Almost Optimal Inapproximability of Multidimensional Packing Problems

Sai Sandeep

CMU

June 10, 2021

Plan

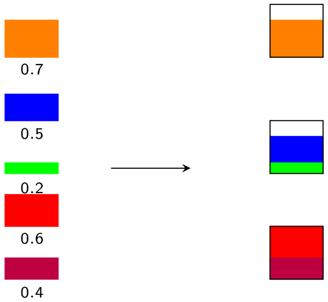
1. Introduction
2. Results
3. Vector Bin Packing
4. Summary
5. Vector Scheduling
6. Vector Bin Covering

Packing Problems

- Input: n items with sizes $a_1, a_2, \ldots, a_n \in [0, 1]$.
- Goal: Pack the items into bins efficiently.

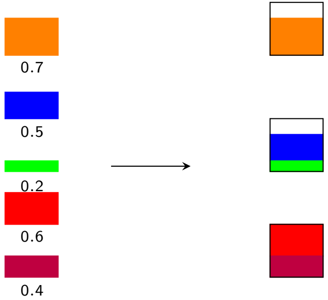
Packing Problems

- Input: n items with sizes $a_1, a_2, \ldots, a_n \in [0, 1]$.
- Goal: Pack the items into bins efficiently.
- Bin Packing: Each bin has capacity 1, Minimize the number of bins.



Packing Problems

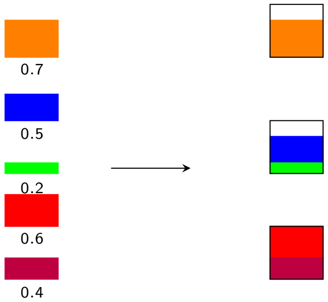
- Input: n items with sizes $a_1, a_2, \ldots, a_n \in [0, 1]$.
- Goal: Pack the items into bins efficiently.
- Bin Packing: Each bin has capacity 1, Minimize the number of bins.



- Multiprocessor Scheduling: Fixed number of bins, Minimize the maximum load.

Packing Problems

- Input: n items with sizes $a_1, a_2, \ldots, a_n \in [0, 1]$.
- Goal: Pack the items into bins efficiently.
- Bin Packing: Each bin has capacity 1, Minimize the number of bins.



- Multiprocessor Scheduling: Fixed number of bins, Minimize the maximum load.
- Bin Covering: Maximize the number of bins, each bin should get at least 1 load.

Approximation Algorithms

- The problems are NP-Hard.

Approximation Algorithms

- The problems are NP-Hard.
- Approximation algorithms: Our goal is to output a solution with cost at most α times the optimal value.

Approximation Algorithms

- The problems are NP-Hard.
- Approximation algorithms: Our goal is to output a solution with cost at most α times the optimal value.
- All of them have a Polynomial Time Approximation Scheme (PTAS): $(1+\epsilon)$-factor approximation algorithm running in time $\text{poly}(n, \frac{1}{\epsilon})$.

Approximation Algorithms

- The problems are NP-Hard.
- Approximation algorithms: Our goal is to output a solution with cost at most α times the optimal value.
- All of them have a Polynomial Time Approximation Scheme (PTAS): $(1+\epsilon)$-factor approximation algorithm running in time poly$(n, \frac{1}{\epsilon})$.
- Asymptotic approximation for Bin Packing: We study the setting when the optimal value is large enough.

Multidimensional Packing Problems

- Multidimensional problems: each job is a vector in $[0, 1]^d$–Vector Bin Packing, Vector Scheduling, Vector Bin Covering.

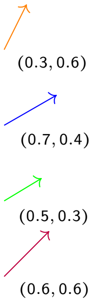
Multidimensional Packing Problems

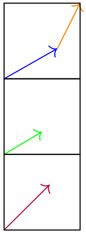
- Multidimensional problems: each job is a vector in $[0,1]^d$–Vector Bin Packing, Vector Scheduling, Vector Bin Covering.
- Motivation:
  1. Fundamental problems in theory, well studied in the approximation algorithms community.
  2. Most of the scheduling problems in practice are multidimensional.

Vector Bin Packing

Vector Bin Packing

- Input: n d-dimensional vectors $v_1, v_2, \ldots, v_n \in [0,1]^d$.
- Assign the jobs to minimum number of machines such that in each machine, the sum is at most 1 in every coordinate.





Vector Bin Packing

- $d = 1$: Bin Packing.

Vector Bin Packing

- $d = 1$: Bin Packing.
- $d = 2$: No PTAS (Woeginger, IPL 1997; Ray, 2021)

Vector Bin Packing

- $d = 1$: Bin Packing.
- $d = 2$: No PTAS (Woeginger, IPL 1997; Ray, 2021)
- $O(d)$-factor approximation algorithm. (De La Vega, Lueker, Combinatorica 1981)

Vector Bin Packing

- $d = 1$: Bin Packing.
- $d = 2$: No PTAS (Woeginger, IPL 1997; Ray, 2021)
- $O(d)$-factor approximation algorithm. (De La Vega, Lueker, Combinatorica 1981)
- When d is part of the input, essentially tight : $\Omega(d^{1-\epsilon})$ NP-hardness (Chekuri, Khanna, SICOMP 2004)

Vector Bin Packing: Fixed dimension

- When d is fixed, the algorithms can now run in $n^{f(d)}$ time, for some function f.

Vector Bin Packing: Fixed dimension

- When d is fixed, the algorithms can now run in $n^{f(d)}$ time, for some function f.
- $O(\ln d)$ factor algorithm (Chekuri, Khanna, SICOMP 2004)
- Improved to $\ln d + O(1)$ by (Bansal, Caprara, Sviridenko, SICOMP 2009) and (Bansal, Eliáš, Khan, SODA 2016)

Vector Bin Packing: Fixed dimension

- When d is fixed, the algorithms can now run in $n^{f(d)}$ time, for some function f.
- $O(\ln d)$ factor algorithm (Chekuri, Khanna, SICOMP 2004)
- Improved to $\ln d + O(1)$ by (Bansal, Caprara, Sviridenko, SICOMP 2009) and (Bansal, Eliáš, Khan, SODA 2016)
- Best Hardness: No PTAS for $d = 2$.

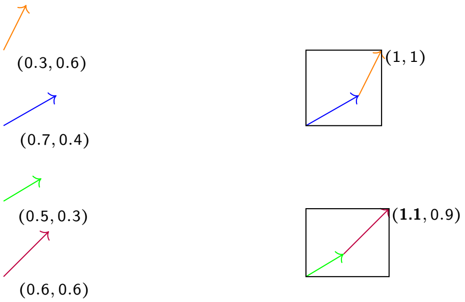
Vector Bin Packing: Fixed dimension

- When d is fixed, the algorithms can now run in $n^{f(d)}$ time, for some function f.
- $O(\ln d)$ factor algorithm (Chekuri, Khanna, SICOMP 2004)
- Improved to $\ln d + O(1)$ by (Bansal, Caprara, Sviridenko, SICOMP 2009) and (Bansal, Eliáš, Khan, SODA 2016)
- Best Hardness: No PTAS for $d = 2$.
- Question: Is there a constant factor approximation for Vector Bin Packing when the dimension is fixed?

Vector Scheduling

Vector Scheduling

- Input is $v_1, v_2, \ldots, v_n \in [0,1]^d$ and number of machines m.
- Find $f : [n] \to [m]$. Load vector on a machine $j \in [m] : \sum_{i \in [n]: f(i)=j} v_i$.
- Objective: Minimize the maximum ℓ_∞ norm of the load vectors.



Vector Scheduling

- $d = 1$: Multiprocessor Scheduling.

Vector Scheduling

- $d = 1$: Multiprocessor Scheduling.
- PTAS for fixed d. (Chekuri, Khanna, SICOMP 2004)
- When d is part of the input, $O\left(\frac{\log d}{\log \log d}\right)$-factor algorithms. (Harris, Srinivasan, JACM 2019) and (Im, Kell, Kulkarni, Panigrahi, SICOMP 2019).

Vector Scheduling

- $d = 1$: Multiprocessor Scheduling.
- PTAS for fixed d. (Chekuri, Khanna, SICOMP 2004)
- When d is part of the input, $O\left(\frac{\log d}{\log \log d}\right)$-factor algorithms. (Harris, Srinivasan, JACM 2019) and (Im, Kell, Kulkarni, Panigrahi, SICOMP 2019).
- Hardness: No constant factor approximation algorithm. (Chekuri, Khanna, SICOMP 2004)

Vector Bin Covering

Vector Bin Covering

- Input: n d-dimensional vectors $v_1, v_2, \ldots, v_n \in [0,1]^d$.
- Objective: Partition the vectors into the maximum number of parts such that in each part, the sum of the vectors is at least 1 in every coordinate.

Vector Bin Covering

Vector Bin Covering

- Input: n d-dimensional vectors $v_1, v_2, \ldots, v_n \in [0,1]^d$.
- Objective: Partition the vectors into the maximum number of parts such that in each part, the sum of the vectors is at least 1 in every coordinate.

- $d = 1$: Bin Covering.

Vector Bin Covering

Vector Bin Covering

- Input: n d-dimensional vectors $v_1, v_2, \ldots, v_n \in [0,1]^d$.
- Objective: Partition the vectors into the maximum number of parts such that in each part, the sum of the vectors is at least 1 in every coordinate.

- $d = 1$: Bin Covering.
- General d: $O(\log d)$ factor approximation algorithm by (Alon et al., Algorithmica 1998).

Vector Bin Covering

Vector Bin Covering

- Input: n d-dimensional vectors $v_1, v_2, \dots, v_n \in [0,1]^d$.
- Objective: Partition the vectors into the maximum number of parts such that in each part, the sum of the vectors is at least 1 in every coordinate.

- $d = 1$: Bin Covering.
- General d: $O(\log d)$ factor approximation algorithm by (Alon et al., Algorithmica 1998).
- Hardness: APX-Hard when $d = 2$. (Ray, 2021)

Plan

Our results

Vector Bin Packing

Assuming $P \neq NP$, when d is fixed, Vector Bin Packing is hard to approximate within $\Omega(\log d)$ factor.

Our results

Vector Bin Packing

Assuming $P \neq NP$, when d is fixed, Vector Bin Packing is hard to approximate within $\Omega(\log d)$ factor.

Vector Scheduling

Assuming NP has no quasipolynomial time algorithms, Vector Scheduling has no $\Omega((\log d)^{1-\epsilon})$ factor polynomial time algorithms for all $\epsilon > 0$.

Our results

Vector Bin Packing

Assuming $P \neq NP$, when d is fixed, Vector Bin Packing is hard to approximate within $\Omega(\log d)$ factor.

Vector Scheduling

Assuming NP has no quasipolynomial time algorithms, Vector Scheduling has no $\Omega((\log d)^{1-\epsilon})$ factor polynomial time algorithms for all $\epsilon > 0$.

Vector Bin Covering

Assuming $P \neq NP$, Vector Bin Packing has no polynomial time algorithm with approximation factor $\Omega\left(\frac{\log d}{\log \log d}\right)$.

Our results

Problem	Subcase	Best Algorithm	Best Hardness
VBP	$d = 1$	PTAS	NP-Hard
	Fixed d	$\ln d + O(1)$	$\Omega(\log d)$
	Arbitrary d	$1 + \epsilon d + O\left(\ln \frac{1}{\epsilon}\right)$	$d^{1-\epsilon}$
VS	$d = 1$	PTAS	NP-Hard
	Fixed d	PTAS	NP-Hard
	Arbitrary d	$O\left(\frac{\log d}{\log \log d}\right)$	$\Omega\left((\log d)^{1-\epsilon}\right)$ $\left(\mathrm{NP} \not\subseteq \mathrm{ZPTIME}\left(n^{(\log n)^{O(1)}}\right)\right)$
VBC	$d = 1$	FPTAS	NP-Hard
	Arbitrary d	$O(\log d)$	$\Omega\left(\frac{\log d}{\log \log d}\right)$

Plan

Our results

Vector Bin Packing

Assuming $P \neq NP$, when d is fixed, Vector Bin Packing is hard to approximate within $\Omega(\log d)$ factor.

Vector Scheduling

Assuming NP has no quasipolynomial time algorithms, Vector Scheduling has no $\Omega((\log d)^{1-\epsilon})$ factor polynomial time algorithms for all $\epsilon > 0$.

Vector Bin Covering

Assuming $P \neq NP$, Vector Bin Packing has no polynomial time algorithm with approximation factor $\Omega\left(\frac{\log d}{\log \log d}\right)$.

Vector Bin Packing

- Recall the problem statement: Input is $v_1, v_2, \ldots, v_n \in [0,1]^d$. Objective: Partition them into minimum number of parts such that in each part, the sum of the vectors is at most 1 in every coordinate.

Vector Bin Packing

- Recall the problem statement: Input is $v_1, v_2, \ldots, v_n \in [0,1]^d$. Objective: Partition them into minimum number of parts such that in each part, the sum of the vectors is at most 1 in every coordinate.
- Configuration: A subset $\{i_1, i_2, \ldots, i_k\} \subseteq [n]$ is called a configuration if

$$\left\| v_{i_1} + v_{i_2} + \ldots + v_{i_k} \right\|_\infty \leq 1$$

- Objective is to use the minimum number of configurations to cover $[n]$.

Vector Bin Packing

- Recall the problem statement: Input is $v_1, v_2, \ldots, v_n \in [0,1]^d$. Objective: Partition them into minimum number of parts such that in each part, the sum of the vectors is at most 1 in every coordinate.
- Configuration: A subset $\{i_1, i_2, \ldots, i_k\} \subseteq [n]$ is called a configuration if

$$\|v_{i_1} + v_{i_2} + \ldots + v_{i_k}\|_\infty \leq 1$$

- Objective is to use the minimum number of configurations to cover $[n]$.
- Using the minimum number of sets from a given family to cover all the elements: Set Cover Problem.

$$\boxed{\text{Vector Bin Packing}} \longrightarrow \boxed{\text{Set Cover}}$$

Reversing the Reduction

Reversing the Reduction

Reversing the Reduction

Which Set Cover instances can be formulated as d-dimensional Vector Bin Packing instances?

Reversing the Reduction

Reversing the Reduction

Which Set Cover instances can be formulated as d-dimensional Vector Bin Packing instances?

- Given a set family $\mathcal{F} \subseteq 2^{[n]}$, goal is to find vectors $v_1, v_2, \ldots, v_n \in [0,1]^d$ such that for every set $S \subseteq [n]$, $S \in \mathcal{F}$ if and only if

$$\left\| \sum_{i \in S} v_i \right\|_\infty \leq 1$$

Packing Dimension

- Packing Dimension of $\mathcal{F} \subseteq 2^{[n]}$ pdim($\mathcal{F}$) is the smallest positive integer d such that there exist vectors $v_1, v_2, \ldots, v_n \in [0,1]^d$ with

$$\left\| \sum_{i \in S} v_i \right\|_\infty \leq 1 \text{ if and only if } S \in \mathcal{F}$$

- If no such d exists, $\text{pdim}(\mathcal{F}) = \infty$.

Packing Dimension

- Packing Dimension of $\mathcal{F} \subseteq 2^{[n]}$ $\mathrm{pdim}(\mathcal{F})$ is the smallest positive integer d such that there exist vectors $v_1, v_2, \ldots, v_n \in [0,1]^d$ with

$$\left\| \sum_{i \in S} v_i \right\|_\infty \leq 1 \text{ if and only if } S \in \mathcal{F}$$

- If no such d exists, $\mathrm{pdim}(\mathcal{F}) = \infty$.
- $\mathrm{pdim}(\mathcal{F})$ being finite requires two simple conditions:

Packing Dimension

- Packing Dimension of $\mathcal{F} \subseteq 2^{[n]}$ pdim$(\mathcal{F})$ is the smallest positive integer d such that there exist vectors $v_1, v_2, \ldots, v_n \in [0,1]^d$ with

$$\left\| \sum_{i \in S} v_i \right\|_{\infty} \leq 1 \text{ if and only if } S \in \mathcal{F}$$

- If no such d exists, pdim$(\mathcal{F}) = \infty$.
- pdim$(\mathcal{F})$ being finite requires two simple conditions:
  - $\mathcal{F}$ is downward-closed i.e., if $S \in \mathcal{F}$, then $T \in \mathcal{F}$ for all $T \subseteq S$.

Packing Dimension

- Packing Dimension of $\mathcal{F} \subseteq 2^{[n]}$ $\mathrm{pdim}(\mathcal{F})$ is the smallest positive integer d such that there exist vectors $v_1, v_2, \ldots, v_n \in [0,1]^d$ with

$$\left\| \sum_{i \in S} v_i \right\|_\infty \leq 1 \text{ if and only if } S \in \mathcal{F}$$

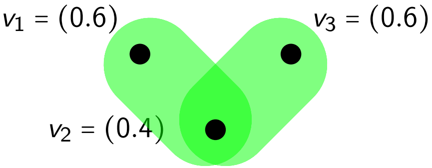
- If no such d exists, $\mathrm{pdim}(\mathcal{F}) = \infty$.
- $\mathrm{pdim}(\mathcal{F})$ being finite requires two simple conditions:
  - $\mathcal{F}$ is downward-closed i.e., if $S \in \mathcal{F}$, then $T \in \mathcal{F}$ for all $T \subseteq S$.
  - No isolated elements: For every $i \in [n]$, there exists $S \in \mathcal{F} : i \in S$.

Packing Dimension

- Packing Dimension of $\mathcal{F} \subseteq 2^{[n]}$ $\mathrm{pdim}(\mathcal{F})$ is the smallest positive integer d such that there exist vectors $v_1, v_2, \ldots, v_n \in [0,1]^d$ with

$$\left\| \sum_{i \in S} v_i \right\|_\infty \leq 1 \text{ if and only if } S \in \mathcal{F}$$

- If no such d exists, $\mathrm{pdim}(\mathcal{F}) = \infty$.
- $\mathrm{pdim}(\mathcal{F})$ being finite requires two simple conditions:
  - $\mathcal{F}$ is downward-closed i.e., if $S \in \mathcal{F}$, then $T \in \mathcal{F}$ for all $T \subseteq S$.
  - No isolated elements: For every $i \in [n]$, there exists $S \in \mathcal{F} : i \in S$.
- These two conditions are sufficient.



$\mathcal{F} = \{\phi, \{1\}, \{2\}, \{3\}, \{1,2\}, \{2,3\}\}$

Hardness of VBP

- Observation: If $\mathrm{pdim}(\mathcal{F}) \leq d$, the Set Cover problem on $\mathcal{F}$ is a d-dimensional Vector Bin Packing problem.

Hardness of VBP

- Observation: If $\mathrm{pdim}(\mathcal{F}) \leq d$, the Set Cover problem on $\mathcal{F}$ is a d-dimensional Vector Bin Packing problem.

Packing Dimension of $\mathcal{F}$ is at most d

Set Cover is Hard to approximate within $f(d)$ on $\mathcal{F}$

$\downarrow$

VBP is Hard to approximate within $f(d)$

Choosing the Set Family

- Set Cover: $\Omega(\ln n)$ hardness. (Feige, JACM 1998)

Choosing the Set Family

- Set Cover: $\Omega(\ln n)$ hardness. (Feige, JACM 1998)
- Packing Dimension of these hard instances grows with n ☹

Choosing the Set Family

- Set Cover: $\Omega(\ln n)$ hardness. (Feige, JACM 1998)
- Packing Dimension of these hard instances grows with n ☹
- Question: Are there structured set families that have small Packing Dimension but the Set Cover problem is hard on them?

Choosing the Set Family

- Set Cover: $\Omega(\ln n)$ hardness. (Feige, JACM 1998)
- Packing Dimension of these hard instances grows with n ☹
- Question: Are there structured set families that have small Packing Dimension but the Set Cover problem is hard on them?
- Answer: Simple Bounded Set Families. ☺

Simple Bounded Set Families

- Simple: Any two sets intersect in at most one element.

Simple Bounded Set Families

- Simple: Any two sets intersect in at most one element.
- (k, Δ)-Bounded: Each set has cardinality at most k, and each element appears in at most Δ sets.

Simple Bounded Set Families

- Simple: Any two sets intersect in at most one element.
- (k, Δ)-Bounded: Each set has cardinality at most k, and each element appears in at most Δ sets.

Set Cover on Simple Bounded Set Families

Set Cover is hard to approximate within $\Omega(\ln k)$ factor on simple (k, Δ)-bounded set families with $\Delta = k^{O(1)}$, when k is a large constant.

- Proof: Essentially (Anil Kumar, Arya, Ramesh, ICALP 2000), start with a modified Label Cover instance.

Packing Dimension of Simple Bounded set families

- What's left: Proving that Simple Bounded set families have small Packing Dimension.

Packing Dimension of Simple Bounded set families

- What's left: Proving that Simple Bounded set families have small Packing Dimension.
- Small caveat: It's actually the downward-closure $\mathcal{F}^{\downarrow}$ of Simple Bounded set family $\mathcal{F}$ that has a small Packing Dimension.

$$\mathcal{F}^{\downarrow} = \{T : \exists S \in \mathcal{F}, T \subseteq S\}$$

Packing Dimension of Simple Bounded set families

- What's left: Proving that Simple Bounded set families have small Packing Dimension.
- Small caveat: It's actually the downward-closure $\mathcal{F}^{\downarrow}$ of Simple Bounded set family $\mathcal{F}$ that has a small Packing Dimension.

$$\mathcal{F}^{\downarrow} = \{T : \exists S \in \mathcal{F}, T \subseteq S\}$$

Packing Dimension of Simple Bounded Set Families

Suppose that $\mathcal{F}$ is a simple (k, Δ)-bounded set family with no isolated elements. Then,

$$\mathrm{pdim}(\mathcal{F}^{\downarrow}) \leq (k\Delta)^{O(1)}$$

A property of Packing Dimension

Sub-additivity of Packing Dimension

$\mathrm{pdim}(\mathcal{F}_1 \cap \mathcal{F}_2)$ is at most $\mathrm{pdim}(\mathcal{F}_1) + \mathrm{pdim}(\mathcal{F}_2)$.

A property of Packing Dimension

Sub-additivity of Packing Dimension

$\mathrm{pdim}(\mathcal{F}_1 \cap \mathcal{F}_2)$ is at most $\mathrm{pdim}(\mathcal{F}_1) + \mathrm{pdim}(\mathcal{F}_2)$.

- Proof: $f_1 : [n] \to [0,1]^{d_1}$ such that for every $S \subseteq [n]$, $S \in \mathcal{F}_1$ if and only if

$$\left\| \sum_{i \in S} f_1(i) \right\|_\infty \leq 1$$

Similarly, $f_2 : [n] \to [0,1]^{d_2}$.

A property of Packing Dimension

Sub-additivity of Packing Dimension

$\mathrm{pdim}(\mathcal{F}_1 \cap \mathcal{F}_2)$ is at most $\mathrm{pdim}(\mathcal{F}_1) + \mathrm{pdim}(\mathcal{F}_2)$.

- Proof: $f_1 : [n] \to [0,1]^{d_1}$ such that for every $S \subseteq [n]$, $S \in \mathcal{F}_1$ if and only if

$$\left\| \sum_{i \in S} f_1(i) \right\|_\infty \leq 1$$

  Similarly, $f_2 : [n] \to [0,1]^{d_2}$.
- Define $f : [n] \to [0,1]^{d_1+d_2}$ as $(f_1(i), f_2(i))$. For every $S \subseteq [n]$,

$$\left\| \sum_{i \in S} f(i) \right\|_\infty \leq 1$$

  if and only if $S \in \mathcal{F}_1$ and $S \in \mathcal{F}_2$.

Sunflower Bouquets

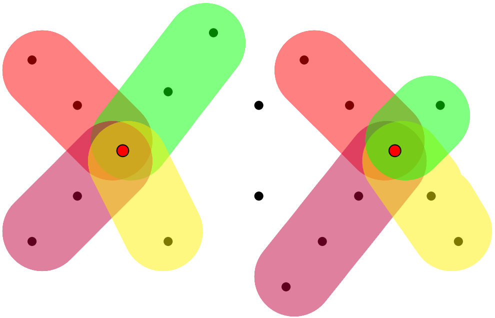
- Writing a simple bounded set family as an intersection of small number of structured set families with small Packing Dimension?

Sunflower Bouquets

- Writing a simple bounded set family as an intersection of small number of structured set families with small Packing Dimension?
- Yes, using Sunflower-Bouquets.

(k, Δ)-Sunflower Bouquet with core U

1. Every set $S \in \mathcal{F}$ intersects with U exactly once.
2. Intersection of any two sets is either empty or is in U.
3. Each set has cardinality at most k, each element appears in at most Δ sets.



Embedding of Sunflower Bouquets

Embedding of Sunflower Bouquets (Main Technical Lemma)

Suppose that $\mathcal{F} \subseteq 2^{[n]}$ is a (k, Δ)-Sunflower with core U. Then, there exists an embedding $f : [n] \to [0,1]^d$ with $d = \text{poly}(k, \Delta)$ such that

1. For every set $S \in \mathcal{F}$, $\|\sum_{i \in S} f(i)\|_\infty \le 1$.

Embedding of Sunflower Bouquets

Embedding of Sunflower Bouquets (Main Technical Lemma)

Suppose that $\mathcal{F} \subseteq 2^{[n]}$ is a (k, Δ)-Sunflower with core U. Then, there exists an embedding $f : [n] \to [0,1]^d$ with $d = \mathrm{poly}(k, \Delta)$ such that

1. For every set $S \in \mathcal{F}$, $\|\sum_{i \in S} f(i)\|_\infty \leq 1$.
2. For every set $S \notin \mathcal{F}^{\downarrow}$ with $S \cap U \neq \phi$, $\|\sum_{i \in S} f(i)\|_\infty > 1$.

Embedding of Sunflower Bouquets

Embedding of Sunflower Bouquets (Main Technical Lemma)

Suppose that $\mathcal{F} \subseteq 2^{[n]}$ is a (k, Δ)-Sunflower with core U. Then, there exists an embedding $f : [n] \to [0,1]^d$ with $d = \text{poly}(k, \Delta)$ such that

1. For every set $S \in \mathcal{F}$, $\|\sum_{i \in S} f(i)\|_\infty \leq 1$.
2. For every set $S \notin \mathcal{F}^{\downarrow}$ with $S \cap U \neq \phi$, $\|\sum_{i \in S} f(i)\|_\infty > 1$.
3. For every set S with $S \cap U \neq \phi$ and $|S| \leq k$, $\|\sum_{i \in S} f(i)\|_\infty \leq 1$.

Embedding of Sunflower Bouquets

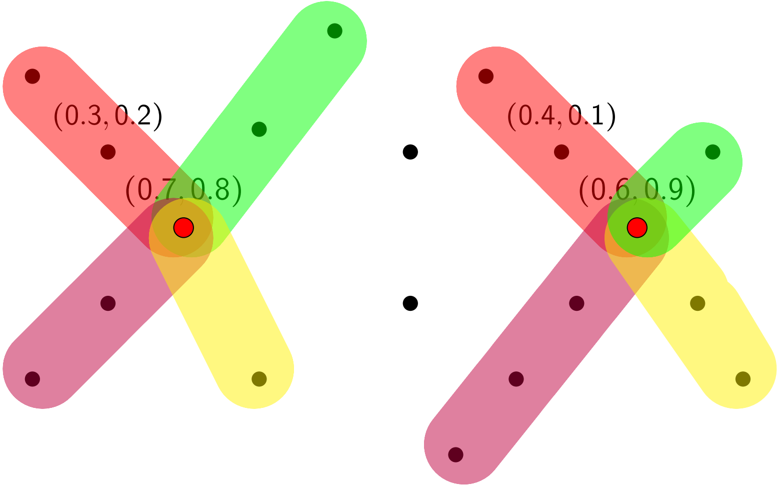
Embedding of Sunflower Bouquets (Main Technical Lemma)

Suppose that $\mathcal{F} \subseteq 2^{[n]}$ is a (k, Δ)-Sunflower with core U. Then, there exists an embedding $f : [n] \to [0,1]^d$ with $d = \text{poly}(k, \Delta)$ such that

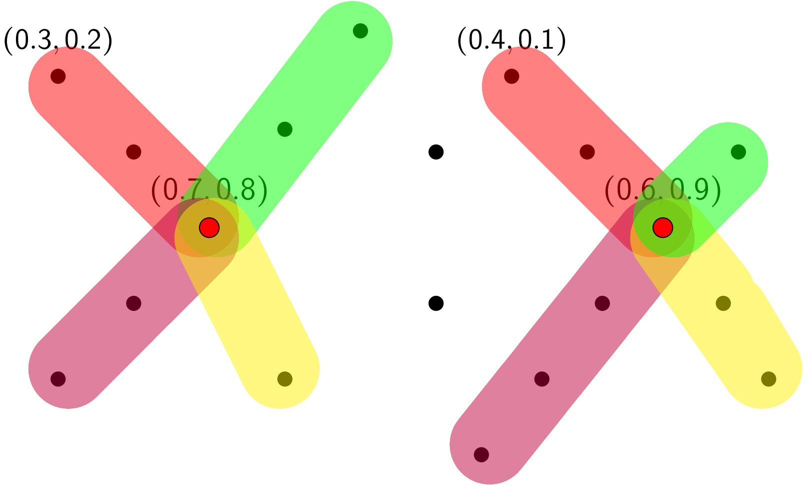
1. For every set $S \in \mathcal{F}$, $\|\sum_{i \in S} f(i)\|_\infty \leq 1$.
2. For every set $S \notin \mathcal{F}^\downarrow$ with $S \cap U \neq \phi$, $\|\sum_{i \in S} f(i)\|_\infty > 1$.
3. For every set S with $S \cap U \neq \phi$ and $|S| \leq k$, $\|\sum_{i \in S} f(i)\|_\infty \leq 1$.

- High level idea: many embeddings, each satisfying 1. and 3.
  1. Eliminating "inter-sunflower" sets.
  2. Pinning the "intra-sunflower" sets.

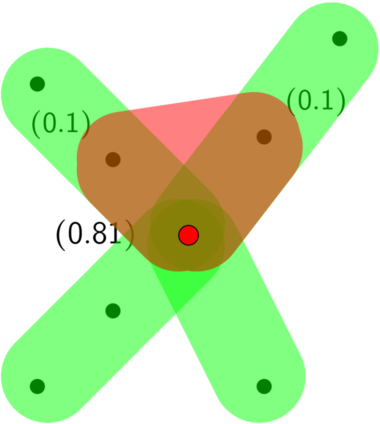
Inter-Sunflower sets



Inter-Sunflower sets



Intra-Sunflower sets



- For every minimal set $S \notin \mathcal{F}^{\downarrow}$ with $S \cap U \neq \phi$, we create a coordinate such that the sum is greater than 1.
- There are only $O((k\Delta)^2)$ such minimal sets in a sunflower.

Packing Dimension of Simple Bounded Set Families

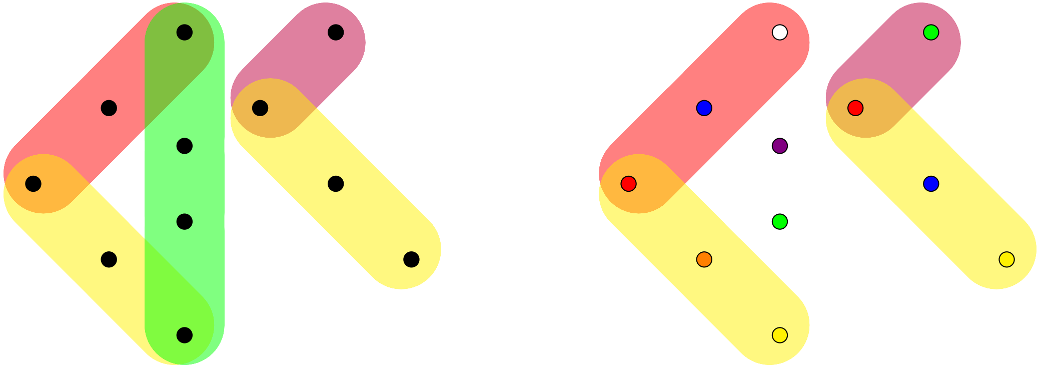
- $\mathcal{F} \subseteq 2^{[n]}$: Simple (k, Δ)-Bounded set family.

Packing Dimension of Simple Bounded Set Families

- $\mathcal{F} \subseteq 2^{[n]}$: Simple (k, Δ)-Bounded set family.
- Color $[n]$ with L colors such that
  1. All the elements in an edge are assigned distinct colors.
  2. If two edges intersect, all their elements are assigned distinct colors.
- Note: $L = O\left((k\Delta)^2\right)$ suffices.

Packing Dimension of Simple Bounded Set Families

- $\mathcal{F} \subseteq 2^{[n]}$: Simple (k, Δ)-Bounded set family.
- Color $[n]$ with L colors such that
  1. All the elements in an edge are assigned distinct colors.
  2. If two edges intersect, all their elements are assigned distinct colors.
- Note: $L = O\left((k\Delta)^2\right)$ suffices.
- $\mathcal{F}_i \subseteq \mathcal{F}$: all the sets that hit the ith color. $\mathcal{F}_i$ is a Sunflower-Bouquet!



Packing Dimension of Simple Bounded Set Families

- Use the Embedding of Sunflower-Bouquets for every $\mathcal{F}_i$, $f_i : [n] \to [0,1]^d$ with $d = \text{poly}(k, \Delta)$ such that
  1. For every set $S \in \mathcal{F}_i$, $\|\sum_{i \in S} f(i)\|_\infty \leq 1$.
  2. For every set $S \notin \mathcal{F}_i^\downarrow$ with $S \cap U_i \neq \phi$, $\|\sum_{i \in S} f(i)\|_\infty > 1$.
  3. For every set S with $S \cap U_i \neq \phi$ and $|S| \leq k$, $\|\sum_{i \in S} f(i)\|_\infty \leq 1$.

Packing Dimension of Simple Bounded Set Families

- Use the Embedding of Sunflower-Bouquets for every $\mathcal{F}_i$, $f_i : [n] \to [0,1]^d$ with $d = \text{poly}(k, \Delta)$ such that
  1. For every set $S \in \mathcal{F}_i$, $\|\sum_{i \in S} f(i)\|_\infty \leq 1$.
  2. For every set $S \notin \mathcal{F}_i^{\downarrow}$ with $S \cap U_i \neq \phi$, $\|\sum_{i \in S} f(i)\|_\infty > 1$.
  3. For every set S with $S \cap U_i \neq \phi$ and $|S| \leq k$, $\|\sum_{i \in S} f(i)\|_\infty \leq 1$.
- Final embedding $f = (f_1, f_2, \dots, f_L)$
  1. For every set $S \in \mathcal{F}$, $\|\sum_{i \in S} f(i)\|_\infty \leq 1$.
  2. For every set $S \notin \mathcal{F}^{\downarrow}$, $\|\sum_{i \in S} f(i)\|_\infty > 1$.
- Completes the proof that $\text{pdim}(\mathcal{F})$ is at most $\text{poly}(k, \Delta)$.

Hardness of Vector Bin Packing

- The Packing Dimension of Simple (k, Δ)-Bounded set families is at most $\text{poly}(k, \Delta)$.
- Set Cover on Simple (k, Δ)-Bounded set families is hard to approximate within $\Omega(\log k)$ when $\Delta = k^{O(1)}$ and k is a large constant.

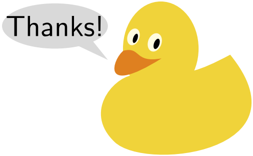
Hardness of Vector Bin Packing

- The Packing Dimension of Simple (k, Δ)-Bounded set families is at most poly(k, Δ).
- Set Cover on Simple (k, Δ)-Bounded set families is hard to approximate within $\Omega(\log k)$ when $\Delta = k^{O(1)}$ and k is a large constant.
- Both together prove that d-dimensional Vector Bin Packing is hard to approximate within $\Omega(\log d)$ factor when d is a large constant.

Plan

Summary

- Hardness of Multidimensional Packing Problems: Vector Bin Packing, Vector Scheduling, Vector Bin Covering.
- Vector Bin Packing: Packing Dimension.
- Open Problems: Other applications of Packing Dimension? Hardness of Geometric Bin Packing? Better hardness for 2-dimensional Bin Packing?



Plan

Our results

Vector Bin Packing

Assuming $P \neq NP$, when d is fixed, Vector Bin Packing is hard to approximate within $\Omega(\log d)$ factor.

Vector Scheduling

Assuming NP has no quasipolynomial time algorithms, Vector Scheduling has no $\Omega((\log d)^{1-\epsilon})$ factor polynomial time algorithms for all $\epsilon > 0$.

Vector Bin Covering

Assuming $P \neq NP$, Vector Bin Packing has no polynomial time algorithm with approximation factor $\Omega\left(\frac{\log d}{\log \log d}\right)$.

Vector Scheduling

- Input: $v_1, v_2, \ldots, v_n \in [0,1]^d$, and m, the number of machines.
- Find $f : [n] \to [m]$. Load vector on a machine $j \in [m] : \sum_{i \in [n] : f(i) = j} v_i$.
- Objective: Minimize the maximum ℓ_∞ norm of the load vectors.

Monochromatic Clique

- Monochromatic Clique(k, B): Given a graph $G = ([n], E)$, and parameters $k := k(n), B := B(n)$, the goal is to distinguish between
  1. G is k-colorable.
  2. In any assignment of k-colors to the vertices of G, there is a monochromatic clique of size B.

Monochromatic Clique

- Monochromatic Clique(k, B): Given a graph $G = ([n], E)$, and parameters $k := k(n), B := B(n)$, the goal is to distinguish between
  1. G is k-colorable.
  2. In any assignment of k-colors to the vertices of G, there is a monochromatic clique of size B.
- $B = 2$: Standard k-coloring of graphs.

Monochromatic Clique

- Monochromatic Clique(k, B): Given a graph $G = ([n], E)$, and parameters $k := k(n), B := B(n)$, the goal is to distinguish between
  1. G is k-colorable.
  2. In any assignment of k-colors to the vertices of G, there is a monochromatic clique of size B.
- $B = 2$: Standard k-coloring of graphs.
- Problem gets easier as B increases.
- When $B = \sqrt{n}$, the problem can be solved in polynomial time.

Hardness of Vector Scheduling from Monochromatic Clique

- Suppose that Monochromatic Clique(k, B) is hard.

Hardness of Vector Scheduling from Monochromatic Clique

- Suppose that Monochromatic Clique(k, B) is hard.
- Given a graph $G = ([n], E)$, order all the cliques of size at most B of G as $T_1, T_2, \ldots, T_d$ with $d \leq n^B$.

Hardness of Vector Scheduling from Monochromatic Clique

- Suppose that Monochromatic Clique(k, B) is hard.
- Given a graph $G = ([n], E)$, order all the cliques of size at most B of G as $T_1, T_2, \ldots, T_d$ with $d \leq n^B$.
- Define $\mathcal{V} = \{v_1, v_2, \ldots, v_n\} \subseteq \{0, 1\}^d$ as

$$(v_i)_j = \begin{cases} 1 & \text{if } i \in T_j \\ 0 & \text{otherwise.} \end{cases}$$

  The number of machines is equal to k.

Hardness of Vector Scheduling from Monochromatic Clique

- Suppose that Monochromatic Clique(k, B) is hard.
- Given a graph $G = ([n], E)$, order all the cliques of size at most B of G as $T_1, T_2, \ldots, T_d$ with $d \leq n^B$.
- Define $\mathcal{V} = \{v_1, v_2, \ldots, v_n\} \subseteq \{0,1\}^d$ as

$$(v_i)_j = \begin{cases} 1 & \text{if } i \in T_j \\ 0 & \text{otherwise.} \end{cases}$$

  The number of machines is equal to k.
- Completeness: If G is k-colorable, there is an assignment with maximum ℓ_∞ value 1.

Hardness of Vector Scheduling from Monochromatic Clique

- Suppose that Monochromatic Clique(k, B) is hard.
- Given a graph $G = ([n], E)$, order all the cliques of size at most B of G as $T_1, T_2, \dots, T_d$ with $d \leq n^B$.
- Define $\mathcal{V} = \{v_1, v_2, \dots, v_n\} \subseteq \{0,1\}^d$ as

$$(v_i)_j = \begin{cases} 1 & \text{if } i \in T_j \\ 0 & \text{otherwise.} \end{cases}$$

  The number of machines is equal to k.
- Completeness: If G is k-colorable, there is an assignment with maximum ℓ_∞ value 1.
- Soundness: If in any assignment of k-colors to the vertices of G there exists a clique of size B, the load is at least B in any scheduling.

Hardness of Monochromatic Clique

- For every constant B, there exists $k := k(n)$ such that Monochromatic Clique(k, B) is NP-Hard. (Chekuri, Khanna, SICOMP 2004)

Hardness of Monochromatic Clique

- For every constant B, there exists $k := k(n)$ such that Monochromatic Clique(k, B) is NP-Hard. (Chekuri, Khanna, SICOMP 2004)
- Question: Can we prove improved hardness of Monochromatic Clique (k, B) when B is larger?

Hardness of Monochromatic Clique

- For every constant B, there exists $k := k(n)$ such that Monochromatic Clique(k, B) is NP-Hard. (Chekuri, Khanna, SICOMP 2004)
- Question: Can we prove improved hardness of Monochromatic Clique (k, B) when B is larger?
- Answer: Yes, using Lexicographic graph product based amplification.

Hardness of Vector Scheduling

Graph Coloring Hardness of Approximation (Khot, FOCS 2001)

↓ Elementary Ramsey Theoretic Argument

Strong Monochromatic Clique Hardness

↓ Amplification using Lexicographic Product

Monochromatic Clique Hardness

↓ Above reduction

Vector Scheduling Hardness of Approximation

Strong Monochromatic Clique

- Strong Monochromatic Clique(k, B, C): A generalization of Monochromatic Clique: Given a graph $G = ([n], E)$, parameters $k := k(n), B := B(n), C$, the goal is to distinguish between
    1. G is k-colorable.
    2. In any k^C-coloring of G, there is a monochromatic clique of size B.
- $C = 1$: Monochromatic Clique(k, B).

Strong Monochromatic Clique

- Strong Monochromatic Clique(k, B, C): A generalization of Monochromatic Clique: Given a graph $G = ([n], E)$, parameters $k := k(n), B := B(n), C$, the goal is to distinguish between
  1. G is k-colorable.
  2. In any k^C-coloring of G, there is a monochromatic clique of size B.
- $C = 1$: Monochromatic Clique(k, B).

Hardness of Strong Monochromatic Clique

Assuming NP has no quasipolynomial time algorithm, there exist constant $\gamma > 0$ and $k := k(n)$ such that Strong Monochromatic Clique $(k, (\log n)^{\gamma}, C)$ has no quasipolynomial time algorithm for all integers $C \geq 1$.

Lexicographic Product

- $G = G_1 \times G_2$. Vertex set of G is $V_1 \times V_2$.
- Two vertices (u_1, u_2) and (v_1, v_2) are adjacent in G if
  - (u_1, v_1) are adjacent in G_1, or
  - $u_1 = v_1$, and (u_2, v_2) are adjacent in G_2

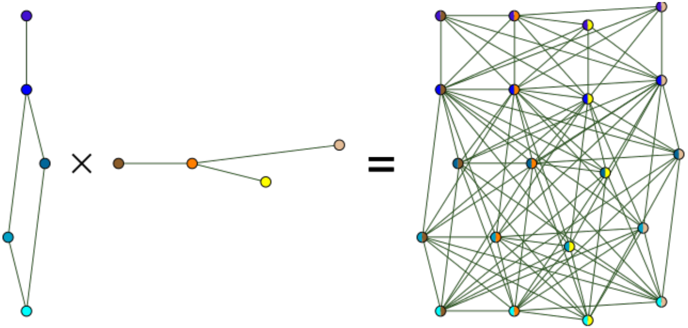


Image Credit: Wikipedia, Author=David Eppstein

Hardness Amplication using Lexicographic Product

- Reduction from Strong Monochromatic Clique (k, B, C) to Monochromatic Clique (k^C, B^C).

Hardness Amplication using Lexicographic Product

- Reduction from Strong Monochromatic Clique (k, B, C) to Monochromatic Clique (k^C, B^C).
- Let $G^2 = G \times G$.
- Completeness: If $\chi(G) \le k$, then $\chi(G^2) \le k^2$. If $c : V(G) \to [k]$ is a proper k-coloring of G, simply assign $(c(u_1), c(u_2))$ to (u_1, u_2).

Hardness Amplication using Lexicographic Product

- Reduction from Strong Monochromatic Clique (k, B, C) to Monochromatic Clique (k^C, B^C).
- Let $G^2 = G \times G$.
- Completeness: If $\chi(G) \le k$, then $\chi(G^2) \le k^2$. If $c : V(G) \to [k]$ is a proper k-coloring of G, simply assign $(c(u_1), c(u_2))$ to (u_1, u_2).
- Soundness: Suppose that in any assignment of k^2 colors to $V(G)$, there is a monochromatic clique of size B.
- Consider an assignment $c : V(G^2) \to [k^2]$.
    - For every $u \in V(G)$, there is a clique of size B $(u, v_1), (u, v_2), \dots, (u, v_B)$ in G^2 that are all assigned the same color $c'(u)$.

Hardness Amplication using Lexicographic Product

- Reduction from Strong Monochromatic Clique (k, B, C) to Monochromatic Clique (k^C, B^C).
- Let $G^2 = G \times G$.
- Completeness: If $\chi(G) \leq k$, then $\chi(G^2) \leq k^2$. If $c : V(G) \to [k]$ is a proper k-coloring of G, simply assign $(c(u_1), c(u_2))$ to (u_1, u_2).
- Soundness: Suppose that in any assignment of k^2 colors to $V(G)$, there is a monochromatic clique of size B.
- Consider an assignment $c : V(G^2) \to [k^2]$.
  - For every $u \in V(G)$, there is a clique of size B $(u, v_1), (u, v_2), \ldots, (u, v_B)$ in G^2 that are all assigned the same color $c'(u)$.
  - c' itself is a coloring of $V(G)$, there is a clique of size B, $u_1, u_2, \ldots, u_B$ that have the same c' value.

Hardness Amplification Using Lexicographic Product

- Let $G' = G^C$.
    1. Completeness: If $\chi(G) \leq k$, then $\chi(G') \leq k^C$.
    2. Soundness: If in any assignment of k^C colors to $V(G)$, there is a monochromatic clique of size B, then in any assignment of k^C colors to $V(G')$, there is a monochromatic clique of size B^C.

Hardness Amplification Using Lexicographic Product

- Let $G' = G^C$.
    1. Completeness: If $\chi(G) \leq k$, then $\chi(G') \leq k^C$.
    2. Soundness: If in any assignment of k^C colors to $V(G)$, there is a monochromatic clique of size B, then in any assignment of k^C colors to $V(G')$, there is a monochromatic clique of size B^C.
- Polynomial time reduction from Strong Monochromatic Clique (k, B, C) to Monochromatic Clique (k^C, B^C).

Hardness Amplification Using Lexicographic Product

- Let $G' = G^C$.
  1. Completeness: If $\chi(G) \le k$, then $\chi(G') \le k^C$.
  2. Soundness: If in any assignment of k^C colors to $V(G)$, there is a monochromatic clique of size B, then in any assignment of k^C colors to $V(G')$, there is a monochromatic clique of size B^C.
- Polynomial time reduction from Strong Monochromatic Clique (k, B, C) to Monochromatic Clique (k^C, B^C).

Hardness of Monochromatic Clique

Assuming NP has no quasipolynomial time algorithm, for every C, there exists $k := k(n)$ such that Monochromatic Clique $(k, (\log n)^C)$ has no quasipolynomial time algorithm.

Hardness Amplification Using Lexicographic Product

- Let $G' = G^C$.
  1. Completeness: If $\chi(G) \le k$, then $\chi(G') \le k^C$.
  2. Soundness: If in any assignment of k^C colors to $V(G)$, there is a monochromatic clique of size B, then in any assignment of k^C colors to $V(G')$, there is a monochromatic clique of size B^C.
- Polynomial time reduction from Strong Monochromatic Clique (k, B, C) to Monochromatic Clique (k^C, B^C).

Hardness of Monochromatic Clique

Assuming NP has no quasipolynomial time algorithm, for every C, there exists $k := k(n)$ such that Monochromatic Clique $(k, (\log n)^C)$ has no quasipolynomial time algorithm.

Hardness of Vector Scheduling

Assuming NP has no quasipolynomial time algorithm, Vector Scheduling has no polynomial time algorithm with approximation ratio $\Omega((\log d)^{1-\epsilon})$ for all $\epsilon > 0$.

Our results

Vector Bin Packing

Assuming $P \neq NP$, when d is fixed, Vector Bin Packing is hard to approximate within $\Omega(\log d)$ factor.

Vector Scheduling

Assuming NP has no quasipolynomial time algorithms, Vector Scheduling has no $\Omega((\log d)^{1-\epsilon})$ factor polynomial time algorithms for all $\epsilon > 0$.

Vector Bin Covering

Assuming $P \neq NP$, Vector Bin Packing has no polynomial time algorithm with approximation factor $\Omega\left(\frac{\log d}{\log \log d}\right)$.

Plan

Vector Bin Covering

- Input: n vectors $v_1, v_2, \ldots, v_n \in [0,1]^d$. Objective: Partition them into the maximum number of parts such that in each part, the sum is at least 1 in every coordinate.

Vector Bin Covering

- Input: n vectors $v_1, v_2, \ldots, v_n \in [0,1]^d$. Objective: Partition them into the maximum number of parts such that in each part, the sum is at least 1 in every coordinate.
- Hard instances: $\{0,1\}^d$.

Vector Bin Covering

- Input: n vectors $v_1, v_2, \ldots, v_n \in [0,1]^d$. Objective: Partition them into the maximum number of parts such that in each part, the sum is at least 1 in every coordinate.
- Hard instances: $\{0,1\}^d$.
- View each vector as a vertex of a hypergraph and each coordinate as an edge of the hypergraph.

Vector Bin Covering

- Input: n vectors $v_1, v_2, \ldots, v_n \in [0,1]^d$. Objective: Partition them into the maximum number of parts such that in each part, the sum is at least 1 in every coordinate.
- Hard instances: $\{0,1\}^d$.
- View each vector as a vertex of a hypergraph and each coordinate as an edge of the hypergraph.
- Given a hypergraph, objective is to partition the vertex set into maximum number of parts such that every part hits each edge.

Vector Bin Covering

- Input: n vectors $v_1, v_2, \ldots, v_n \in [0,1]^d$. Objective: Partition them into the maximum number of parts such that in each part, the sum is at least 1 in every coordinate.
- Hard instances: $\{0,1\}^d$.
- View each vector as a vertex of a hypergraph and each coordinate as an edge of the hypergraph.
- Given a hypergraph, objective is to partition the vertex set into maximum number of parts such that every part hits each edge.
- Such a coloring of hypergraphs: Rainbow Coloring.

Hypergraph Rainbow Coloring

- Hypergraph $H = (V, E)$, find $c : V \to [k]$ such that

$$\bigcup_{v \in e} c(v) = [k] \quad \forall e \in E$$

Hypergraph Rainbow Coloring

- Hypergraph $H = (V, E)$, find $c : V \to [k]$ such that

$$\bigcup_{v \in e} c(v) = [k] \quad \forall e \in E$$

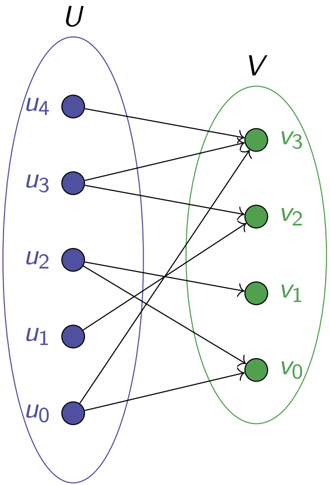
- When $k = 2$, same as 2-coloring of hypergraphs. NP-Hard.

Approximate Rainbow Coloring Hardness

Given a hypergraph H with m edges, it is NP-Hard to distinguish between

1. H is 2-colorable.
2. H cannot be rainbow colored with $\Omega\left(\frac{\log m}{\log \log m}\right)$ colors.

Hardness of approximate Rainbow Coloring



Label Cover

In the Label Cover problem, the input is a bipartite graph $U \cup V, E$ with projection constraints $\Pi_e : \Sigma \to \Sigma$ on each edge.

- The objective is to assign labels from Σ to the vertices to satisfy as many constraints as possible.
- Very hard to approximate: NP-Hard to find a labeling satisfying ϵ fraction of the constraints on fully satisfiable instances.
- "Mother of most optimal inapproximability results".

Approximate Rainbow Coloring Hardness

- Gadget reduction based on Label Cover-Long Code framework.

Approximate Rainbow Coloring Hardness

- Gadget reduction based on Label Cover-Long Code framework.
- Difference from the previous results: we now use very weak Label Cover hardness.
- We just use NP-Hardness of Label Cover, so alphabet size is $O(1)$.