The Submodular Santa Claus Problem in the Restricted Assignment Case

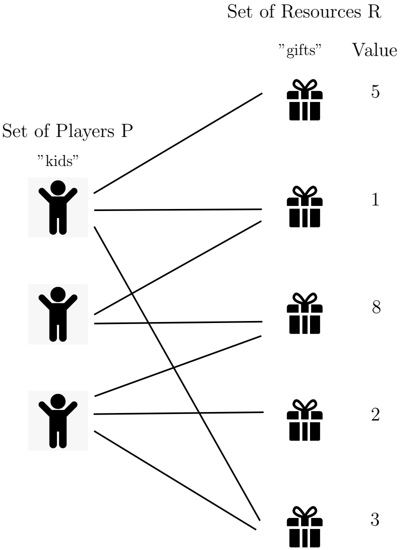
Étienne Bamas, Paritosh Garg, Lars Rohwedder

École polytechnique fédérale de Lausanne

EPFL

ICALP2021

The (Linear) Santa Claus Problem

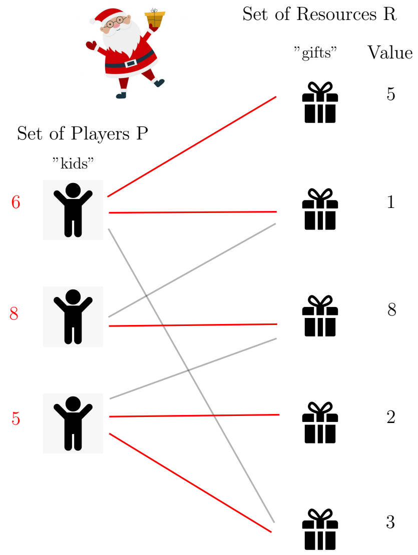


- A set of resources R and a set of players P. Each resource j has a value p_j.
- Assignment restrictions given by some bipartite graph.
- **Goal:** Find an assignment $\sigma : R \mapsto P$ respecting the restrictions such that

$$\min_{i \in P} \sum_{j \in \sigma^{-1}(i)} p_j$$

is maximized, i.e. make the least happy kid as happy as possible!

The (Linear) Santa Claus Problem



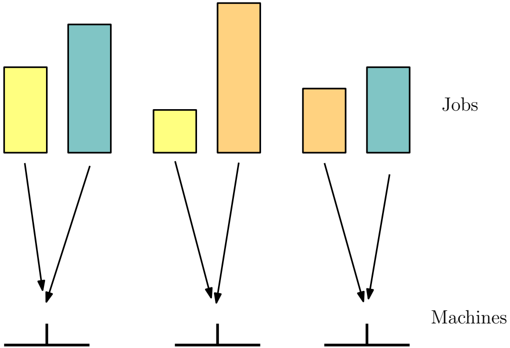
Santa is only as happy as the least happy kid.

The (Linear) Santa Claus Problem

- A very natural problem.

The (Linear) Santa Claus Problem

- A very natural problem.
- "Dual" problem of the famous makespan minimization problem. Minmax becomes maxmin.



The **Submodular** Santa Claus Problem

An equivalent formulation of the **linear** Santa Claus:

Given a **linear** function $f : 2^R \mapsto \mathbb{R}_+$, find an assignment (with assignment restrictions) $\sigma : R \mapsto P$ such that

$$\min_{i \in P} \quad f\left(\sigma^{-1}(i)\right)$$

is maximized.

What happens if f becomes a **submodular** function?

The **Submodular** Santa Claus Problem

An equivalent formulation of the **linear** Santa Claus:

Given a **linear** function $f : 2^R \mapsto \mathbb{R}_+$, find an assignment (with assignment restrictions) $\sigma : R \mapsto P$ such that

$$\min_{i \in P} \quad f\left(\sigma^{-1}(i)\right)$$

is maximized.

What happens if f becomes a **submodular** function?

Submodular Santa Claus problem (with assignment restrictions).

Why submodular functions?

A very natural property in economics: *diminishing returns*.

Why submodular functions?

A very natural property in economics: *diminishing returns*.

For all $X, Y \subseteq R$, with $X \subseteq Y$ and every $j \in R \setminus Y$,

$$f(Y \cup \{j\}) - f(Y) \leq f(X \cup \{j\}) - f(X).$$

Why submodular functions?

A very natural property in economics: *diminishing returns*.

For all $X, Y \subseteq R$, with $X \subseteq Y$ and every $j \in R \setminus Y$,

$$f(Y \cup \{j\}) - f(Y) \leq f(X \cup \{j\}) - f(X).$$

You are alone without food in the desert, do you think that your happiness f satisfies

$f(10\times$  $) = 10 \times f($  $)?$

Why submodular functions?

A very natural property in economics: *diminishing returns*.

For all $X, Y \subseteq R$, with $X \subseteq Y$ and every $j \in R \setminus Y$,

$$f(Y \cup \{j\}) - f(Y) \leq f(X \cup \{j\}) - f(X).$$

You are alone without food in the desert, do you think that your happiness f satisfies

$f(10^6 \times$  $) = 10^6 \times f($  $)$?

Why submodular functions?

A very natural property in economics: *diminishing returns*.

For all $X, Y \subseteq R$, with $X \subseteq Y$ and every $j \in R \setminus Y$,

$$f(Y \cup \{j\}) - f(Y) \leq f(X \cup \{j\}) - f(X).$$

You are alone without food in the desert, do you think that your happiness f satisfies

$f(10^6 \times$  $) = 10^6 \times f($  $)?$

Some problems become more difficult with submodular functions, but also more interesting! For instance, maximizing global welfare.

Previous results

In the **linear** case, very well studied problem.

- Introduced by Bansal and Srividenko (STOC'06) who gives an $O(\log\log(m)/\log\log\log(m))$-approximation algorithm (with $m = |P|$).
- Numerous improvements over the years on the approximation guarantee, the technique and/or the running time. The current best approximation is a $(4+\epsilon)$-approximation in polynomial time (Davies, Rothvoss, and Zhang SODA'20, Cheng and Mao ICALP'19).

Previous results

In the **linear** case, very well studied problem.

- Introduced by Bansal and Srividenko (STOC'06) who gives an $O(\log\log(m)/\log\log\log(m))$-approximation algorithm (with $m = |P|$).
- Numerous improvements over the years on the approximation guarantee, the technique and/or the running time. The current best approximation is a $(4+\epsilon)$-approximation in polynomial time (Davies, Rothvoss, and Zhang SODA'20, Cheng and Mao ICALP'19).

In the **submodular** case, a more general result by Goemans, Harvey, Iwata, and Mirrokni (SODA'09) implies a $O(n^{1/2+\epsilon})$-approximation in polynomial time (where $n = |R|$) in the restricted assignment case.

Previous results

In the **linear** case, very well studied problem.

- Introduced by Bansal and Srividenko (STOC'06) who gives an $O(\log\log(m)/\log\log\log(m))$-approximation algorithm (with $m = |P|$).
- Numerous improvements over the years on the approximation guarantee, the technique and/or the running time. The current best approximation is a $(4+\epsilon)$-approximation in polynomial time (Davies, Rothvoss, and Zhang SODA'20, Cheng and Mao ICALP'19).

In the **submodular** case, a more general result by Goemans, Harvey, Iwata, and Mirrokni (SODA'09) implies a $O(n^{1/2+\epsilon})$-approximation in polynomial time (where $n = |R|$) in the restricted assignment case.

Our result:

Theorem

There exists a $O(\log\log(n))$-approximation algorithm running in polynomial time for the Submodular Santa Claus in the Restricted Assignment case.

The Configuration LP

The Configuration LP introduced by Bansal and Srividenko in 2006.

Guess the optimum is T. Then a configuration $C \in \mathcal{C}(i, T)$ is a subset of resources that player i values to at least T.

$$\sum_{C \in \mathcal{C}(i,T)} x_{i,C} \geq 1 \quad \text{for all } i \in P \quad \textit{each player gets enough value}$$

$$\sum_{i \in P} \sum_{C \in \mathcal{C}(i,T): j \in C} x_{i,C} \leq 1 \quad \text{for all } j \in R \quad \textit{no resource is taken more than once}$$

$$x_{i,C} \geq 0 \quad \text{for all } i \in P, C \in \mathcal{C}(i, T)$$

Previous Techniques

Theorem (Bansal and Srividenko)

The configuration LP can be solved within a factor $(1 + \epsilon)$ in polynomial time.

Two rounding techniques against the Configuration LP in the **linear** case:

- Bansal and Srividenko (STOC'06) used Lovász Local Lemma.
- Asadpour, Feige, and Saberi (APPROX'08) introduced a Local Search Technique.

Both of them are based on finding a matching in some hypergraph.

The approach of Bansal and Srividenko

1. Compute x^* a feasible solution to the configuration LP with objective T^*.

The approach of Bansal and Srividenko

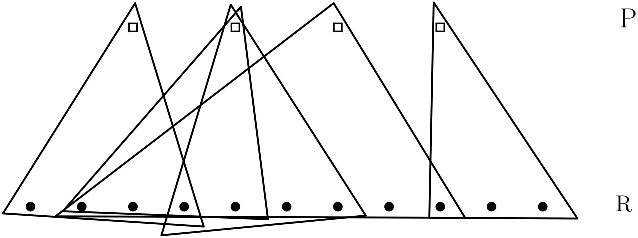
1. Compute x^* a feasible solution to the configuration LP with objective T^*.
2. Preprocess the solution x^* to reduce to some problem in which we have $\log(n)$ candidate configurations per player, with **unit** size resources.

The approach of Bansal and Srividenko

1. Compute x^* a feasible solution to the configuration LP with objective T^*.
2. Preprocess the solution x^* to reduce to some problem in which we have $\log(n)$ candidate configurations per player, with **unit** size resources.
3. Find a good choice of configurations using Lovász Local Lemma.

The approach of Bansal and Srividenko

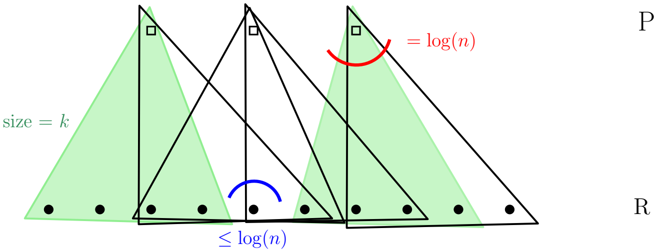
A hypergraph $\mathcal{H} = (P \cup R, \mathcal{C})$ is **bipartite** if for all hyperedges $C \in \mathcal{C}$ we have $|C \cap P| = 1$.



A bipartite hypergraph matching problem

Given a **regular** and **uniform** bipartite hypergraph, find for each vertex $i \in P$ one hyperedge C_i such that:

1. $i \in C_i$, and player i is assigned a good fraction of resources in C_i.
2. No resource is taken more than $\log\log(n)$ times.



Given a **regular** and **uniform** bipartite hypergraph, find for each vertex $i \in P$ one hyperedge C_i such that:

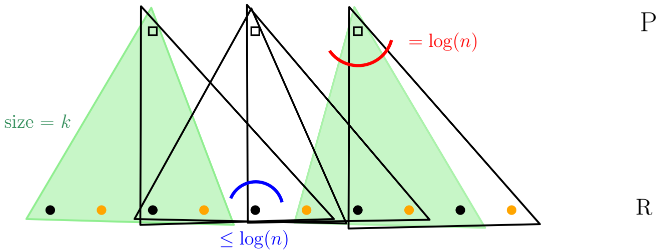
1. $i \in C_i$, and player i is assigned a good fraction of resources in C_i.
2. No resource is taken more than $\log \log(n)$ times.

Given a **regular** and **uniform** bipartite hypergraph, find for each vertex $i \in P$ one hyperedge C_i such that:

1. $i \in C_i$, and player i is assigned a good fraction of resources in C_i.
2. No resource is taken more than $\log\log(n)$ times.

Solution:

1. Keep each resource in R with probability $\log(n)/k$.
2. Sample one hyperedge for each player using LLL.

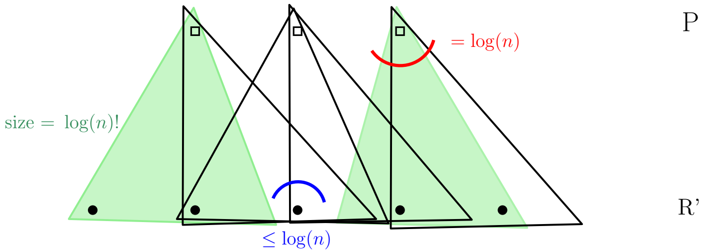


Given a **regular** and **uniform** bipartite hypergraph, find for each vertex $i \in P$ one hyperedge C_i such that:

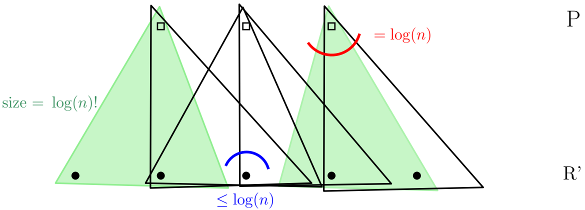
1. $i \in C_i$, and player i is assigned a good fraction of resources in C_i.
2. No resource is taken more than $\log\log(n)$ times.

Solution:

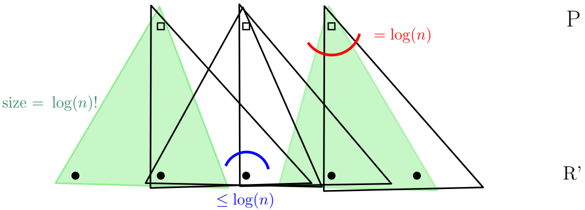
1. Keep each resource in R with probability $\log(n)/k$.
2. Sample one hyperedge for each player using LLL.



Why does it work?

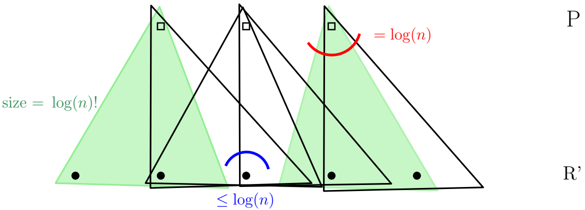


Why does it work?



Sampling: Each player i selects one of his $\log(n)$ configurations uniformly at random.

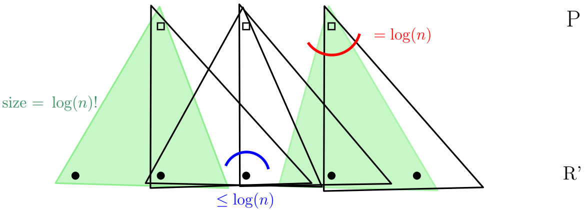
Why does it work?



Sampling: Each player i selects one of his $\log(n)$ configurations uniformly at random.

Bad event: $B_j = \{$resource j is taken more than $\log\log(n)$ times$\}$.

Why does it work?

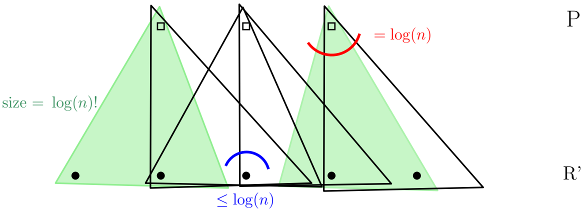


Sampling: Each player i selects one of his $\log(n)$ configurations uniformly at random.

Bad event: $B_j = \{$resource j is taken more than $\log\log(n)$ times$\}$.

1. $\mathbb{P}(B_j) \leq 1/\text{polylog}(n)$ (the expectation is at most 1).

Why does it work?

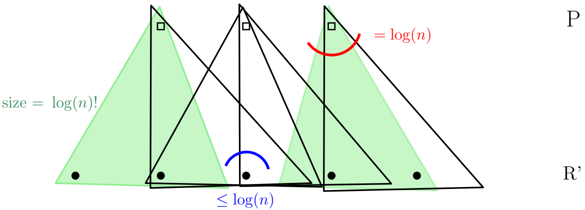


Sampling: Each player i selects one of his $\log(n)$ configurations uniformly at random.

Bad event: $B_j = \{$resource j is taken more than $\log\log(n)$ times$\}$.

1. $\mathbb{P}(B_j) \leq 1/\text{polylog}(n)$ (the expectation is at most 1).
2. B_j depends on $\text{polylog}(n)$ other $B_{j'}$. Apply Lovász Local Lemma with 1 and 2.

Why does it work?



Sampling: Each player i selects one of his $\log(n)$ configurations uniformly at random.

Bad event: $B_j = \{$resource j is taken more than $\log\log(n)$ times$\}$.

1. $\mathbb{P}(B_j) \leq 1/\text{polylog}(n)$ (the expectation is at most 1).
2. B_j depends on polylog(n) other $B_{j'}$. Apply Lovász Local Lemma with 1 and 2.
3. There is a good solution **after** sampling down if and only if there is a good solution **before** sampling down. I.e. R' is **representative** enough of R.

The submodular case

Theorem

The Configuration LP can be solved in polynomial time within constant factor in the case where f is submodular.

The submodular case

Theorem

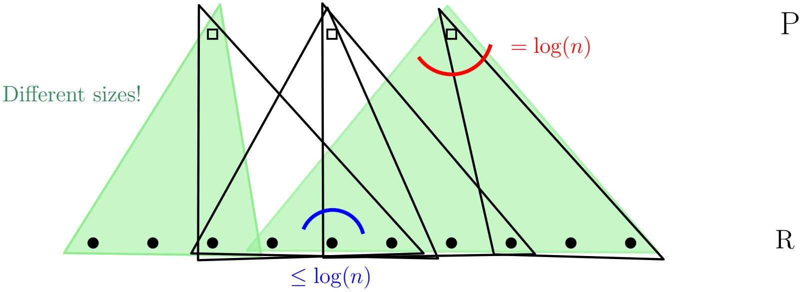
The Configuration LP can be solved in polynomial time within constant factor in the case where f is submodular.

By **submodularity**, we can preprocess the solution x^* in a similar way as Bansal and Srividenko to arrive at some hypergraph problem.

A **new** bipartite hypergraph matching problem

Given a **regular** and **non-uniform** bipartite hypergraph, find for each vertex $i \in P$ one hyperedge C_i such that:

1. $i \in C_i$, and player i is assigned a good fraction of resources in C_i.
2. No resource is taken more than $\log\log(n)$ times in $\cup_{i \in P} C_i$.



A **new** bipartite hypergraph matching problem

Non-uniformity introduces significant problems in the approach of Bansal and Srividenko. How do we sample down?

- Sampling down too aggressively might create false positive. R' is not representative of R anymore.
- Not being aggressive enough fails to reduce dependencies enough, hence LLL does not work!

A **new** bipartite hypergraph matching problem

Non-uniformity introduces significant problems in the approach of Bansal and Srividenko. How do we sample down?

- Sampling down too aggressively might create false positive. R' is not representative of R anymore.
- Not being aggressive enough fails to reduce dependencies enough, hence LLL does not work!

General intuition of our solution:

- Select hyperedges so that every hyperedge C intersects other hyperedges $C', |C'| \leq |C|$ not too many times.
- Then iterate from bigger to smaller hyperedges. When small hyperedges arrive, allow them to steal resources from big hyperedges that appeared earlier.

Our solution

- Partition the hyperedges according to their size, $\mathcal{C} = \mathcal{C}^{(1)} \cup \mathcal{C}^{(2)} \cup \ldots \cup \mathcal{C}^{(\log(n))}$ where $C^{(k)}$ contains all the hyperedges such that $|C| \in [\log^{k-1}(n), \log^{k}(n))$.

Our solution

- Partition the hyperedges according to their size, $\mathcal{C} = \mathcal{C}^{(1)} \cup \mathcal{C}^{(2)} \cup \ldots \cup \mathcal{C}^{(\log(n))}$ where $C^{(k)}$ contains all the hyperedges such that $|C| \in [\log^{k-1}(n), \log^{k}(n))$.
- Build a hierarchy of resources sets $R_0 (= R), R_1, R_2, \ldots, R_{\log(n)}$ where each item from R_i survives into R_{i+1} with probability $1/\log(n)$.

Our solution

- Partition the hyperedges according to their size, $\mathcal{C} = \mathcal{C}^{(1)} \cup \mathcal{C}^{(2)} \cup \ldots \cup \mathcal{C}^{(\log(n))}$ where $C^{(k)}$ contains all the hyperedges such that $|C| \in [\log^{k-1}(n), \log^{k}(n))$.
- Build a hierarchy of resources sets $R_0(= R), R_1, R_2, \ldots, R_{\log(n)}$ where each item from R_i survives into R_{i+1} with probability $1/\log(n)$.
- Use LLL to sample one hyperedge for each player such that, for all $C \in \mathcal{C}^{(k)}$,

$$\sum_{C' \in \mathcal{C}^{(h)}, C' \text{ sampled}} |C \cap C' \cap R_h|$$

is not too big for all $h \leq k$.

Our solution

Why does it work?

- With high probability, $|C' \cap R_h| \leq \log(n)$ for all $C' \in \mathcal{C}^{(h)}$. It recovers the **low dependencies** property of Bansal and Srividenko and is enough to apply LLL.

Our solution

Why does it work?

- With high probability, $|C' \cap R_h| \leq \log(n)$ for all $C' \in \mathcal{C}^{(h)}$. It recovers the **low dependencies** property of Bansal and Srividenko and is enough to apply LLL.

- The intersection with other configurations is still big enough so that it is **representative**.

Conclusion

- **Submodularity** is captured by the **non-uniformity** of our hypergraph.
- We obtain an $O(\log\log(n))$-approximate solution in polynomial time.
- Getting $O(1)$-approximation is an interesting open problem.
- What about the local search technique?
- What about more general valuation functions?

Thank you for your attention!