Locally Fair Position Assignments

Simon Rey

Joint work with Giovanni Varricchione and Ronald de Haan

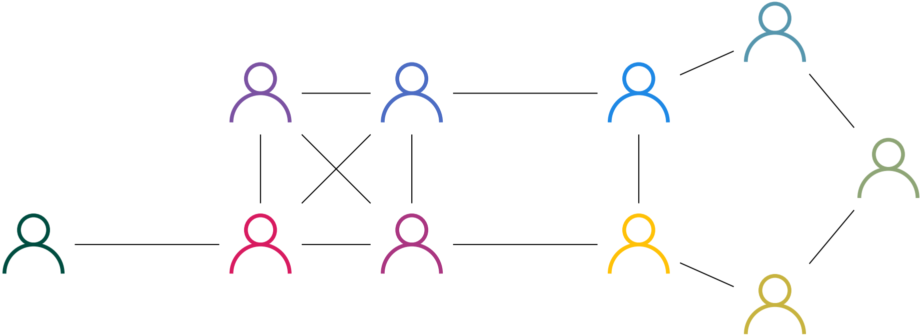
Institute for Logic, Language and Computation (ILLC)
University of Amsterdam

Intelligent Systems Group
University of Utrecht

2022 SSCW Meeting – Computational and Topological Social Choice Workshop

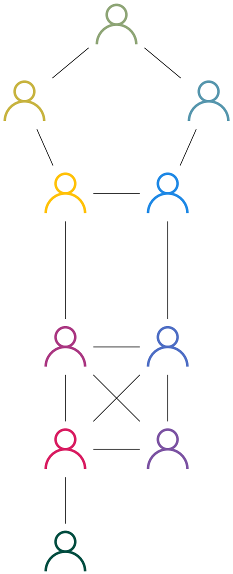
1. The Task at Hand

Positioning Agents on a Graph

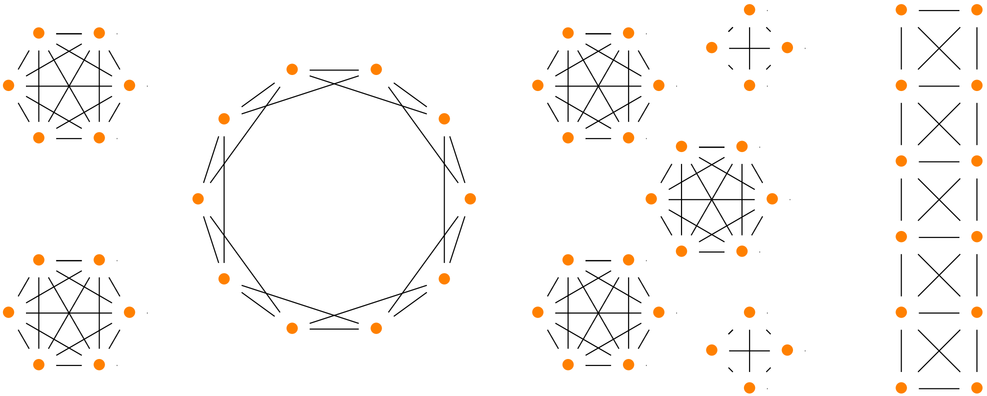


Fair Position Assignment

The utility of the agents is
a function of their neighborhood

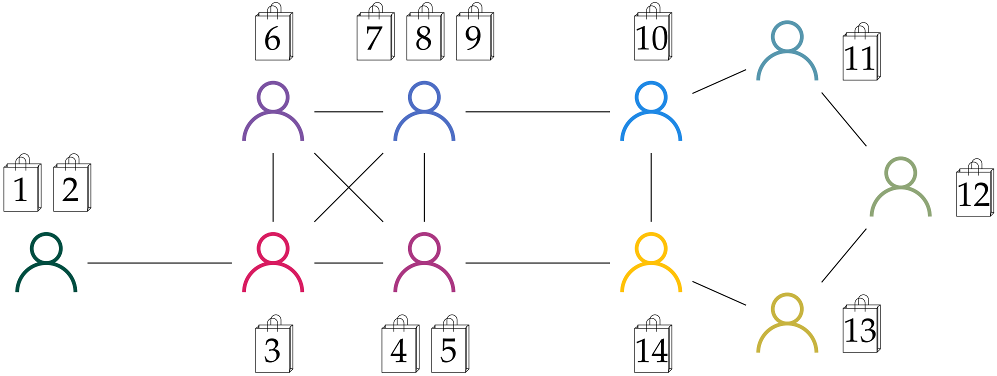


Application – Wedding Tables



Finding a position assignment such that no one is sitting next to someone they do not like.

Application – Fair Allocation on a Social Network



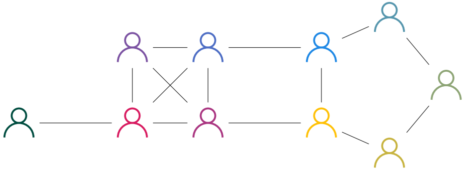
Finding a fair position assignment assuming the items have already been distributed.

Finding a fair position assignment and item allocation.

2. Formalities

Fair Allocation on a Social Network

A social network

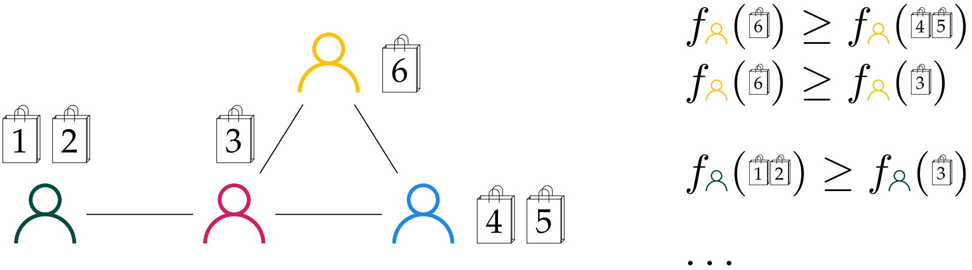


A set of items

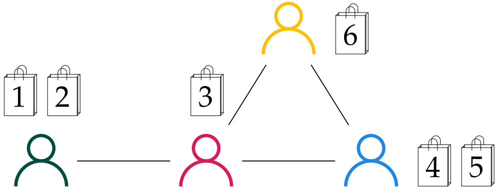
Additive utility functions

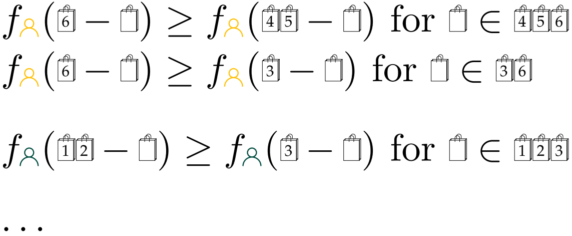
Local Envy-Freeness

Local Envy-Freeness (LEF): no agent envies another agent from their neighborhood.



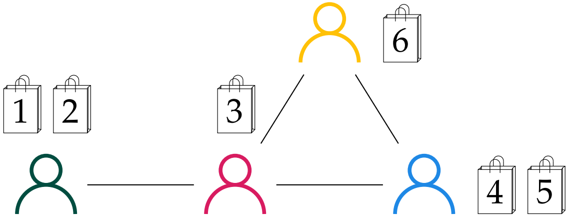
Local Envy-Freeness up to one Item (LEF1): agents can envy their neighbors but by no more than one item.





Local Proportionality

Local Proportionality: The utility of an agent is at least $1/n'+1$ the utility they would have by also owning all the items of their n' neighbors (in addition to their owns).



$$f\left(\boxed{3}\right) \geq 1/4\, f\left(\boxed{1}\boxed{2}\boxed{3}\boxed{4}\boxed{5}\boxed{6}\right)$$

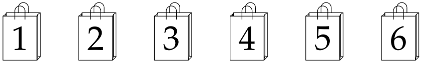
$$f\left(\boxed{4}\boxed{5}\right) \geq 1/3\, f\left(\boxed{3}\boxed{4}\boxed{5}\boxed{6}\right)$$

$$f\left(\boxed{6}\right) \geq 1/3\, f\left(\boxed{3}\boxed{4}\boxed{5}\boxed{6}\right)$$

$$f\left(\boxed{1}\boxed{2}\right) \geq 1/2\, f\left(\boxed{1}\boxed{2}\boxed{3}\right)$$

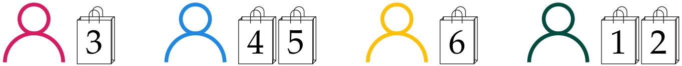
A set of agents

A set of items

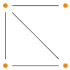


Additive utility functions

An allocation



A graph



Is there a position assignment of the agents to the graph that satisfy $\mathcal{F}$ given the allocation?

3. Classical Complexity Analysis

Easy and Hard Problems

We are interested in understanding how to classify the problem based on graph structure:

Easy case

There exists a polynomial time (in n and m) algorithm deciding whether a suitable position assignment exists

Hard case

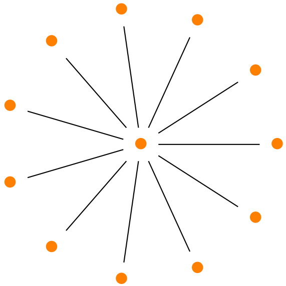
We can only hope for exponential time (in n and m) algorithm deciding whether a suitable position assignment exists (unless $\mathtt{P} = \mathtt{NP}$)

An Easy Start: Stars

On a star, a position assignment is LEF if and only if there exists an agent who does not envy any other agent, and who is not envied by any other agent

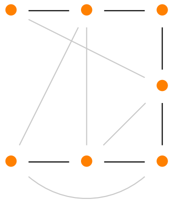
Same idea for LEF1 and LPROP

EASY CASE



Which Turns Short: Lines

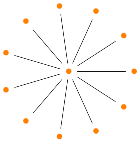
We can find an equivalence between finding a Hamiltonian path in a graph and an LEF/LEF1/LPROP position assignment



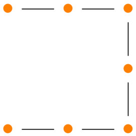
Hard case

The Picture so Far

Easy case



Hard case



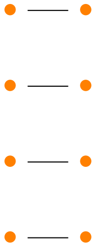
Connected graphs in general

Let's have a look at disconnected graphs!

Another Easy Case: Matchings

Envy-Free Graph: There is an edge between two agents if and only if none of the two envies the other

On a matching, a position assignment is LEF if and only if it corresponds to a perfect matching of the envy-free graph. Finding a perfect matching in a graph can be standardly done in polynomial time.



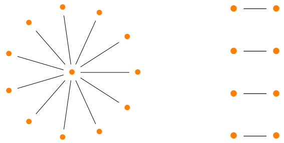
In this case LEF and LPROP are equivalent.

The same idea can be used for LEF1.

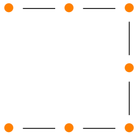
Easy case

The Final Picture

EASY CASE



HARD CASE



(Dis)-Connected graphs

Let's look at a more fine-grained analysis now!

4. Fine-Grained Analysis

Moving from Classical Complexity to Parameterized Complexity

We search for algorithms polynomial in n *and* m. What if we allow for exponential running time but only in one of the two (or other parameters)?

A First Trivial Algorithm

Previous Results: The problem is hard when considering any graph as input.

Naive algorithm polynomial in m and 2^n: run through all position assignments.

↳ Can we reduce the exponential factor?

Yes We Can! On Trees

Vertex type: Two vertices have the same vertex type if the two subtrees they induce are identical up to a bijection.

Agent type: Two agents have the same agent type if they envy and are envied by the same agents.

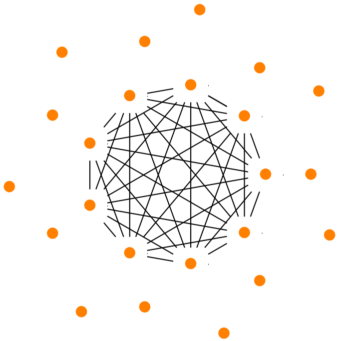
For trees, there exists an algorithm for LEF and LEF1 running in time polynomial in m and $2^{n_{vt}+n_{at}}$ where n_{vt} is the number of vertex type and n_{at} is the number of agent types.

Note: There is no hope for an algorithm polynomial in only $2^{n_{vt}}$.

↳ Is this specific to trees?

A Hardness Result

For graphs consisting of the union of a complete graph and some isolated vertices, there is no hope for algorithms for LEF and LEF1 running in time polynomial in m and 2^{n_i} where n_i is the number of isolated vertices.



5. Conclusion

Wrapping-Up

We have presented a typical analysis of a problem using classical and parameterized complexity. Focusing on the problem of position assignment we have:

- Presented easy cases;
- Discussed hard cases; and,
- Explored hard and easy parameterized algorithms.

The idea of this talk was mainly to showcase a typical computer science approach. Several extensions can be considered.

THANKS