On a Parameterized Problem in Access Control

Rémi Watrigant

Royal Holloway University of London

Joint work with: Jason Crampton and Gregory Gutin

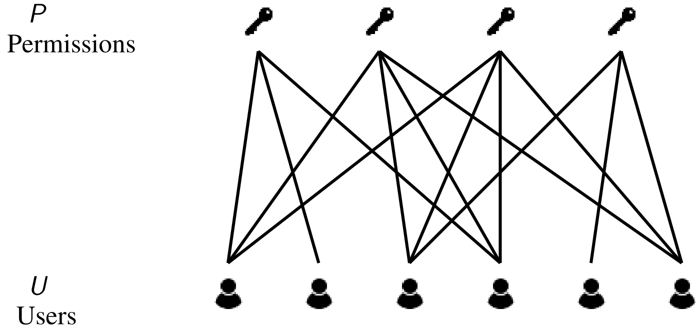
Séminaire COATI - Sophia Antipolis
March 15th, 2016.

1. Definition of the problem
2. Appetizer: easy observations
3. FPT algorithms
4. Efficient algorithms and implementation
5. Conclusion

Resiliency Checking Problem (RCP)

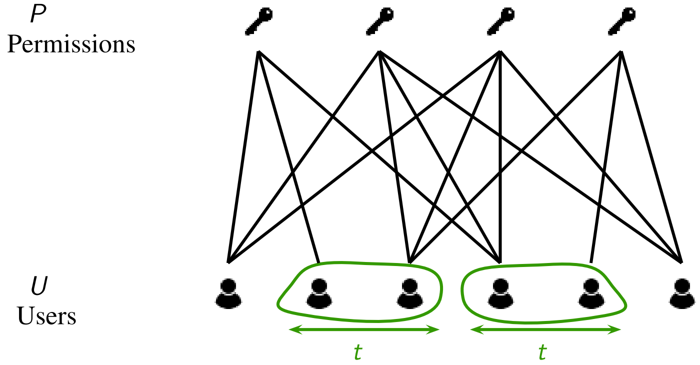
Input: an authorization policy: $UP \subseteq U \times P$

$s, d, t \in \mathbb{N}$



Resiliency Checking Problem (RCP)

Input: an authorization policy: $UP \subseteq U \times P$
$s, d, t \in \mathbb{N}$

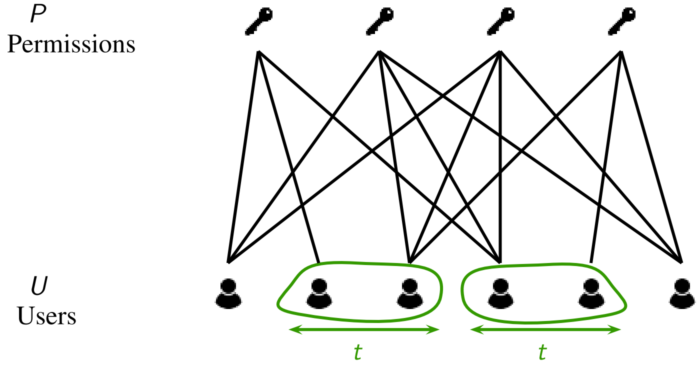


Set of teams: d mutually disjoint sets of t users having collectively all permissions.

Resiliency Checking Problem (RCP)

Input: an authorization policy: $UP \subseteq U \times P$
$s, d, t \in \mathbb{N}$

Output: decide whether upon removal of any set of s users, there still exists a set of d teams of size t

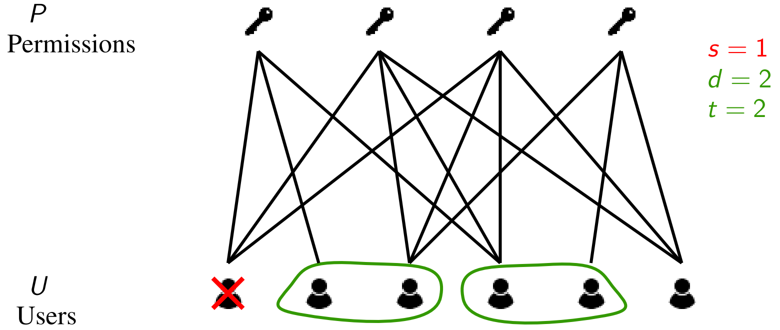


Set of teams: d mutually disjoint sets of t users having collectively all permissions.

Resiliency Checking Problem (RCP)

Input: an authorization policy: $UP \subseteq U \times P$
$s, d, t \in \mathbb{N}$

Output: decide whether upon removal of any set of s users, there still exists a set of d teams of size t

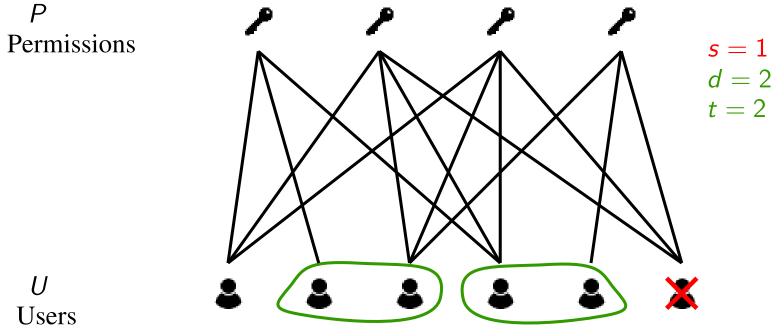


Set of teams: d mutually disjoint sets of t users having collectively all permissions.

Resiliency Checking Problem (RCP)

Input: an authorization policy: $UP \subseteq U \times P$
$s, d, t \in \mathbb{N}$

Output: decide whether upon removal of any set of s users, there still exists a set of d teams of size t

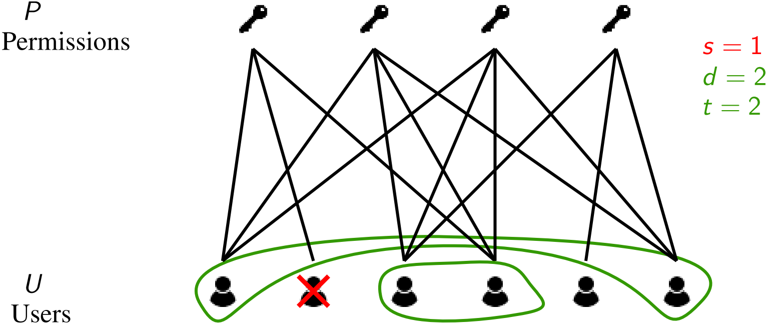


Set of teams: d mutually disjoint sets of t users having collectively all permissions.

Resiliency Checking Problem (RCP)

Input: an authorization policy: $UP \subseteq U \times P$
$s, d, t \in \mathbb{N}$

Output: decide whether upon removal of any set of s users, there still exists a set of d teams of size t

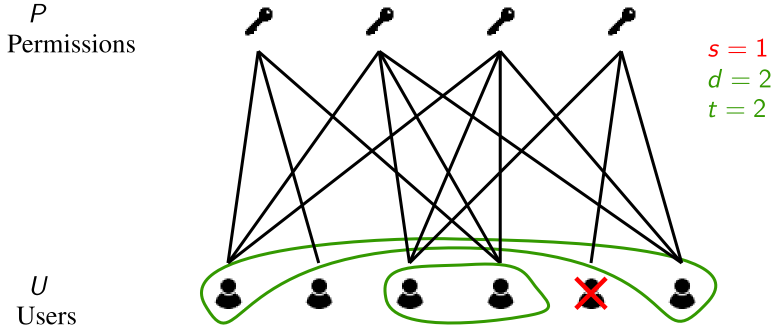


Set of teams: d mutually disjoint sets of t users having collectively all permissions.

Resiliency Checking Problem (RCP)

Input: an authorization policy: $UP \subseteq U \times P$
$s, d, t \in \mathbb{N}$

Output: decide whether upon removal of any set of s users, there still exists a set of d teams of size t



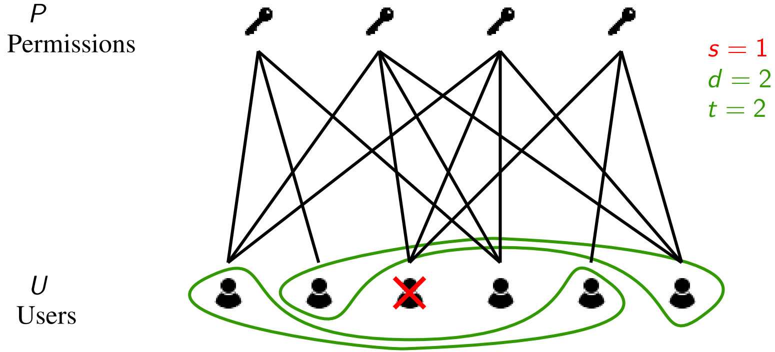
Set of teams: d mutually disjoint sets of t users having collectively all permissions.

Resiliency Checking Problem (RCP)

Input: an authorization policy: $UP \subseteq U \times P$

$s, d, t \in \mathbb{N}$

Output: decide whether upon removal of any set of s users, there still exists a set of d teams of size t

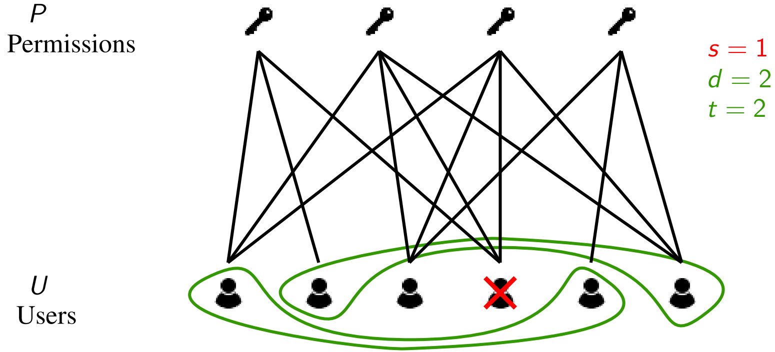


Set of teams: d mutually disjoint sets of t users having collectively all permissions.

Resiliency Checking Problem (RCP)

Input: an authorization policy: $UP \subseteq U \times P$
$s, d, t \in \mathbb{N}$

Output: decide whether upon removal of any set of s users, there still exists a set of d teams of size t



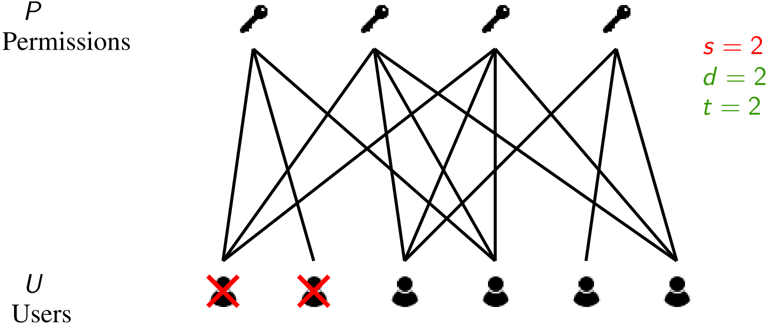
Set of teams: d mutually disjoint sets of t users having collectively all permissions.

Resiliency Checking Problem (RCP)

Input: an authorization policy: $UP \subseteq U \times P$

$s, d, t \in \mathbb{N}$

Output: decide whether upon removal of any set of s users, there still exists a set of d teams of size t



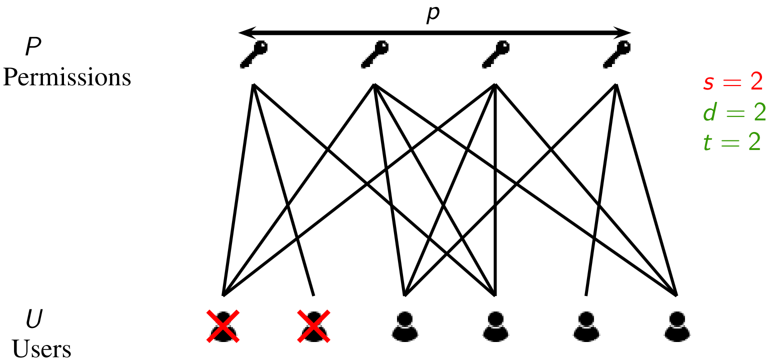
Set of teams: d mutually disjoint sets of t users having collectively all permissions.

Blocker set: set of users which intersects every set of teams.

Resiliency Checking Problem (RCP)

Input: an authorization policy: $UP \subseteq U \times P$
$s, d, t \in \mathbb{N}$

Output: decide whether upon removal of any set of s users, there still exists a set of d teams of size t



Set of teams: d mutually disjoint sets of t users having collectively all permissions.
Blocker set: set of users which intersects every set of teams.

Parameterized algorithms

For a problem instance x coming with its parameter k:

Parameterized algorithms

For a problem instance x coming with its parameter k:

- *XP* if you can solve it in $O(|x|^{f(k)})$

Parameterized algorithms

For a problem instance x coming with its parameter k:

- *XP* if you can solve it in $O(|x|^{f(k)})$
- *FPT* if you can solve it in $O(f(k)|x|^{O(1)})$

Parameterized algorithms

For a problem instance x coming with its parameter k:

- *XP* if you can solve it in $O(|x|^{f(k)})$
- para-NP-hard: NP-hard when k is fixed to some constant
  $\implies$ no *XP* algorithm unless $P = NP$
- *FPT* if you can solve it in $O(f(k)|x|^{O(1)})$

Parameterized algorithms

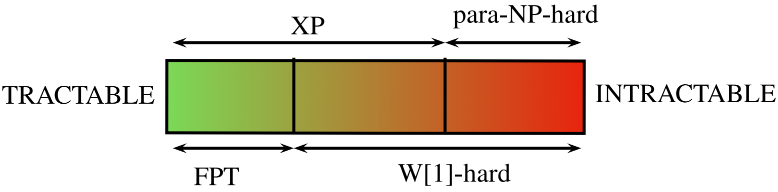
For a problem instance x coming with its parameter k:

- XP if you can solve it in $O(|x|^{f(k)})$
- para-NP-hard: NP-hard when k is fixed to some constant
  $\implies$ no XP algorithm unless $P = NP$
- FPT if you can solve it in $O(f(k)|x|^{O(1)})$
- $W[1]$-hardness: parameter-preserving reduction from a $W[1]$-hard problem
  $\implies$ no FPT algorithm unless $FPT = W[1]$

Parameterized algorithms

For a problem instance x coming with its parameter k:

- XP if you can solve it in $O(|x|^{f(k)})$
- para-NP-hard: NP-hard when k is fixed to some constant
  $\implies$ no XP algorithm unless $P = NP$
- FPT if you can solve it in $O(f(k)|x|^{O(1)})$
- $W[1]$-hardness: parameter-preserving reduction from a $W[1]$-hard problem
  $\implies$ no FPT algorithm unless $FPT = W[1]$



Related work

RCP

Input: $UP \subseteq U \times P$, $s, d, t \in \mathbb{N}$

Output: upon removal of any set of s users, are there still d teams of size t ?

Related work

RCP

Input: $UP \subseteq U \times P$, $s, d, t \in \mathbb{N}$

Output: upon removal of any set of s users, are there still d teams of size t ?

Li, Tripunitara, Wang, 2009

- RCP$<>$, RCP$<d = 1>$ and RCP$<t = \infty>$ are NP-hard and in $coNP^{NP}$.
- RCP$<s = 0,\ d = 1>$ and RCP$<s = 0,\ t = \infty>$ are NP-hard.
- RCP$<d = 1,\ t = \infty>$ is linear-time solvable.

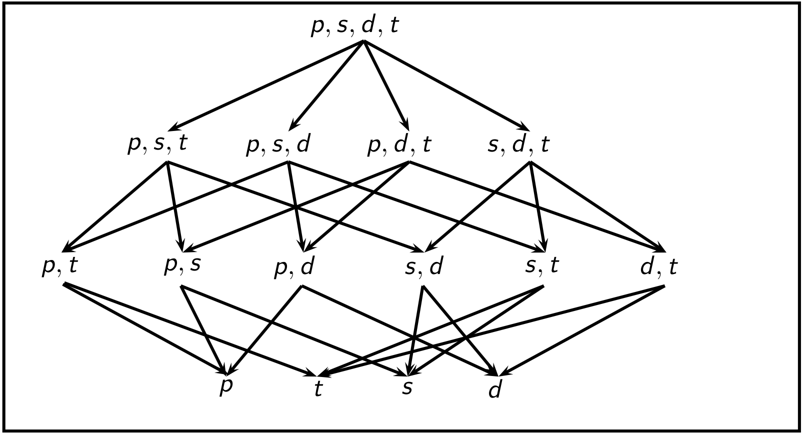
And they present an implementation of an algorithm relying on a SAT formulation.

Related work

RCP

Input: $UP \subseteq U \times P$, $s, d, t \in \mathbb{N}$

Output: upon removal of any set of s users, are there still d teams of size t ?



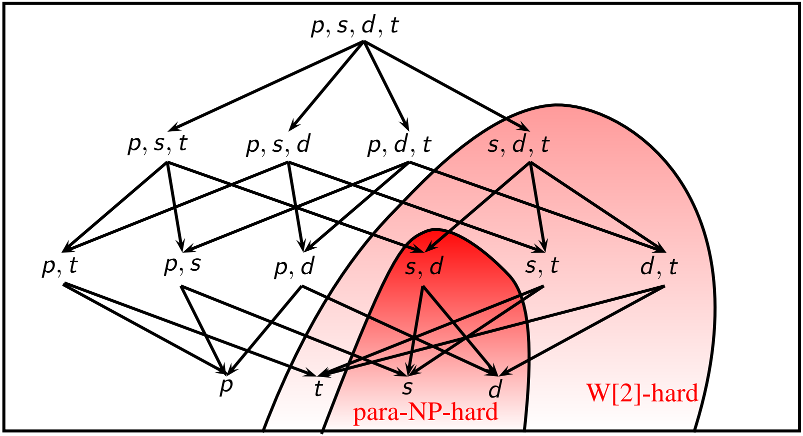
Related work

RCP

Input: $UP \subseteq U \times P,\ s, d, t \in \mathbb{N}$

Output: upon removal of any set of s users, are there still d teams of size t ?

RCP<$s = 0,\ d = 1$> is equivalent to the Hitting Set problem



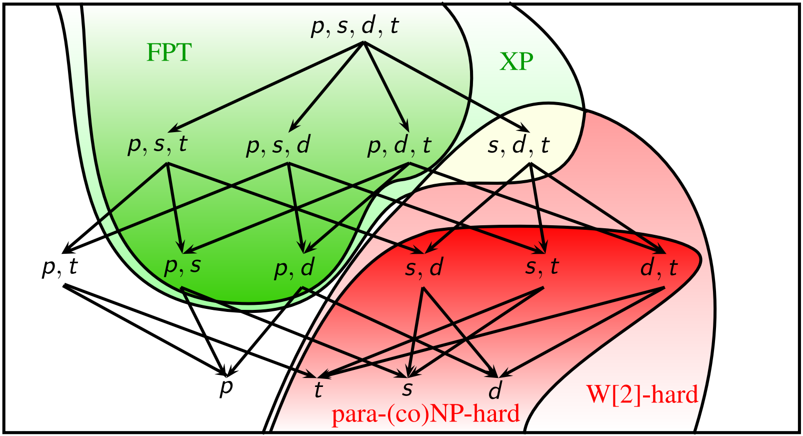
Related work

RCP

Input: $UP \subseteq U \times P$, $s, d, t \in \mathbb{N}$

Output: upon removal of any set of s users, are there still d teams of size t ?

Results obtained for RCP<> [Crampton, Gutin, W.]



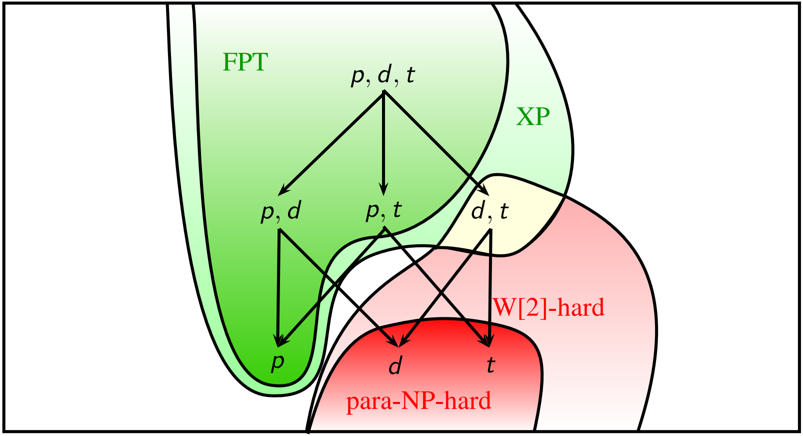
Related work

RCP

Input: $UP \subseteq U \times P$, $s, d, t \in \mathbb{N}$

Output: upon removal of any set of s users, are there still d teams of size t ?

Results obtained for RCP<$s = 0$> [Crampton, Gutin, W.]



Easy Observations

RCP

Input: $UP \subseteq U \times P$, $s, d, t \in \mathbb{N}$

Output: upon removal of any set of s users, are there still d teams of size t ?

- RCP$<s = 0>$ is in XP parameterized by (d, t) (brute force)

Easy Observations

RCP

Input: $UP \subseteq U \times P$, $s, d, t \in \mathbb{N}$

Output: upon removal of any set of s users, are there still d teams of size t ?

- RCP<$s = 0$> is in XP parameterized by (d, t) (brute force)
- $\Rightarrow$ RCP<> is in XP parameterized by (s, d, t) (branching)

Easy, but:

Easy Observations

RCP

Input: $UP \subseteq U \times P$, $s, d, t \in \mathbb{N}$

Output: upon removal of any set of s users, are there still d teams of size t ?

- $\text{RCP}{<}s = 0{>}$ is in XP parameterized by (d, t) (brute force)
- $\Rightarrow$ $\text{RCP}{<}{>}$ is in XP parameterized by (s, d, t) (branching)

Easy, but:

- $W[2]$-hard parameterized by (s, d, t)

Easy Observations

RCP

Input: $UP \subseteq U \times P$, $s, d, t \in \mathbb{N}$

Output: upon removal of any set of s users, are there still d teams of size t ?

- RCP<$s = 0$> is in XP parameterized by (d, t) (brute force)
- $\Rightarrow$ RCP<> is in XP parameterized by (s, d, t) (branching)

Easy, but:

- $W[2]$-hard parameterized by (s, d, t)
- para-NP-hard parameterized by (s, d), (d, t) (s, t)

Easy Observations

RCP

Input: $UP \subseteq U \times P$, $s, d, t \in \mathbb{N}$

Output: upon removal of any set of s users, are there still d teams of size t ?

- RCP<$s = 0$> is in XP parameterized by (d, t) (brute force)
- $\Rightarrow$ RCP<> is in XP parameterized by (s, d, t) (branching)

Easy, but:

- $W[2]$-hard parameterized by (s, d, t)
- para-NP-hard parameterized by (s, d), (d, t) (s, t)

What about replacing t by p ? (we may assume $t \leq p$)

Easy Observations

RCP

Input: $UP \subseteq U \times P$, $s, d, t \in \mathbb{N}$

Output: upon removal of any set of s users, are there still d teams of size t ?

- RCP<$s = 0$> is in XP parameterized by (d, t) (brute force)
- $\Rightarrow$ RCP<> is in XP parameterized by (s, d, t) (branching)

Easy, but:

- $W[2]$-hard parameterized by (s, d, t)
- para-NP-hard parameterized by (s, d), (d, t) (s, t)

What about replacing t by p ? (we may assume $t \leq p$)

- RCP<> is FPT parameterized by $(p, min\{s, d\})$

Easy Observations

RCP

Input: $UP \subseteq U \times P$, $s, d, t \in \mathbb{N}$

Output: upon removal of any set of s users, are there still d teams of size t ?

- RCP<$s = 0$> is in XP parameterized by (d, t) (brute force)
- $\Rightarrow$ RCP<> is in XP parameterized by (s, d, t) (branching)

Easy, but:

- $W[2]$-hard parameterized by (s, d, t)
- para-NP-hard parameterized by (s, d), (d, t) (s, t)

What about replacing t by p ? (we may assume $t \leq p$)

- RCP<> is FPT parameterized by $(p, min\{s, d\})$
- first: let us show that RCP<$s = 0$> is FPT parameterized by p only

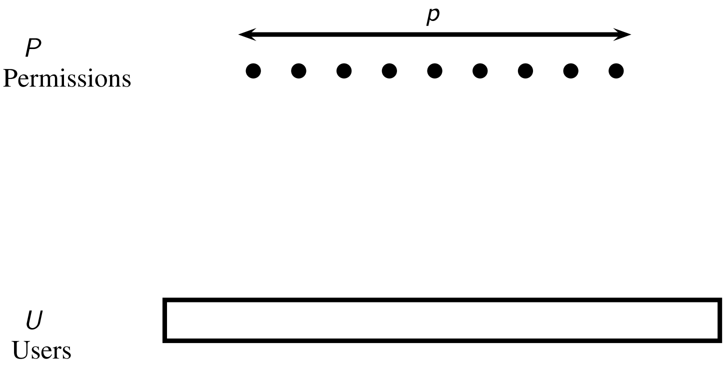
RCP<$s = 0$> is FPT parameterized by p

RCP<$s = 0$> is FPT parameterized by p

Theorem [Lenstra, 1983]+[Kannan, 1987]+[Frank and Tardos, 1987]

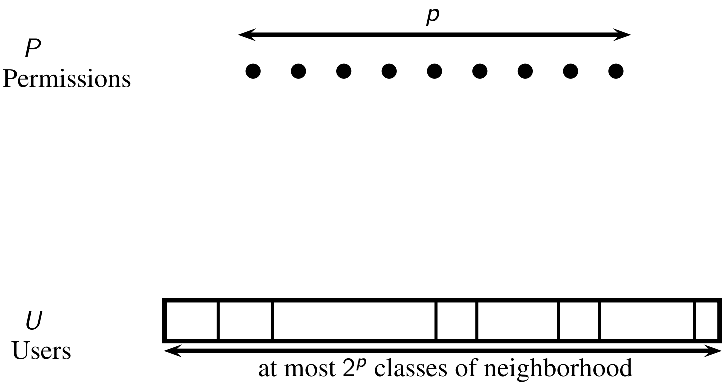
Whether a given ILP has a non-empty solution set can be decided in $O^*(n^{2.5n+o(n)})$ time and polynomial space, where n is the number of variables.

RCP<$s = 0$> is FPT parameterized by p



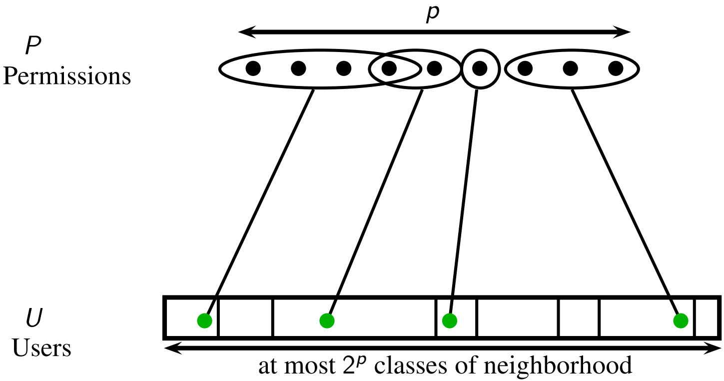
RCP<$s = 0$> is FPT parameterized by p

- partition U into at most 2^p groups of users of same neighborhood



RCP<$s = 0$> is FPT parameterized by p

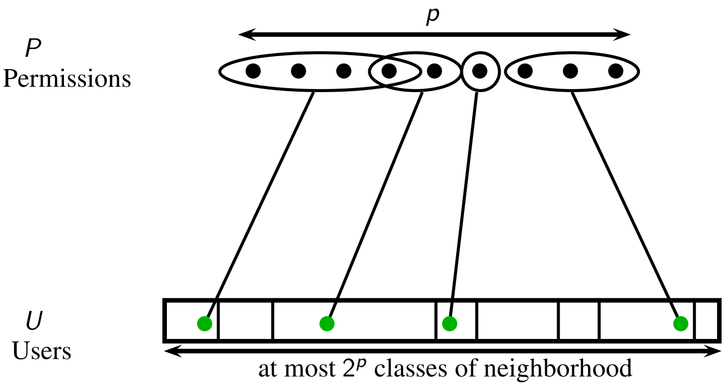
- partition U into at most 2^p groups of users of same neighborhood
- a team $\equiv$ a set of $\leq t$ subsets of P, called configurations:



RCP<$s = 0$> is FPT parameterized by p

- partition U into at most 2^p groups of users of same neighborhood
- a team $\equiv$ a set of $\leq t$ subsets of P, called configurations:

$$\mathcal{C} = \left\{ \{N_1, \ldots, N_b\} : b \leq t, N_i \subseteq P \text{ s.t. } \bigcup_{i=1}^{b} N_i = P \right\}$$



RCP< $s = 0$ > is FPT parameterized by p

- partition U into at most 2^p groups of users of same neighborhood
- a team $\equiv$ a set of $\leq t$ subsets of P, called configurations:

$$\mathcal{C} = \left\{ \{N_1, \ldots, N_b\} : b \leq t, N_i \subseteq P \text{ s.t. } \bigcup_{i=1}^{b} N_i = P \right\}$$

- variables of the ILP:
  for $c \in \mathcal{C}$, $x_c \in [0, d]$ is the number of teams with configuration c

RCP<$s = 0$> is FPT parameterized by p

- partition U into at most 2^p groups of users of same neighborhood
- a team $\equiv$ a set of $\leq t$ subsets of P, called configurations:

$$\mathcal{C} = \left\{ \{N_1, \ldots, N_b\} : b \leq t, N_i \subseteq P \text{ s.t. } \bigcup_{i=1}^{b} N_i = P \right\}$$

- variables of the ILP:
  for $c \in \mathcal{C}$, $x_c \in [0, d]$ is the number of teams with configuration c
- First constraint:

$$\sum_{c \in \mathcal{C}} x_c = d$$

RCP<$s = 0$> is FPT parameterized by p

- partition U into at most 2^p groups of users of same neighborhood
- a team $\equiv$ a set of $\leq t$ subsets of P, called configurations:

$$\mathcal{C} = \left\{ \{N_1, \ldots, N_b\} : b \leq t, N_i \subseteq P \text{ s.t. } \bigcup_{i=1}^{b} N_i = P \right\}$$

- variables of the ILP:
  for $c \in \mathcal{C}$, $x_c \in [0, d]$ is the number of teams with configuration c
- First constraint:

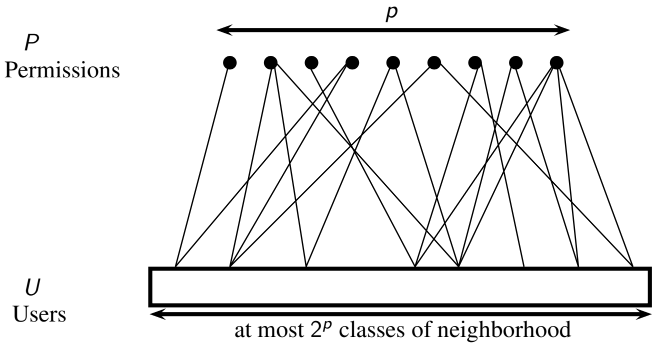
$$\sum_{c \in \mathcal{C}} x_c = d$$

- Second constraint:

$$\sum_{c \in \mathcal{C}[N]} x_c \leq |U[N]| \quad \forall N \subseteq P$$

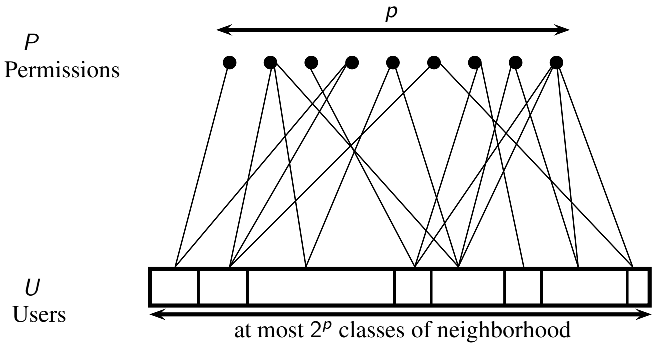
  where:
  - $\mathcal{C}[N]$ are the configurations involving N
  - $U[N]$ users having neighborhood N

RCP<> is FPT parameterized by $(p, min\{s, d\})$



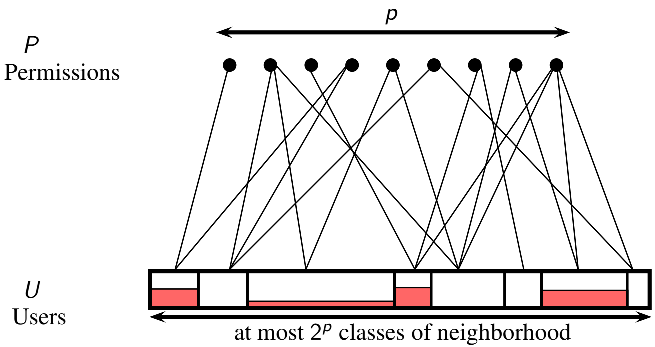
RCP<> is FPT parameterized by $(p, min\{s, d\})$

- partition U into at most 2^p groups of users of same neighborhood



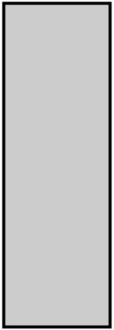
RCP<> is FPT parameterized by $(p, min\{s, d\})$

- partition U into at most 2^p groups of users of same neighborhood
- let $S \subseteq U$ be a blocker set



RCP<> is FPT parameterized by $(p, min\{s, d\})$

- partition U into at most 2^p groups of users of same neighborhood
- let $S \subseteq U$ be a blocker set

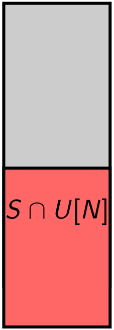


class U[N] (users of neighborhood N)

RCP<> is FPT parameterized by $(p, min\{s, d\})$

- partition U into at most 2^p groups of users of same neighborhood
- let $S \subseteq U$ be a blocker set

Claim: $\forall N \subseteq P,\ U[N] \cap S \neq \emptyset \implies |U[N] \setminus S| \leq d - 1$



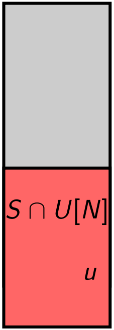
class U[N] (users of neighborhood N)

RCP<> is FPT parameterized by $(p, min\{s, d\})$

- partition U into at most 2^p groups of users of same neighborhood
- let $S \subseteq U$ be a blocker set

Claim: $\forall N \subseteq P,\ U[N] \cap S \neq \emptyset \implies |U[N] \setminus S| \leq d - 1$

- For all $u \in U[N] \cap S$, there exists a set of teams $V_1, \ldots, V_t$ such that:
  - $(\cup V_i) \cap S = \{u\}$
  - $|(\cup V_i) \cap U[N]| \leq d$



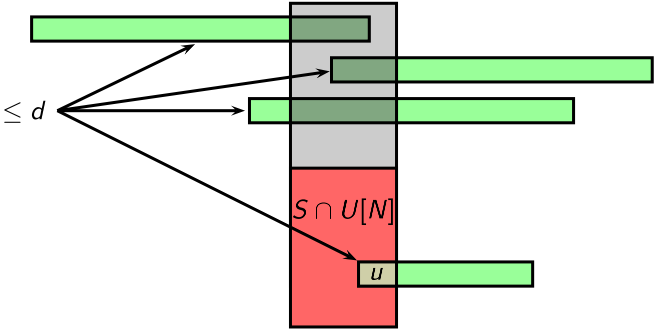
class U[N] (users of neighborhood N)

RCP<> is FPT parameterized by $(p, min\{s, d\})$

- partition U into at most 2^p groups of users of same neighborhood
- let $S \subseteq U$ be a blocker set

Claim: $\forall N \subseteq P,\ U[N] \cap S \neq \emptyset \implies |U[N] \setminus S| \leq d - 1$

- For all $u \in U[N] \cap S$, there exists a set of teams $V_1, \ldots, V_t$ such that:
  - $(\cup V_i) \cap S = \{u\}$
  - $|(\cup V_i) \cap U[N]| \leq d$



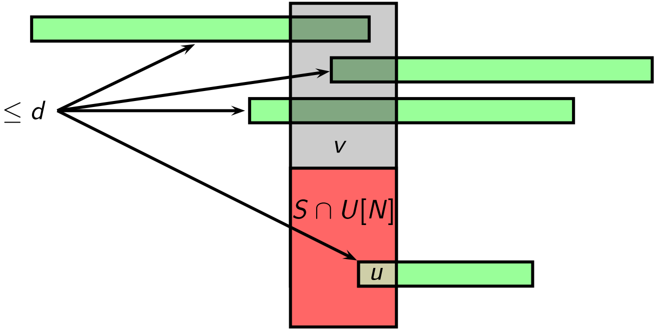
class U[N] (users of neighborhood N)

RCP<> is FPT parameterized by $(p, min\{s, d\})$

- partition U into at most 2^p groups of users of same neighborhood
- let $S \subseteq U$ be a blocker set

Claim: $\forall N \subseteq P$, $U[N] \cap S \neq \emptyset \implies |U[N] \setminus S| \leq d - 1$

- Now, if $|U[N] \setminus S| \geq d$ there exists $v \in U[N] \setminus S$ such that $v \notin \cup V_i$



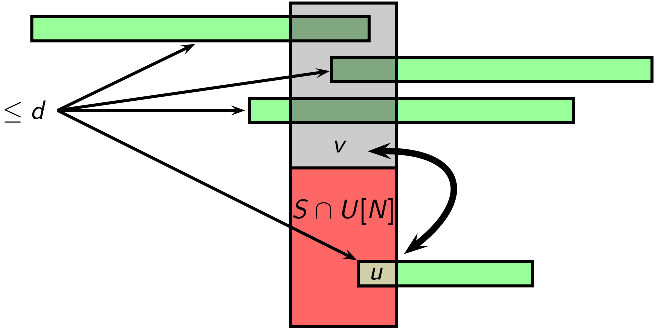
class U[N] (users of neighborhood N)

RCP<> is FPT parameterized by $(p, min\{s, d\})$

- partition U into at most 2^p groups of users of same neighborhood
- let $S \subseteq U$ be a blocker set

Claim: $\forall N \subseteq P,\ U[N] \cap S \neq \emptyset \implies |U[N] \setminus S| \leq d - 1$

- Now, if $|U[N] \setminus S| \geq d$ there exists $v \in U[N] \setminus S$ such that $v \notin \cup V_i$
- Replacing u by v creates another set of teams which does not intersect S: impossible!



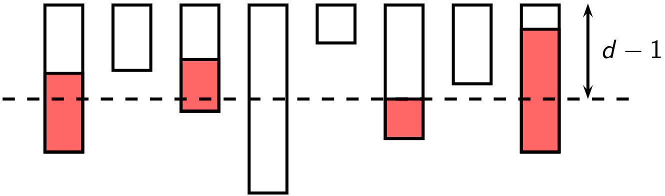
class U[N] (users of neighborhood N)

RCP<> is FPT parameterized by $(p, min\{s, d\})$

- partition U into at most 2^p groups of users of same neighborhood
- let $S \subseteq U$ be a blocker set

<u>Claim:</u> $\forall N \subseteq P,\ U[N] \cap S \neq \emptyset \implies |U[N] \setminus S| \leq d - 1$

- Conclusion: it is sufficient to enumerate:

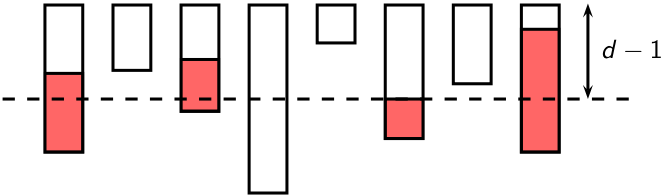


RCP<> is FPT parameterized by $(p, min\{s, d\})$

- partition U into at most 2^p groups of users of same neighborhood
- let $S \subseteq U$ be a blocker set

Claim: $\forall N \subseteq P,\ U[N] \cap S \neq \emptyset \implies |U[N] \setminus S| \leq d - 1$

- Conclusion: it is sufficient to enumerate:
  - which classes S intersects $\implies O(2^{2^p})$



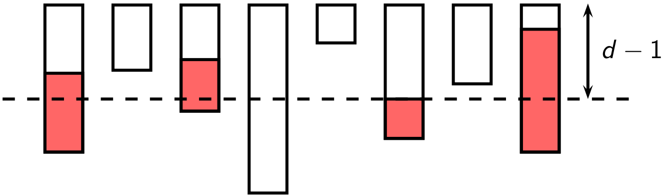
RCP<> is FPT parameterized by $(p, min\{s, d\})$

- partition U into at most 2^p groups of users of same neighborhood
- let $S \subseteq U$ be a blocker set

Claim: $\forall N \subseteq P,\ U[N] \cap S \neq \emptyset \implies |U[N] \setminus S| \leq d - 1$

- Conclusion: it is sufficient to enumerate:
  - which classes S intersects $\implies O(2^{2^p})$
  - how much we take in addition to what we already know

$\implies \leq 2^p$ numbers taking value in $[0, min\{d-1, s\}]$

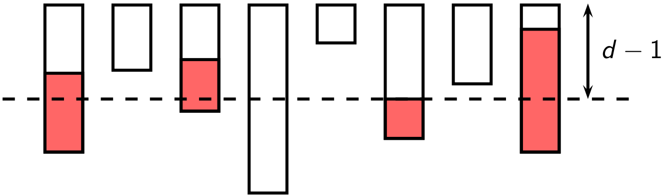


RCP<> is FPT parameterized by $(p, min\{s, d\})$

- partition U into at most 2^p groups of users of same neighborhood
- let $S \subseteq U$ be a blocker set

<u>Claim:</u> $\forall N \subseteq P,\ U[N] \cap S \neq \emptyset \implies |U[N] \setminus S| \leq d - 1$

- Conclusion: it is sufficient to enumerate:
  - which classes S intersects $\implies O(2^{2^p})$
  - how much we take in addition to what we already know $\implies \leq 2^p$ numbers taking value in $[0, min\{d-1, s\}]$
- And then: for each candidate S, test whether it is a blocker set by solving RCP<$s = 0$> with user set $U \setminus S$



RCP<> is FPT parameterized by $(p, min\{s, d\})$

- partition U into at most 2^p groups of users of same neighborhood
- let $S \subseteq U$ be a blocker set

<u>Claim:</u> $\forall N \subseteq P,\ U[N] \cap S \neq \emptyset \implies |U[N] \setminus S| \leq d - 1$

- Conclusion: it is sufficient to enumerate:
  - which classes S intersects $\implies O(2^{2^p})$
  - how much we take in addition to what we already know
    $\implies \leq 2^p$ numbers taking value in $[0, min\{d-1, s\}]$
- And then: for each candidate S, test whether it is a blocker set by solving RCP<$s = 0$> with user set $U \setminus S$

Theorem [Crampton, Gutin, W.]

RCP<> is FPT parameterized by $(p, min\{s, d\})$.

RCP<> is FPT parameterized by $(p, min\{s, d\})$

- partition U into at most 2^p groups of users of same neighborhood
- let $S \subseteq U$ be a blocker set

Claim: $\forall N \subseteq P,\ U[N] \cap S \neq \emptyset \implies |U[N] \setminus S| \leq d - 1$

- Conclusion: it is sufficient to enumerate:
  - which classes S intersects $\implies O(2^{2^p})$
  - how much we take in addition to what we already know
    $\implies \leq 2^p$ numbers taking value in $[0, min\{d-1, s\}]$
- And then: for each candidate S, test whether it is a blocker set by solving RCP<$s = 0$> with user set $U \setminus S$

Theorem [Crampton, Gutin, W.]

RCP<> is FPT parameterized by $(p, min\{s, d\})$.

Open questions:

- what about parameterized by p only ?
- better running time ? (combinatorial algorithm for RCP<$s = 0$>)

Contents

1. Definitions

2. Appetizer: easy observations

3. FPT algorithms

4. Efficient algorithms and implementation

5. Conclusion

Efficient algorithms?

Li, Tripunitara, Wang's approach for solving RCP<>:

Efficient algorithms?

Li, Tripunitara, Wang's approach for solving RCP<>:

- enumerate all subsets of users of size $\leq s$

Efficient algorithms?

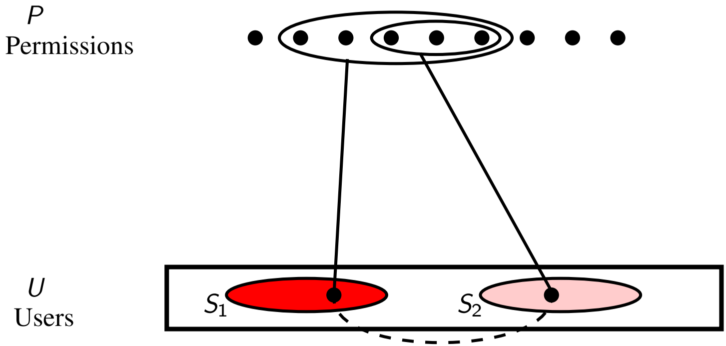
Li, Tripunitara, Wang's approach for solving RCP<>:

- enumerate all subsets of users of size $\leq s$
- for each such S, solve RCP$<s = 0>$ with users $U \setminus S$ (using a SAT solver)

Efficient algorithms?

Li, Tripunitara, Wang's approach for solving RCP<>:

- enumerate all subsets of users of size $\leq s$
- ~~for each such S~~, solve RCP$<s = 0>$ with users $U \setminus S$ (using a SAT solver)
- S_1 dominates S_2:
  $\exists$ a set of teams in $U \setminus S_1 \implies \exists$ a set of teams in $U \setminus S_2$



Efficient algorithms?

Li, Tripunitara, Wang's approach for solving RCP<>:

- enumerate all subsets of users of size $\leq s$
- ~~for each such S~~, solve RCP$<s = 0>$ with users $U \setminus S$ (using a SAT solver)
- S_1 dominates S_2:
  $\exists$ a set of teams in $U \setminus S_1 \implies \exists$ a set of teams in $U \setminus S_2$
- enough to run RCP$<s = 0>$ only when removing non-dominated sets

Efficient algorithms?

Li, Tripunitara, Wang's approach for solving RCP$<>$:

- enumerate all subsets of users of size $\leq s$
- ~~for each such S~~, solve RCP$<s = 0>$ with users $U \setminus S$ (using a SAT solver)
- S_1 dominates S_2:
  $\exists$ a set of teams in $U \setminus S_1 \implies \exists$ a set of teams in $U \setminus S_2$
- enough to run RCP$<s = 0>$ only when removing non-dominated sets
- the bottleneck comes from the SAT solver
  $\implies$ what about a fast algorithm for RCP$<s = 0>$?

Efficient algorithms?

Li, Tripunitara, Wang's approach for solving RCP$<>$:

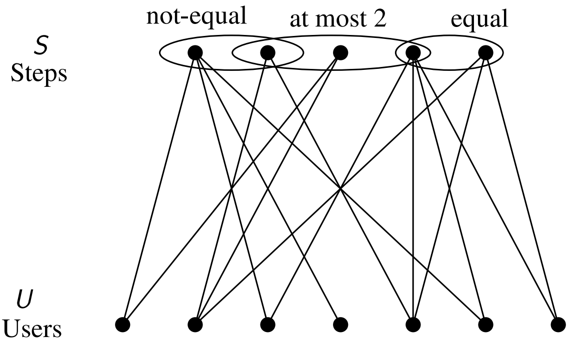
- enumerate all subsets of users of size $\leq s$
- ~~for each such S~~, solve RCP$<s = 0>$ with users $U \setminus S$ (using a SAT solver)
- S_1 dominates S_2:
  $\exists$ a set of teams in $U \setminus S_1 \implies \exists$ a set of teams in $U \setminus S_2$
- enough to run RCP$<s = 0>$ only when removing non-dominated sets
- the bottleneck comes from the SAT solver
  $\implies$ what about a fast algorithm for RCP$<s = 0>$?
- better not to use:
  - Lenstra's algorithm/ILP (FPT param. by p only)
  - dynamic programming ($O^*(2^{dp})$ time and space)

Efficient algorithms?

Workflow Satisfaction Problem (WSP)

Input: a set of steps S, a set of users U
authorization policy $A \subseteq U \times S$, a set of constraints
Output: a plan $\pi : S \to U$ such that:

- $(\pi(s), s) \in A$ for all $s \in S$
- π does not violate any constraint

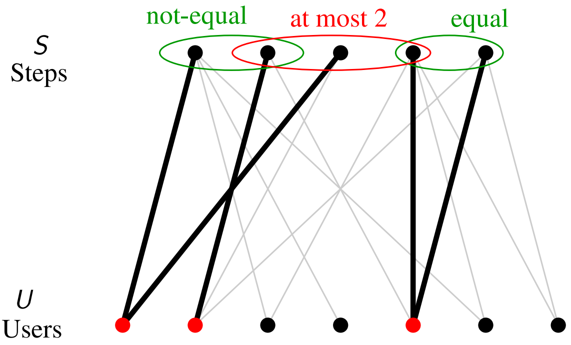


Efficient algorithms?

Workflow Satisfaction Problem (WSP)

Input: a set of steps S, a set of users U
authorization policy $A \subseteq U \times S$, a set of constraints
Output: a plan $\pi : S \rightarrow U$ such that:

- $(\pi(s), s) \in A$ for all $s \in S$
- π does not violate any constraint

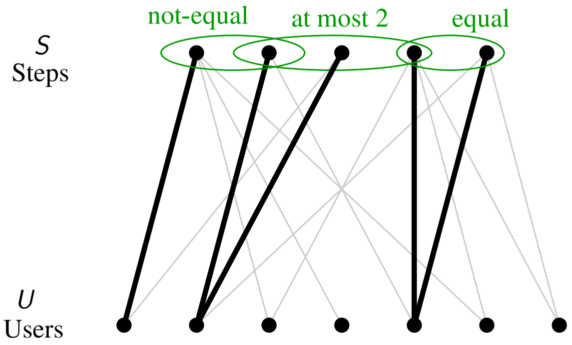


Efficient algorithms?

Workflow Satisfaction Problem (WSP)

Input: a set of steps S, a set of users U
authorization policy $A \subseteq U \times S$, a set of constraints
Output: a plan $\pi : S \rightarrow U$ such that:

- $(\pi(s), s) \in A$ for all $s \in S$
- π does not violate any constraint

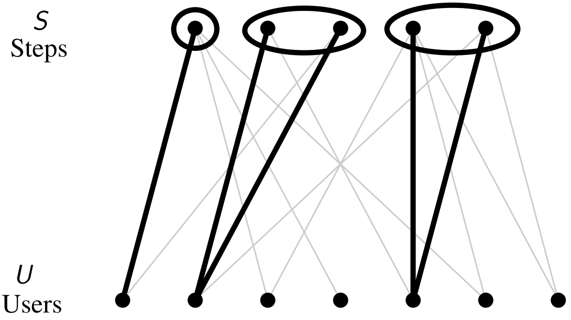


Efficient algorithms?

Workflow Satisfaction Problem (WSP)

Input: a set of steps S, a set of users U
authorization policy $A \subseteq U \times S$, a set of constraints
Output: a plan $\pi : S \to U$ such that:

- $(\pi(s), s) \in A$ for all $s \in S$
- π does not violate any constraint

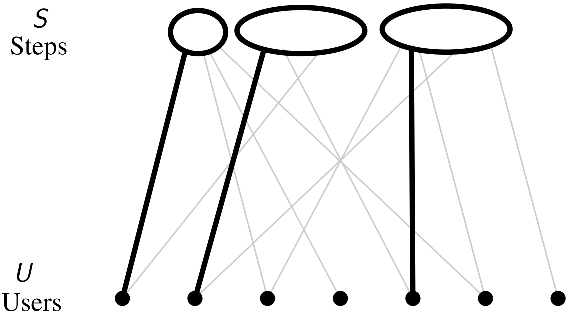


Efficient algorithms?

Workflow Satisfaction Problem (WSP)

Input: a set of steps S, a set of users U
authorization policy $A \subseteq U \times S$, a set of constraints
Output: a plan $\pi : S \to U$ such that:

- $(\pi(s), s) \in A$ for all $s \in S$
- π does not violate any constraint



Efficient algorithms?

Workflow Satisfaction Problem (WSP)

Input: a set of steps S, a set of users U
authorization policy $A \subseteq U \times S$, a set of constraints
Output: a plan $\pi : S \to U$ such that:

- $(\pi(s), s) \in A$ for all $s \in S$
- π does not violate any constraint

Theorem [Karapetyan, Gagarin, Gutin, 2015]

WSP can be solved in $O^*(2^{k \log k})$, where k is the number of steps.

More importantly: an efficient implementation of the algorithm can solve instances with up to 60 steps !

Efficient algorithms ?

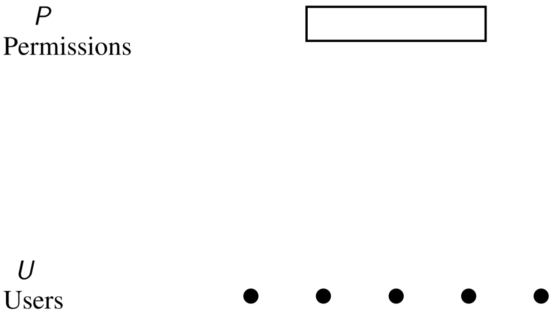
Theorem [Crampton, Gutin, W.]

There is a reduction from RCP$<s = 0>$ to WSP with dp steps.

Efficient algorithms ?

Theorem [Crampton, Gutin, W.]

There is a reduction from RCP<$s = 0$> to WSP with dp steps.

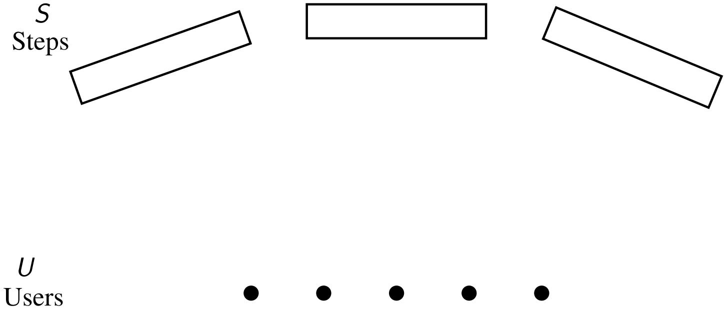


Efficient algorithms ?

Theorem [Crampton, Gutin, W.]

There is a reduction from RCP<$s = 0$> to WSP with dp steps.

- duplicate the set of permissions d times

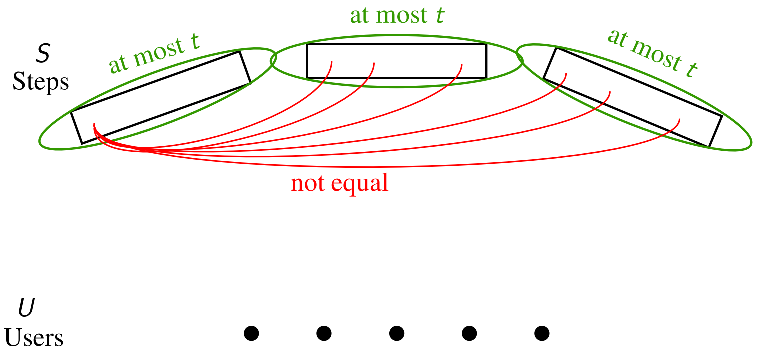


Efficient algorithms ?

Theorem [Crampton, Gutin, W.]

There is a reduction from RCP<$s = 0$> to WSP with dp steps.

- duplicate the set of permissions d times
- add **at-most-t** constraints to preserve team sizes

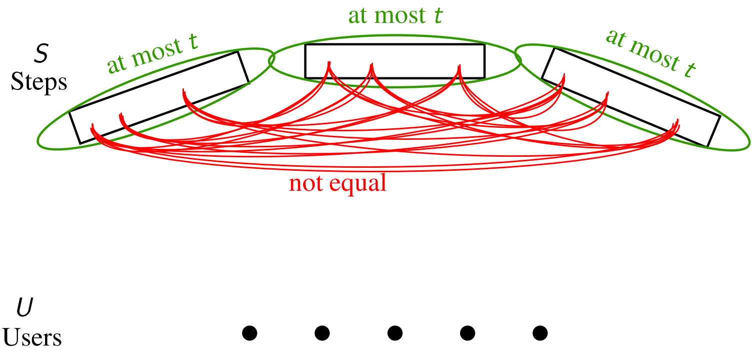


Efficient algorithms ?

Theorem [Crampton, Gutin, W.]

There is a reduction from RCP<$s = 0$> to WSP with dp steps.

- duplicate the set of permissions d times
- add **at-most-t** constraints to preserve team sizes
- add **not-equal** constraints to preserve disjointness

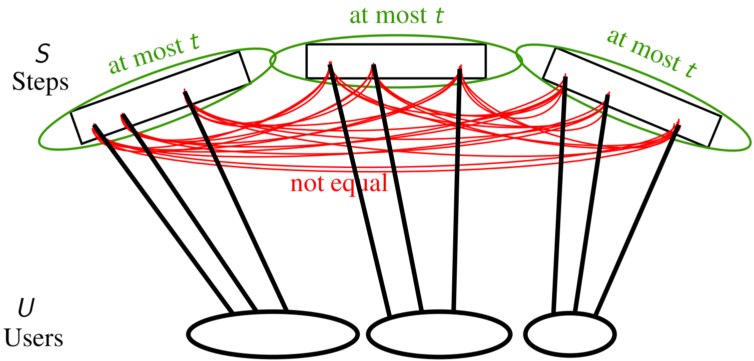


Efficient algorithms ?

Theorem [Crampton, Gutin, W.]

There is a reduction from RCP<$s = 0$> to WSP with dp steps.

- duplicate the set of permissions d times
- add **at-most-t** constraints to preserve team sizes
- add **not-equal** constraints to preserve disjointness



Efficient algorithms ?

Theorem [Crampton, Gutin, W.]

There is a reduction from RCP$<s = 0>$ to WSP with dp steps.

Corollary

RCP$<s = 0>$ can be solved in $O^*(2^{dp \log(dp)})$.

(with an efficient algorithm!)

Efficient algorithms ?

Theorem [Crampton, Gutin, W.]

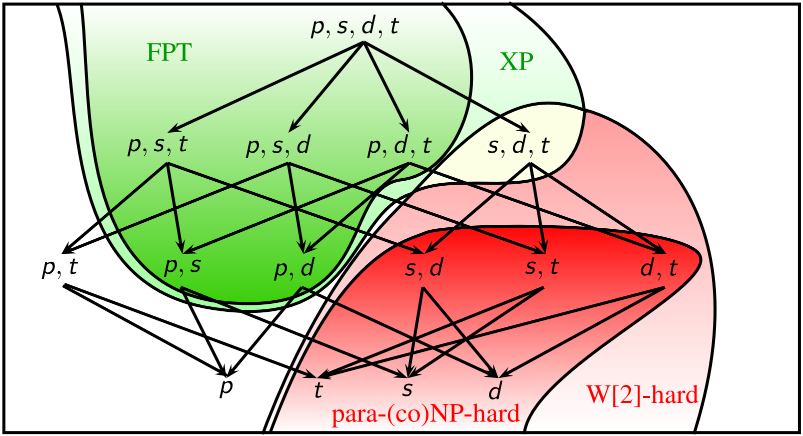
There is a reduction from RCP$<s = 0>$ to WSP with dp steps.

Corollary

RCP$<s = 0>$ can be solved in ~~$O^*(2^{dp \log(dp)})$~~ $O^*(2^{dp \log(p)})$.

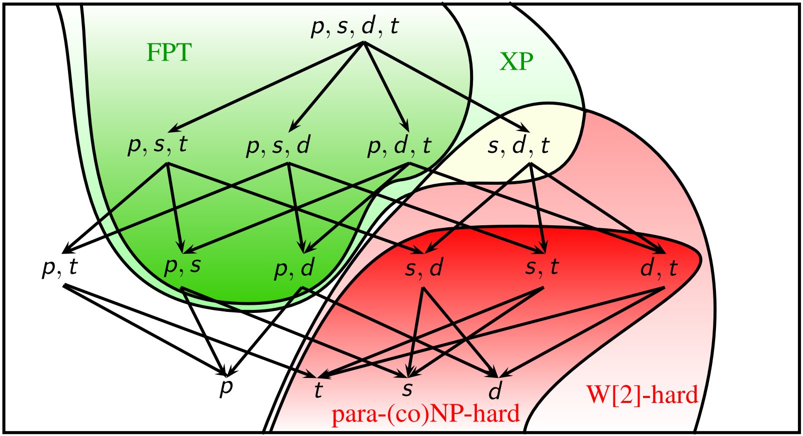
(with an efficient algorithm!)

Conclusion



- RCP<> parameterized by p only ?

Conclusion



- RCP<> parameterized by p only ?
- resiliency w.r.t. other problems ?

Merci !