Fair Division of A Graph

Authors: S. Bouveret, K. Cechlarova, E. Elkind, A. Igarashi, D. Peters

arXiv:1705.10239

Era Sarda, mt1200801@iitd.ac.in

November 17, 2023

Prof. Rohit Vaish

Outline

Recap

- **Example Scenario:** department of Mathematics at University X that is about to move to a new building and need adjacent offices.
- **Fair division of indivisible items** under connectivity constraints : each agent's share has to be connected in this graph.
- **Undirected graph**: as a tool to capture the connectivity requirement.

Connected Fair Division

An instance of the CFD is a triple $I = (G, N, U)$, where

- $G = (V, E)$ is an undirected graph with m vertices.
- $N = \{1, 2, \ldots, n\}$ is a set of agents
- U is an n-tuple of utility functions $u_i : V \to \mathbb{R}_{\geq 0}$ where $\sum_{v \in V} u_i(v) = 1$ for each $i \in N$.

Framework for fair division

- proportionality
- envy-freeness+completeness
- maximin share

Along with connectivity constraints

Some Terms

- A problem is **slice-wise polynomial** (XP) with respect to a parameter k if each instance I of this problem can be solved in time where f is a computable function.
- A problem is **fixed parameter tractable** (FPT) with respect to a parameter k if each instance I of this problem can be solved in time f(k)poly($|I|$).
- The Agents are said to have the **same type** when they have the same preferences over all the items.

Main Results of the Paper

- Finding PROP-CFD allocations is NP-complete even for paths (reducible to X3C).

Main Results of the Paper

- Finding PROP-CFD allocations is NP-complete even for paths (reducible to X3C).
- PROP-CFD is solvable in polynomial time if G is a star.

Main Results of the Paper

- Finding PROP-CFD allocations is NP-complete even for paths (reducible to X3C).
- PROP-CFD is solvable in polynomial time if G is a star.
- Finding complete EF-CFD allocations is NP-complete for both paths and stars (reducible to X3C and Independent Set problems, respectively).

Main Results of the Paper

- Both PROP and complete EF allocations can be found efficiently when the graph is a path and agents can be classified into a small number of types (XP problem).

Main Results of the Paper

- Both PROP and complete EF allocations can be found efficiently when the graph is a path and agents can be classified into a small number of types (XP problem).
- A small number of player types plus a small number of players result in an efficient algorithm for finding proportional allocations on arbitrary trees.

Main Results of the Paper

- Both PROP and complete EF allocations can be found efficiently when the graph is a path and agents can be classified into a small number of types (XP problem).
- A small number of player types plus a small number of players result in an efficient algorithm for finding proportional allocations on arbitrary trees.
- An MMS (Maximal Minsum Share) allocation always exists if the underlying graph is a tree and can be computed efficiently in polynomial time.

Main Results of the Paper

- **Both PROP and complete EF allocations can be found efficiently when the graph is a path and agents can be classified into a small number of types (XP problem).**

Proportional Connected Fair Division

When G is a path and number of agent types is small.

PROP-CFD

Denote $k :=$ Max no. of connected components each agent can get in proportional allocation.
$\nu_0 :=$ Minimum valuation of each connected component an agent gets assigned.

Theorem

PROP-CFD is in XP with respect to the number of player types p if G is a path. Here k=1 and $\nu_0 = \frac{1}{n}$.

PROP-CFD for k=1

- For $i = 0, \ldots, m$, and $0 \leq j_l \leq n$ for each $l \in [p]$.

  Let $A_i[j_1, \ldots, j_p] =$
  $\begin{cases} 1 & \exists \text{ a valid partial allocation } \pi \text{ of } V_i \text{ with } j_l \text{ happy agents of type } l \\ 0 & \textit{otherwise} \end{cases}$

Note: $A_0[j_1, \ldots, j_p] = 1 \Leftrightarrow j_k = 0$ for all $k \in [p]$.

- For $i = 1, \ldots, m$,

$$A_i[j_1, \ldots, j_p] = 1 \Leftrightarrow$$

$\exists s < i$ and $t \in [p]$ such that $A_s[j_1, \ldots, j_t - 1, \ldots, j_p] = 1$ and $\text{val}(t, \{v_{s+1}, v_{s+2}, \ldots, v_i\}) \geq \frac{1}{n}$.

PROP-CFD for k=1

$$A_m[j_1, \ldots, j_p] = 1?$$

for $n_l \leq j_l \leq n$

PROP-CFD for k=1

$$A_m[j_1, \ldots, j_p] = 1?$$

for $n_l \leq j_l \leq n$

$$O(mp) \times (m+1)(n+1)^p$$

PROP-CFD for k=1

$$A_m[j_1, \ldots, j_p] = 1?$$

for $n_l \leq j_l \leq n$

$$O(mp) \times (m+1)(n+1)^p$$

XP w.r.t. p

PROP-CFD for k=1 Algorithm

Algorithm 1 checkV(t, S, n)

```
if val(t, S) ≥ 1/n then
    return true
else
    return false
end if
```

PROP-CFD for k=1 Algorithm

Algorithm 2 checkA($i, j_1, j_2, \ldots, j_p,\ n$)

```
if i == 0 then
    k = 1
    while k ≤ p do
        if j_k ≠ 0 then
            return false
        end if
        k++
    end while
    return true
end if
k = 1
while k < i do
    t = 1
    while t ≤ p do
        ans = (ans | {checkA(k, j_1, j_2, ..., j_{t-1}, j_t - 1, j_{t+1}, ..., j_p) & checkV(t, {v_{k+1}, ..., v_i}, n)})
        t++
    end while
    k++
end while
if ans == 1 then
    return true
else
    return false
end if
```

PROP-CFD for $k > 1$

Inspiration: Mathemtics Department at IIT Delhi has 2 connected components allocated for faculty offices in the campus. One is in the Main building other one is in Mechanical Department.

Initial intuition for the allocation: increase the number of players from n to kn, and run the same alogorithm.

Problem: It will give each connected components valuation greater than or equal to $\frac{1}{kn}$.

What if we want each connected components minimum valuation less than or greater than $\frac{1}{kn}$, i.e. ν_0?

PROP-CFD for $m > k > 1$ and $\frac{1}{n} > \nu_0 \geq 0$

Recursive function checkA(i,$j_1, j_2, ...j_p$,n):
Call checkA(m,$n_1, n_2, ...n_p$,n);

1. For l from 1 to i-1
2. S=$\{v_{l+1}, v_{l+2}, \ldots, v_i\}$
3. Check for each player type 1 to p, if there's a player which can be allocated the connected component S and our allocation is valid
    1. S can be allocated to the player of type t say j_t^{th} player, if $val(t, S) \geq \nu_0$ and the current no. of allocated components to that player is less than k and the player is not **satisfied**.
    2. If above is true, then need to check if S is allocated to the player, then will it be satisfied.
        1. If yes, then update $value(t, j_t)$, and check if $A_l[j_1, j_2, ...j_t - 1, ..j_p]$ is 1 or 0. If 1, then valid allocation, $perans = perans|true$. If 0, then invalid allocation, re-update $value(t, j_t)$.
        2. If no, then update $value(t, j_t)$, and check if $A_l[j_1, j_2, ...j_t, ..j_p]$ is 1 or 0. If 1, then valid allocation, $perans = perans|true$. If 0, then invalid allocation, re-update $value(t, j_t)$.
    3. If not true, then invalid allocation.

Contd.

- Above algorithm returns true if $A_0[j_1, j_2, \dots, j_p] = 0 \Leftrightarrow j_t = 0$ for $t = 1, 2, ..p$

Algorithm 3 k-PROP-CFD

```
int k[][] = 0 for all t = 1 to p and j_t = 1 to max{j_t; t = 1 to p};
int value[][] = 0 for all t = 1 to p and j_t = 1 to max{j_t; t = 1 to p};
checkA(m, j_1, j_2, ..., j_p);            ▷ m is the number of vertices in the path G
function CHECKA(i, j_1, j_2, ..., j_p)
    if i == 0 then
        l = 1
        while l ≤ p do
            if j_l ≠ 0 then
                return false
            end if
            l++
        end while
        return true
    end if
    perans = false
    l = 1
```

Algorithm: k-PROP-CFD (Part 2)

Algorithm 4 k-PROP-CFD

while $l < i$ **do**
    $t = 1$
    while $t \leq p$ **do**
        $S = \{v_{l+1}, \ldots, v_i\}$
        if $(t, S) \geq v_0$ **then**
            if $(t, S) \geq \frac{1}{n} - \text{value}(t, j_t)$ **then**
                $\text{value}(t, j_t) += (t, S)$; $\text{ans} = \text{checkA}(k, j_1, j_2, \ldots, j_t - 1, j_{t+1}, \ldots, j_p)$;
                if $\text{ans} == 1$ **then**
                    if $k(t, j_t) \geq k$ **then**
                        $\text{value}(t, j_t) -= (t, S)$;$\text{perans} = \text{perans} \mid \text{false}$;
                    else
                        $\text{perans} = \text{perans} \mid \text{true}$; $k(t) ++$;
                    end if
                else
                    $\text{value}(t, j_t) -= (t, S)$; $\text{perans} = \text{perans} \mid \text{false}$;
                end if

Algorithm: k-PROP-CFD (Part 3)

Algorithm 5 k-PROP-CFD

if . **then**..

  if . **then**..

  else

    value$(t, j_t)+ = (t, S)$; ans $=$ checkA$(k, j_1, j_2, \ldots, j_t, j_{t+1}, \ldots, j_p)$;

    if ans $== 1$ **then**

      if $k(t, j_t) \geq k$ **then**

        value$(t, j_t)- = (t, S)$; perans $=$ perans | false;

      else

        perans $=$ perans | true; $k(t) + +$;

      end if

    else

      value$(t, j_t)- = (t, S)$; perans $=$ perans | false ;

    end if

  end if

else

  perans $=$ perans | false

end if

$t + +$; **end while**; $l + +$; **end while;** return **perans;**

Conclusion

k-PROP-CFD is of order $O(mp) \times (m+1)(min[k, \frac{1}{\nu_0}] \times n + 1)^p$ in worst case

k-PROP-CFD with flexible ν_0 has not a very bad complexity compared to k=1, and has a benefit of increasing the possibilities of allocation of vertices.

Thank You for Listening!

Conclusion

Any Questions?