Beyond Exact Fairness: Envy-Free Incomplete Connected Fair Division

Ajaykrishnan E S
University of California, Santa Barbara

Joint work with Daniel Lokshtanov

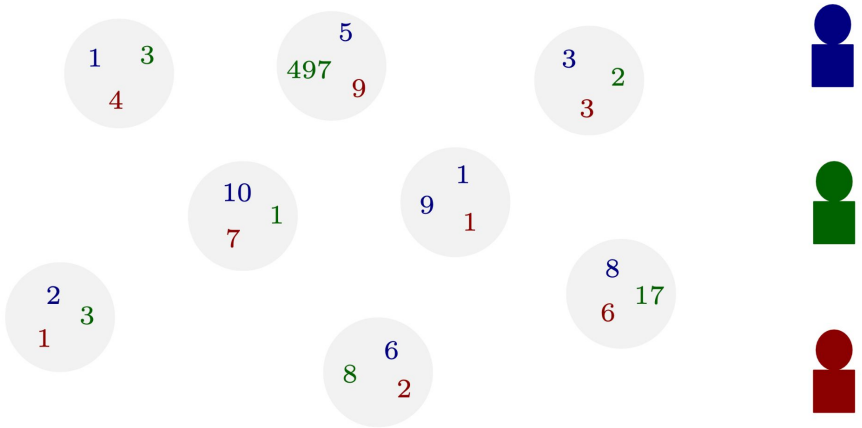
Envy-Free Fair Division

Input

[m] items

[n] agents

A utility function u_i per agent

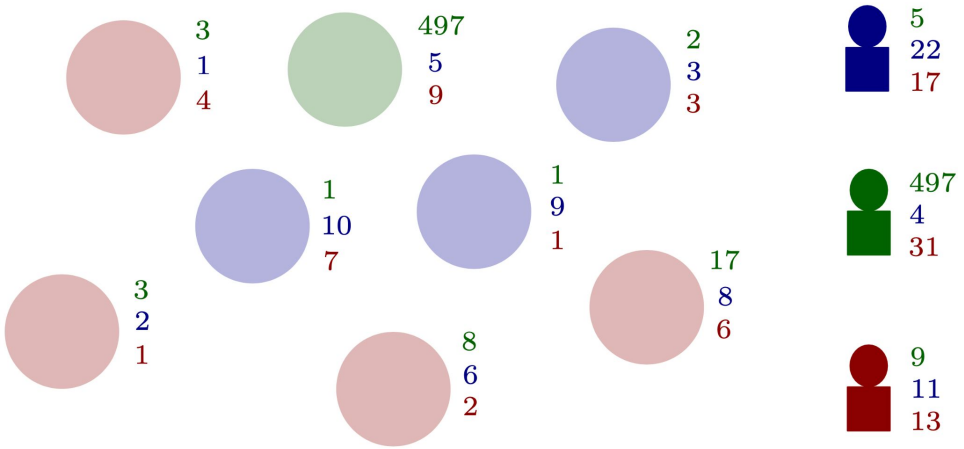


Envy-Free Fair Division

Envy-Free Allocation : $\mathbf{\Pi} = (\boldsymbol{\pi}_1 \boldsymbol{\pi}_2 \ldots \boldsymbol{\pi}_n)$

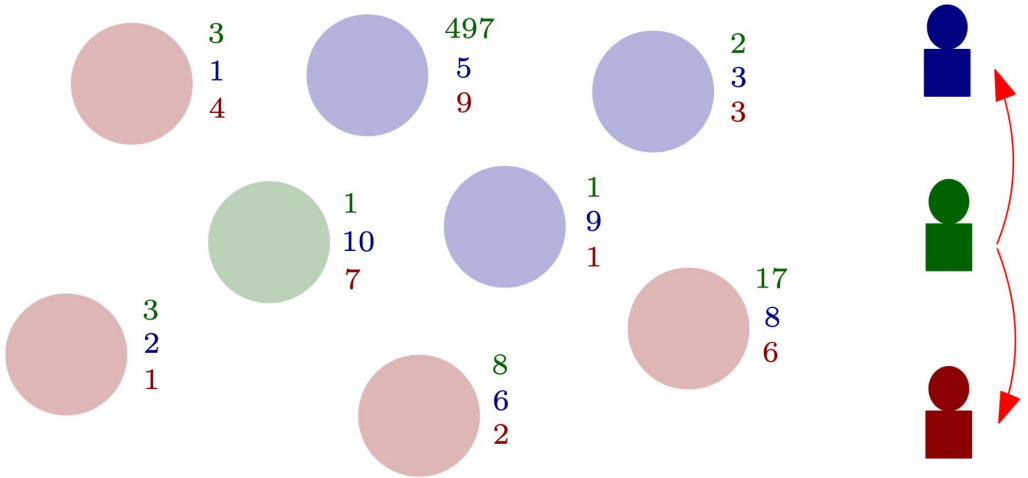
Partition the set of items

Each agent thinks they got the best part



Envy-Free Fair Division

Non-Example



Literature

Envy-Free Fair Division models tons of real life situations! Particularly, any task that entails allocating responsibilities or rewards. Eg: Inheritance.

Fair division of indivisible goods: Recent progress and open questions.

G. Amanatidis, H. Aziz, G. Birmpas, A. Filos-Ratsikas, B. Li, H. Moulin, A. A Voudouris, and X. Wu.

Envy-Free Incomplete Connected Fair Division

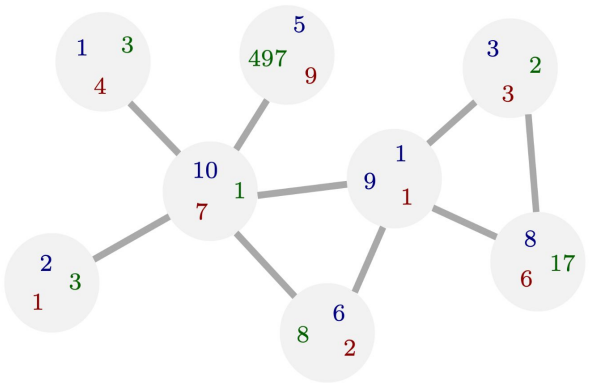
Input

A graph on m vertices

An integer p

[n] agents

A utility function u_i per agent



Envy-Free Incomplete Connected Fair Division

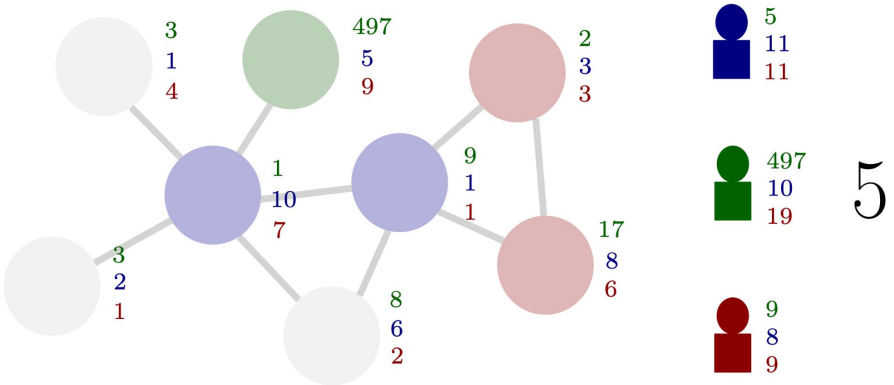
Valid and Envy-Free Allocation : $\mathbf{\Pi} = (\boldsymbol{\pi}_1 \boldsymbol{\pi}_2 \ldots \boldsymbol{\pi}_n)$

Assign disjoint subsets of items

Each agent thinks they got the best part

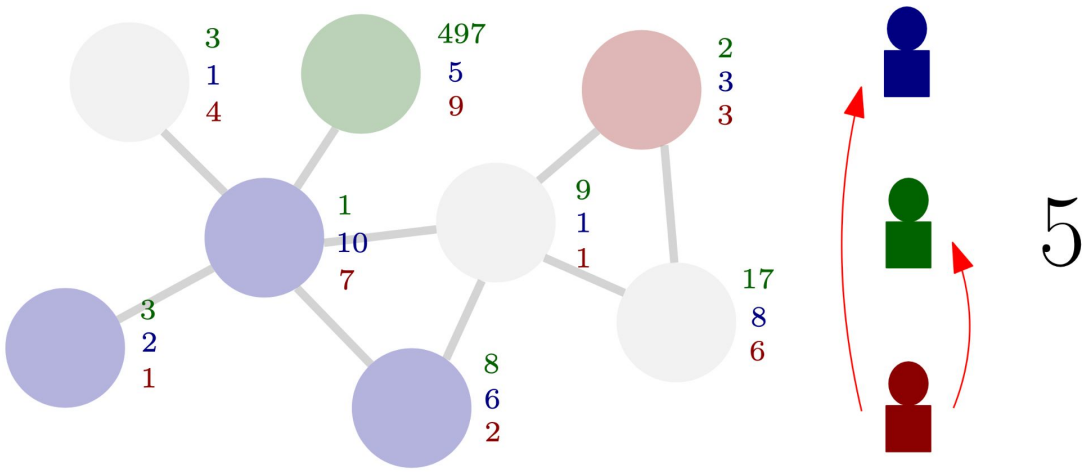
Every agent gets a connected part

p items are allocated in total



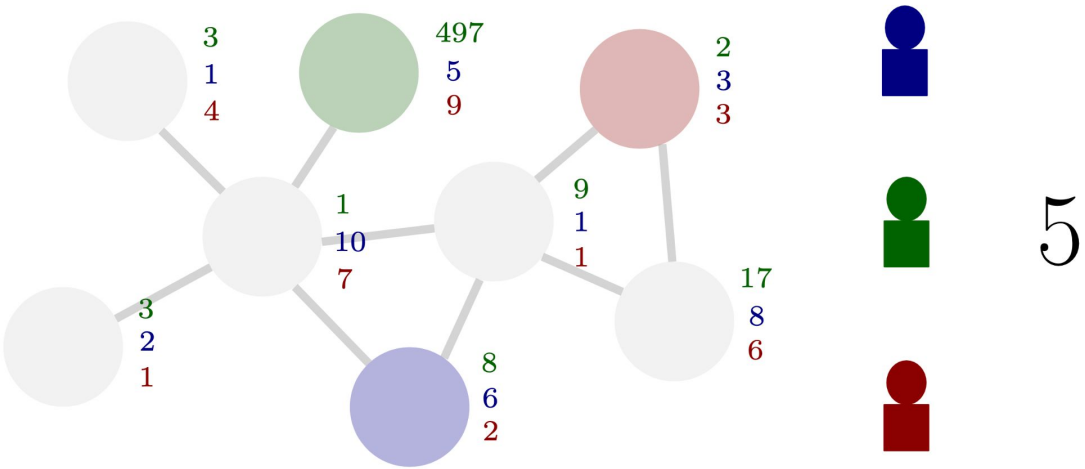
Envy-Free Incomplete Connected Fair Division

Non-Example 1



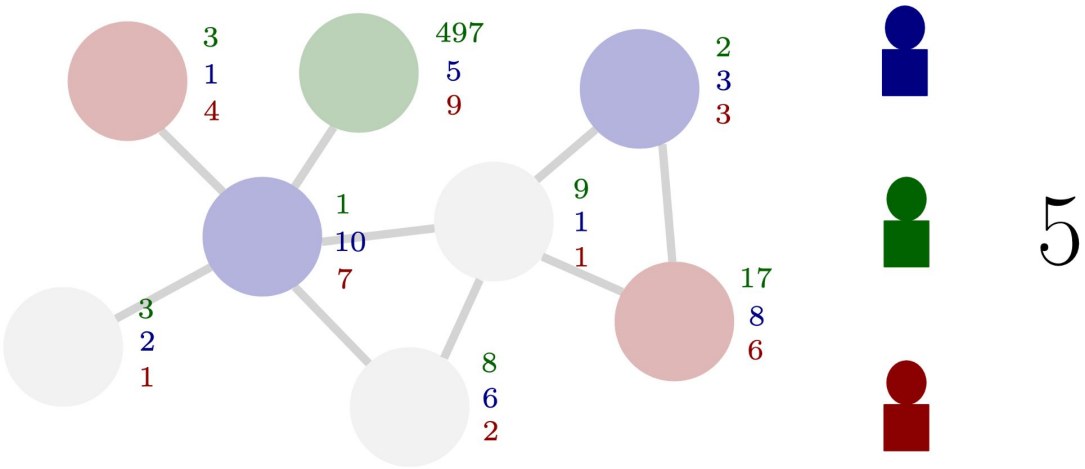
Envy-Free Incomplete Connected Fair Division

Non-Example 2



Envy-Free Incomplete Connected Fair Division

Non-Example 3



Literature

Real-life has way more complicated scenarios.

For example, connectivity requirement: maybe the inheritance involves dividing a plot of land and it makes less sense to give someone a lot of disconnected pieces!

Fair division of a graph

S. Bouveret, K. Cechlárová,
E. Elkind,
A. Igarashi, and D. Peters.

Literature

Real-life has way more complicated scenarios.

For example, incompleteness: maybe the inheritance involves dividing only a small subplot of land and donating the rest to charity!

Parameterized Complexity of Incomplete Connected Fair Division.

H. Gahlawat and M. Zehavi

Literature

Parameterized Complexity of Incomplete Connected Fair Division.

Results (for EF-ICFD) :

- W[1] Hardness parameterized by $p+|A|+vc(G)$ for binary input
- Exponential Kernal parameterized by $p+|A|+vc(G)+val$
- No Polynomial Kernal parameterized by $p+|A|+vc(G)+val$

Open Problem

Does there exist an algorithm that runs in FPT time parameterized by $p + |A|$, that solves EF-ICFD when the input numbers are provided in *unary* representation?

Results

W[1] hardness:

Parameterized by $p + |A|$

For Unary Input

In Star Graphs

Results

W[1] hardness:

Parameterized by $p + |A|$
For Unary Input
In Star Graphs

Parameterized Approximation:

Approximation - $(1+\varepsilon)$
Time - $f(p, \varepsilon, |\mathscr{A}|)$ poly(input)
For Binary Input
In General Graphs

Results

W[1] hardness:

Parameterized by $p + |A|$
For Unary Input
In Star Graphs

Parameterized Approximation:

Approximation - $(1+\varepsilon)$
Time - $f(p, \varepsilon, |\mathcal{A}|)$ poly(input)
For Binary Input
In General Graphs

W[1] - hardness

(k,M)-Vector Sum *(Multidimensional Subset Sum)*

Input:

Set (W) of d-dimensional integer vectors, with entries from [M]

Target vector (t),

Integer (k)

Question:

Does there exists a subset of W of size exactly k such that their sum equals the target vector t?

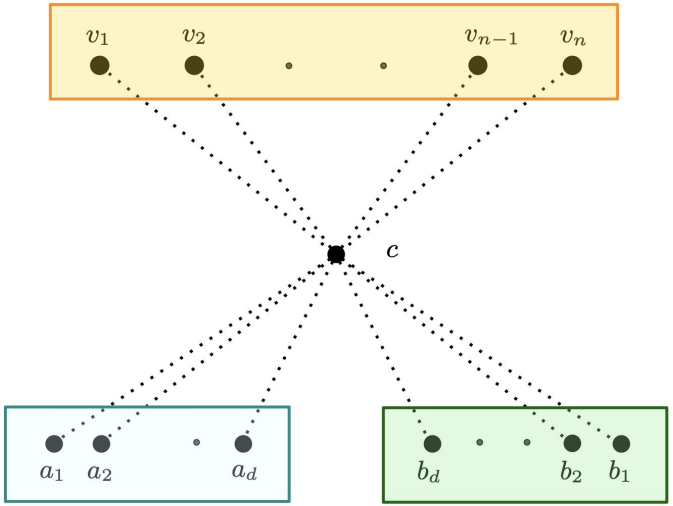
W[1] - hardness

(k,M)-Vector Sum is W[1]-hard parameterized by k + d, even when $M = k\, n^{1+o(1)}$

Losing Weight by Gaining Edges

A. Abboud, K. Lewi, and R. Williams

W[1] - hardness



d + d + 1 agents,

- A_1 A_2 ... A_d
- B_1 B_2 ... B_d
- C

p = d + d + (k + 1)

Results

W[1] hardness:

Parameterized by $p + |A|$
For Unary Input
In Star Graphs

Parameterized Approximation:

Approximation - $(1+\varepsilon)$
Time - $f(p, \varepsilon, |\mathcal{A}|)$ poly(input)
For Binary Input
In General Graphs

Results

W[1] hardness:

Parameterized by $p + |A|$
For Unary Input
In Star Graphs

Parameterized Approximation:

Approximation - $(1+\varepsilon)$
Time - $f(p, \varepsilon, |\mathcal{A}|)$ poly(input)
For Binary Input
In General Graphs

Algorithm

What does “(1+ ε)-approximation” or “ε-envy-free” mean?

Envy-Free = $u_i(\pi_i) \geq u_i(\pi_j)$ for every pair of agents i, j

ε-Envy-Free = $(1+\varepsilon)u_i(\pi_i) \geq u_i(\pi_j)$ for every pair of agents i, j

Algorithm

What can we directly do in FPT time?

1) Apply *color coding* to partition the graph into p parts
   (such that only one vertex is assigned from each part)

2) Guess which part is assigned to which agent

   The task now reduces to finding an allocation that respects these guesses

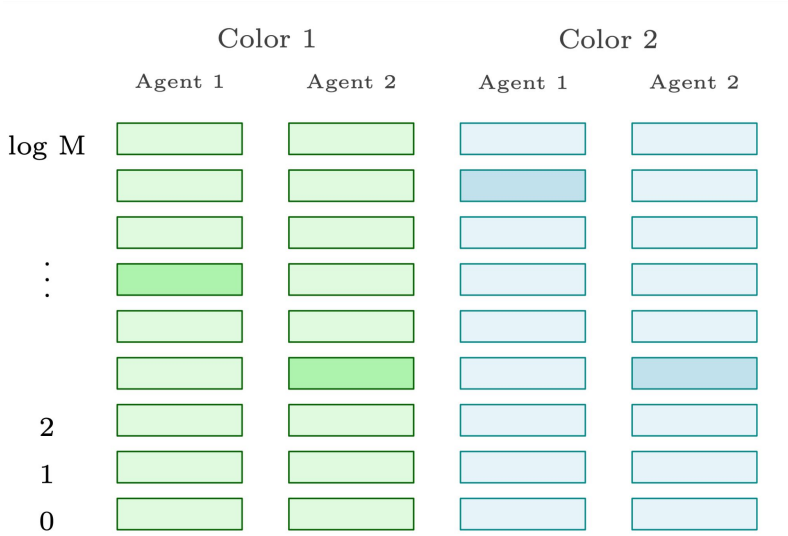
Algorithm

What can we directly do in FPT time?

Note that if we are allowed “unary” time i.e., if $(\log M)^{g(p,|A|)}$ is FPT;

Then we can *bucket* the values into powers of $(1+\varepsilon)$ and guess things.

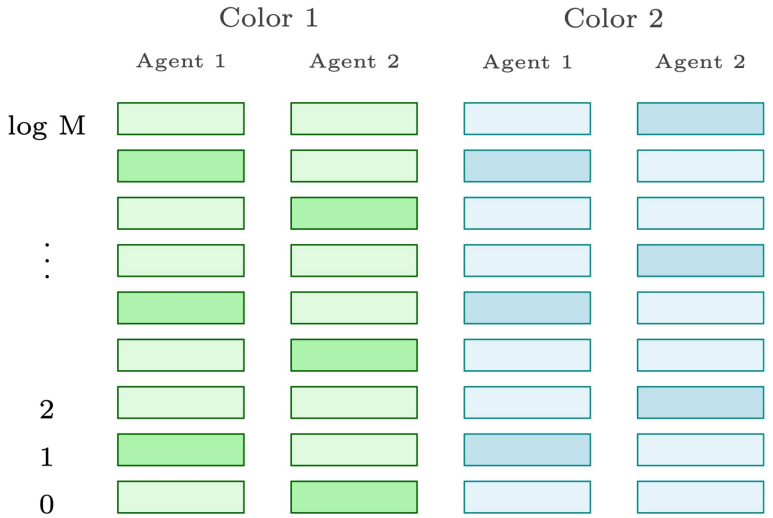
eg : The bucket of $u_i(v_c)$, for each color c and agent i.



Algorithm

Since we are only allowed “binary” time; we can only guess things, modulo some function of p, |A| and ε

How can we use this limited information?



Inspiration

We draw inspiration from the polynomial time algorithm for the "*house allocation*" problem

Envy-freeness in house allocation problems

J. Gan, W. Suksompong, and A. A Voudouris

Inspiration

To Illustrate a version of their idea, consider the following *simplification* of our current setup:

Each agent has a single unique color, and must be assigned one vertex of that color

Inspiration

Step 1: If there exists an agent such that none of their most-valued items have their color, then discard these items.

Step 2: If for every agent there exists at least one most-valued item with their color, then return the allocation that assigns each agent such an item.

Inspiration

Key Idea:

Either restrict the search space considerably

Or find an envy-free allocation

Algorithm

Our algorithm follows the same framework.

To encode the restriction of search space, we use a *target* vector.

$$\bar{\mu} = (\mu_1, \ldots, \mu_{|A|})$$

μ_i represents the bucket in which $u_i(\pi_i)$ belongs.

Note: This is NOT something we guess / iterate over.

Algorithm

We guess some features of the solution, which for every agent i and color c tells us how big the bucket corresponding to $u_i(v_c)$ is, when compared to $u_i(\pi_i)$

And try to find a valid and ε-envy-free allocation that respects these guesses, *assuming* that the bucket of $u_i(\pi_i)$ is μ_i

Algorithm

Either we succeed and return the solution we found

Or we fail and can point to a target that is set too high!
so we reduce that entry in the target vector, and iterate.

Results

W[1] hardness:

Parameterized by $p + |A|$
For Unary Input
In Star Graphs

Parameterized Approximation:

Approximation - $(1+\varepsilon)$
Time - $f(p, \varepsilon, |\mathcal{A}|)$ poly(input)
For Binary Input
In General Graphs

Thank You!