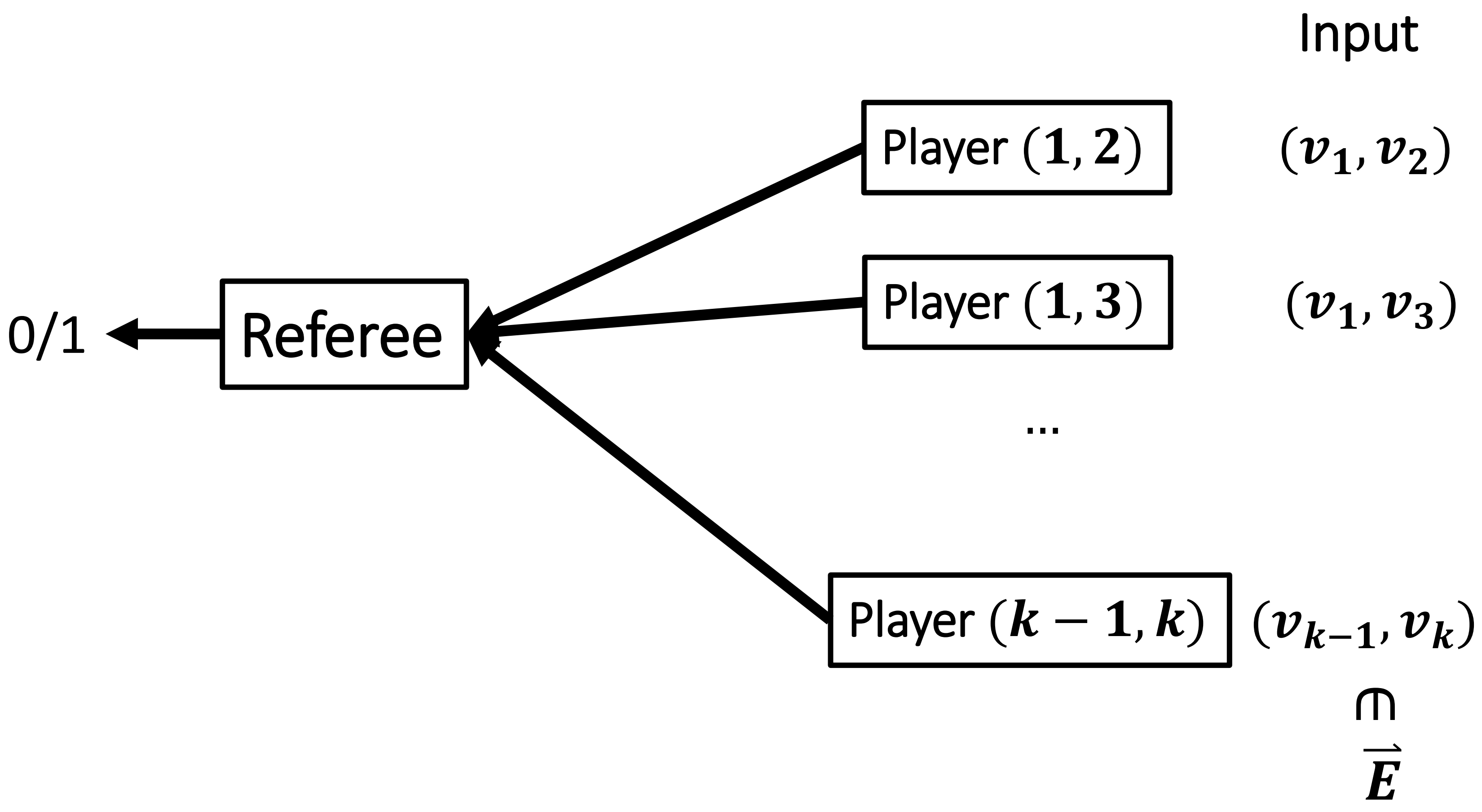
f "checks that the edges form a clique"

More formally:

$$f\big((v_1, v_2), (v_1, v_3), \dots, (v_{k-1}, v_k)\big) = 1$$

iff, for all $i \in [k]$, v_i's are all equal

Consistent Clique Communication Problem



Informal Goal:
"Check that the edges form a valid clique"

Goal: Output 1 iff for all $i \in [k]$, v_i received by different players are the same

Consistent Clique Communication Problem

Public randomness $\boldsymbol{r} \in [\ell]$

Input

Player $(\mathbf{1},\mathbf{2})$	$(\boldsymbol{v_1}, \boldsymbol{v_2})$
Player $(\mathbf{1},\mathbf{3})$	$(\boldsymbol{v_1}, \boldsymbol{v_3})$
…	
Player $(\boldsymbol{k}-\mathbf{1},\boldsymbol{k})$	$(\boldsymbol{v_{k-1}}, \boldsymbol{v_k})$

$\in$

$\overrightarrow{\boldsymbol{E}}$

$\boldsymbol{\phi}(\boldsymbol{v_1})_r, \boldsymbol{\phi}(\boldsymbol{v_2})_r$

$\boldsymbol{\phi}(\boldsymbol{v_1})_r, \boldsymbol{\phi}(\boldsymbol{v_3})_r$

$\boldsymbol{\phi}(\boldsymbol{v_k})_r, \boldsymbol{\phi}(\boldsymbol{v_{k-1}})_r$

0/1 ← Referee

Informal Goal:
"Check that the edges form a valid clique"

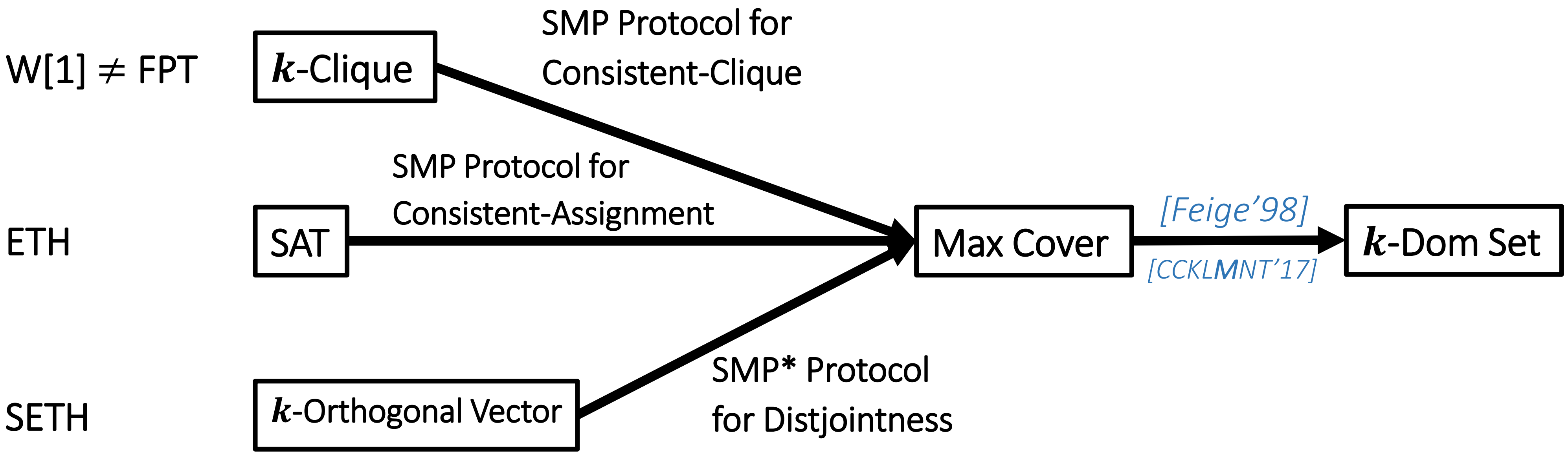
Goal: Output 1 iff for all $i \in [k]$, v_i received by different players are the same

Error Correcting Codes
$\phi: [n] \to \{0,1\}^\ell$ such that $\Delta(\phi(x), \phi(y)) > 0.1 \cdot \ell$ for all $x \neq y$.

SMP Protocol:

1. Selects random position $\boldsymbol{r} \in [\ell]$
2. Player $(\boldsymbol{i}, \boldsymbol{j})$ sends $(\boldsymbol{\phi}(\boldsymbol{v_i})_r, \boldsymbol{\phi}(\boldsymbol{v_j})_r)$ to referee
3. Referee checks that $\boldsymbol{\phi}(\boldsymbol{v_i})_r$ are all equal

A General Framework



[Lin'18] Alternative approach that doesn't use communication protocols…

Part III:

Repeated Gap Amplification

Dinur's Proof of PCP Theorem

PCP Theorem Clique is NP-hard to approximate to within 2 factor

Gap Amplification Step

Clique $\mathbf{G}$
Size: n
Gap: $1 + \varepsilon$

→

Clique $\mathbf{G}'$
Size: $100n$
Gap: $1 + 2\varepsilon$

Only works for non-parameterized (NP-hard) regime!

NP-hard

Clique $\mathbf{G_1}$
Size: n
Gap: $1 + 1/n$

$\xrightarrow{\text{Gap Amplification}}$

NP-hard

Clique $\mathbf{G_2}$
Size: $100n$
Gap: $1 + 2/n$

$\underbrace{\longrightarrow \cdots \longrightarrow}_{\log n \text{ times}}$

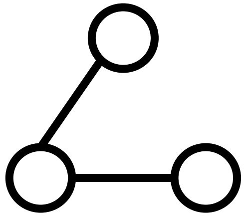
NP-hard

Clique $\mathbf{G_{\log n}}$
Size: $\mathrm{poly}(n)$
Gap: 2

Gap Amplification for k-Clique?

Parameterized PCP Theorem? Clique is W[1]-hard to approximate to within 2 factor

$G = (V, E)$

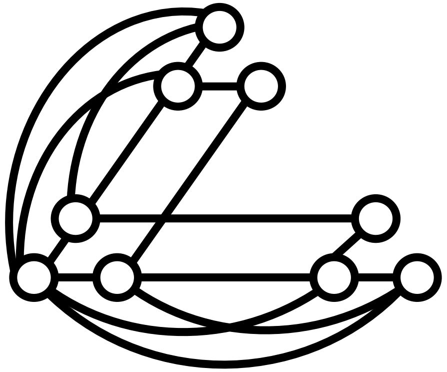


Tensor Product

$\boldsymbol{Clique(G^{\otimes 2})}$
$\boldsymbol{= Clique(G)^2}$

$G^{\otimes 2} = (V \times V, E^{\otimes 2})$

$(u_1, u_2) \sim (v_1, v_2)$ iff $u_1 \sim u_2$ and $v_1 \sim v_2$



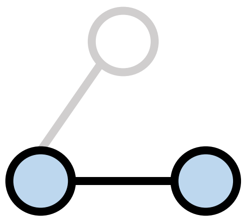
Gap Amplification for k-Clique?

<u>Parameterized PCP Theorem?</u> Clique is W[1]-hard to approximate to within 2 factor

Gap Amplification Step

$\boldsymbol{k}$-Clique $\mathbf{G}$	→	$\boldsymbol{k^2}$-Clique $\mathbf{G'}$
Size: n		Size: n^2
Gap: $1+\varepsilon$		Gap: $1+2\varepsilon$

$G = (V, E)$



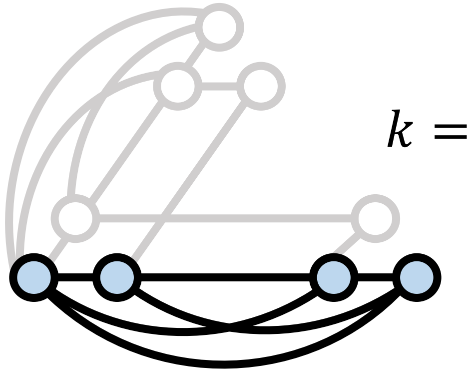
$k = 2$

Tensor Product

$\boldsymbol{Clique(G^{\otimes 2}) = Clique(G)^2}$

$G^{\otimes 2} = (V \times V, E^{\otimes 2})$

$(u_1, u_2) \sim (v_1, v_2)$ iff $u_1 \sim u_2$ and $v_1 \sim v_2$



$k = 4$

W[1]-hard		***W[1]-hard***			***W[1]-hard***
$\boldsymbol{k}$-Clique $\mathbf{G_1}$	Gap Amplification →	$\boldsymbol{k^2}$-Clique $\mathbf{G_2}$	$\log k$ times → … →		$\boldsymbol{k^k}$-Clique $\mathbf{G_{\log k}}$
Size: n		Size: n^2			Size: n^k
Gap: $1 + 1/k$		Gap: $1 + 2/k$			Gap: 2

Too large!
Doesn't give any lower bound

Gap Amplification: a success story

<u>Theorem</u> *[Wlodarczyk'19]* k-Steiner Orientation (k-SO) is W[1]-hard to approximate to within $(\log k)^{o(1)}$ factor

[Dinur'07]

Gap Amplification Step

Clique $\mathbf{G}$		Clique $\mathbf{G}'$
Size: n	$\rightarrow$	Size: $100n$
Gap: $1+\varepsilon$		Gap: $1+2\varepsilon$

[Wlodarczyk'19]

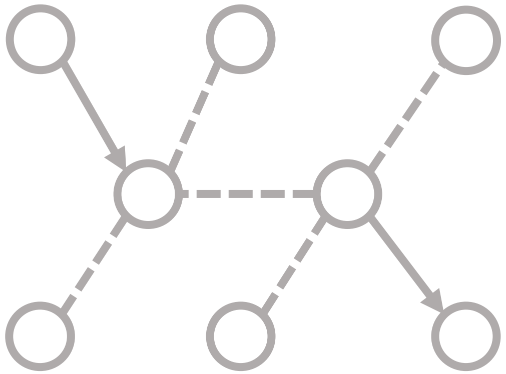
Gap Amplification Step

$\boldsymbol{k}$-SO $\boldsymbol{I}$		$\boldsymbol{h(k)}$-SO $\boldsymbol{I'}$
Size: n	$\rightarrow$	Size: $f(k)\cdot n$
Gap: $1+\varepsilon$		Gap: $1+2\varepsilon$

W[1]-hard		*W[1]-hard*	$\overbrace{\qquad\qquad}^{\log k \text{ times}}$	*W[1]-hard*
$\boldsymbol{k}$-SO $\boldsymbol{I_1}$	Gap Amplification	$\boldsymbol{h(k)}$-SO $\boldsymbol{I_2}$		$\boldsymbol{h(\dots(h(k))\dots)}$-SO $\mathbf{G}_{\log k}$
Size: n	$\longrightarrow$	Size: $f(k)\cdot n$	$\longrightarrow \cdots \longrightarrow$	Size: $f(h(\dots(h(k))\dots))\cdot\ldots\cdot f(k)\cdot n$
Gap: $1+1/k$		Gap: $1+2/k$		Gap: 2

k-Steiner Orientation (k-SO)

Input: A mixed graph $G = (V, E)$,
k terminal pairs $(s_1, t_1), \dots, (s_k, t_k)$

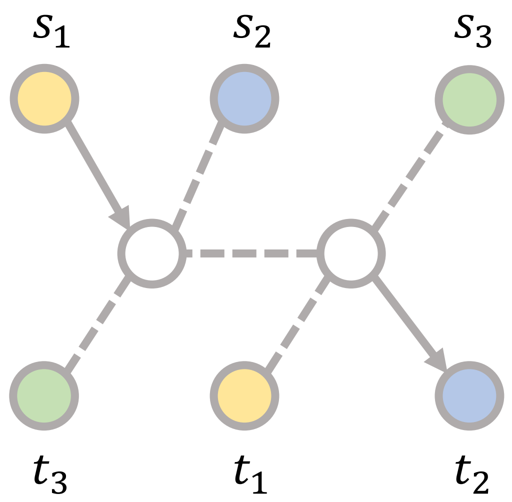


k-Steiner Orientation (k-SO)

Input: A mixed graph $G = (V, E)$,
k terminal pairs $(s_1, t_1), \dots, (s_k, t_k)$

Parameter: k

Output: Orientation of undirected edges that maximizes pairs $s_i \to t_i$

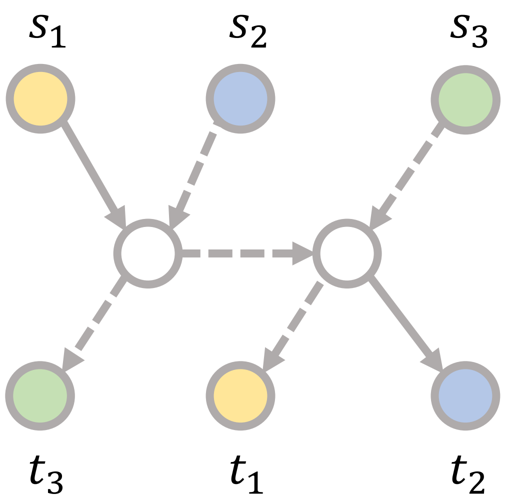


k-Steiner Orientation (k-SO)

Input: A mixed graph $G = (V, E)$,
k terminal pairs $(s_1, t_1), \dots, (s_k, t_k)$

Parameter: k

Output: Orientation of undirected edges that maximizes pairs $s_i \to t_i$

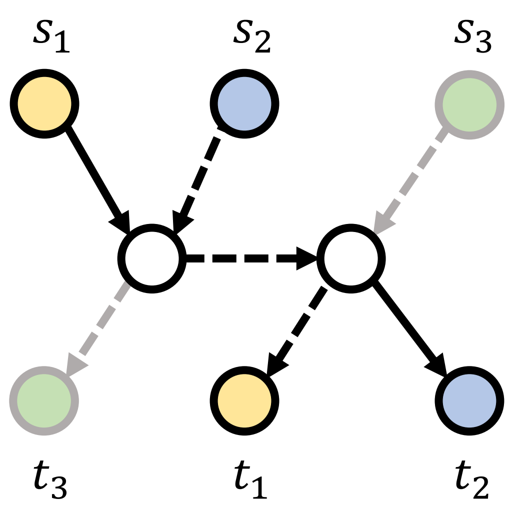


k-Steiner Orientation (k-SO)

Input: A mixed graph $G = (V, E)$,
k terminal pairs $(s_1, t_1), \dots, (s_k, t_k)$

Parameter: k

Output: Orientation of undirected edges that maximizes pairs $s_i \to t_i$



[Cygan-Kortsarz-Nutov'13]
k-Steiner Orientation is solvable in $n^{O(k)}$ time

[Pilipczuk-Wahlstrom'16]
k-Steiner Orientation is W[1]-hard

[Wlodarczyk'19]
k-Steiner Orientation is in W[1]

[Wlodarczyk'19]
k-Steiner Orientation is in W[1]-hard to approximate to within $(\log k)^{o(1)}$ factor

Wlodarczyk's Gap Amplification Step

[Wlodarczyk'19]

Gap Amplification Step

$\boldsymbol{k}$-SO $\boldsymbol{I}$	→	$\boldsymbol{h(k)}$-SO $\boldsymbol{I'}$
Size: n		Size: $f(k) \cdot n$
Gap: $1 + \varepsilon$		Gap: $1 + 2\varepsilon$

Wlodarczyk’s Gap Amplification Step

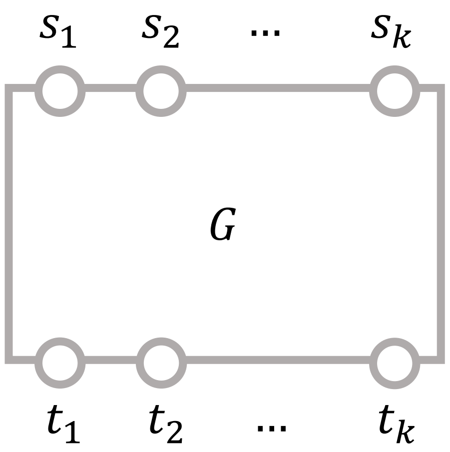
[Wlodarczyk’19]

Gap Amplification Step

$\boldsymbol{k}$-SO $\boldsymbol{I}$		$\boldsymbol{k^{k+1}}$-SO $\boldsymbol{I'}$
Size: n	$\rightarrow$	Size: $O(k^k \cdot n)$
Gap: $1 + 1/k$		Gap: $1 + 1.1/k$

SO Instance $\boldsymbol{I}$		SO Instance $\boldsymbol{I'}$
$\boldsymbol{G}, \{(\boldsymbol{s_i}, \boldsymbol{t_i})\}_{i \in [\boldsymbol{k}]}$		$\boldsymbol{G'}, \{(\boldsymbol{s'}_i, \boldsymbol{t'}_i)\}_{i \in [\boldsymbol{k'}]}$
Size: n	$\Rightarrow$	Size: $O(k^k \cdot n)$
Parameter: k		Parameter: $k' = k^{k+1}$
YES: $OPT(I) = k$		YES: $OPT(I') = k'$
NO: $OPT(I) \leq k - 1$		NO: $OPT(I') \leq k'(1 - 1.1/k)$

Wlodarczyk’s Gap Amplification Step



SO Instance $\boldsymbol{I}$

$\boldsymbol{G}, \{(\boldsymbol{s_i}, \boldsymbol{t_i})\}_{i \in [k]}$

Size: n

Parameter: k

YES: $OPT(I) = k$

NO: $OPT(I) \leq k - 1$

SO Instance $\boldsymbol{I'}$

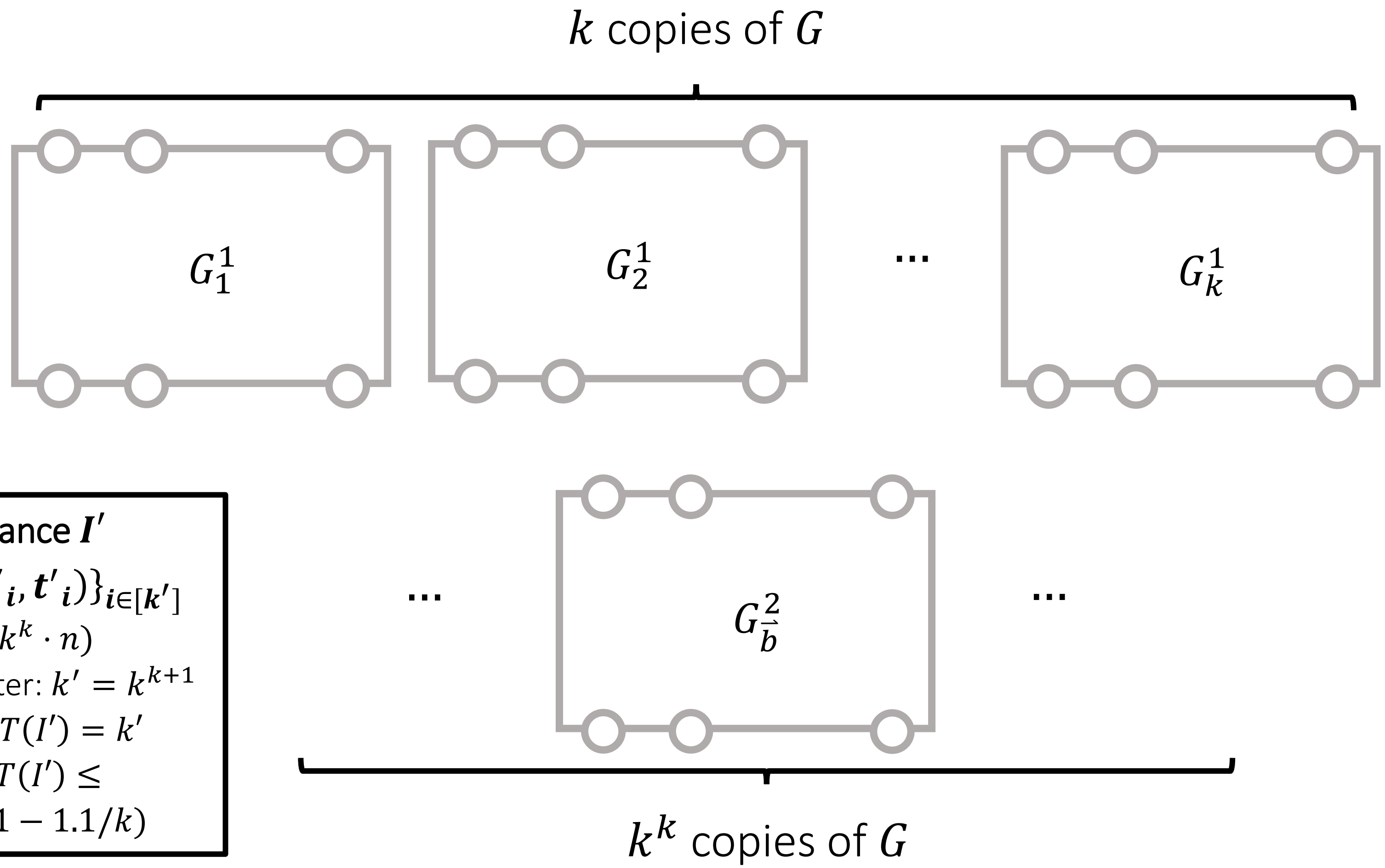
$\boldsymbol{G'}, \{(\boldsymbol{s'_i}, \boldsymbol{t'_i})\}_{i \in [k']}$

Size: $O(k^k \cdot n)$

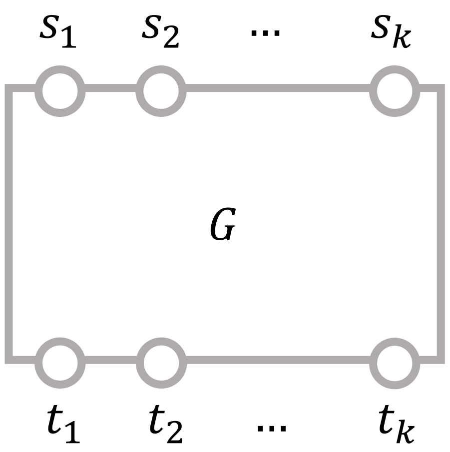
Parameter: $k' = k^{k+1}$

YES: $OPT(I') = k'$

NO: $OPT(I') \leq k'(1 - 1.1/k)$



Wlodarczyk's Gap Amplification Step



SO Instance $\boldsymbol{I}$

$\boldsymbol{G}, \{(\boldsymbol{s_i}, \boldsymbol{t_i})\}_{\boldsymbol{i \in [k]}}$

Size: n

Parameter: k

YES: $OPT(I) = k$

NO: $OPT(I) \leq k - 1$

SO Instance $\boldsymbol{I'}$

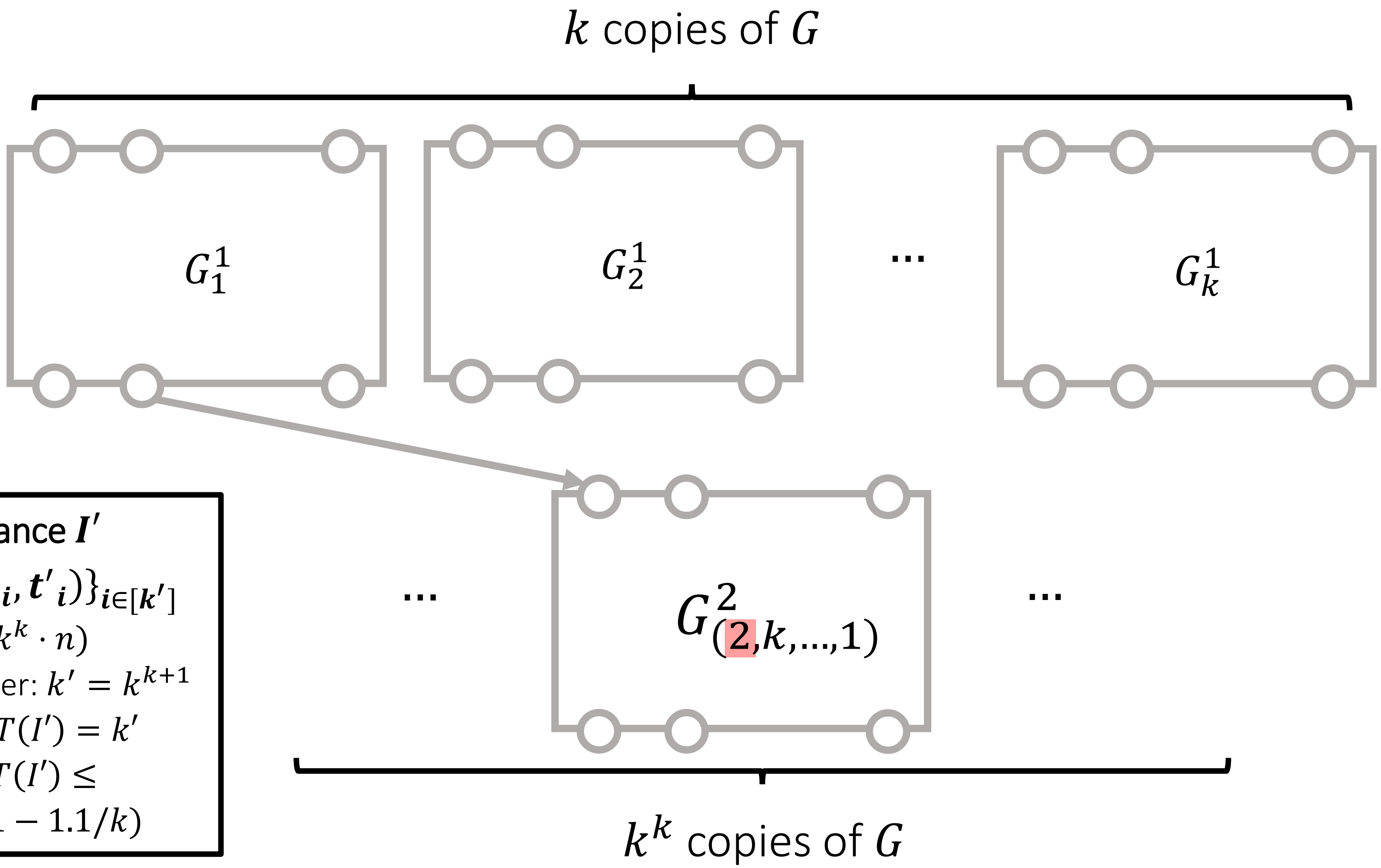
$\boldsymbol{G'}, \{(\boldsymbol{s'_i}, \boldsymbol{t'_i})\}_{\boldsymbol{i \in [k']}}$

Size: $O(k^k \cdot n)$

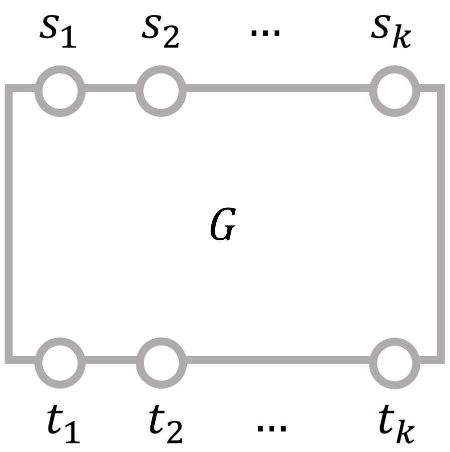
Parameter: $k' = k^{k+1}$

YES: $OPT(I') = k'$

NO: $OPT(I') \leq k'(1 - 1.1/k)$



Wlodarczyk's Gap Amplification Step



SO Instance I

$G, \{(s_i, t_i)\}_{i \in [k]}$

Size: n

Parameter: k

YES: $OPT(I) = k$

NO: $OPT(I) \le k - 1$

SO Instance I'

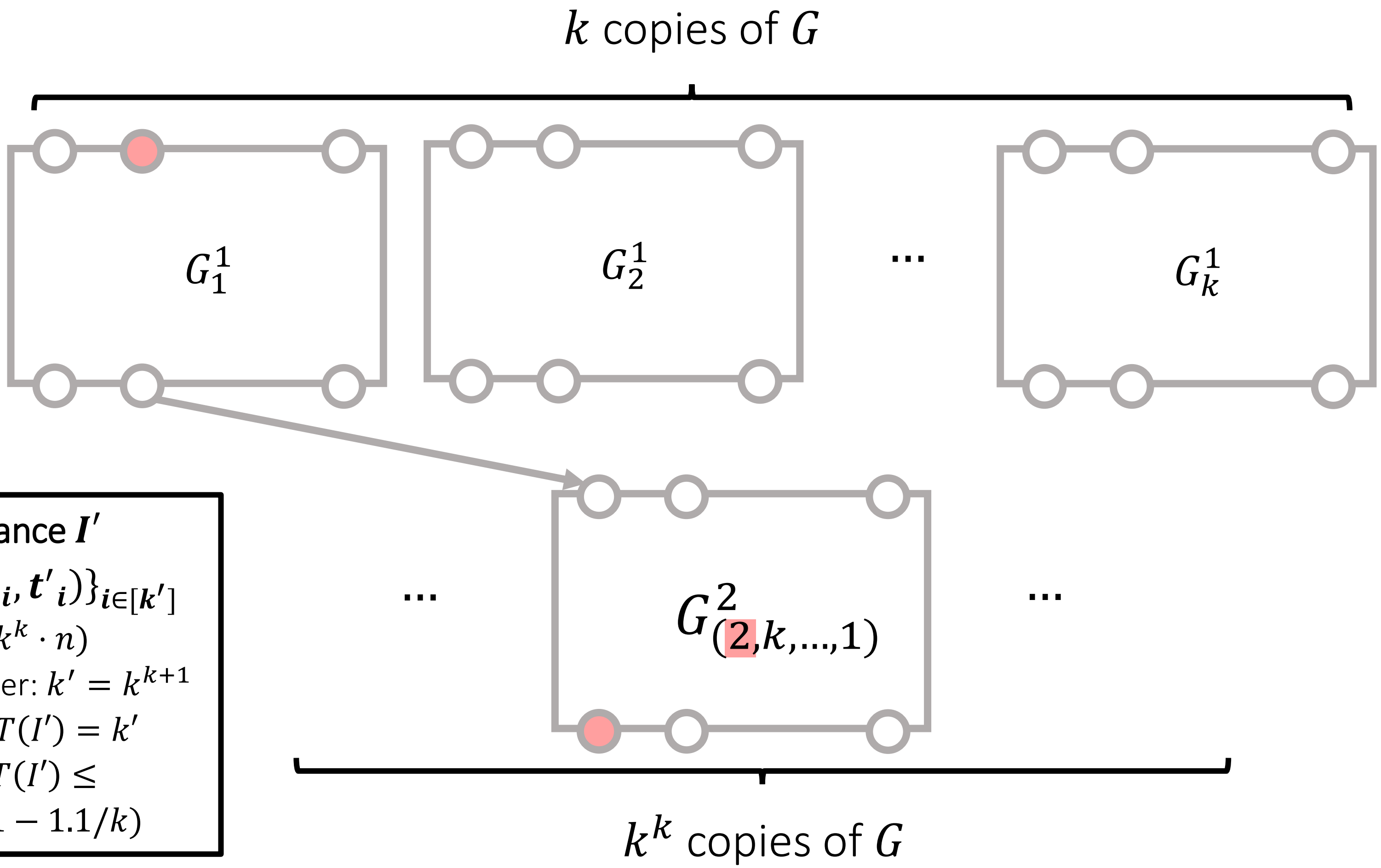
$G', \{(s'_i, t'_i)\}_{i \in [k']}$

Size: $O(k^k \cdot n)$

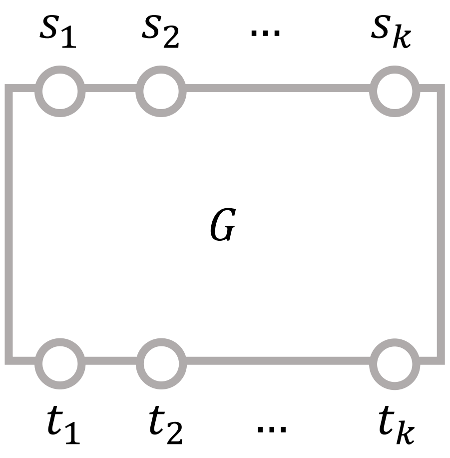
Parameter: $k' = k^{k+1}$

YES: $OPT(I') = k'$

NO: $OPT(I') \le k'(1 - 1.1/k)$



Wlodarczyk's Gap Amplification Step



SO Instance I

$G, \{(s_i, t_i)\}_{i \in [k]}$

Size: n

Parameter: k

YES: $OPT(I) = k$

NO: $OPT(I) \leq k - 1$

SO Instance I'

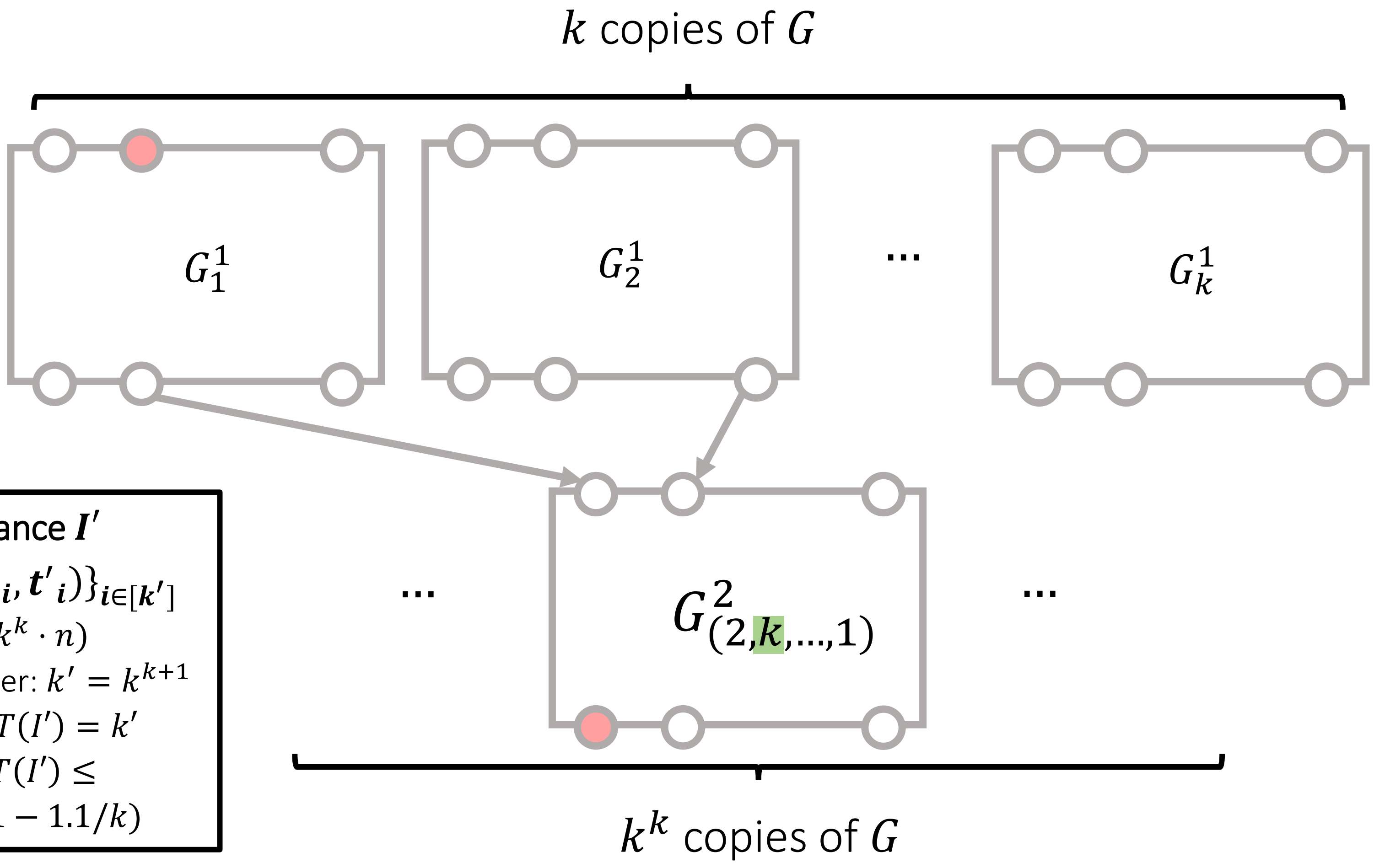
$G', \{(s'_i, t'_i)\}_{i \in [k']}$

Size: $O(k^k \cdot n)$

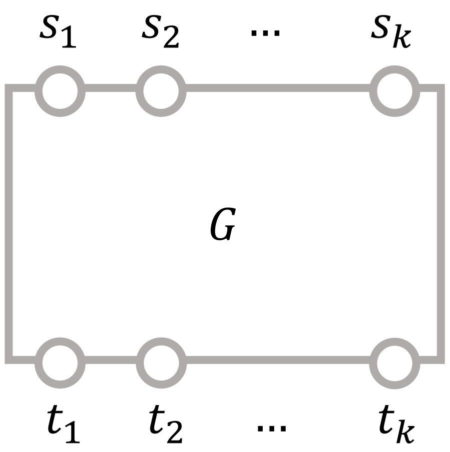
Parameter: $k' = k^{k+1}$

YES: $OPT(I') = k'$

NO: $OPT(I') \leq$
$k'(1 - 1.1/k)$



Wlodarczyk's Gap Amplification Step



SO Instance $\boldsymbol{I}$

$\boldsymbol{G}, \{(\boldsymbol{s_i}, \boldsymbol{t_i})\}_{i \in [\boldsymbol{k}]}$

Size: n

Parameter: k

YES: $OPT(I) = k$

NO: $OPT(I) \leq k - 1$

SO Instance $\boldsymbol{I'}$

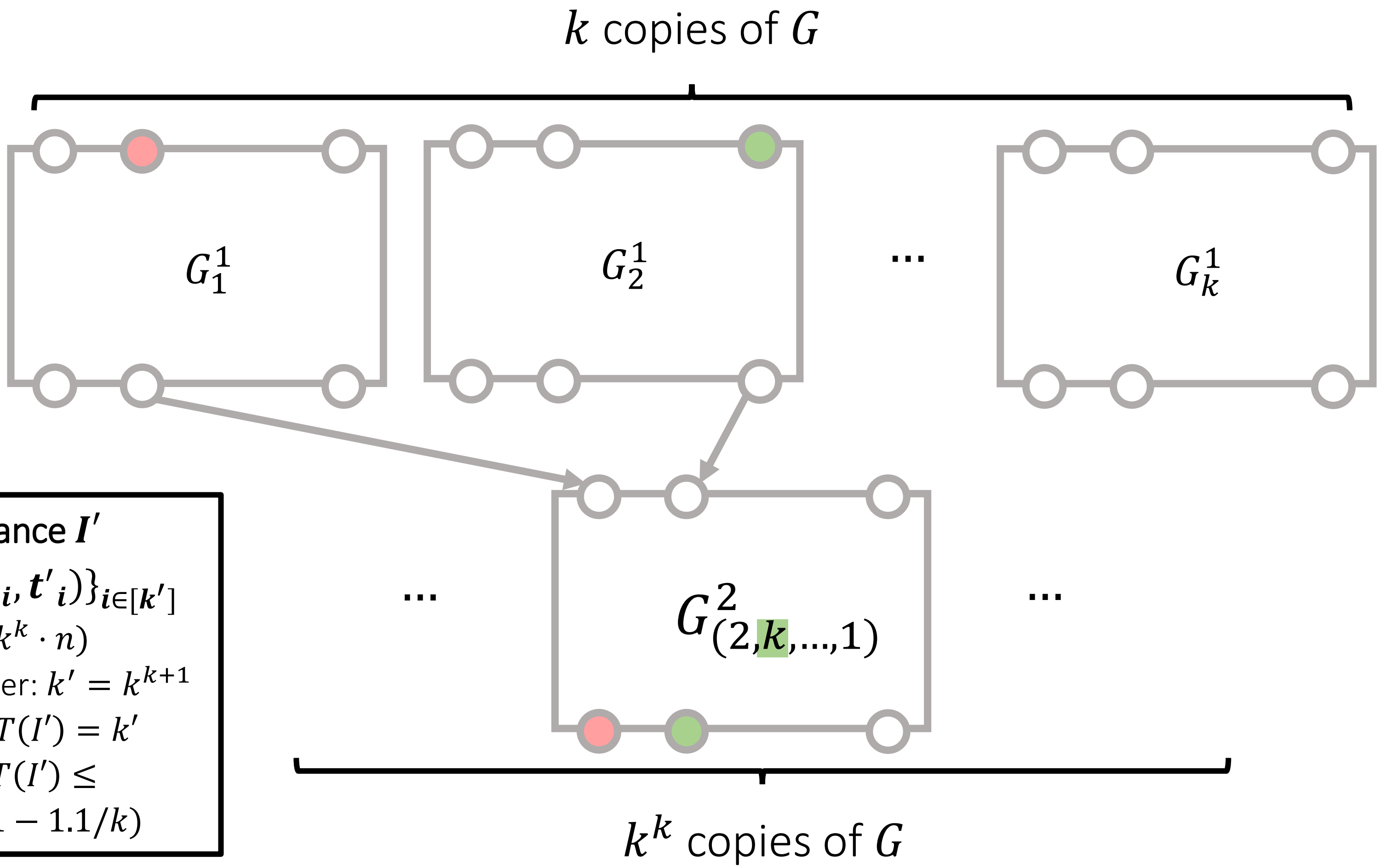
$\boldsymbol{G'}, \{(\boldsymbol{s'_i}, \boldsymbol{t'_i})\}_{i \in [\boldsymbol{k'}]}$

Size: $O(k^k \cdot n)$

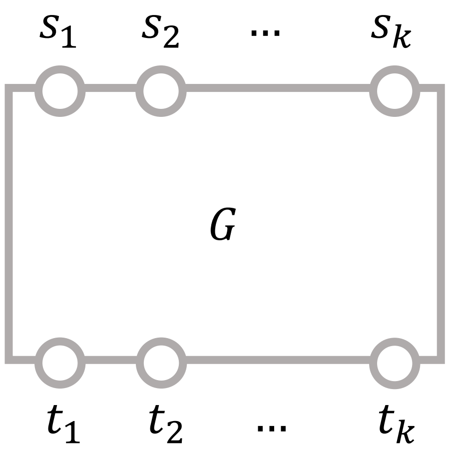
Parameter: $k' = k^{k+1}$

YES: $OPT(I') = k'$

NO: $OPT(I') \leq k'(1 - 1.1/k)$



Wlodarczyk's Gap Amplification Step



SO Instance $\boldsymbol{I}$

$\boldsymbol{G}, \{(\boldsymbol{s_i}, \boldsymbol{t_i})\}_{i \in [\boldsymbol{k}]}$

Size: n

Parameter: k

YES: $OPT(I) = k$

NO: $OPT(I) \leq k - 1$

SO Instance $\boldsymbol{I'}$

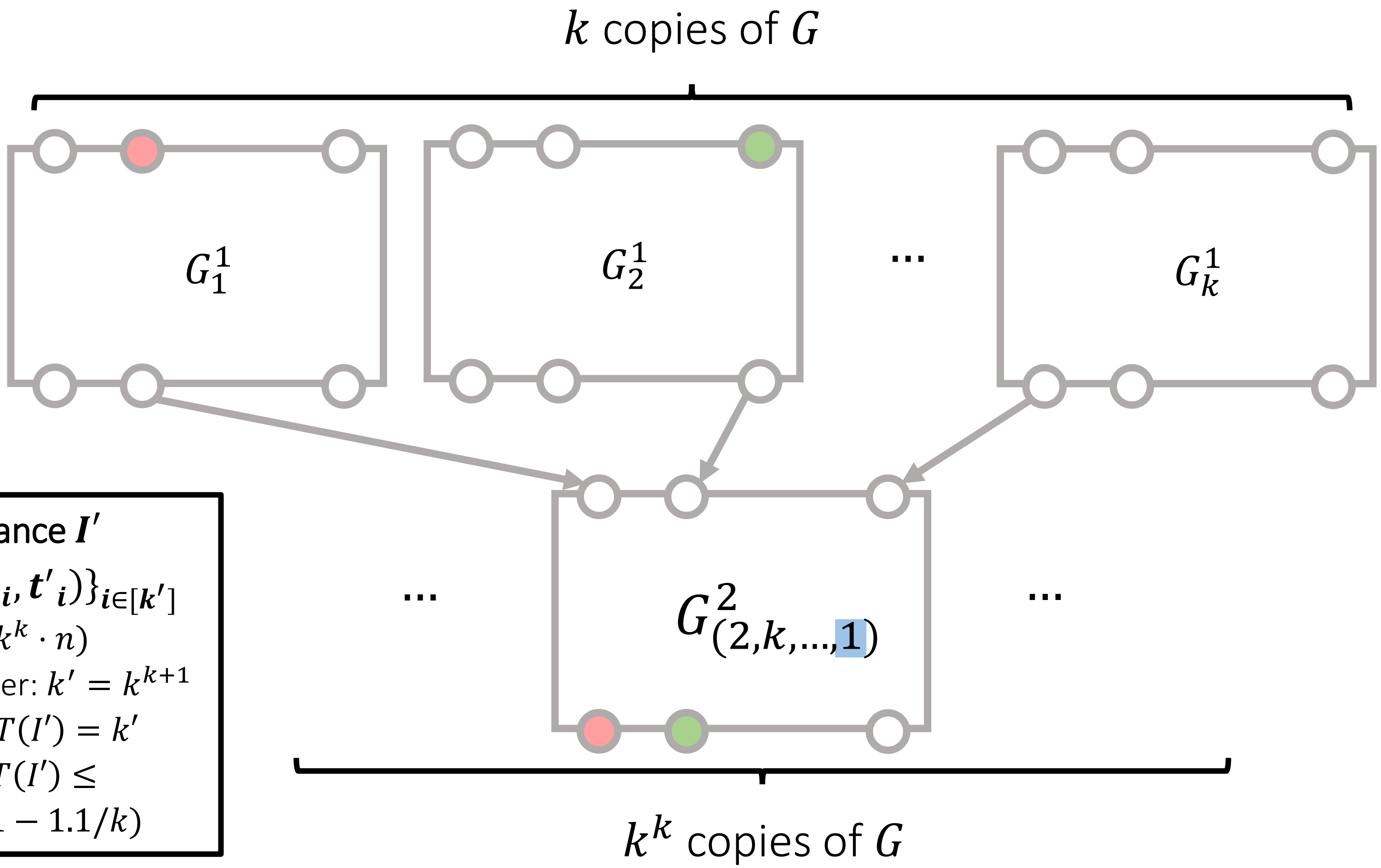
$\boldsymbol{G'}, \{(\boldsymbol{s'_i}, \boldsymbol{t'_i})\}_{i \in [\boldsymbol{k'}]}$

Size: $O(k^k \cdot n)$

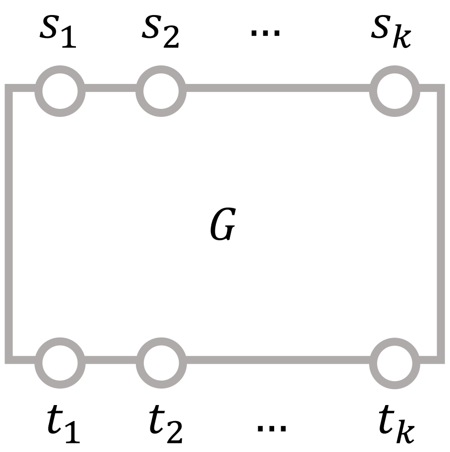
Parameter: $k' = k^{k+1}$

YES: $OPT(I') = k'$

NO: $OPT(I') \leq k'(1 - 1.1/k)$



Wlodarczyk's Gap Amplification Step



SO Instance I

$\boldsymbol{G, \{(s_i, t_i)\}_{i \in [k]}}$

Size: n

Parameter: k

YES: $OPT(I) = k$

NO: $OPT(I) \leq k - 1$

SO Instance I'

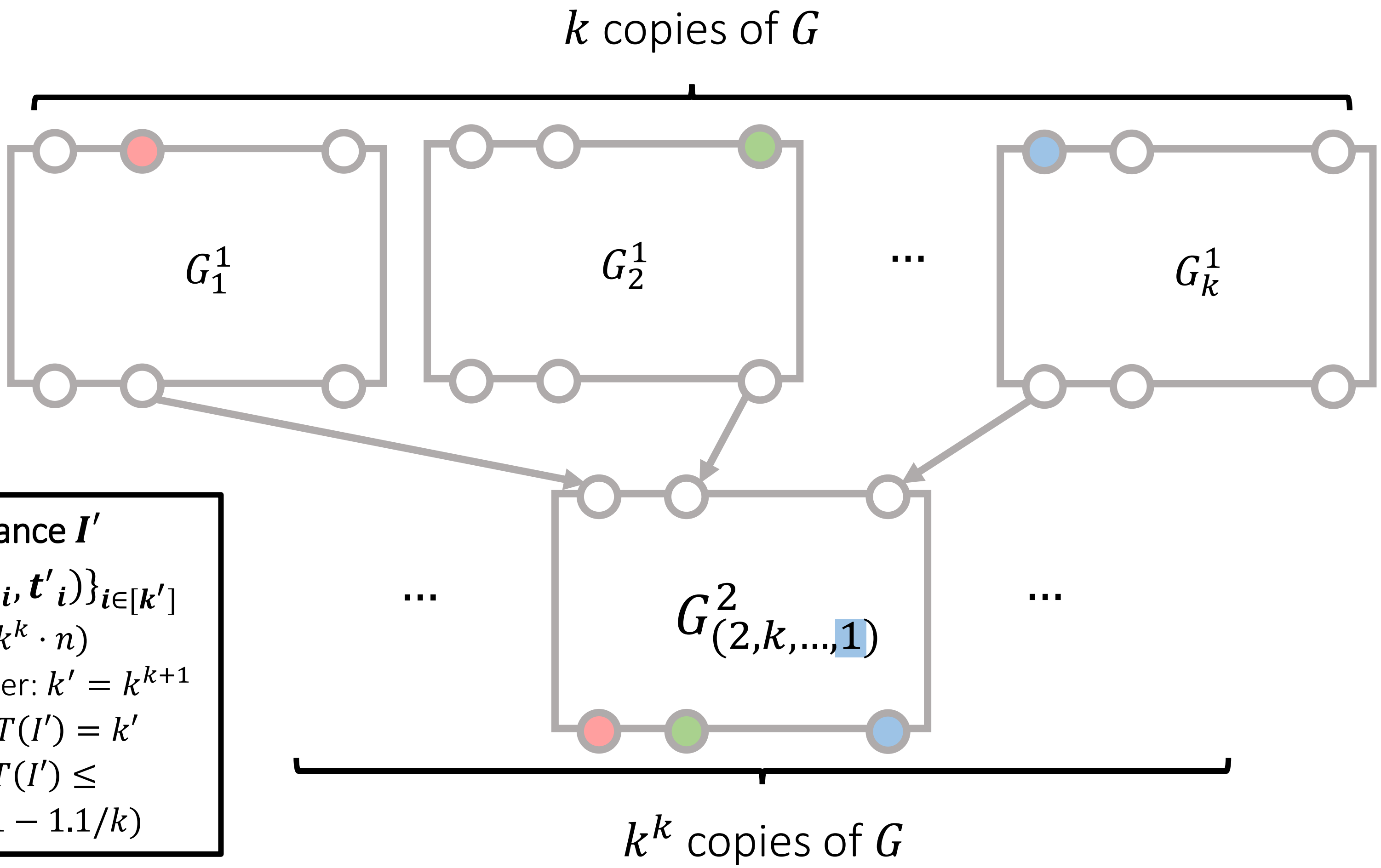
$\boldsymbol{G', \{(s'_i, t'_i)\}_{i \in [k']}}$

Size: $O(k^k \cdot n)$

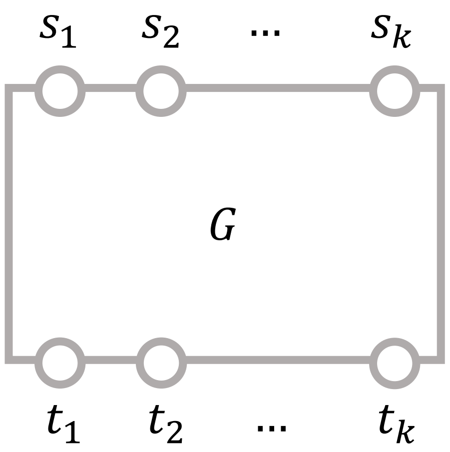
Parameter: $k' = k^{k+1}$

YES: $OPT(I') = k'$

NO: $OPT(I') \leq k'(1 - 1.1/k)$



Wlodarczyk's Gap Amplification Step



SO Instance I

$\boldsymbol{G}, \{(\boldsymbol{s_i}, \boldsymbol{t_i})\}_{i \in [k]}$

Size: n

Parameter: k

YES: $OPT(I) = k$

NO: $OPT(I) \leq k - 1$

SO Instance I'

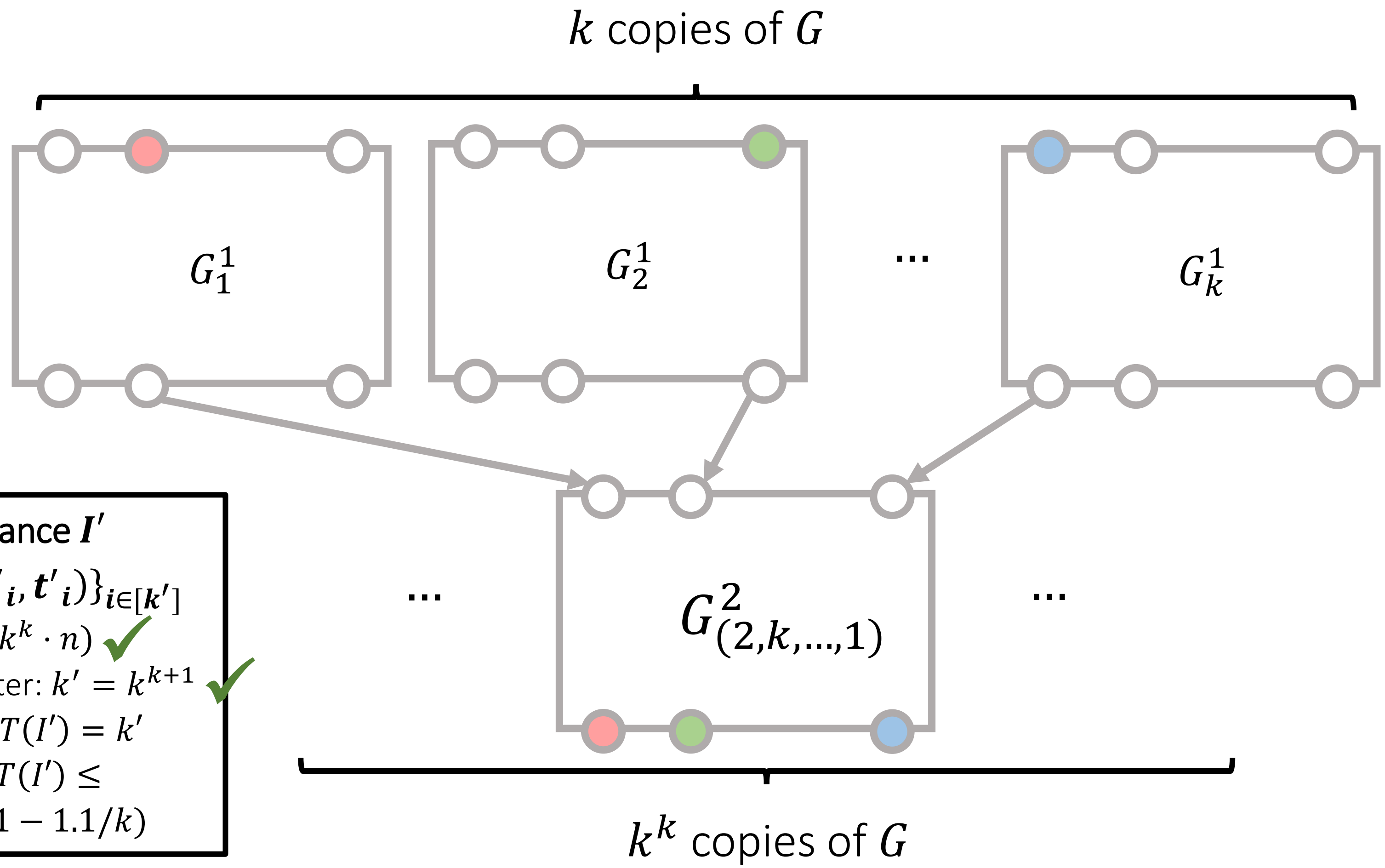
$\boldsymbol{G'}, \{(\boldsymbol{s'_i}, \boldsymbol{t'_i})\}_{i \in [k']}$

Size: $O(k^k \cdot n)$ ✓

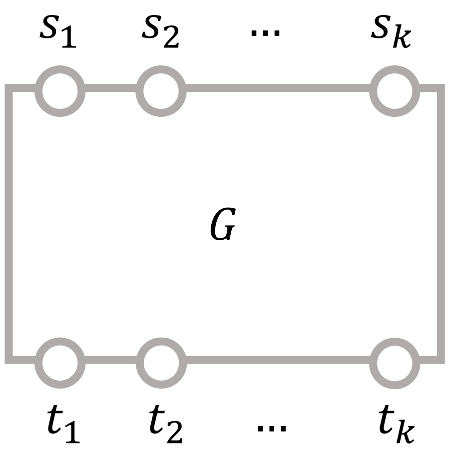
Parameter: $k' = k^{k+1}$ ✓

YES: $OPT(I') = k'$

NO: $OPT(I') \leq k'(1 - 1.1/k)$



Wlodarczyk's Gap Amplification Step



SO Instance I

$\boldsymbol{G, \{(s_i, t_i)\}_{i \in [k]}}$

Size: n

Parameter: k

YES: $OPT(I) = k$

NO: $OPT(I) \leq k - 1$

SO Instance I'

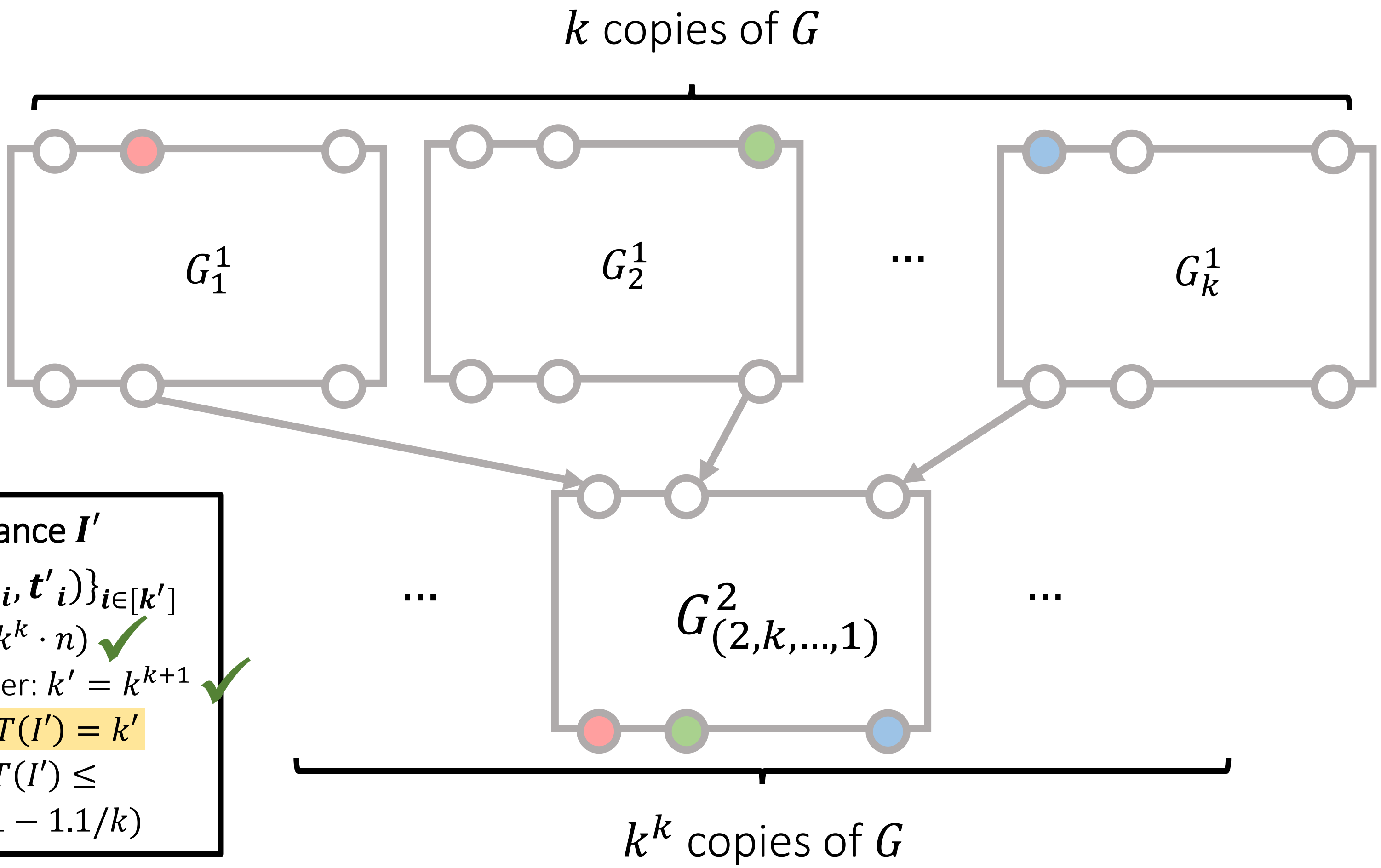
$\boldsymbol{G', \{(s'_i, t'_i)\}_{i \in [k']}}$

Size: $O(k^k \cdot n)$ ✓

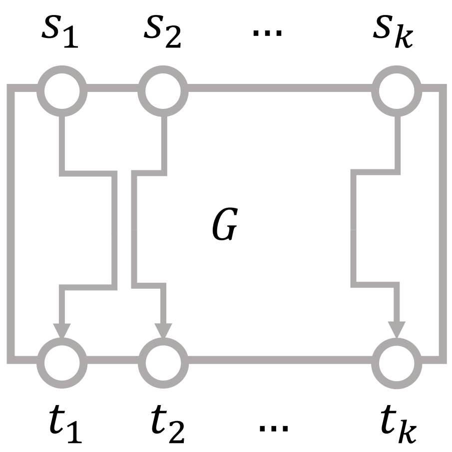
Parameter: $k' = k^{k+1}$ ✓

YES: $OPT(I') = k'$

NO: $OPT(I') \leq k'(1 - 1.1/k)$



Wlodarczyk’s Gap Amplification Step



SO Instance I

$G, \{(s_i, t_i)\}_{i \in [k]}$

Size: n

Parameter: k

YES: $OPT(I) = k$

NO: $OPT(I) \leq k - 1$

SO Instance I'

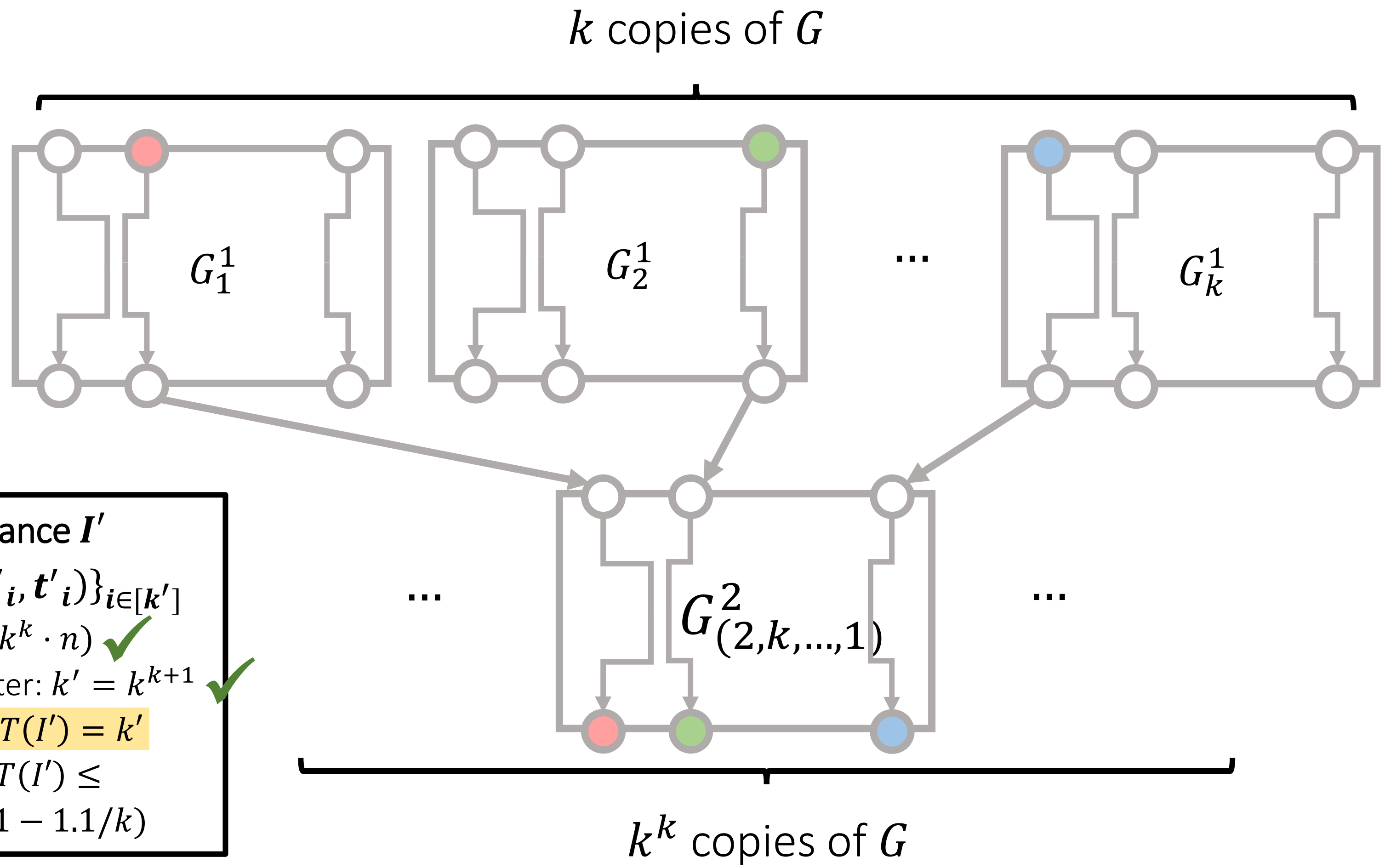
$G', \{(s'_i, t'_i)\}_{i \in [k']}$

Size: $O(k^k \cdot n)$

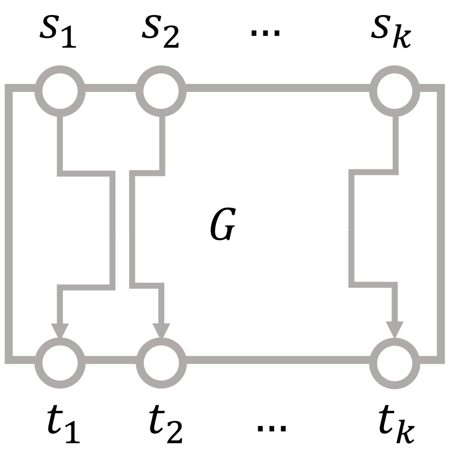
Parameter: $k' = k^{k+1}$

YES: $OPT(I') = k'$

NO: $OPT(I') \leq k'(1 - 1.1/k)$



Wlodarczyk’s Gap Amplification Step



SO Instance I

$G, \{(s_i, t_i)\}_{i\in[k]}$

Size: n

Parameter: k

YES: $OPT(I) = k$

NO: $OPT(I) \leq k - 1$

SO Instance I'

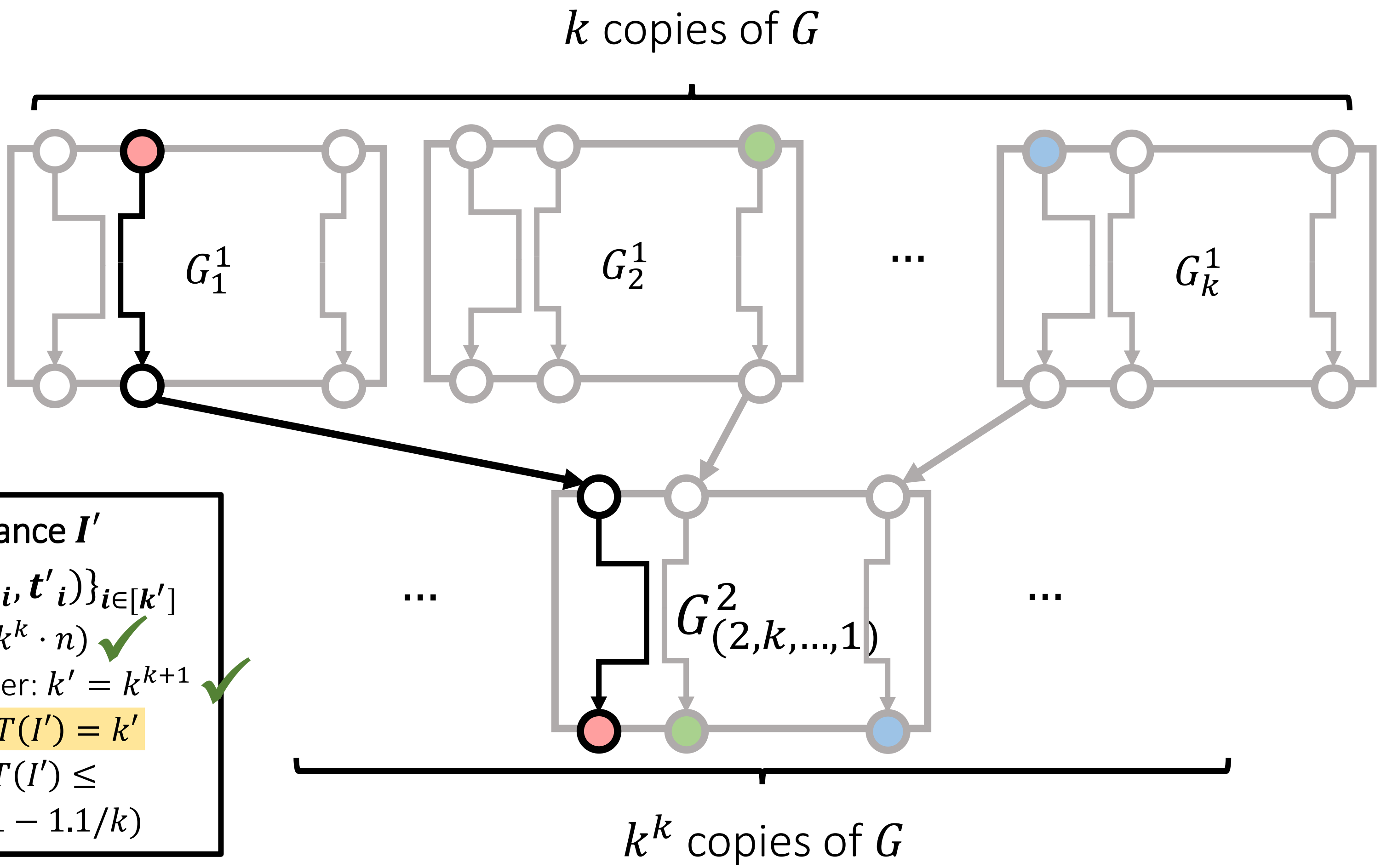
$G', \{(s'_i, t'_i)\}_{i\in[k']}$

Size: $O(k^k \cdot n)$ ✓

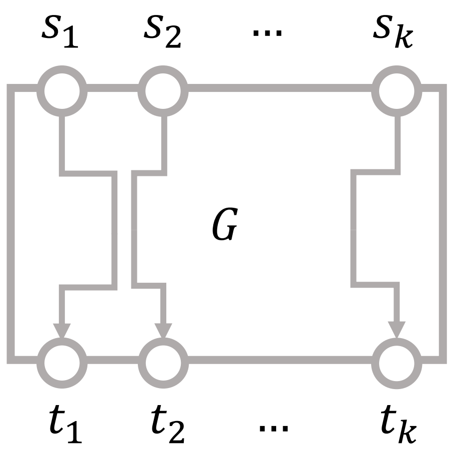
Parameter: $k' = k^{k+1}$ ✓

YES: $OPT(I') = k'$

NO: $OPT(I') \leq k'(1 - 1.1/k)$



Wlodarczyk's Gap Amplification Step



SO Instance I

$G, \{(s_i, t_i)\}_{i \in [k]}$

Size: n

Parameter: k

YES: $OPT(I) = k$

NO: $OPT(I) \le k - 1$

SO Instance I'

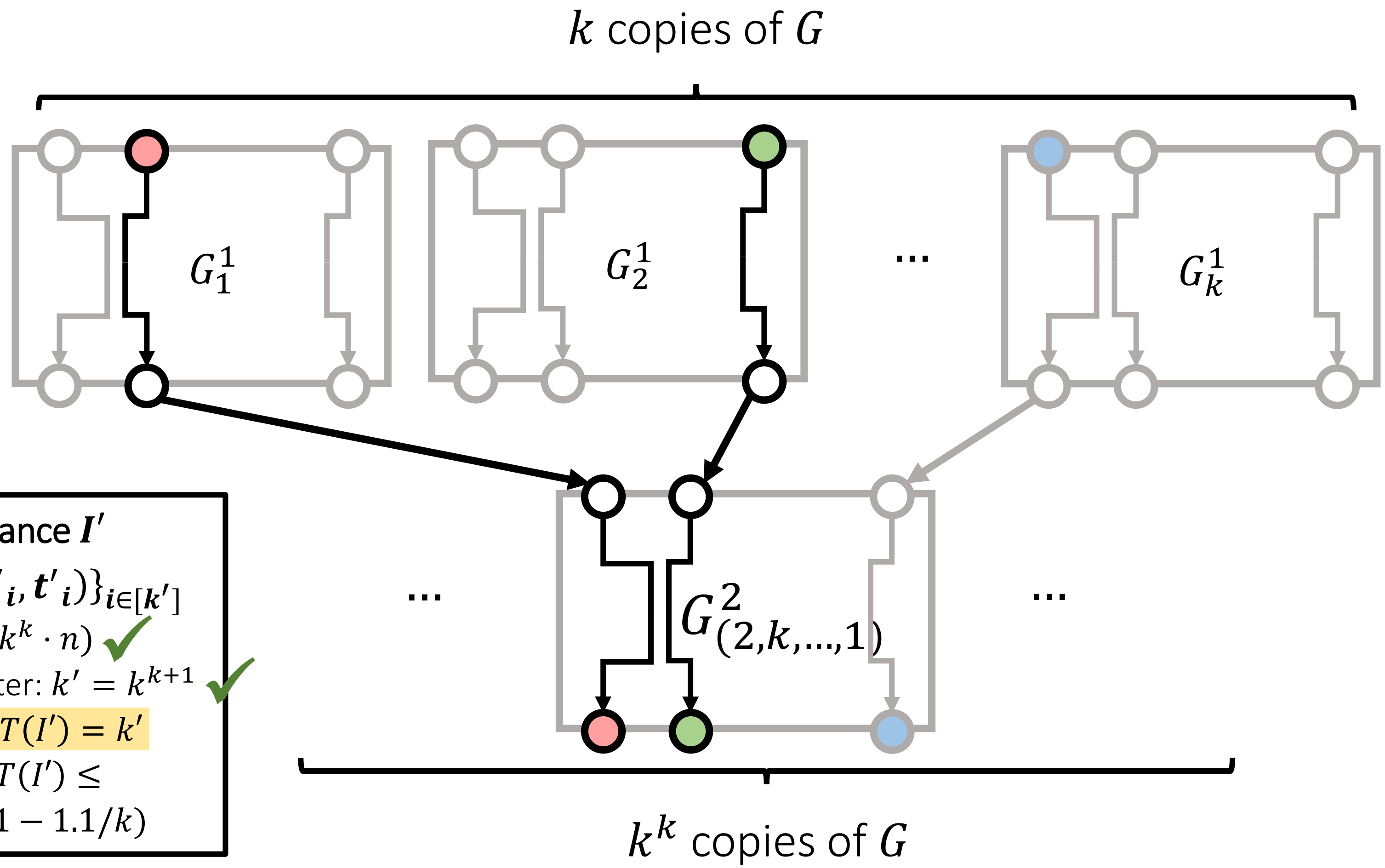
$G', \{(s'_i, t'_i)\}_{i \in [k']}$

Size: $O(k^k \cdot n)$ ✓

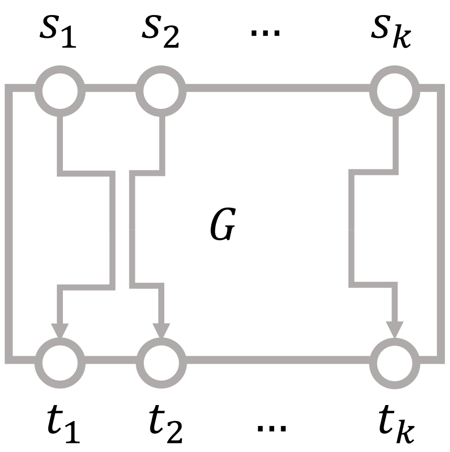
Parameter: $k' = k^{k+1}$ ✓

YES: $OPT(I') = k'$

NO: $OPT(I') \le k'(1 - 1.1/k)$



Wlodarczyk’s Gap Amplification Step



SO Instance I

$G, \{(s_i, t_i)\}_{i \in [k]}$

Size: n

Parameter: k

YES: $OPT(I) = k$

NO: $OPT(I) \leq k - 1$

SO Instance I'

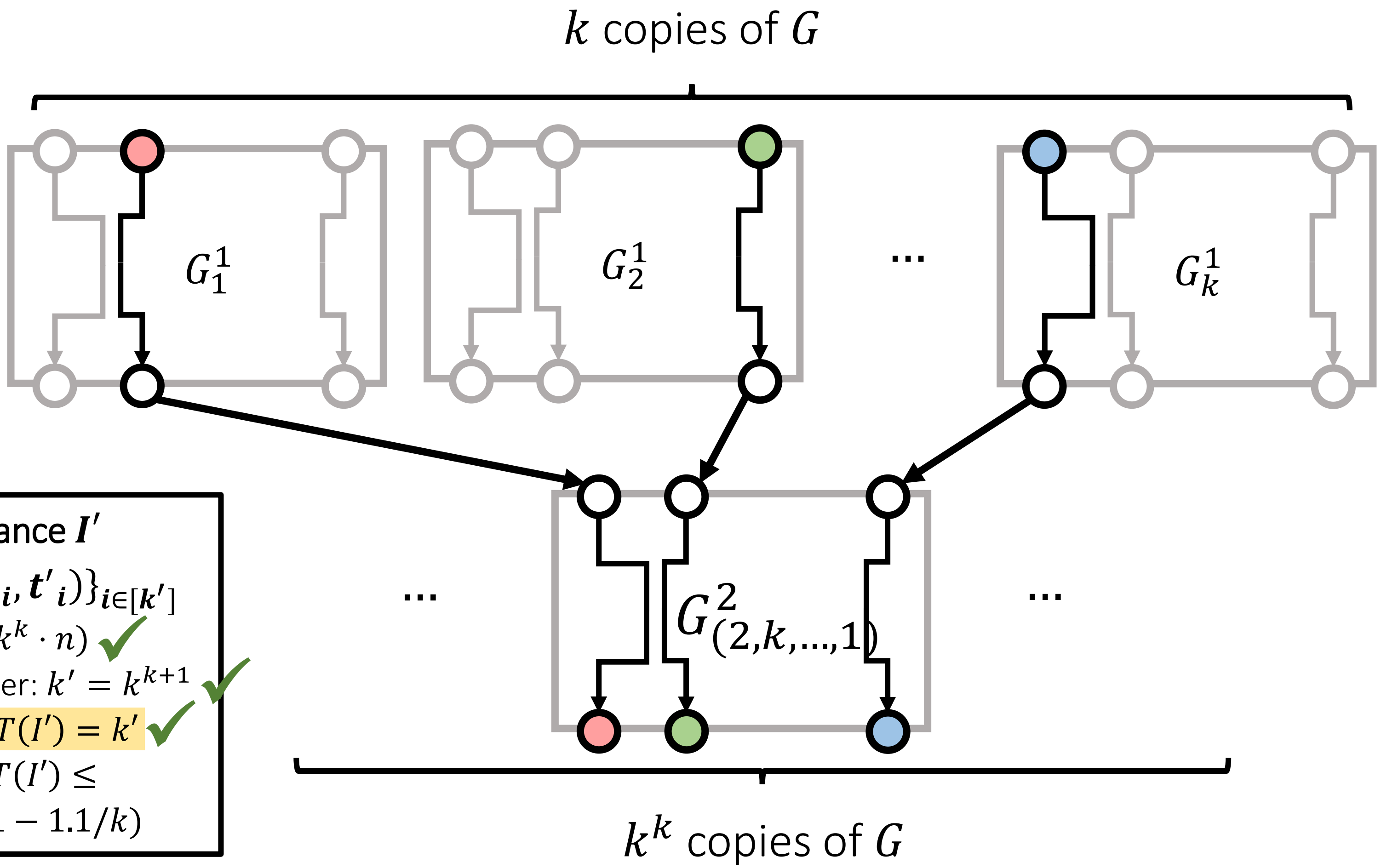
$G', \{(s'_i, t'_i)\}_{i \in [k']}$

Size: $O(k^k \cdot n)$

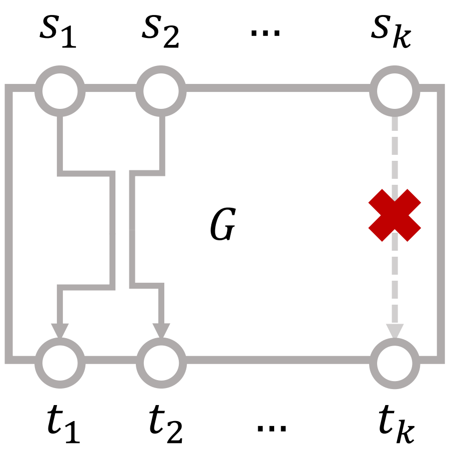
Parameter: $k' = k^{k+1}$

YES: $OPT(I') = k'$

NO: $OPT(I') \leq k'(1 - 1.1/k)$



Wlodarczyk's Gap Amplification Step



SO Instance I

$G, \{(s_i, t_i)\}_{i \in [k]}$

Size: n

Parameter: k

YES: $OPT(I) = k$

NO: $OPT(I) \leq k - 1$

SO Instance I'

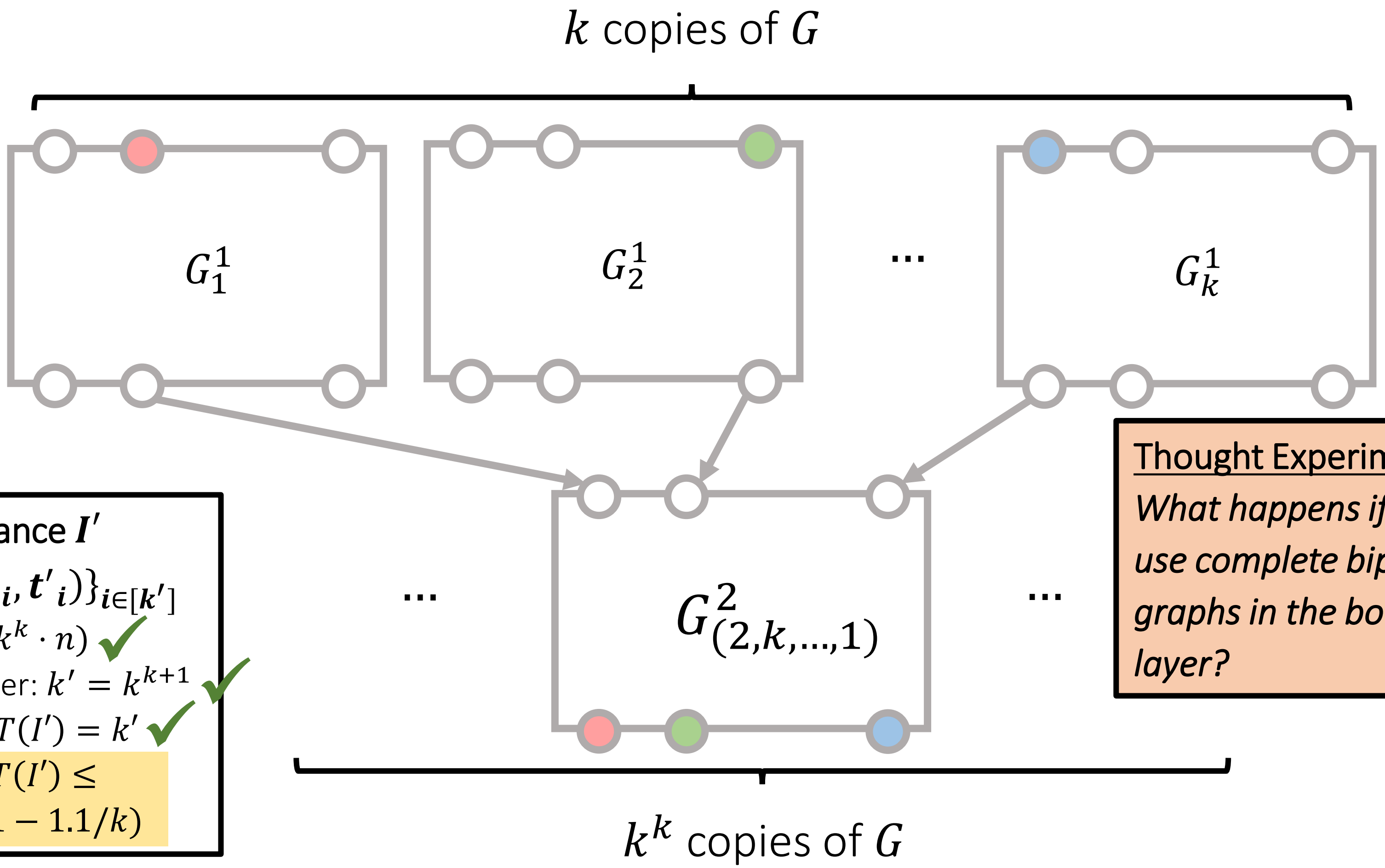
$G', \{(s'_i, t'_i)\}_{i \in [k']}$

Size: $O(k^k \cdot n)$ ✓

Parameter: $k' = k^{k+1}$ ✓

YES: $OPT(I') = k'$ ✓

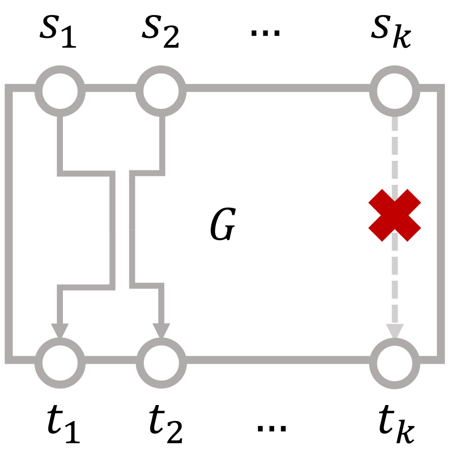
NO: $OPT(I') \leq k'(1 - 1.1/k)$



Thought Experiment

What happens if we use complete bipartite graphs in the bottom layer?

Wlodarczyk's Gap Amplification Step



SO Instance I

$G, \{(s_i, t_i)\}_{i \in [k]}$

Size: n

Parameter: k

YES: $OPT(I) = k$

NO: $OPT(I) \leq k - 1$

SO Instance I'

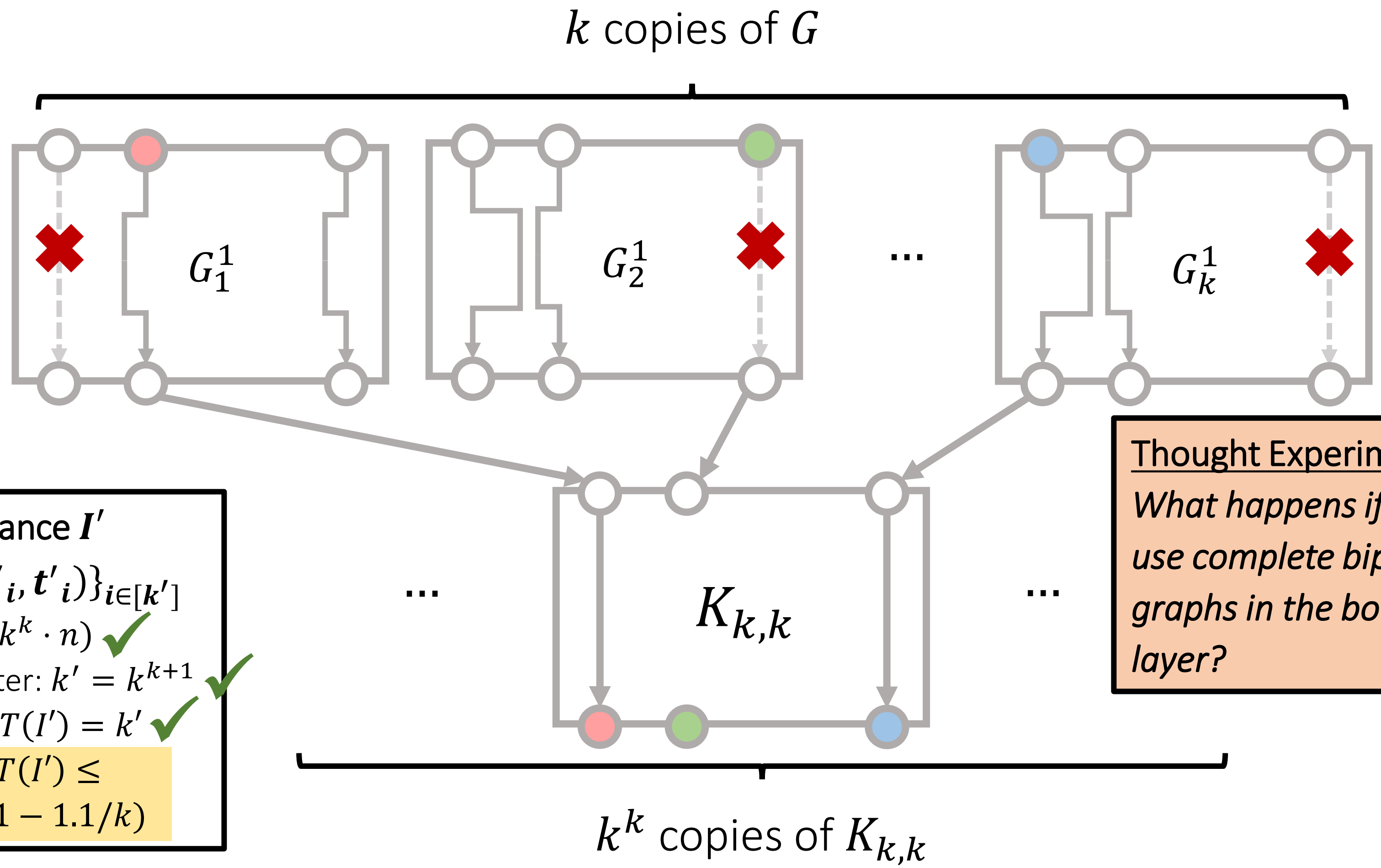
$G', \{(s'_i, t'_i)\}_{i \in [k']}$

Size: $O(k^k \cdot n)$

Parameter: $k' = k^{k+1}$

YES: $OPT(I') = k'$

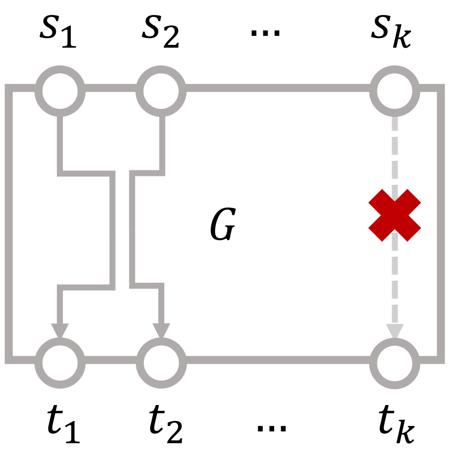
NO: $OPT(I') \leq k'(1 - 1.1/k)$



Thought Experiment

What happens if we use complete bipartite graphs in the bottom layer?

Wlodarczyk's Gap Amplification Step



SO Instance $\boldsymbol{I}$

$\boldsymbol{G}, \{(\boldsymbol{s_i}, \boldsymbol{t_i})\}_{i \in [k]}$

Size: n

Parameter: k

YES: $OPT(I) = k$

NO: $OPT(I) \leq k - 1$

SO Instance $\boldsymbol{I'}$

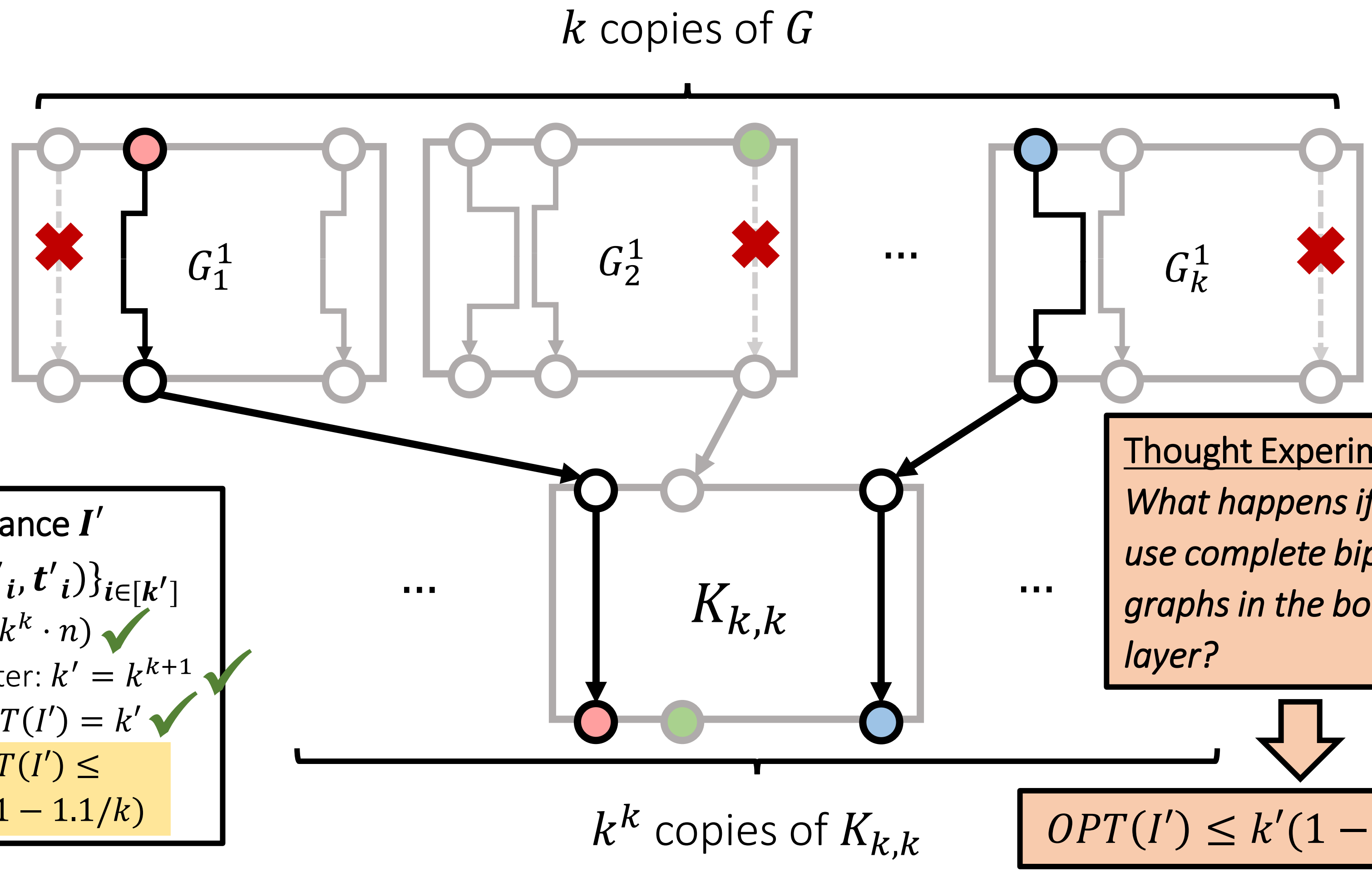
$\boldsymbol{G'}, \{(\boldsymbol{s'}_i, \boldsymbol{t'}_i)\}_{i \in [k']}$

Size: $O(k^k \cdot n)$ ✓

Parameter: $k' = k^{k+1}$ ✓

YES: $OPT(I') = k'$ ✓

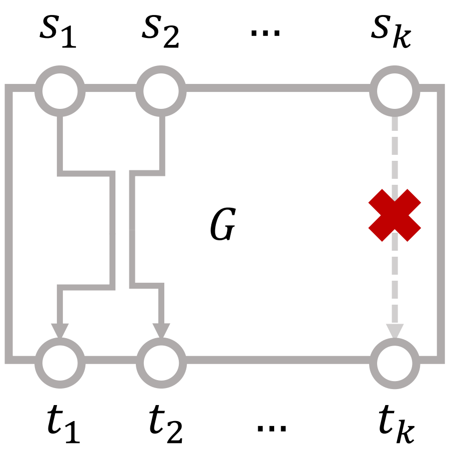
NO: $OPT(I') \leq$
$k'(1 - 1.1/k)$



<u>Thought Experiment</u>
What happens if we use complete bipartite graphs in the bottom layer?

$OPT(I') \leq k'(1 - 1/k)$

Wlodarczyk's Gap Amplification Step



SO Instance I

$G, \{(s_i, t_i)\}_{i \in [k]}$

Size: n

Parameter: k

YES: $OPT(I) = k$

NO: $OPT(I) \leq k - 1$

SO Instance I'

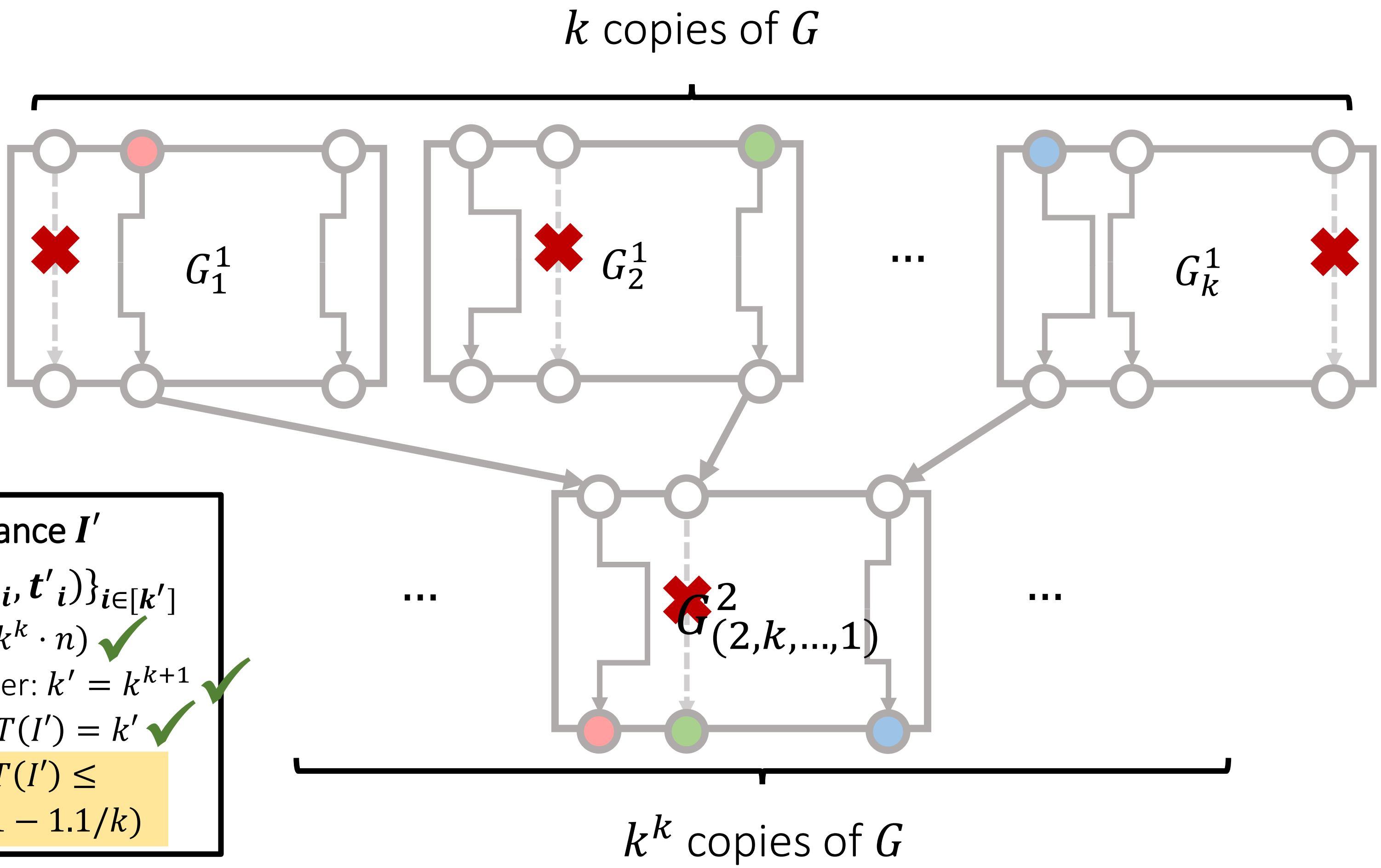
$G', \{(s'_i, t'_i)\}_{i \in [k']}$

Size: $O(k^k \cdot n)$ ✓

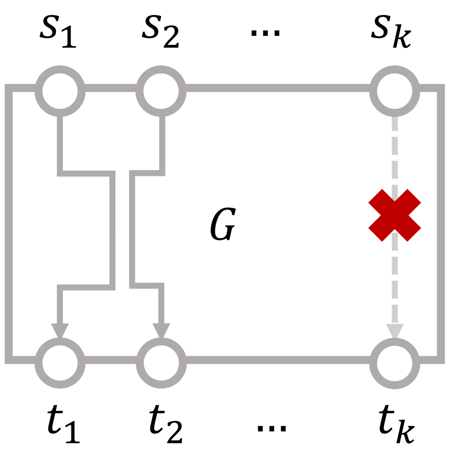
Parameter: $k' = k^{k+1}$ ✓

YES: $OPT(I') = k'$ ✓

NO: $OPT(I') \leq k'(1 - 1.1/k)$



Wlodarczyk's Gap Amplification Step



SO Instance I

$G, \{(s_i, t_i)\}_{i \in [k]}$

Size: n

Parameter: k

YES: $OPT(I) = k$

NO: $OPT(I) \leq k - 1$

SO Instance I'

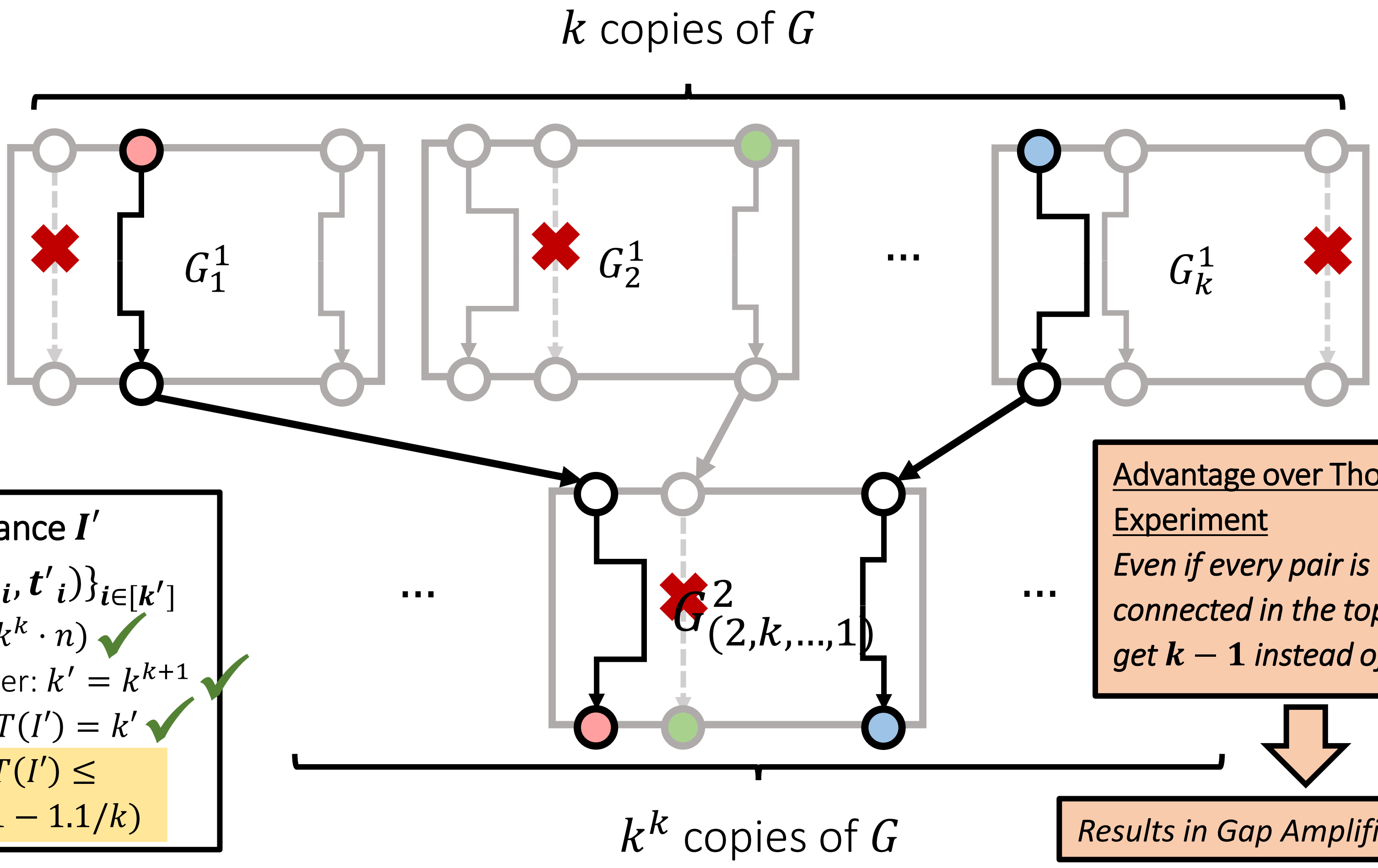
$G', \{(s'_i, t'_i)\}_{i \in [k']}$

Size: $O(k^k \cdot n)$ ✓

Parameter: $k' = k^{k+1}$ ✓

YES: $OPT(I') = k'$ ✓

NO: $OPT(I') \leq k'(1 - 1.1/k)$



<u>Advantage over Thought Experiment</u>

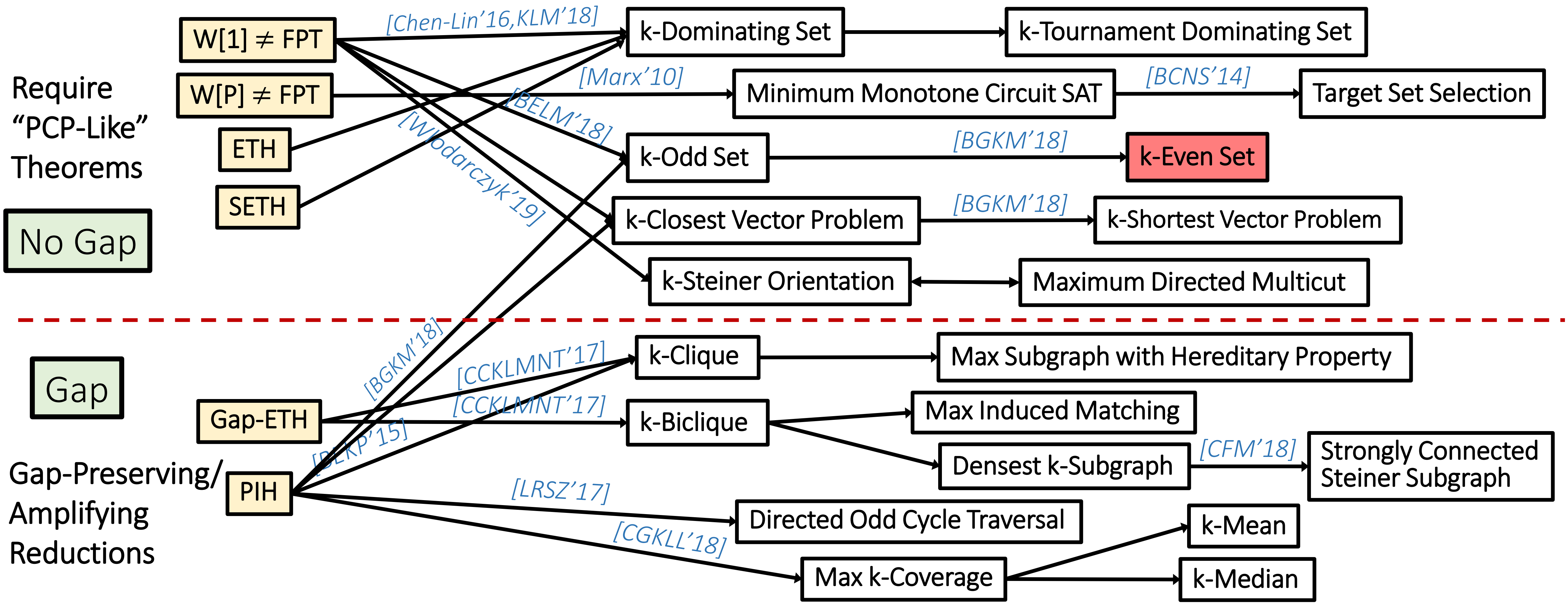
*Even if every pair is connected in the top layer, get **$k-1$** instead of **k***

Results in Gap Amplification

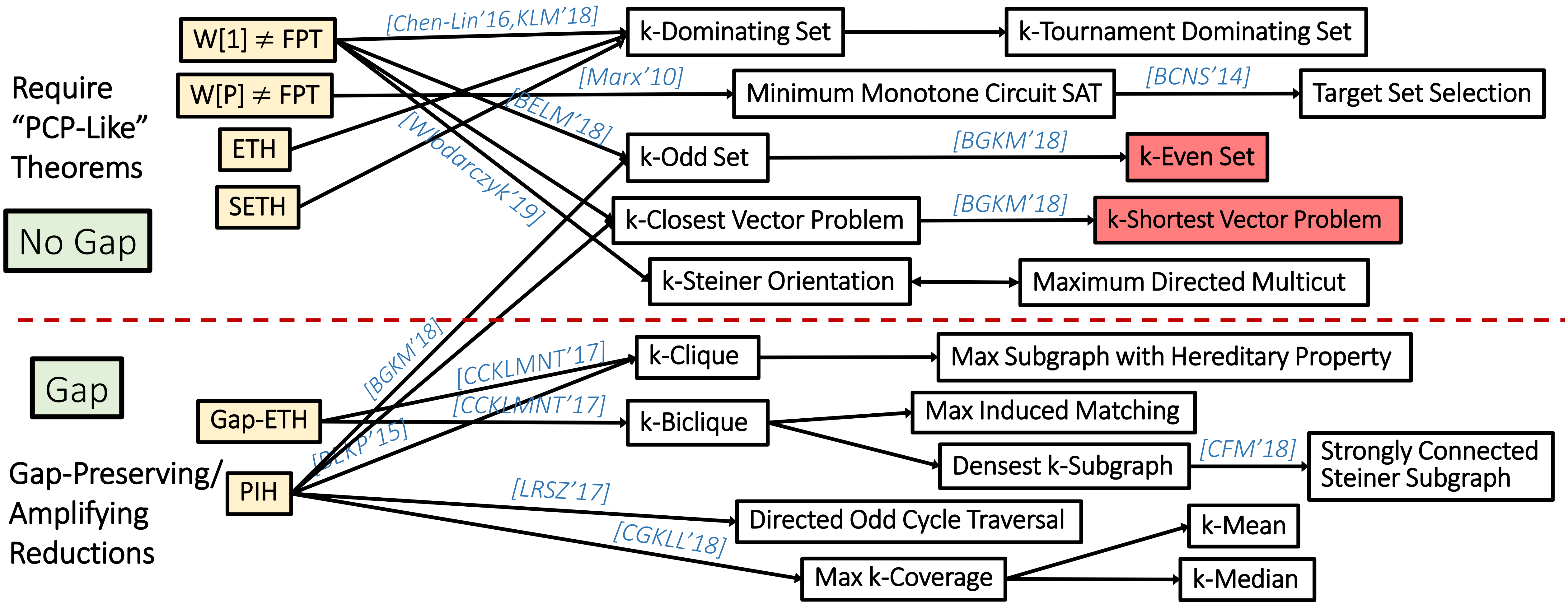
Concluding Remarks

- Useful not just for ruling out approximation algorithms

Parameterized Inapproximability: *Recent Developments*



Parameterized Inapproximability: *Recent Developments*



Concluding Remarks

- Useful not just for ruling out approximation algorithms
- Many open problems:
  - Directed Odd Cycle Traversal, Minimum k-Cut, …
- W[1]-Completeness of Approximating k-Clique?
- W[2]-Completeness of Approximating k-Domset?
- Unify Gap-Producing Techniques?
- Try to understand the hypotheses better

Complexity Assumptions

Exponential Time Hypotheses

Exponential time Hypothesis (ETH) *[Impagliazzo, Paturi'01]*
No $2^{o(n)}$-time algo can decide whether a 3CNF formula is satisfiable

Strong ETH (SETH) *[Impagliazzo, Paturi'01]*
For any constant $\varepsilon > 0$, no $O(2^{(1-\varepsilon)n})$-time algorithm can decide whether a CNF formula is satisfiable

Gap ETH (Gap-ETH) *[Dinur'16] [M, Raghavendra'16]*
For some constant $\delta > 0$, no $2^{o(n)}$-time algorithm can distinguish between a satisfiable CNF formula and one which is not even $(1-\delta)$-satisfiable.

Parameterized Complexity

W[1] $\neq$ FPT Assumption *[Downey, Fellows'92]*
For any function f, no $f(k)n^{O(1)}$-time algo can decide whether a given graph contains a k-clique

W[2] $\neq$ FPT Assumption *[Downey, Fellows'92]*
For any function f, no $f(k)n^{O(1)}$-time algo can decide whether a given graph contains a dominating set of size k

Parameterized Inapproximability Hypothesis (PIH) *[Lokshtanov, Ramanujan, Saurabh, Zehavi'17]*
For some $\delta > 0$ and any function f, no $f(k)n^{O(1)}$-time algo can distinguish between a satisfiable 2CSP instance with k variables and alphabet size n and one which is not even $(1 - \delta)$-satisfiable.

Complexity Assumptions

Exponential Time Hypotheses

Exponential time Hypothesis (ETH) *[Impagliazzo, Paturi'01]*
No $2^{o(n)}$-time algo can decide whether a 3CNF formula is satisfiable

Strong ETH (SETH) *[Impagliazzo, Paturi'01]*
For any constant $\varepsilon > 0$, no $O(2^{(1-\varepsilon)n})$-time algorithm can decide whether a CNF formula is satisfiable

Equivalence?

Gap ETH (Gap-ETH) *[Dinur'16] [M, Raghavendra'16]*
For some constant $\delta > 0$, no $2^{o(n)}$-time algorithm can distinguish between a satisfiable CNF formula and one which is not even $(1-\delta)$-satisfiable.

Parameterized Complexity

W[1] $\neq$ FPT Assumption *[Downey, Fellows'92]*
For any function f, no $f(k)n^{O(1)}$-time algo can decide whether a given graph contains a k-clique

W[2] $\neq$ FPT Assumption *[Downey, Fellows'92]*
For any function f, no $f(k)n^{O(1)}$-time algo can decide whether a given graph contains a dominating set of size k

Parameterized Inapproximability Hypothesis (PIH) *[Lokshtanov, Ramanujan, Saurabh, Zehavi'17]*
For some $\delta > 0$ and any function f, no $f(k)n^{O(1)}$-time algo can distinguish between a satisfiable 2CSP instance with k variables and alphabet size n and one which is not even $(1-\delta)$-satisfiable.

Complexity Assumptions

Exponential Time Hypotheses

Exponential time Hypothesis (ETH) *[Impagliazzo, Paturi'01]*
No $2^{o(n)}$-time algo can decide whether a 3CNF formula is satisfiable

Strong ETH (SETH) *[Impagliazzo, Paturi'01]*
For any constant $\varepsilon > 0$, no $O(2^{(1-\varepsilon)n})$-time algorithm can decide whether a CNF formula is satisfiable

Equivalence?

Gap ETH (Gap-ETH) *[Dinur'16] [M, Raghavendra'16]*
For some constant $\delta > 0$, no $2^{o(n)}$-time algorithm can distinguish between a satisfiable CNF formula and one which is not even $(1-\delta)$-satisfiable.

Parameterized Complexity

W[1] $\neq$ FPT Assumption *[Downey, Fellows'92]*
For any function f, no $f(k)n^{O(1)}$-time algo can decide whether a given graph contains a k-clique

W[2] $\neq$ FPT Assumption *[Downey, Fellows'92]*
For any function f, no $f(k)n^{O(1)}$-time algo can decide whether a given graph contains a dominating set of size k

Equivalence?

Parameterized Inapproximability Hypothesis (PIH) *[Lokshtanov, Ramanujan, Saurabh, Zehavi'17]*
For some $\delta > 0$ and any function f, no $f(k)n^{O(1)}$-time algo can distinguish between a satisfiable 2CSP instance with k variables and alphabet size n and one which is not even $(1-\delta)$-satisfiable.

THANK YOU

Main References

[Chalermsook-Cygan-Kortsarz-Laekhanukit-Manurangsi-Nanongkai-Trevisan'17]

From Gap-ETH to FPT-Inapproximability: Clique, Dominating Set, and More. *FOCS'17*

[Karthik-Laekhanukit-Manurangsi'18]

On the parameterized complexity of approximating dominating set. *STOC'18*

[Wlodarczyk'19]

Inapproximability within W[1]: the case of Steiner Orientation. arXiv:1907.06529.