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 $(s_1,t_1),..., (s_k,t_k)$.

**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

- Constant $k$: efficiently solvable [Robertson, Seymour ’90]
- Running time: $f(k) \cdot n^2$ [Kawarabayashi, Kobayashi, Reed]

$$f(k) = 2^{2^k}.$$
Complexity of NDP

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

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

Plan:
• get polylog(n)-approximation for grids
• extend to planar graphs
• look into general graphs
Approximation Status of NDP from 2015

- $O(\sqrt{n})$-approximation algorithm
  - 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 $\epsilon$ [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
- $\tilde{O}(n^{9/19})$-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 $\epsilon$ [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 algorithm
  - 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 $\epsilon$ [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, Nimavat ’17]

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

unless

$NP \subseteq \text{RTIME}(n^{\text{poly log } n})$
Almost polynomial Hardness

Hardness of NDP on grids: [C, Kim, Nimavat '17]

- Hardness for any constant
  \[2^{(\log n)^{1-\epsilon}}\]
- Hardness under randomized ETH
  \[n^{\Omega(1/(\log \log n)^2)}\]
Almost polynomial Hardness

Hardness of NDP on grids: [C, Kim, Nimavat ’17]

- $2^{O(\log n)}$ hardness for any constant $\epsilon$

- $n^{\Omega(1/(\log \log n)^2)}$ hardness

unless

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

unless for every $\delta$

$\text{NP} \subseteq \text{RTIME}(2^{n^\delta})$
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 $\epsilon$ [Andrews, Zhang ‘05], [Andrews, C, Guruswami, Khanna, Talwar, Zhang ‘10]

• But: constant approximation on grids [Aumann, Rabani ‘95], [Kleinberg Tardos ‘95], [Kleinberg, Tardos ‘98]
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

- Congestion $\Omega(\log n / \log \log n)$: constant approximation [Raghavan, Thompson ’87]
- Congestion $c$: $\frac{1}{c}$-approximation [Azar, Regev ’01], [Baveja, Srinivasan ’00], [Kolliopoulos, Stein ’04]
- Congestion $\tilde{\Theta}(\log \log n)$: $\tilde{O}(\log n)$-approximation [Andrews ’10]
- Congestion 2: $\frac{1}{2}$-approximation [Kawarabayashi, Kobayashi ’11]
- Congestion 14: $\text{polylog}(k)$-approximation [C, ’11]
- Congestion 2: $\text{polylog}(k)$-approximation [C, Li ’12]
- $\text{polylog}(k)$-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 $\text{polylog}(k)$-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.

Goal: maximize pieces
Weird Graph Partitioning Problem (WGP)

**Input**: bipartite graph $G=(V,E)$, integers $p, L$.

**Output**: 
- partition $G$ into $p$ vertex-induced subgraphs.
- for each $i$, subset $E_i$ of edges, with $|E_i| \leq h$

**Goal**: maximize

**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)

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 $\sum_i |E_i|$

Intuitive View 1: Balanced Cut

Except:
- Partition into $p$ and not 2 pieces
- Maximize # of surviving 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 \( \sum_{i=1}^{p} |E_i| \)

Intuitive View 2: Densest k-Subgraph

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]
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 $\sum_i |E_i|$. 

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

(up to polylog n factor)
The Reduction
The Reduction

disjoint endpoints

Edge ↔ Demand Pair
The Reduction

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

sources

destinations
The Reduction

all blocks must be far from each other and from the grid boundary
The Reduction
The Reduction

\[ s_1 \]

\[ t_1 \]

\[ G \]
The Reduction
The Reduction
The Reduction

distances between sources chosen strategically!
if too large, can route all pairs
if too small, won’t be able to route what we need to
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
Claim: the reduction preserves solution value to within polylog(n) factor!

suppose we route many demand pairs
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!
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.

Why?
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
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!
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
• 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]
Next Steps

3COL(5) instance G

Constraint Satisfaction Problem instance $\phi(G)$

WGP problem instance $H(G)$

2-prover protocol + parallel repetition
A CSP Instance $\phi(G)$

$r$ – number of repetitions

“edge”-variables

$x_1 
\bullet

x_2 
\bullet

x_3 
\bullet

x_4 
\bullet

\vdots

x_N 
\bullet

“vertex”-variables

\bullet

y_1

\bullet

y_2

\bullet

y_3

\vdots

\bullet

y_M

3COL(5)
A CSP Instance $\phi(G)$

- $r$ – number of repetitions

- "edge"-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

- "vertex"-variables
- $m^r$ variables

A CSP Instance $\phi(G)$

- r – number of repetitions

“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
A CSP Instance $\phi(G)$

- **r** – number of repetitions

- **“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.

need not be consistent across edges
A CSP Instance $\phi(G)$

- **r** – number of repetitions

- **“edge”-variables**
  - need not be consistent across edges

- **“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
A CSP Instance $\phi(G)$

- **r** – number of repetitions
- **“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

need not be consistent across edges
A CSP Instance $\phi(G)$

- **r** – number of repetitions
- **“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.

$6^r$ possible assignments per variable.

need not be consistent across edges.

G
A CSP Instance $\phi(G)$

- $r$ – number of repetitions

- “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 $\phi(G)$

- **r** – number of repetitions

- "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

need not be consistent across vertices
A CSP Instance $\phi(G)$

- **r** – number of repetitions

- **“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.

- Need not be consistent across vertices
- $3^r$ possible assignments per variable
A CSP Instance $\phi(G)$

- $r$ – number of repetitions

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

Diagram showing graph $G$ and labeled edges $e_1$ to $e_r$ and vertices $v_1$ to $v_r$. Each edge is circled in red.
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.
A CSP Instance $\phi(G)$

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

$r$ – number of repetitions

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

A perfect solution
A CSP Instance $\phi(G)$

- $r$ – number of repetitions

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.
A CSP Instance $\phi(G)$

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

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

$\text{r – number of repetitions}$

NP-hard to distinguish

$G$
Bonus Property for Yes Instance!

\[ r \text{ – number of repetitions} \]
Bonus Property for Yes Instance!

$r$ – number of repetitions

Solution 1
Solution 2
Solution 3
...
Solution $6^r$
r – number of repetitions

Bonus Property for Yes Instance!

Solution 1
Solution 2
Solution 3
...
Solution 6^r
Bonus Property for Yes Instance!

$r$ – number of repetitions

$6^r$ possible assignments

Each assignment appears in exactly 1 solution!
Bonus Property for Yes Instance!

r – number of repetitions

Solution 6

Solution 1
Solution 2
Solution 3
...
Solution 6^r
$r$ – number of repetitions

$3^r$ possible assignments

Each assignment in exactly $2^r$ solutions!
• For every edge, each legal coloring appears exactly once
• For each vertex, every coloring appears exactly twice
Next Steps

3COL(5) instance G

2-prover protocol + parallel repetition

Constraint Satisfaction Problem instance $\phi(G)$

WGP problem instance $H(G)$
A CSP Problem Instance $\phi$

$\phi$ is a Yes-Instance, if there is an assignment to variables satisfying all constraints.

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

NP-hard to distinguish + the bonus property
$\phi$ 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) instance $G$

Constraint Satisfaction Problem instance $\phi(G)$

WGP problem instance $H(G)$

2-prover protocol + parallel repetition
2 Graphs for CSP

- Constraint graph
- Assignment graph

1 vertex for each assignment
2 Graphs for CSP

- **Constraint Graph**: 1 vertex for each assignment

- **Assignment Graph**: 2^r vertices for each assignment
2 Graphs for CSP

- **Constraint graph**: Put an edge iff the assignments satisfy the constraint.
- **Assignment graph**: Associated with the constraint graph.

**Clouds**: Represents the set of constraints.
2 Graphs for CSP

- Constraint graph
- Assignment graph
2 Graphs for CSP
Reduction to WGP

... constraint graph

6r vertices

assignment graph

constraint graph
Reduction to WGP

Input to WGP problem

- $p = 6^r$
- $L = \text{#constraints}$

$6^r$ vertices
Yes Case Analysis

Input to WGP problem
• \( p = 6^r \)
• \( L = \#\text{constraints} \)

Solution 1
Solution 2
Solution 3
...
Solution \( 6^r \)
Yes Case Analysis

Input to WGP problem
- \( p=6^r \)
- \( L=\#\text{constraints} \)
Each solution defines a piece in the partition
Yes Case Analysis

Solution 1
Yes Instance Analysis

Solution 1

$G_1$

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

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

In ideal solution, each piece defines assignment to variables

Can only satisfy few constraints, so #edges in each piece very low!
Problem: No-Instance solution does not have to look this way!
No Instance Analysis
Two Extreme Solutions

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

canonical honest solution

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

\[ G_1 \]

\[ x_i \]

\[ y_j \]
Yes—Case solution: constraint will contribute $6^r$ edges to solution.
Yes—Case solution: constraint will contribute $6^r$ edges to solution.

unfair advantage to cheating solutions!

cheating solution may collect a lot more per constraint!
Solution: Cheat
Hardness Proof Plan

3COL(5) instance $G$ → Constraint Satisfaction Problem instance $\phi(G)$ → WGP problem instance $H(G)$ → NDP in grids

2-prover protocol + parallel repetition
Hardness Proof Plan

3COL(5) instance G

2-prover protocol + parallel repetition

Constraint Satisfaction Problem instance $\phi(G)$

Even Weirder Graph Partitioning Problem

EWGP problem instance $H(G)$

NDP in grids
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

... but we still don’t know how to prove that its value is low

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

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 $\alpha$-approximation algorithm $A$ for WGP.

If CSP is a Yes-Instance, each WGP instance has a high-value solution
Assume for contradiction that there is an $\alpha$-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 $\alpha$-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.
Assume for contradiction that there is an $\alpha$-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 $\alpha$-approximation algorithm $A$ for WGP.

will use the algorithm to distinguish yes and no instances of CSP
will always assume that the solution is canonical honest or cheating
Solution value too low?

constraint graph

No Instance!

approx solution

assignment graph/WGP input

G_1

G_2

...
High solution value + canonical honest solution?

Yes Instance!
High solution value + canonical cheating solution?
High solution value + canonical cheating solution?

Solution partitions the constraint graph into many small pieces; keeps most constraints.
Apply same reduction to each piece!

- Build assignment graph for each piece separately
- apply approx. algorithm to each

partition constraint graph into many small pieces; keeps most constraints
The Big Picture

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 with a high solution value and honest solution becomes inactive
- Each piece with a 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
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

Yes Instance!
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!