

On Fair Division With Binary Valuations Respecting Social Networks

Neeldhara Misra and Debanuj Nayak

IIT Gandhinagar

Workshop on Computation & Economics

10th September 2022

OUTLINE

OUTLINE

Background & definitions



OUTLINE

Background & definitions



Summary of results



OUTLINE

Background & definitions



Summary of results



NP-completeness for two agent types



OUTLINE

Background & definitions



Summary of results



NP-completeness for two agent types



Future work

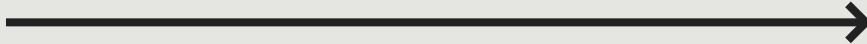


OUTLINE

Background & definitions



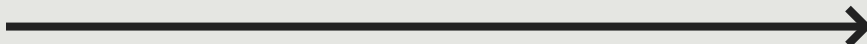
Summary of results



NP-completeness for two agent types



Future work



FAIR DIVISION

#1. Background & definitions

FAIR DIVISION

Task: Distribute m items among n agents.

#1. Background & definitions

FAIR DIVISION

Task: Distribute m items among n agents.  → 

#1. Background & definitions

FAIR DIVISION

Task: Distribute m items among n agents.

Backdrop: Agents have preferences over bundles of items.

FAIR DIVISION

Task: Distribute m items among n agents.  → 

Backdrop: Agents have preferences over bundles of items.

#1. Background & definitions

FAIR DIVISION

Task: Distribute m items among n agents. 🍰 → 😊

Backdrop: Agents have preferences over bundles of items.

#1. Background & definitions

FAIR DIVISION

Task: Distribute m items among n agents. 🚗 → 🙋

Backdrop: Agents have preferences over bundles of items.

#1. Background & definitions

FAIR DIVISION

Task: Distribute m items among n agents.

Backdrop: Agents have preferences over bundles of items.

Goal: Make everyone happy. 🙌

#1. Background & definitions

FAIR DIVISION

Task: Compute an *allocation* $\pi : A \rightarrow 2^R$

Backdrop: Agents have preferences over bundles of items.

Goal: Make everyone happy. 🙌

#1. Background & definitions

No sharing...

FAIR DIVISION

Task: Compute an *allocation* $\pi : A \rightarrow 2^R$

Backdrop: Agents have preferences over bundles of items.

Goal: Make everyone happy. 🙌

#1. Background & definitions

$$\pi(a_i) \cap \pi(a_j) = \emptyset$$

FAIR DIVISION

Task: Compute an *allocation* $\pi : A \rightarrow 2^R$

Backdrop: Agents have preferences over bundles of items.

Goal: Make everyone happy. 🙌🙌

#1. Background & definitions

Complete

FAIR DIVISION

Task: Compute an *allocation* $\pi : A \rightarrow 2^R$

Backdrop: Agents have preferences over bundles of items.

Goal: Make everyone happy. 🙌

#1. Background & definitions

$$\bigcup_{a \in A} \pi(a) = R$$

FAIR DIVISION

Task: Compute an *allocation* $\pi : A \rightarrow 2^R$

Backdrop: Agents have preferences over bundles of items.

Goal: Make everyone happy. 🙌🙌

#1. Background & definitions

FAIR DIVISION

Task: Compute an *allocation* $\pi : A \rightarrow 2^R$

Backdrop: Agents have preferences over bundles of items.

Goal: Make everyone happy. 🙌

#1. Background & definitions

FAIR DIVISION

Task: Compute an *allocation* $\pi : A \rightarrow 2^R$

Backdrop: $\nu_\ell : 2^R \rightarrow \mathbb{Z}$

Goal: Make everyone happy. 🙌

#1. Background & definitions

0/1 Valuations

FAIR DIVISION

Task: Compute an *allocation* $\pi : A \rightarrow 2^R$

Backdrop: $\nu_\ell : 2^R \rightarrow \{0, 1\}$

Goal: Make everyone happy. 🙌🙌

#1. Background & definitions

Additive Valuations

FAIR DIVISION

Task: Compute an *allocation* $\pi : A \rightarrow 2^R$

Backdrop: $\nu_\ell : 2^R \rightarrow \{0, 1\}$

Goal: Make everyone happy. 🙌🙌

#1. Background & definitions

For all subsets $S \subseteq R$, $\nu_\ell(S) = \sum_{o \in S} \nu_\ell(\{o\})$

.

FAIR DIVISION

Task: Compute an *allocation* $\pi : A \rightarrow 2^R$

Backdrop: $\nu_\ell : 2^R \rightarrow \{0, 1\}$

Goal: Make everyone happy. 🙌🙌

#1. Background & definitions

If 🙋 values the items 🍰 and 🌸 at 1 each, then she values {🍰, 🌸} at 2.

FAIR DIVISION

Task: Compute an *allocation* $\pi : A \rightarrow 2^R$

Backdrop: $\nu_\ell : 2^R \rightarrow \{0, 1\}$

Goal: Make everyone happy. 🙌

#1. Background & definitions

FAIR DIVISION

Task: Compute an *allocation* $\pi : A \rightarrow 2^R$

Backdrop: $\nu_\ell : 2^R \rightarrow \{0, 1\}$

Goal: Make everyone happy. 🙌🙌

#1. Background & definitions

FAIR DIVISION

Task: Compute an *allocation* $\pi : A \rightarrow 2^R$

Backdrop: $\nu_\ell : 2^R \rightarrow \{0, 1\}$

Goal: Make everyone happy. 🙌🙌

#1. Background & definitions

FAIR DIVISION

Task: Compute an *allocation* $\pi : A \rightarrow 2^R$

Backdrop: $\nu_\ell : 2^R \rightarrow \{0, 1\}$

Goal: **Make everyone happy.** 🤔

#1. Background & definitions



FAIR DIVISION

Task: Compute an *allocation* $\pi : A \rightarrow 2^R$

Backdrop: $\nu_\ell : 2^R \rightarrow \{0, 1\}$

Goal: **Make everyone happy.** 🤔

#1. Background & definitions

😊 → {🍰, 🍲}

😍 → {🍔, 🍟, ☕}

FAIR DIVISION

Task: Compute an *allocation* $\pi : A \rightarrow 2^R$

Backdrop: $\nu_\ell : 2^R \rightarrow \{0, 1\}$

Goal: **Make everyone happy.** 🤔

#1. Background & definitions

👁️ → {🍰, 🍲}

😍 → {🍔, 🍟, ☕}

FAIR DIVISION

Task: Compute an *allocation* $\pi : A \rightarrow 2^R$

Backdrop: $\nu_\ell : 2^R \rightarrow \{0, 1\}$

Goal: **Make everyone happy.** 🤔

#1. Background & definitions

😬 → {🍰, 🍲}

😍 → {🍔, 🍟, ☕}

FAIR DIVISION

Task: Compute an *allocation* $\pi : A \rightarrow 2^R$

Backdrop: $\nu_\ell : 2^R \rightarrow \{0, 1\}$

Goal: **Make everyone happy.** 🤔

#1. Background & definitions

😡 → {🍰, 🍲}

😍 → {🍔, 🍟, ☕}

FAIR DIVISION

Task: Compute an *allocation* $\pi : A \rightarrow 2^R$

Backdrop: $\nu_\ell : 2^R \rightarrow \{0, 1\}$

Goal: **Make everyone happy.** 🤔

#1. Background & definitions

😭 → {🍰, 🍲}

😍 → {🍔, 🍟, ☕}

FAIR DIVISION

Task: Compute an *allocation* $\pi : A \rightarrow 2^R$

Backdrop: $\nu_\ell : 2^R \rightarrow \{0, 1\}$

Goal: **Make everyone happy.** 🤔

#1. Background & definitions

😭 → {🍰, 🍲}

😍 → {🍔, 🍟, ☕}

FAIR DIVISION

Task: Compute an *allocation* $\pi : A \rightarrow 2^R$

Backdrop: $\nu_\ell : 2^R \rightarrow \{0, 1\}$

Goal: **Make everyone happy.** 🤔

Agent a **envies** b in π if $v_a(\pi(b)) > v_a(\pi(a))$

#1. Background & definitions

😭 → {🍰, 🍲}

😍 → {🍔, 🍟, ☕}

FAIR DIVISION

Task: Compute an *allocation* $\pi : A \rightarrow 2^R$

Backdrop: $\nu_\ell : 2^R \rightarrow \{0, 1\}$

Goal: Make everyone happy. 🤔

Agent a *envies* b in π if $v_a(\pi(b)) > v_a(\pi(a))$

#1. Background & definitions



FAIR DIVISION

Task: Compute an *allocation* $\pi : A \rightarrow 2^R$

Backdrop: $\nu_\ell : 2^R \rightarrow \{0, 1\}$

Goal: The allocation π creates no envy

Agent a *envies* b in π if $v_a(\pi(b)) > v_a(\pi(a))$

#1. Background & definitions

😭 → {🍰, 🍲}

😍 → {🍔, 🍟, ☕}

FAIR DIVISION

Task: Compute an *allocation* $\pi : A \rightarrow 2^R$

Backdrop: $\nu_\ell : 2^R \rightarrow \{0, 1\}$

Goal: The allocation π creates no envy

Agent a *envies* b in π if $v_a(\pi(b)) > v_a(\pi(a))$ and a **knows** b .

#1. Background & definitions

😭 → {🍰, 🍲}

😍 → {🍔, 🍟, ☕}

FAIR DIVISION

Task: Compute an *allocation* $\pi : A \rightarrow 2^R$

Backdrop: $\nu_\ell : 2^R \rightarrow \{0, 1\}$

Goal: The allocation π creates no envy (between friends)

Agent a *envies* b in π if $v_a(\pi(b)) > v_a(\pi(a))$ and a *knows* b .

#1. Background & definitions

😭 → {🍰, 🍲}

😍 → {🍔, 🍟, ☕}

FAIR DIVISION

Task: Compute an *allocation* $\pi : A \rightarrow 2^R$

Backdrop: $\nu_\ell : 2^R \rightarrow \{0, 1\}$ and a social network on A

Goal: The allocation π creates no envy (between friends)

Agent a *envies* b in π if $v_a(\pi(b)) > v_a(\pi(a))$ and a **knows** b .

#1. Background & definitions

Fair Division via Social Comparison

AAMAS 2017

Rediet Abebe,
Jon M. Kleinberg,
and David C. Parkes

ABSTRACT

We study cake cutting on a graph, where agents can only evaluate their shares relative to their neighbors. This is an extension of the classical problem of fair division to incorporate the notion of social comparison from the social sciences. We say an allocation is *locally envy-free* if no agent envies a neighbor's allocation, and *locally proportional* if each agent values its own allocation as much as the average value of its neighbors' allocations. We generalize the classical "Cut and Choose" protocol for two agents to this setting, by fully characterizing the set of graphs for which an oblivious *single-cutter protocol* can give locally envy-free (thus also locally-proportional) allocations. We study the *price of envy-freeness*, which compares the total value of an optimal allocation with that of an optimal, locally envy-free allocation. Surprisingly, a lower bound of $\Omega(\sqrt{n})$ on the price of envy-freeness for global allocations also holds for local envy-freeness in any connected graph, so sparse graphs do not provide more flexibility asymptotically with respect to the quality of envy-free allocations.

#1. Background & definitions

We introduce a graphical framework for fair division in **cake cutting**, where comparisons between agents are limited by an underlying network structure. We generalize the classical fairness notions of envy-freeness and proportionality to this graphical setting. Given a simple undirected graph G , an allocation is *envy-free on G* if no agent envies any of her **neighbor's share**, and is *proportional on G* if every agent values her own share no less than the average among her neighbors, with respect to her own measure. These generalizations open new research directions in developing simple and efficient algorithms that can produce fair allocations under specific graph structures.

On the algorithmic frontier, we first propose a moving-knife algorithm that outputs an envy-free allocation on **trees**. The algorithm is significantly simpler than the discrete and bounded envy-free algorithm recently designed in [Aziz and Mackenzie, 2016a] for complete graphs. Next, we give a discrete and bounded algorithm for computing a proportional allocation on descendant graphs, a class of graphs by taking a rooted tree and connecting all its ancestor-descendant pairs.

Networked fairness in cake cutting

IJCAI 2017

Xiaohui Bei.

Youming Qiao, and
Shengyu Zhang.

#1. Background & definitions

Envy-Free Cake Cutting with Graph Constraints

Preprint (2022)

Ganesh Ghalme, Xin Huang, Nidhi Rathi

We study the classic problem of *fairly* dividing a heterogeneous and divisible resource—represented by a cake, $[0, 1]$ —among n agents. This work considers an interesting variant of the problem where agents are embedded on a graph. The graphical constraint entails that each agent evaluates her allocated share only against her neighbor's share. Given a graph, the goal is to efficiently find a *locally envy-free* allocation where every agent values her share to be at least as much as any of her neighbor's share.

The best known algorithm (by Aziz and Mackenzie [AM16a]) for finding envy-free cake divisions has a hyper-exponential query complexity. One of the key technical contributions of this work is to identify a non-trivial graph structure—tree graphs with depth at-most two (DEPTH2TREE)—on n agents that admits a query efficient cake-cutting protocol (under the Robertson-Webb query model). In particular, we develop a discrete protocol that finds a locally envy-free allocation among n agents on DEPTH2TREE with at-most $O(n^3 \log(n))$ cuts on the cake. For the special case of DEPTH2TREE where every non-root agent is connected to at-most two agents (2-STAR), we show that $O(n^2)$ queries suffice. We complement our algorithmic results with establishing a lower bound of $\Omega(n^2)$ (evaluation) queries for finding a locally envy-free allocation among n agents on a 1-STAR graph (under the assumption that the root agent partitions the cake into n connected pieces).

#1. Background & definitions

We study the fair division problem consisting in allocating one item per agent so as to avoid (or minimize) envy, in a setting where only agents connected in a given social network may experience envy. In a variant of the problem, agents themselves can be located on the network by the central authority. These problems turn out to be difficult even on very simple graph structures, but we identify several tractable cases. We further provide practical algorithms and experimental insights.

Local envy-freeness in house allocation problems.

AAMAS 2018

Aurélie Beynier, Yann Chevaleyre, Laurent Gourvès,
Julien Lesca, Nicolas Maudet, and Anaëlle Wilczynski.

#1. Background & definitions

Finding an *envy-free* allocation of **indivisible resources** to agents is a central task in many multiagent systems. Often, non-trivial envy-free allocations do not exist and determining whether they exist is computationally hard even under highly restricted settings. Classical envy-freeness requires that every agent likes the resources allocated to it at least as much as the resources allocated to *any* other agent. In many situations this assumption can be relaxed since agents often do not even know each other. We **enrich the envy-freeness concept** by taking into account (**directed**) **social networks** of the agents. Thus, we require that every agent likes its own allocation *at least* as much as those of all its (out)neighbors. This leads to a “more local” concept of envy-freeness. We also consider a “strong” variant where every agent must like its own allocation *more than* those of all its (out)neighbors.

We analyze the classical and the parameterized complexity of finding allocations that are complete and, at the same time, envy-free with respect to one of the variants of our new concept. To this end, we study different restrictions of the agents' preferences and of the social network structure. We identify cases that become easier (from Σ_2^P -hard or NP-hard to polynomial-time solvable) and cases that become harder (from polynomial-time solvable to NP-hard) when comparing classical envy-freeness with our graph envy-freeness. Furthermore, we spot cases where graph envy-freeness is easier to decide than strong graph envy-freeness, and vice versa. On the route to one of our fixed-parameter tractability results, we also establish a connection to a directed and colored variant of the classical SUBGRAPH ISOMORPHISM problem, thereby extending a known fixed-parameter tractability result for the latter.

Envy-Free Allocations Respecting Social Networks

Journal of Artificial
Intelligence, 2022

&

AAMAS 2018

Robert Bredereck,
Andrzej Kaczmarczyk,
and Rolf Niedermeier.

#1. Background & definitions

We consider the classical problem of allocating resources among agents in an envy-free (and, where applicable, proportional) way. Recently, the basic model was enriched by introducing the concept of a social network which allows to capture situations where agents might not have full information about the allocation of all resources. We initiate the study of the parameterized complexity of these resource allocation problems by considering natural parameters which capture structural properties of the network and similarities between agents and items. In particular, we show that even very general fragments of the considered problems become tractable as long as the social network has bounded treewidth or bounded clique-width. We complement our results with matching lower bounds which show that our algorithms cannot be substantially improved.

**Parameterized Complexity of
Envy-Free
Resource Allocation
in Social Networks**

AAAI 2020

Eduard Eiben,
Robert Ganian,
Thekla Hamm,
and Sebastian Ordyniak

#1. Background & definitions

Locally EFX Allocations Over a Graph

Preprint (2022)

Justin Payan, Rik Sengupta, Vignesh Viswanathan

The fairness notion of *envy-free up to any good* (EFX) has recently gained popularity in the fair allocation literature. However, few positive results about EFX allocations exist, with the existence of such allocations alone being an open question. In this work, we study an intuitive relaxation of EFX allocations: our allocations are only required to satisfy the EFX constraint for certain pre-defined pairs of agents. Since these problem instances can be represented using an undirected graph where the vertices correspond to the agents and each edge represents an EFX constraint, we call such allocations *graph EFX* (G -EFX) allocations. We show that G -EFX allocations exist for three different classes of graphs — two of them generalize the star $K_{1,n-1}$ and the third generalizes the three-edge path P_4 . We also present and evaluate an algorithm using problem instances from Spliddit to show that G -EFX allocations are likely to exist for larger classes of graphs like long paths P_n .

#1. Background & definitions

Complexity of Locally Fair Allocations on Graphs

MSc Thesis (*Afstudeerscriptie*)

written by

Giovanni Varricchione

(born September 17th, 1996 in Benevento, Italy)

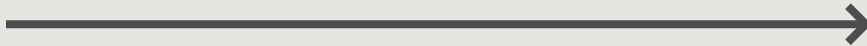
under the supervision of **Dr. Ronald de Haan** and **Drs. Simon Rey**, and
submitted to the Examinations Board in partial fulfillment of the
requirements for the degree of

MSc in Logic

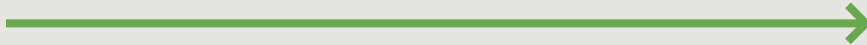
at the *Universiteit van Amsterdam*.

OUTLINE

Background & definitions



Summary of results



NP-completeness for two agent types



Future work



XP Algorithm:

Solve the problem in time $O^*(n^{f(k)})$, for some appropriate *parameter* k .

XP Algorithm:

Solve the problem in time $O^*(n^{f(k)})$, for some appropriate *parameter* k .

FPT Algorithm:

Solve the problem in time $f(k) \cdot n^{O(1)}$, for some appropriate *parameter* k .

XP Algorithm:

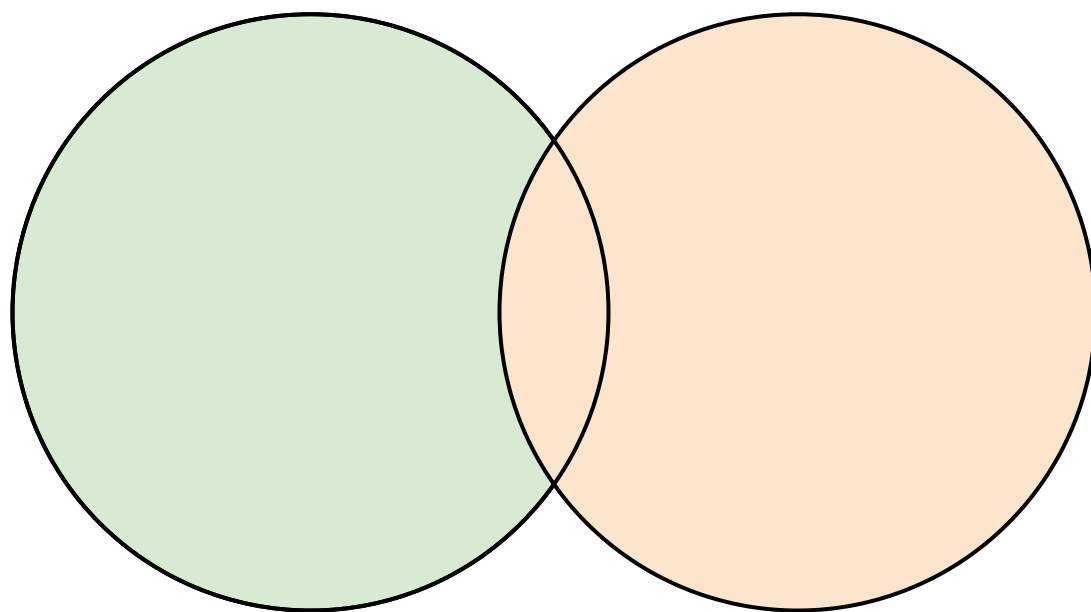
Solve the problem in time $O^*(n^{f(k)})$, for some appropriate *parameter* k .

FPT Algorithm:

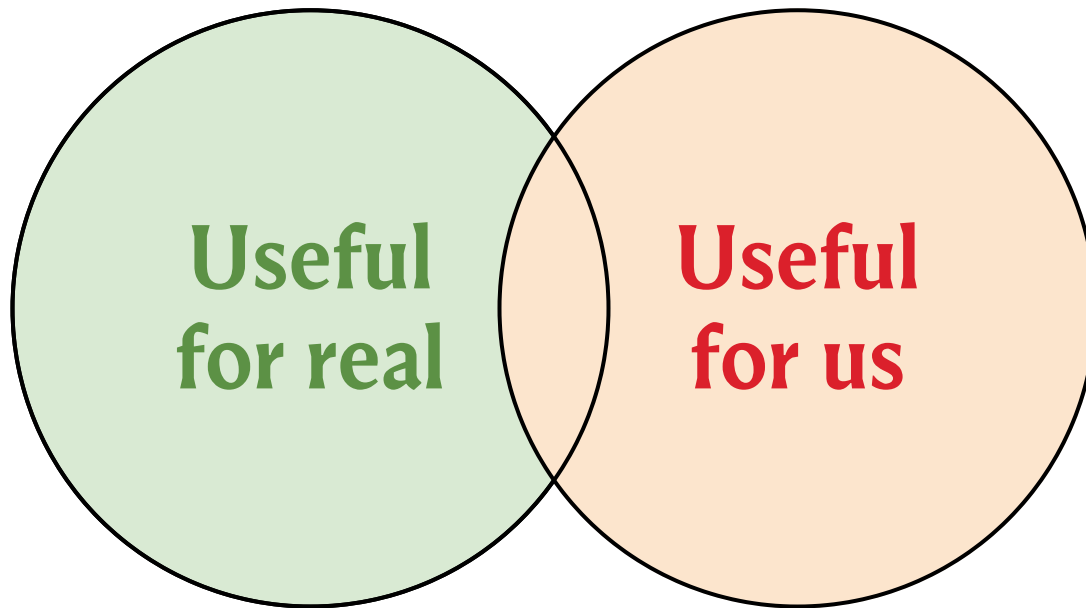
Solve the problem in time $f(k) \cdot n^{O(1)}$, for some appropriate *parameter* k .

Kernelization:

Reduce the problem in time $n^{O(1)}$,
to an **equivalent** instance whose *size* is bounded by $f(k)$,
for some appropriate *parameter* k .



The *art* of parameterization



An example

Finding an efficient (PO) envy-free allocation is fixed-parameter tractable with respect to the combined parameter $(n+z)$.

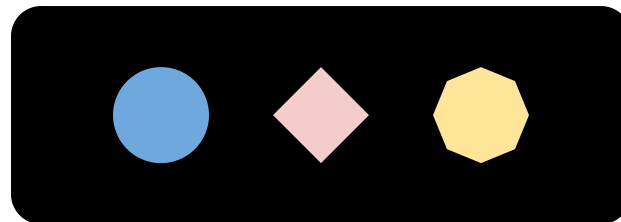
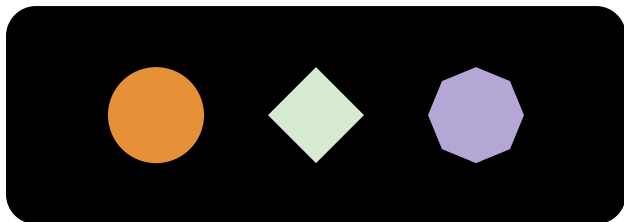
for identical monotonic additive preferences.

- n denotes the number of agents and
- z is the number of different values that occur in the utility functions.

Define the *signature* of a resource to be a n -tuple that collects all the utilities that agents have for said resource.

$$f_j := (u_1(r_j), u_2(r_j), \dots, u_n(r_j)).$$

$$\#\text{signatures} \sim z^n$$



$$f_{\bullet} = (u_1(\bullet), u_2(\bullet), \dots)$$

$$f_{\bullet} = (u_1(\bullet), u_2(\bullet), \dots)$$

Objects with the same signature are
indistinguishable.

For each agent $a_i \in A$ and each fingerprint f :

$x_f^i \rightarrow$ #of resources
with fingerprint f
allocated to agent i .

For each agent $a_i \in A$ and each fingerprint f :

$x_f^i \rightarrow$ #of resources
with fingerprint f
allocated to agent i .

$$\forall 1 \leq p \leq n, 1 \leq q \leq n :$$

For each agent $a_i \in A$ and each fingerprint f :

$x_f^i \rightarrow$ #of resources
with fingerprint f
allocated to agent i .

$$\forall 1 \leq p \leq n, 1 \leq q \leq n :$$

$$\sum_{f \in F} x_p^f \cdot f[p] \geq \sum_{f \in F} x_q^f \cdot f[p]$$

Parameters for Resource Allocation

Parameters for Resource Allocation

number of agents

Parameters for Resource Allocation

number of agents

number of items

Parameters for Resource Allocation

number of agents

number of items

number of agent types

Parameters for Resource Allocation

number of agents

number of items

number of agent types

number of item types

Parameters for Resource Allocation

number of agents

number of items

number of agent types

number of item types

max utility

Parameters for Resource Allocation

number of agents

number of items

number of agent types

number of item types

max utility

#distinct utilities

Parameters for Resource Allocation

number of agents

number of items

number of agent types

number of item types

max utility

#distinct utilities

graph parameters (e.g, vertex cover, treewidth, etc.)

Parameters for Resource Allocation

number of agents

number of items

number of agent types

number of item types

max utility

#distinct utilities

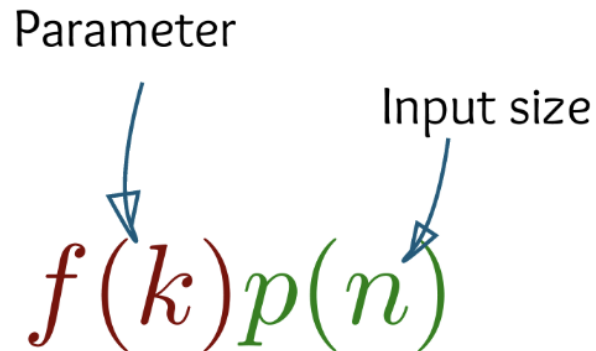
graph parameters (e.g, vertex cover, treewidth, etc.)

domain parameters (e.g, rank/sparsity/etc. of the utility matrix)

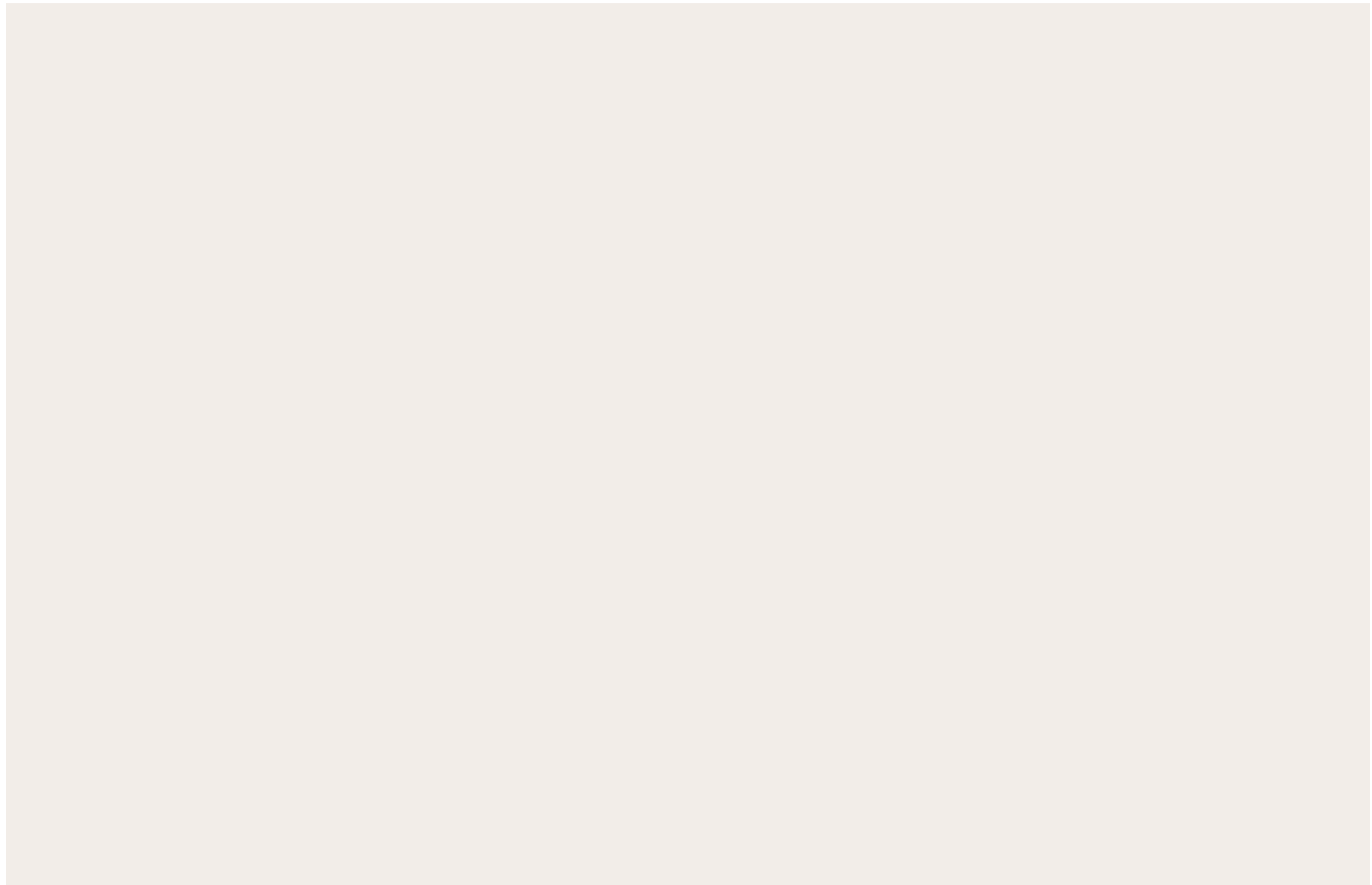
Recap: the parameterized paradigm

Parameterized complexity: acknowledge the presence of additional structure, which manifests as a secondary measurement a parameter.

Design algorithms that restrict the combinatorial explosion to a function of the parameter.



fixed-parameter tractability



#2. Summary of results

Undirected Graphs

Undirected Graphs

Goal: Determine the existence of a complete and locally envy-free allocation.

(even when valuations are binary
and the graph is complete).

Undirected Graphs

Goal: Determine the existence of a complete and locally envy-free allocation.

NP-complete

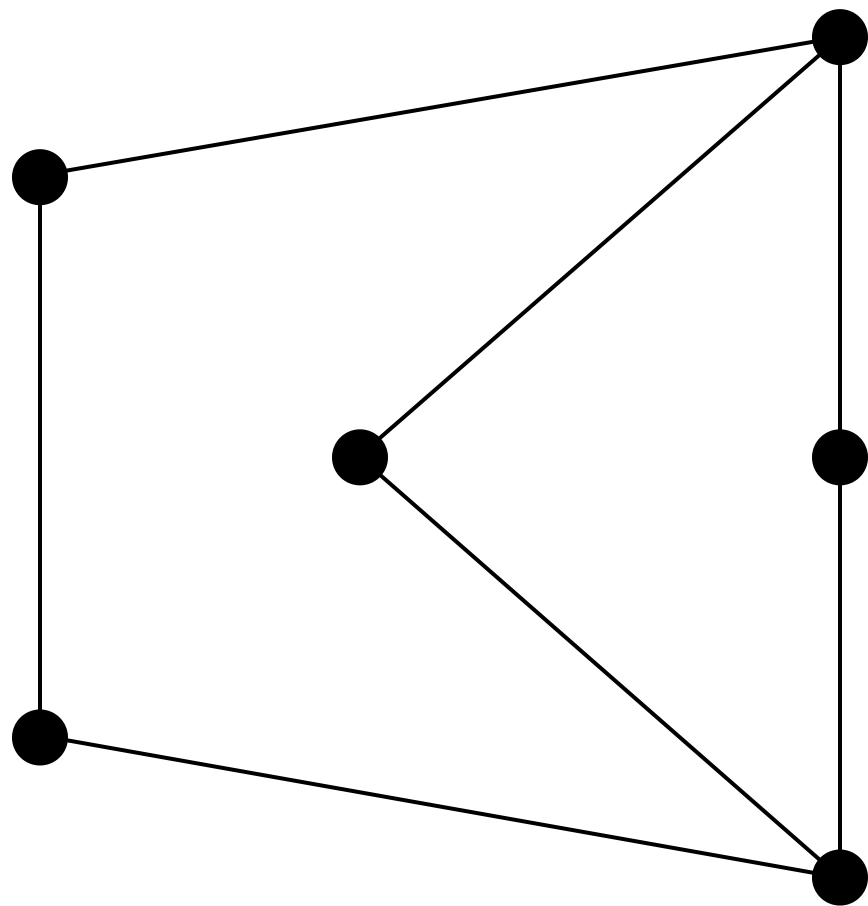
(even when valuations are binary
and the graph is complete).

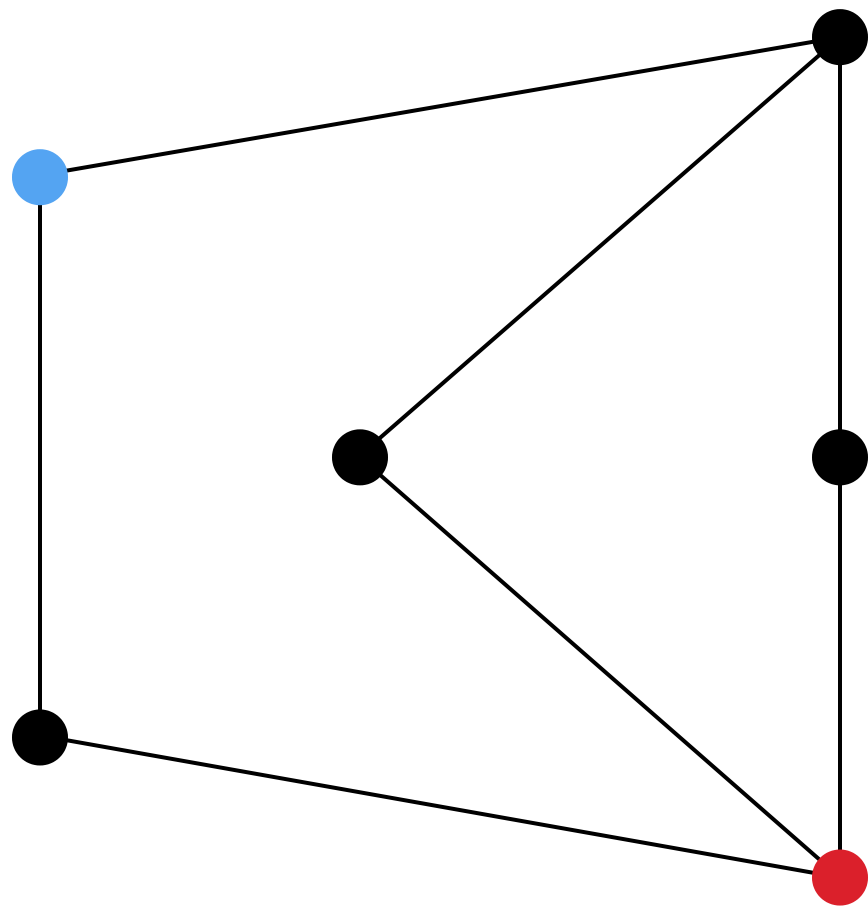
Undirected Graphs

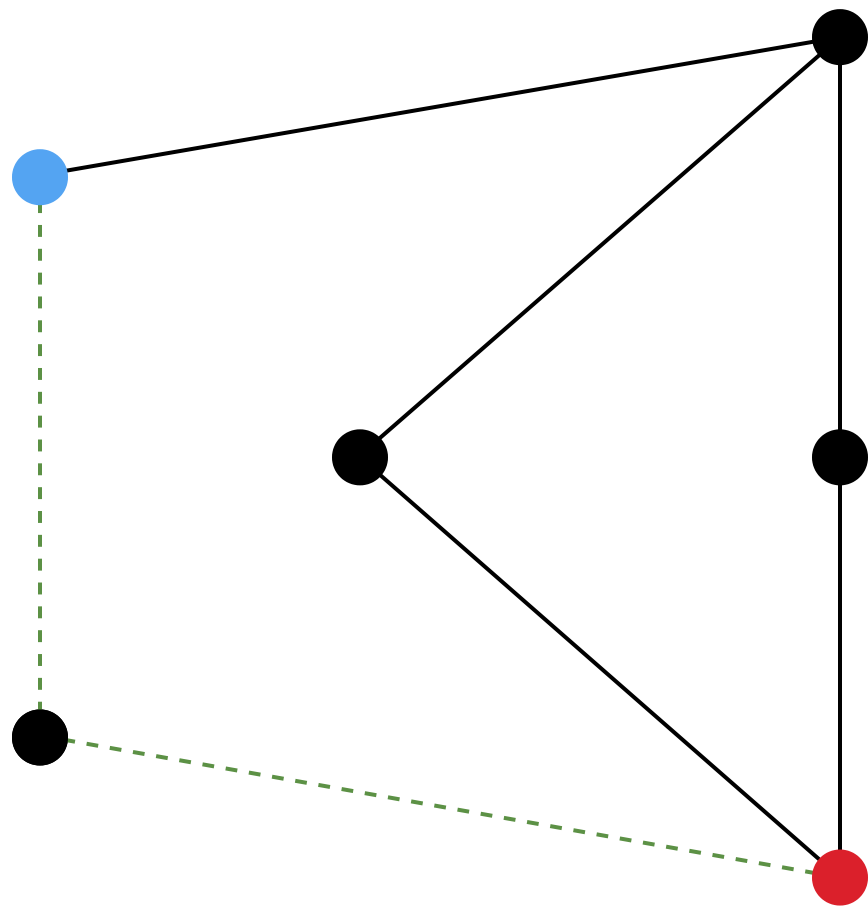
Goal: Determine the existence of a complete and locally envy-free allocation.

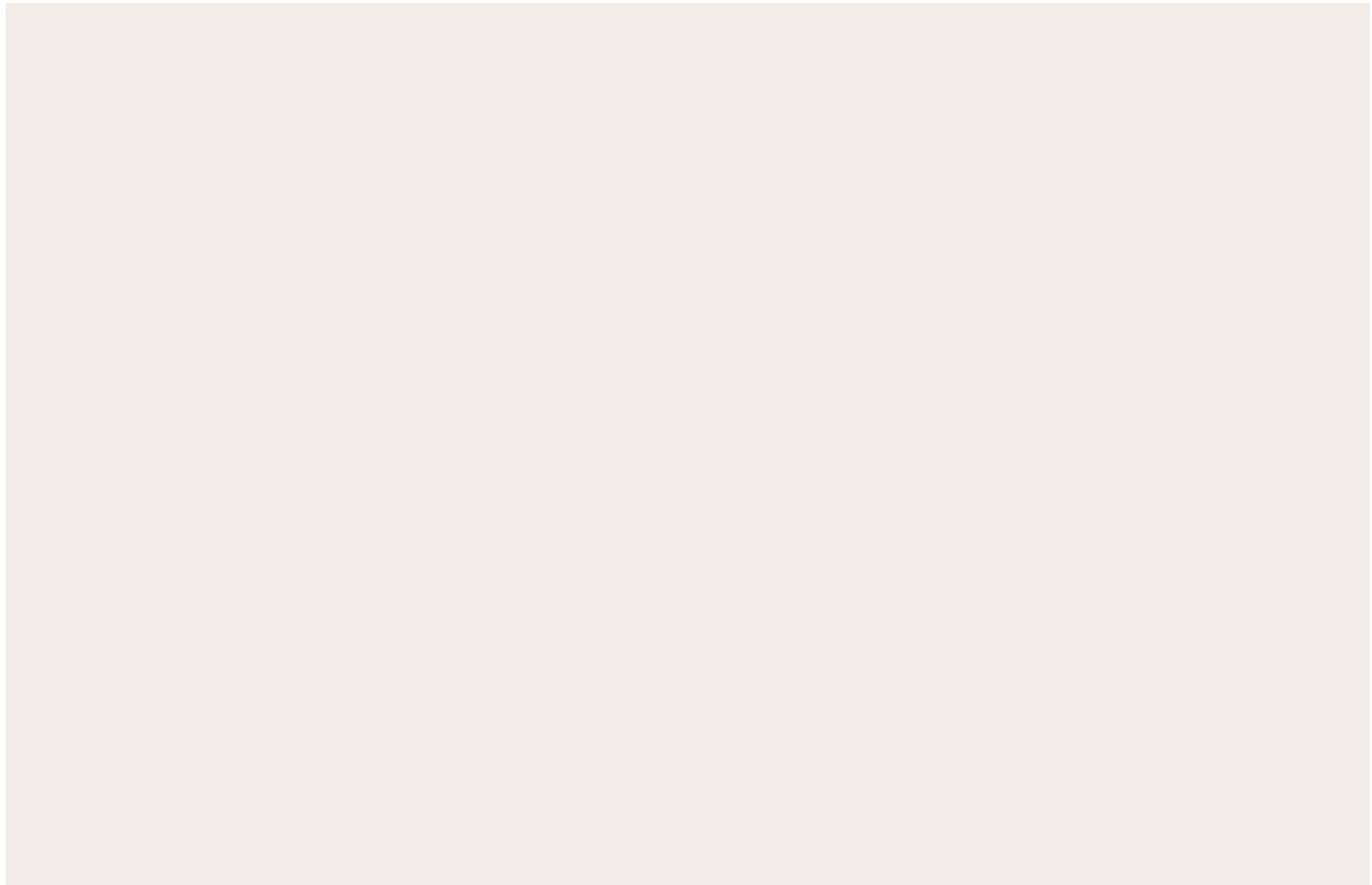
P

~~even~~ when valuations are binary *and* identical.*









#2. Summary of results

Directed Graphs

Directed Graphs

Goal: Determine the existence of a complete and locally envy-free allocation.

Directed Graphs

Goal: Determine the existence of a complete and locally envy-free allocation.

NP-hard

(even when valuations are binary *and* identical).

Undirected Graphs

Goal: Determine the existence of a complete and locally envy-free allocation.

NP-hard

even when valuations are binary *and* **nearly** identical.

Directed Graphs

Goal: Determine the existence of a
efficient and locally envy-free allocation.

P

if the social network is a DAG

Directed Graphs

Goal: Determine the existence of a
efficient and locally envy-free allocation.

P

if the social network is a DAG

Undirected Graphs

Goal: Determine the existence of a
efficient and locally envy-free allocation.

NP-hard

even if the social network is a path

Undirected Graphs

Goal: Determine the existence of a
efficient and locally envy-free allocation.

NP-hard

even if* the social network is a star

*general utilities

Undirected Graphs

Goal: Determine the existence of a
efficient and locally envy-free allocation.

NP-hard

even if* the social network is a star

*general utilities

Undirected Graphs

Goal: Determine the existence of a
efficient and locally envy-free allocation.

W[1]-hard

parameterized by the vertex cover number for 0/1 utilities

#2. Summary of results

Complete Graphs

Goal: Determine the existence of a *efficient* and locally envy-free allocation.

FPT

parameterized by #goods or #agents

Complete Graphs

Goal: Determine the existence of a
efficient and locally envy-free allocation.

FPT

parameterized by #goods or #agents

Undirected Graphs

Goal: Determine the existence of a
efficient and locally envy-free allocation.

FPT

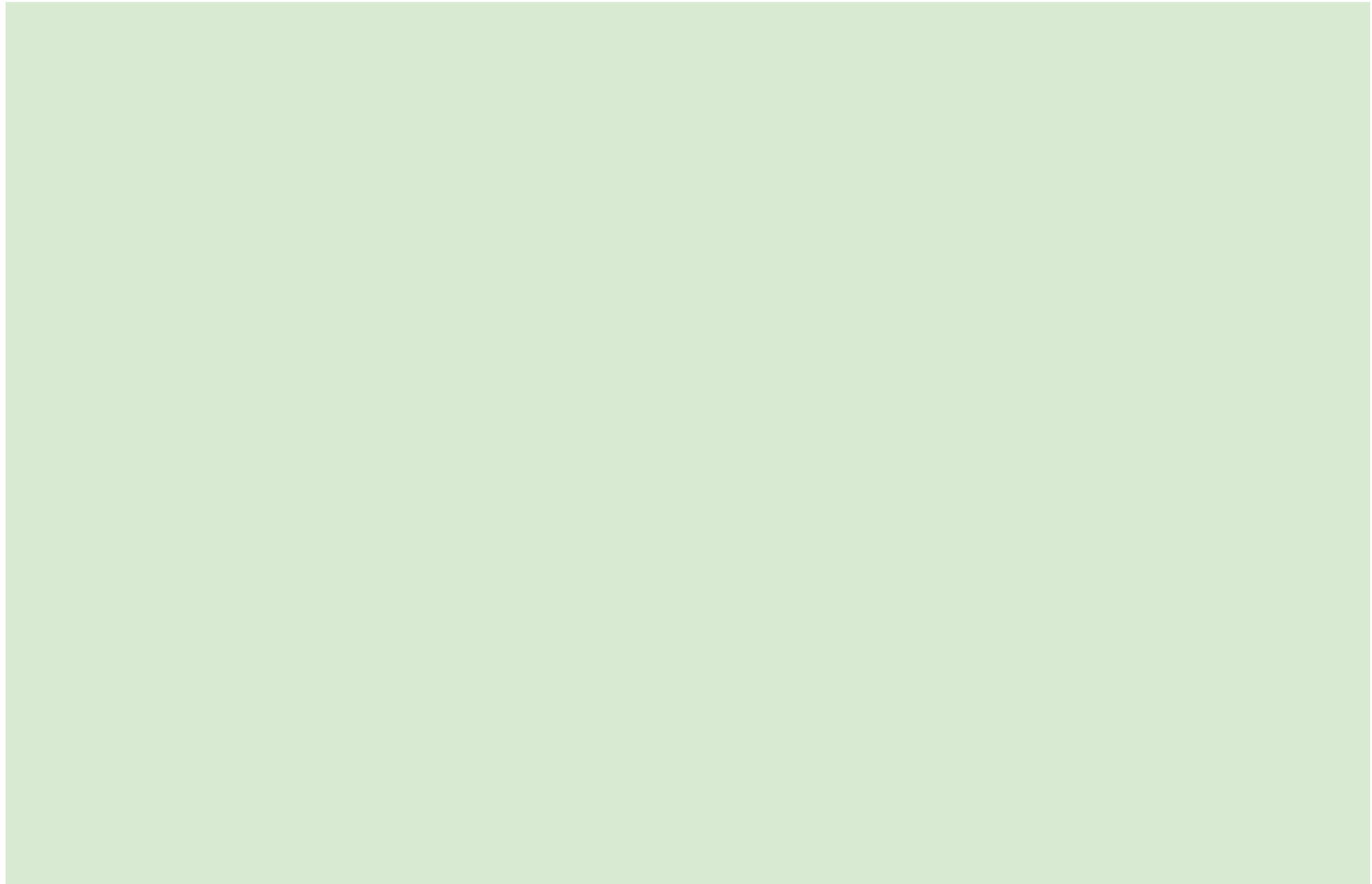
parameterized by ~~#goods~~ or #agents

Undirected Graphs

Goal: Determine the existence of a *efficient* and locally envy-free allocation.

W[1]-hard

parameterized by #goods or ~~#agents~~



#2. Summary of results

TAKEAWAY

TAKEAWAY

Localized notions of envy...

TAKEAWAY

Localized notions of envy...
are a natural and less restrictive model.

TAKEAWAY

undirected graphs

v/s

directed graphs

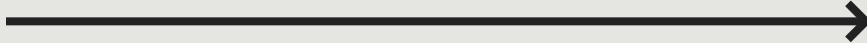
some contrasts, some similarities...

OUTLINE

Background & definitions



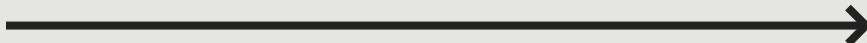
Summary of results



NP-completeness for two agent types



Future work



Reductions: a framework for showing hardness

Historically
annoying
problem

$\leq$

Our
annoying
problem

Recipe for solving
well-known annoying problem (e.g, SAT)
via *any* hypothetical algorithm (A) for solving problem X:

Recipe for solving
well-known annoying problem (e.g, SAT)
via *any* hypothetical algorithm (A) for solving problem X:

Step 1. Reduce instance of SAT to instance of X $\leftrightarrow \mathbf{q}$ 

Recipe for solving
well-known annoying problem (e.g, SAT)
via *any* hypothetical algorithm (A) for solving problem X:

Step 1. Reduce instance of SAT to instance of X $\leftrightarrow \mathbf{q}$ 💥

Step 2. Run the algorithm A on $\mathbf{q}$. Output $\rightarrow o$ 👁👁

Recipe for solving
well-known annoying problem (e.g, SAT)
via *any* hypothetical algorithm (A) for solving problem X:

Step 1. Reduce instance of SAT to instance of X $\leftrightarrow \mathbf{q}$ 💥

Step 2. Run the algorithm A on $\mathbf{q}$. Output $\rightarrow o$ 👁👁

Step 3. Return o . Profit. 💰

CUTTING ℓ VERTICES

G



X

S

Y

Given a graph G , partition its vertex set into three parts: $(X \cup S \cup Y)$ so that:



a) $|S| \leq k$

b) $|X| = \ell$

c) There are no edges between X and Y .

one *trigger* agent

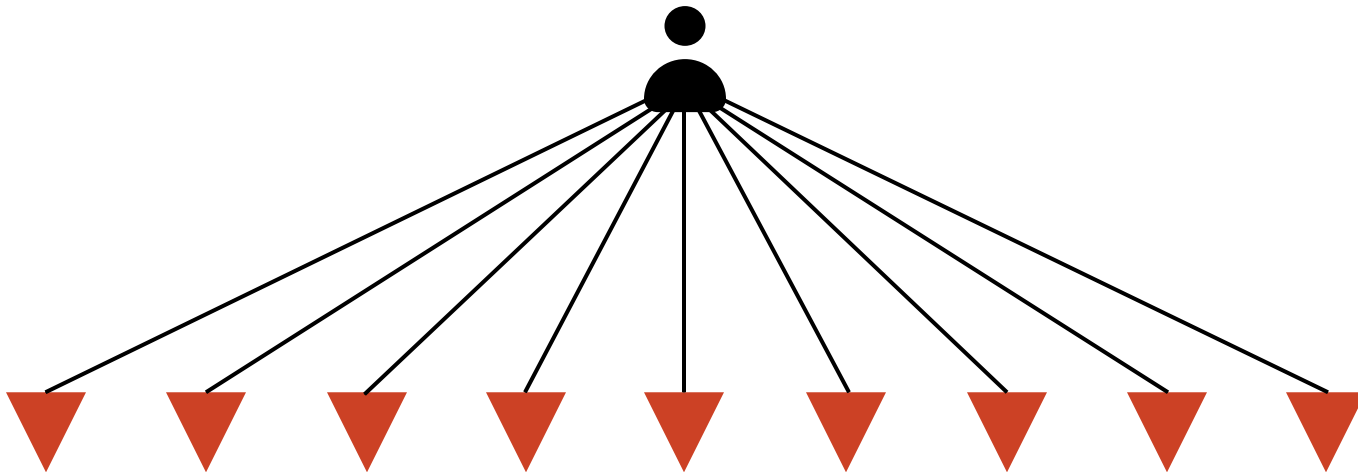


n *greedy* agents



n *happy* agents

one *trigger* agent

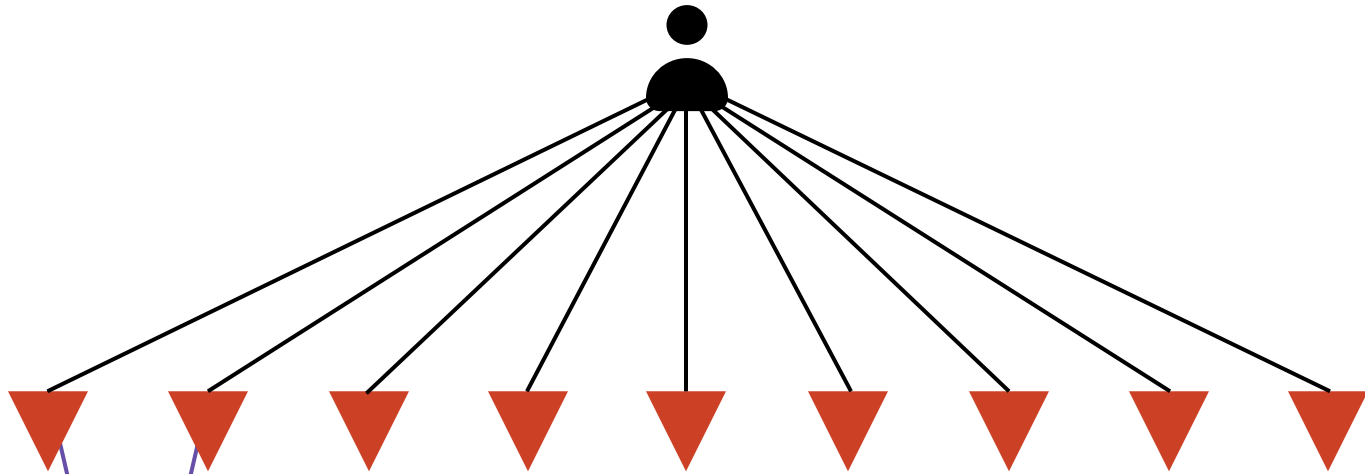


n *greedy* agents

n *happy* agents



one *trigger* agent



n *greedy* agents

...based on the structure of the graph G

n *happy* agents



ℓ coveted items

ℓ coveted items



ℓ coveted items



...liked by all agents.

ℓ *coveted* items



...liked by all agents.

$n - \ell + 1$ *greedy* items

ℓ *coveted* items



...liked by all agents.

$n - \ell + 1$ *greedy* items



ℓ *coveted* items



...liked by all agents.

$n - \ell + 1$ *greedy* items



...liked by all greedy agents and the trigger agent.

ℓ *coveted* items



...liked by all agents.

$n - \ell + 1$ *greedy* items



...liked by all greedy agents and the trigger agent.

$\ell + k$ *happy* items

ℓ *coveted* items



...liked by all agents.

$n - \ell + 1$ *greedy* items



...liked by all greedy agents and the trigger agent.

$\ell + k$ *happy* items



ℓ *coveted* items



...liked by all agents.

$n - \ell + 1$ *greedy* items



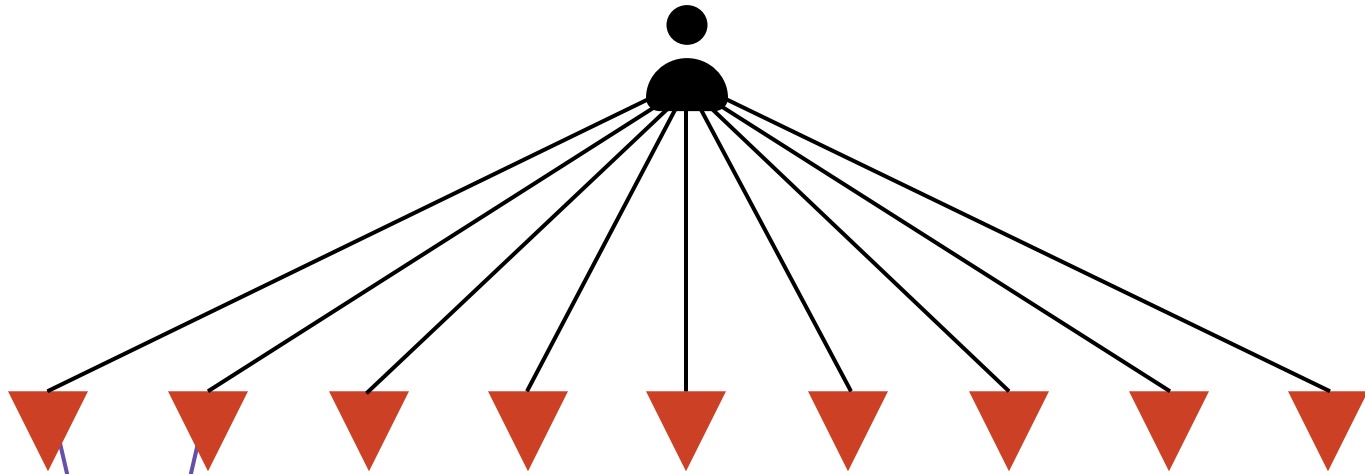
...liked by all greedy agents and the trigger agent.

$\ell + k$ *happy* items



...liked by all happy agents.

one *trigger* agent



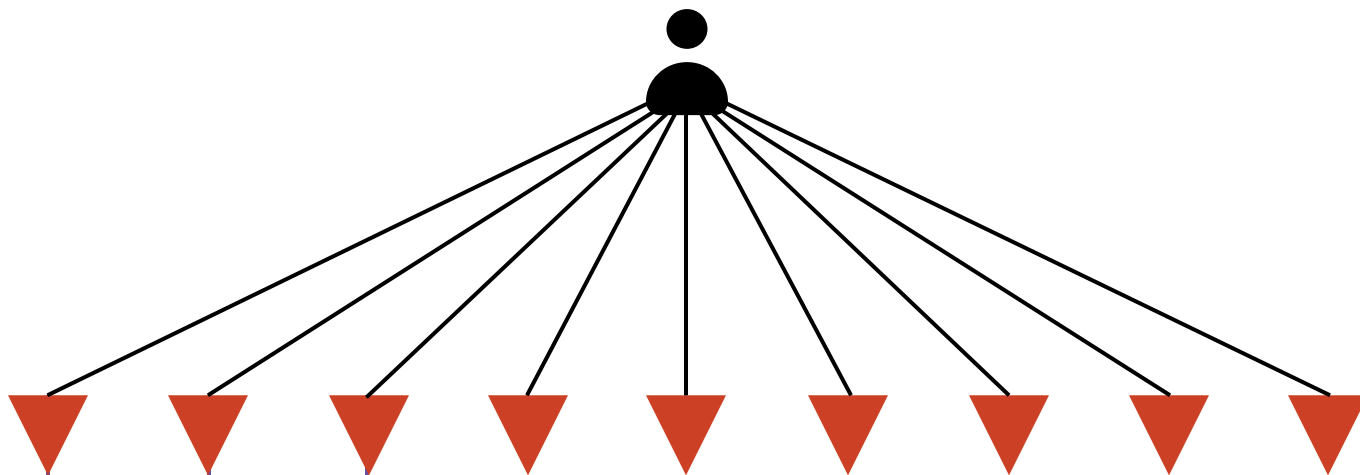
n *greedy* agents

...based on the structure of the graph G

n *happy* agents



one *trigger* agent

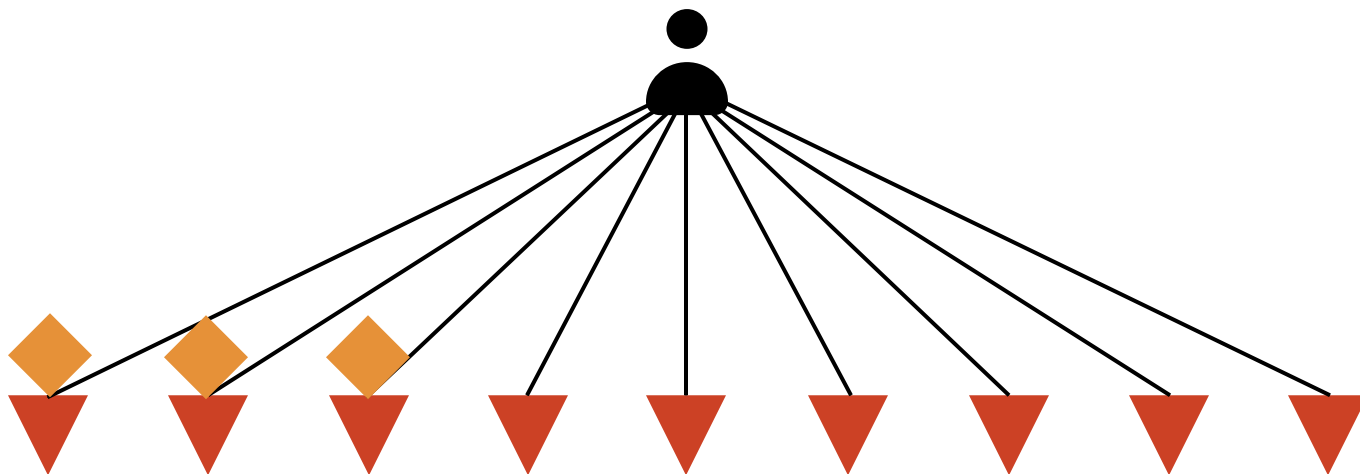


n *greedy* agents

n *happy* agents



one *trigger* agent

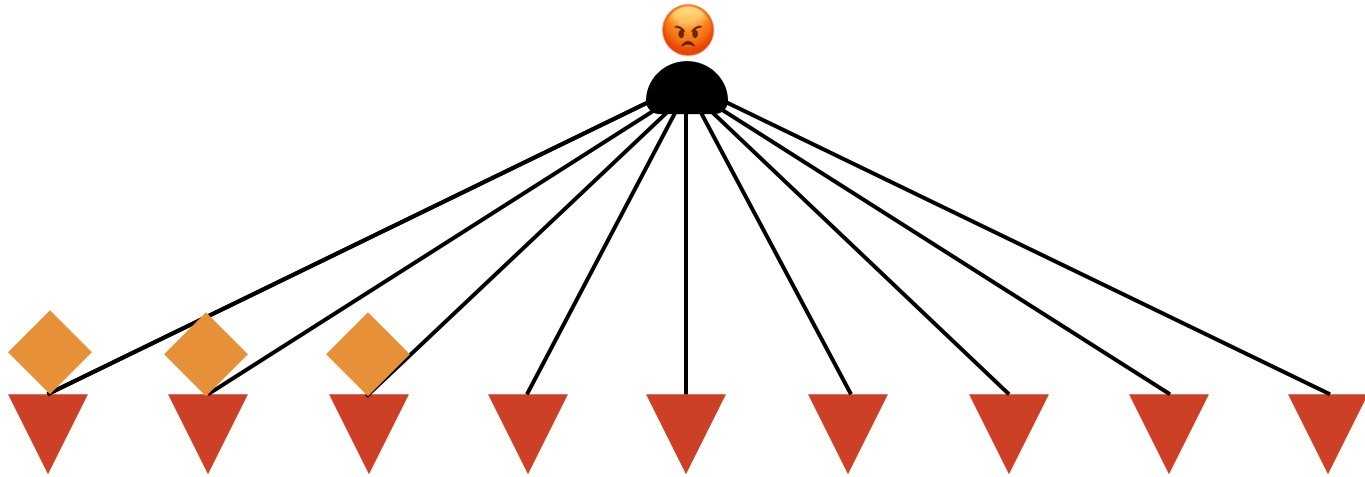


n *greedy* agents

n *happy* agents



one *trigger* agent

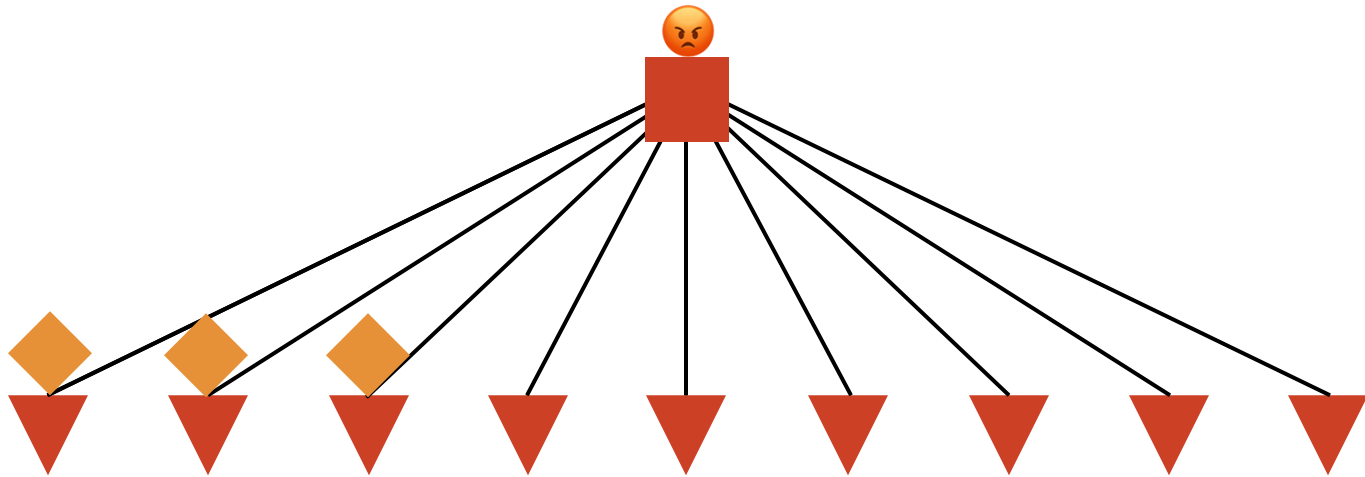


n *greedy* agents

n *happy* agents



one *trigger* agent

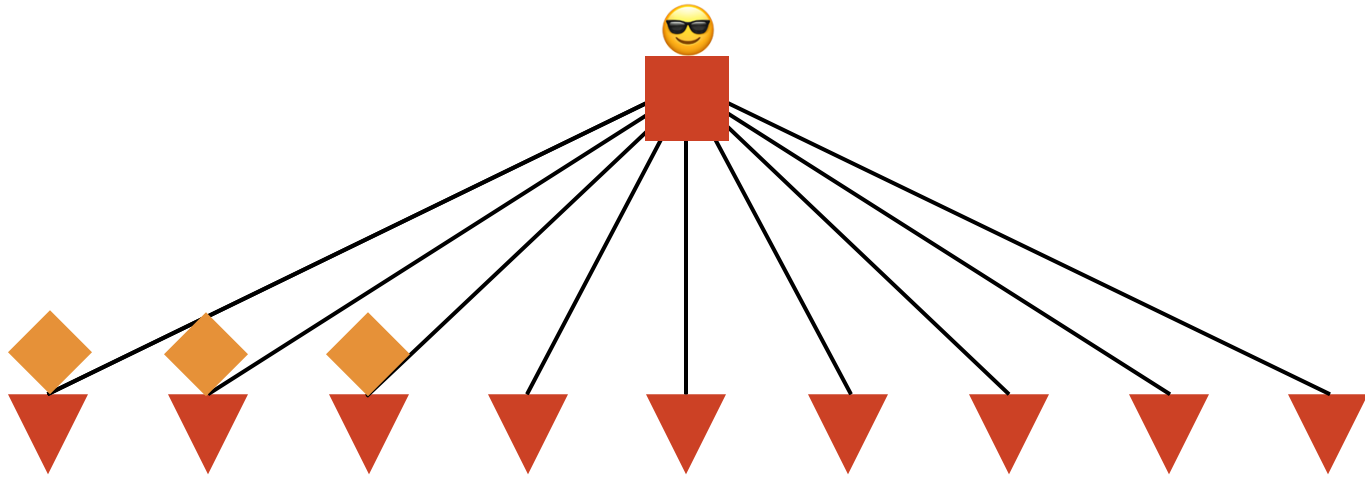


n *greedy* agents

n *happy* agents



one *trigger* agent

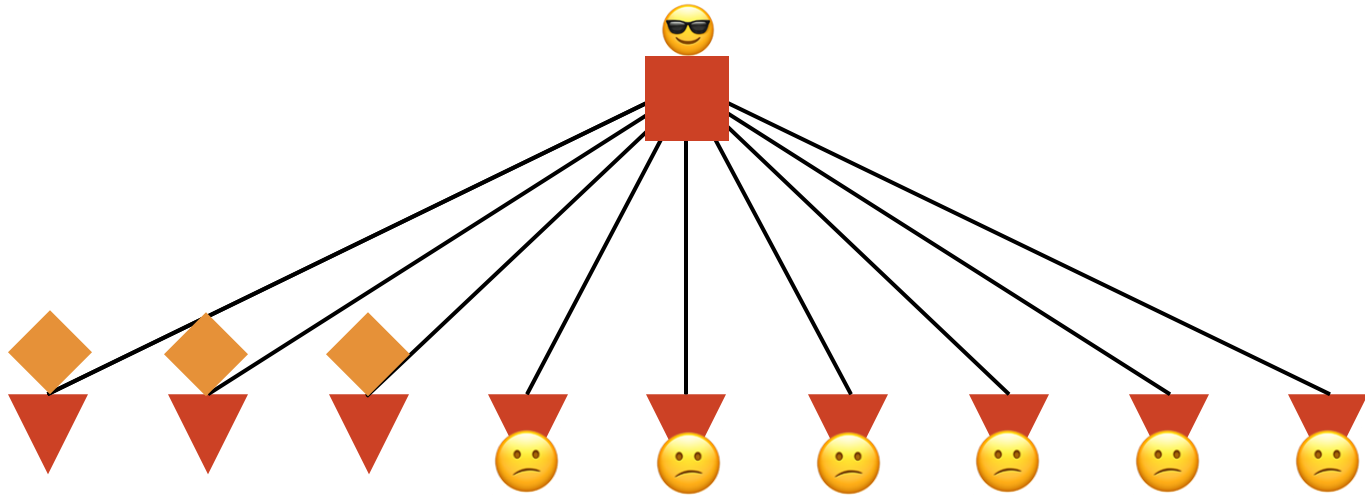


n *greedy* agents

n *happy* agents



one *trigger* agent

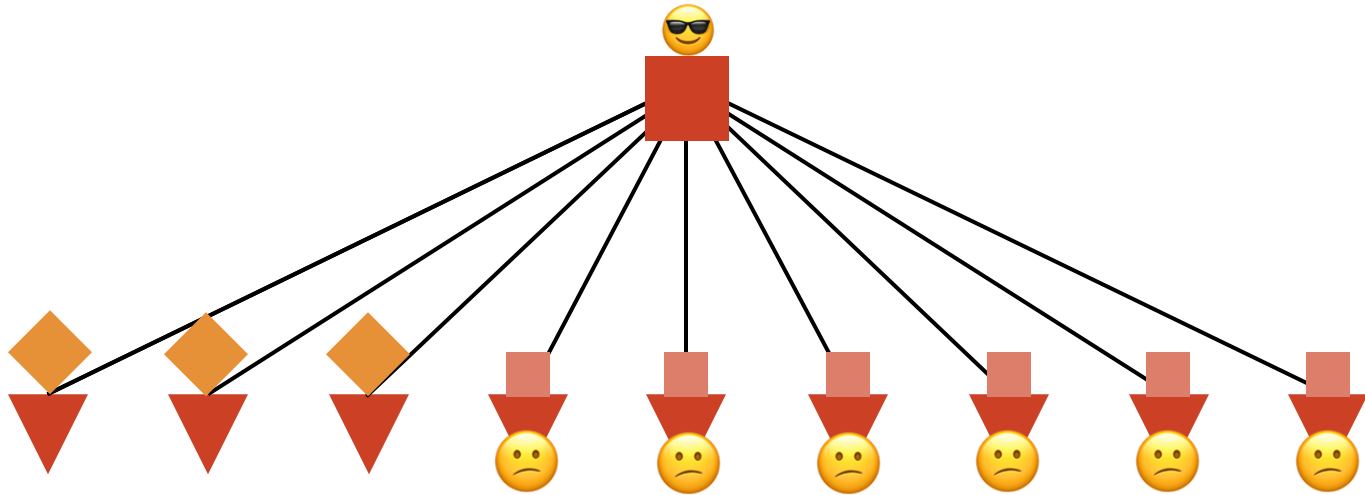


n *greedy* agents

n *happy* agents



one *trigger* agent

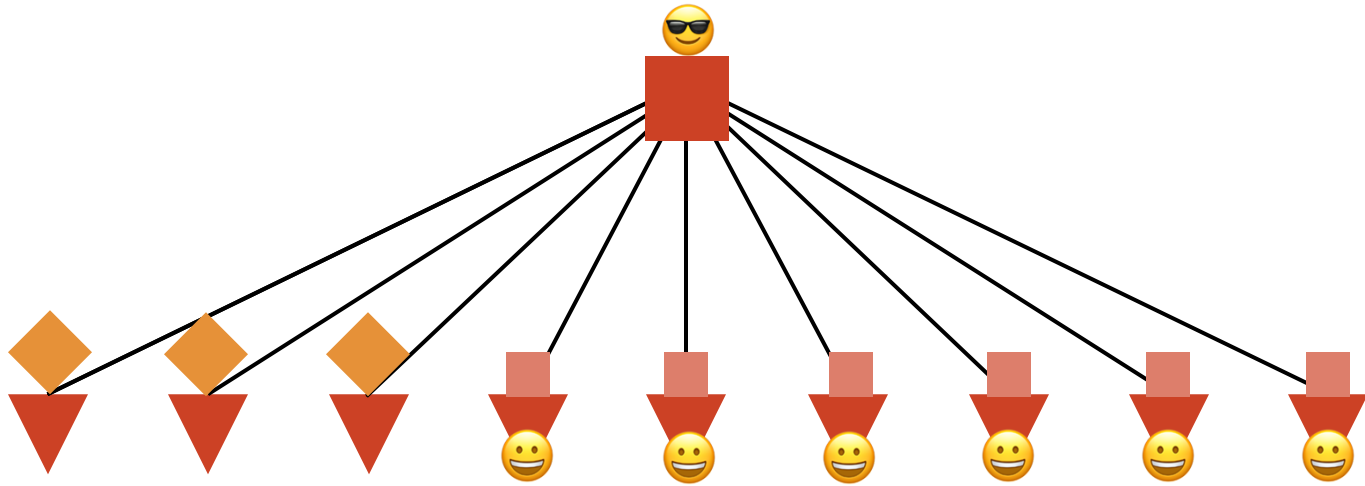


n *greedy* agents

n *happy* agents



one *trigger* agent

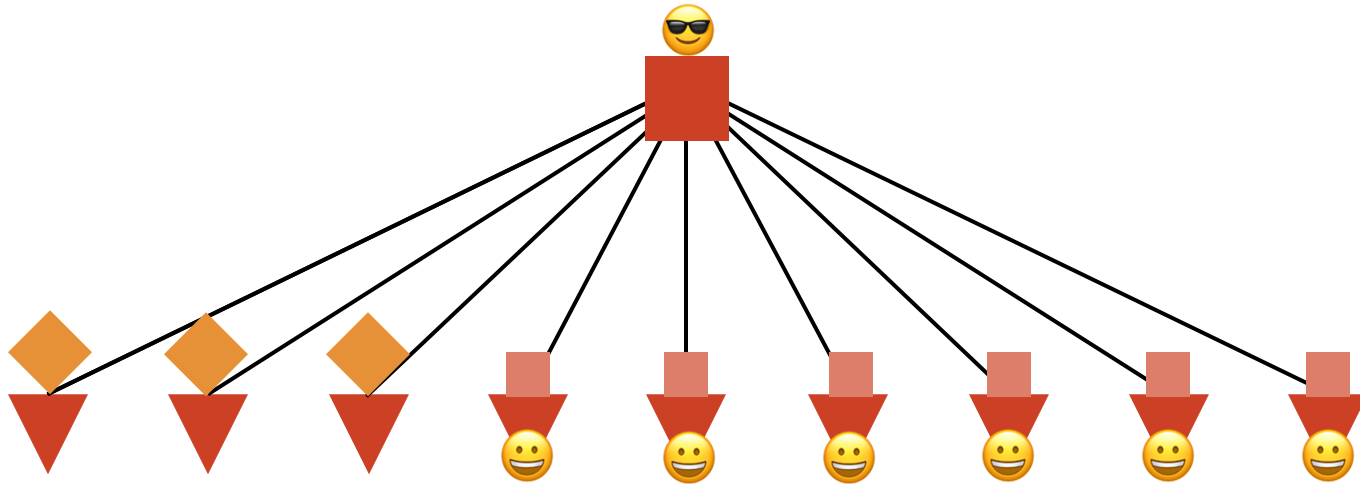


n *greedy* agents

n *happy* agents



one *trigger* agent



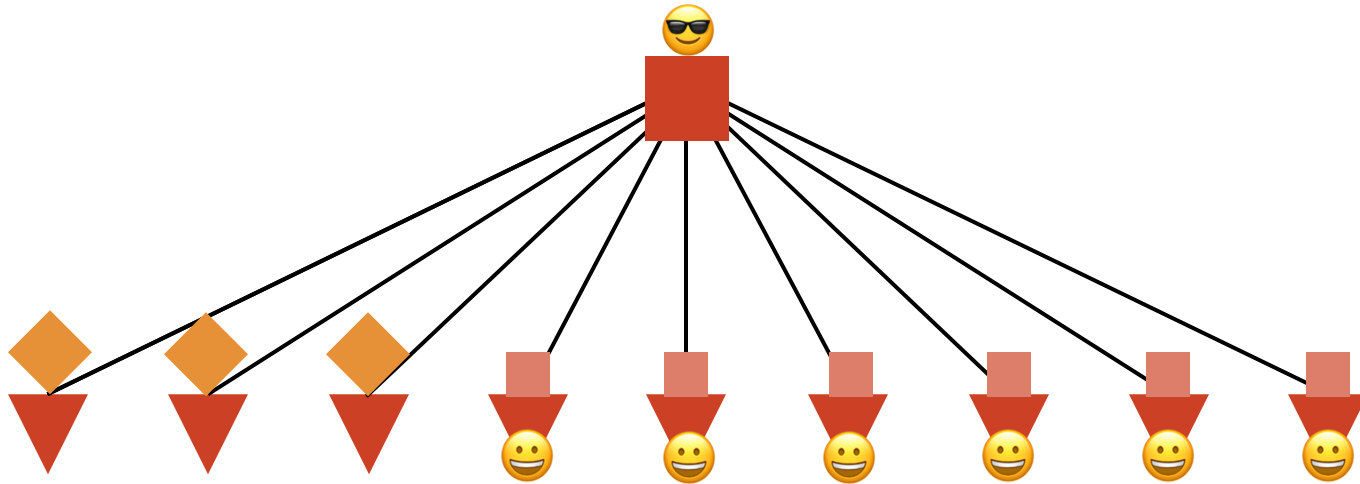
n *greedy* agents

Are any of the happy agents jealous?

n *happy* agents



one *trigger* agent



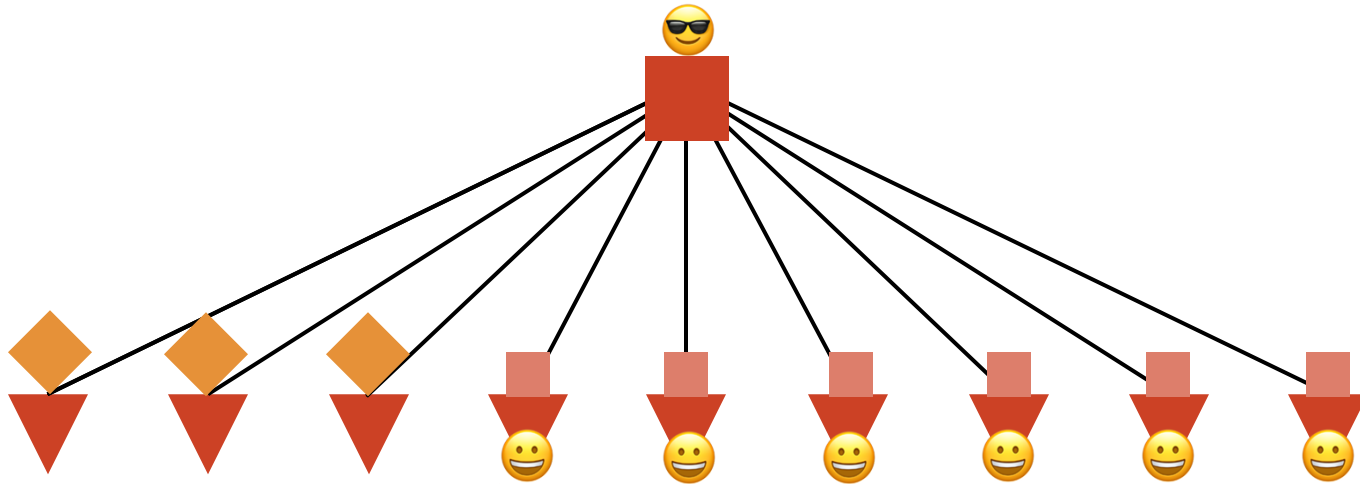
n *greedy* agents

They don't envy the 😊-agents, since they don't value the ■ good.

n *happy* agents



one *trigger* agent



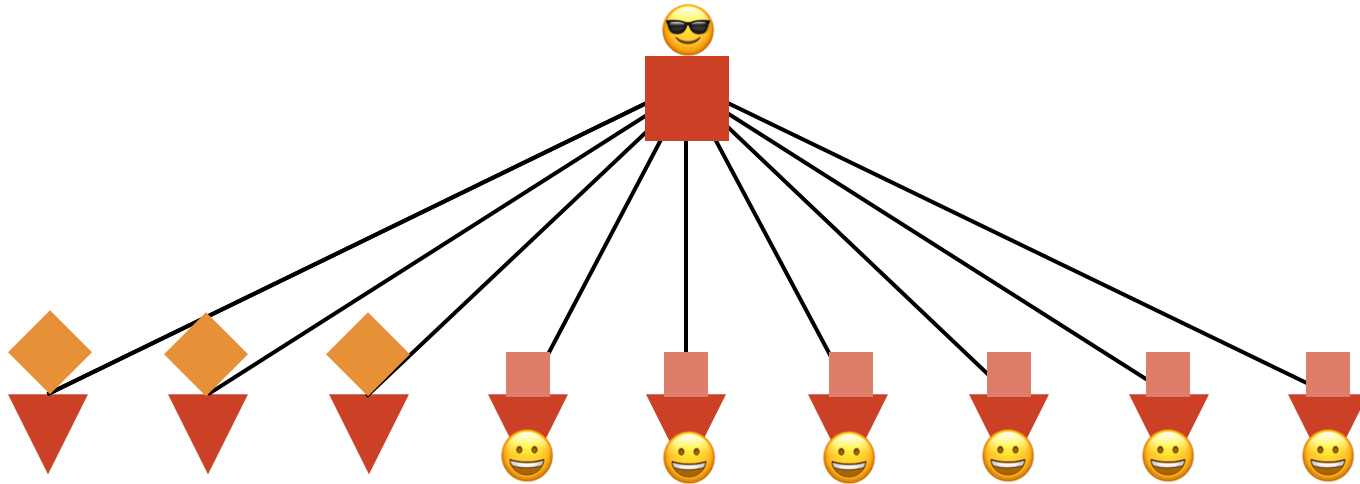
n *greedy* agents

Some of them will envy the agents who got the coveted goods .

n *happy* agents



one *trigger* agent



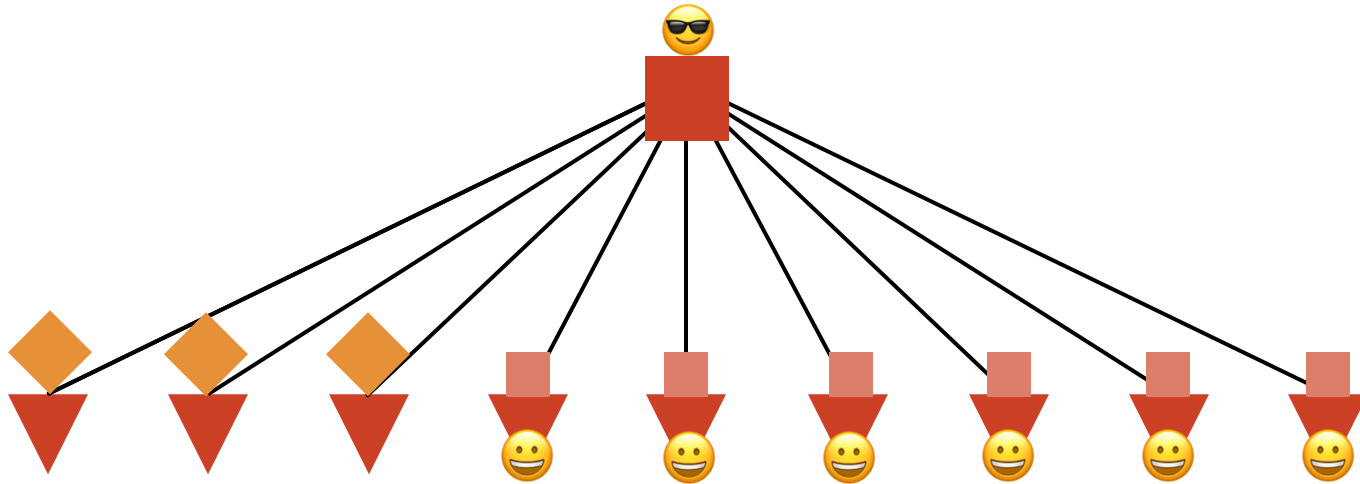
n *greedy* agents

Some of them will envy the agents who got the coveted goods .

n *happy* agents



one *trigger* agent



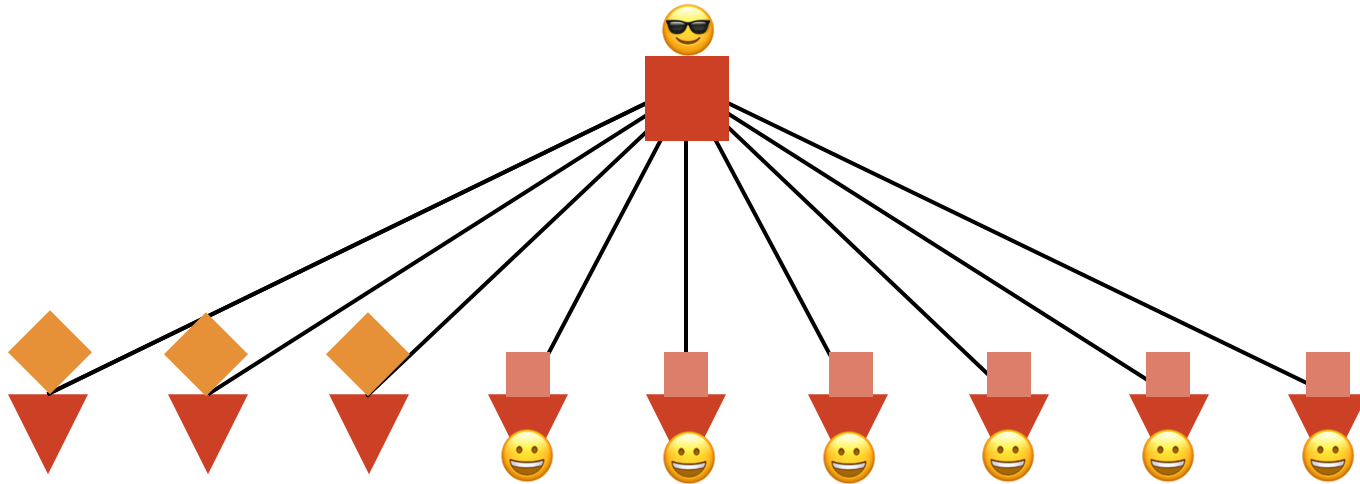
n *greedy* agents

Agents in $X \cup S$ will envy the agents who got the coveted goods .

n *happy* agents



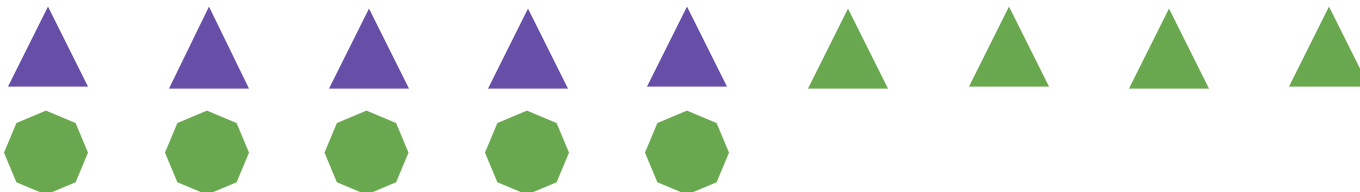
one *trigger* agent



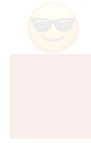
n *greedy* agents

Agents in $X \cup S$ will envy the agents who got the coveted goods .

n *happy* agents



one *trigger* agent



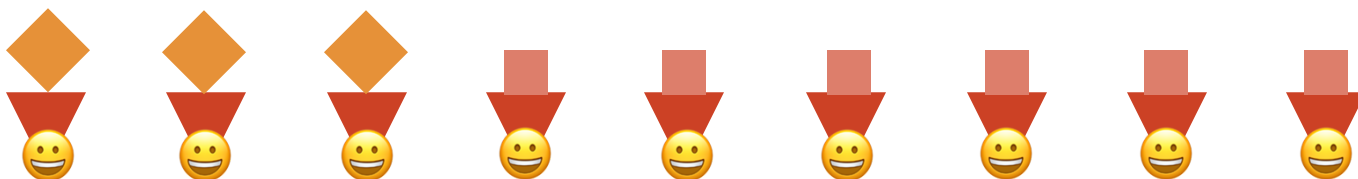
n *greedy* agents

Agents in $X \cup S$ will envy the agents who got the coveted goods .

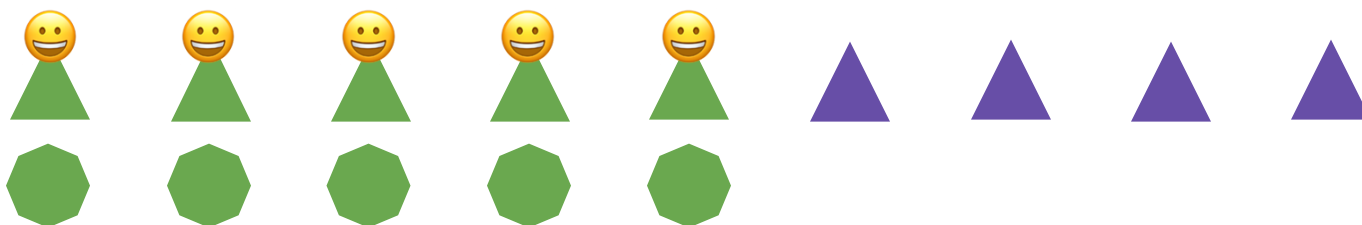
n *happy* agents



one *trigger* agent



n *greedy* agents

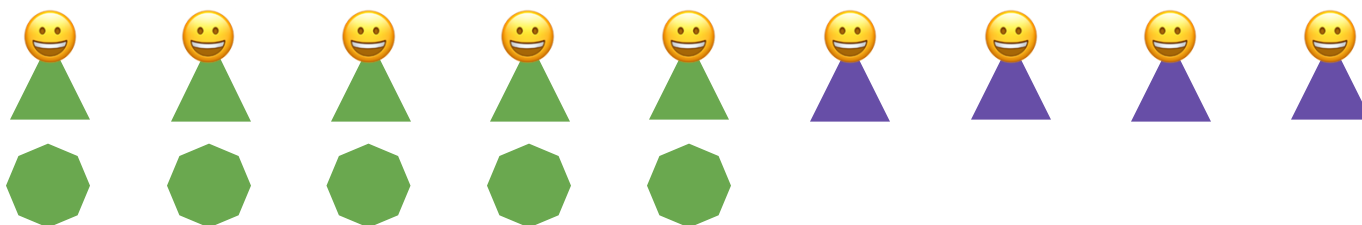


n *happy* agents

one *trigger* agent



n *greedy* agents



n *happy* agents

TAKEAWAY

undirected graphs

v/s

directed graphs

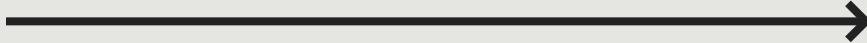
some contrasts, some similarities...

OUTLINE

Background & definitions



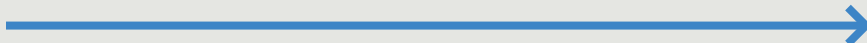
Summary of results



NP-completeness for two agent types



Future work



FUTURE WORK

FUTURE WORK

OPEN PROBLEMS

Is the problem of finding efficient and locally envy-free allocations in XP when parameterized by the vertex cover of the underlying social network?

FUTURE WORK

VALUATIONS

Our study focused completely on 0/1 valuations. More general valuations, including chores, would be interesting to study further!

FUTURE WORK

FAIRNESS CONCEPTS

Note that whenever existence is guaranteed with respect to complete graphs, it is also guaranteed with respect to any social network.