Resiliency problems: algorithms and applications in access control

Rémi Watrigant

Inria Sophia Antipolis Mediterranée*

(* work mainly done when I was at RHUL)
Joint work with: Jason Crampton, Gregory Gutin and Martin Koutecký

Royal Holloway University of London, CS Department seminar
January 10th, 2017.

1. Preliminaries
   - Resiliency, definition of the problem
   - Parameterized complexity
2. Parameterized landscape of the problem
3. Generalization to Integer Linear Programs...
4. ...and applications to other domains

Contents

Resiliency: general idea

Decision problem

Definition of a problem = set of instances, set of positive instances
Goal: Given an instance $\mathcal{I}$, is $\mathcal{I}$ positive ?

Examples of classical (NP-hard) problems:

Resiliency: general idea

Decision problem

Definition of a problem = set of instances, set of positive instances
Goal: Given an instance $\mathcal{I}$, is $\mathcal{I}$ positive ?

Examples of classical (NP-hard) problems:

- $\mathcal{I}$ is a graph, positive iff it has a hamiltonian cycle
- $\mathcal{I}$ is a CNF formula, positive iff it is satisfiable
- $\mathcal{I}$ is a set of subsets of a universe and an integer k, positive iff there exists k sets whose union is the universe
- ...

Resiliency: general idea

Decision problem

Definition of a problem = set of instances, set of positive instances
Goal: Given an instance $\mathcal{I}$, is $\mathcal{I}$ positive ?

Examples of classical (NP-hard) problems:

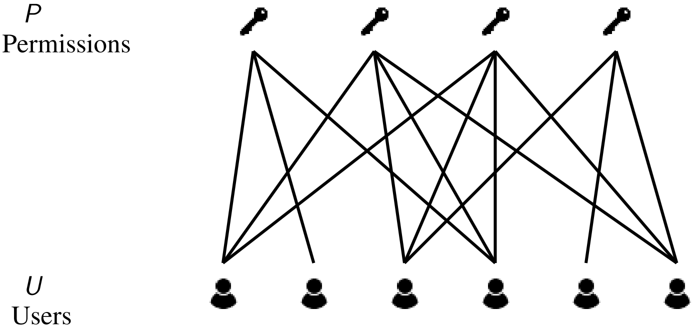
- $\mathcal{I}$ is a graph, positive iff it has a hamiltonian cycle
- $\mathcal{I}$ is a CNF formula, positive iff it is satisfiable
- $\mathcal{I}$ is a set of subsets of a universe and an integer k, positive iff there exists k sets whose union is the universe
- ...

Resiliency problem

Definition of a problem = set of instances, set of positive instances, and for every instance $\mathcal{I}$, a set $Pert(\mathcal{I})$ of perturbed instances
Goal: Given an instance $\mathcal{I}$, is $\mathcal{I}_p$ positive for every $\mathcal{I}_p \in Pert(\mathcal{I})$?

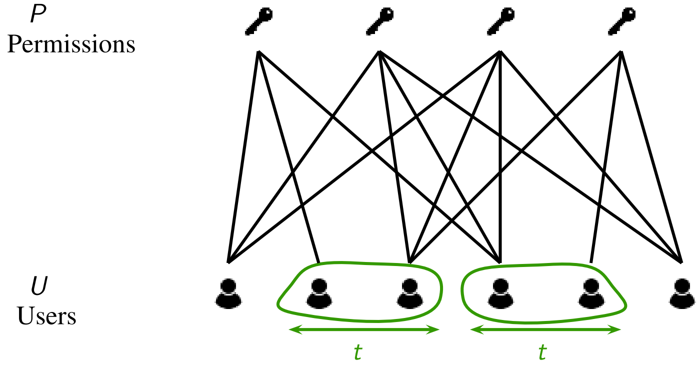
Resiliency Checking Problem (RCP)

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



Resiliency Checking Problem (RCP)

Input: an authorization policy: $UP \subseteq U \times P$
$s, d, t \in \mathbb{N}$
Output: decide whether one can find a set of d teams of size $\leq t$

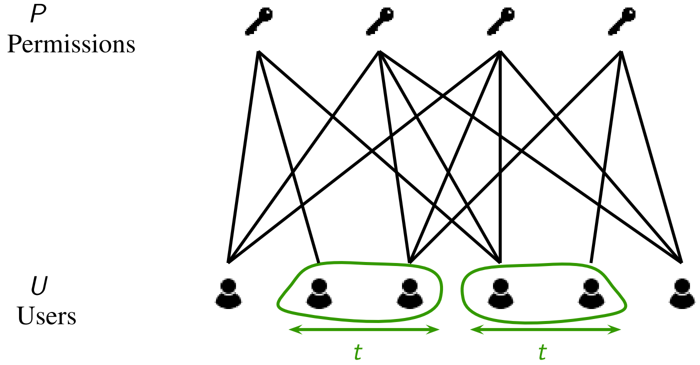


Set of teams: d mutually disjoint sets of $\leq t$ users having collectively all permissions.

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, one can find a set of d teams of size $\leq t$

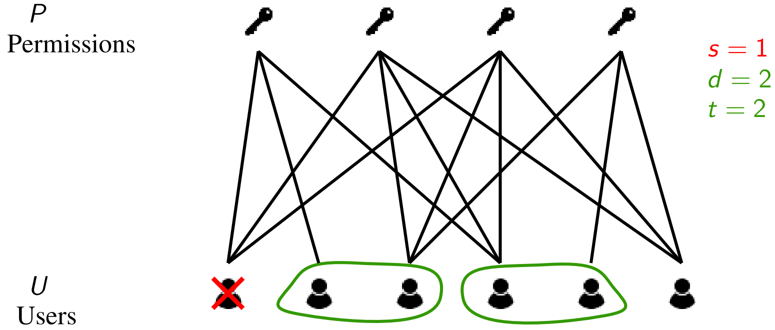


Set of teams: d mutually disjoint sets of $\leq t$ users having collectively all permissions.

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, one can find a set of d teams of size $\leq t$

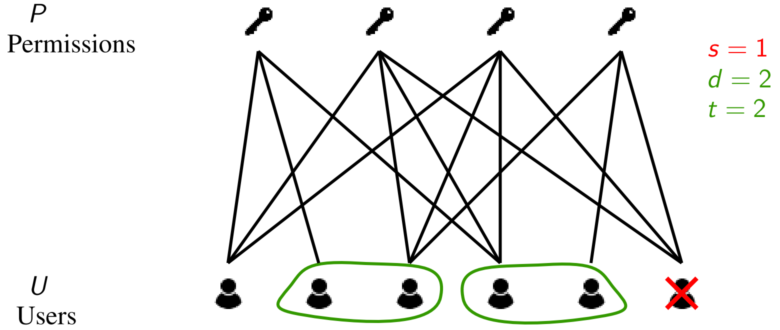


Set of teams: d mutually disjoint sets of $\leq t$ users having collectively all permissions.

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, one can find a set of d teams of size $\leq t$



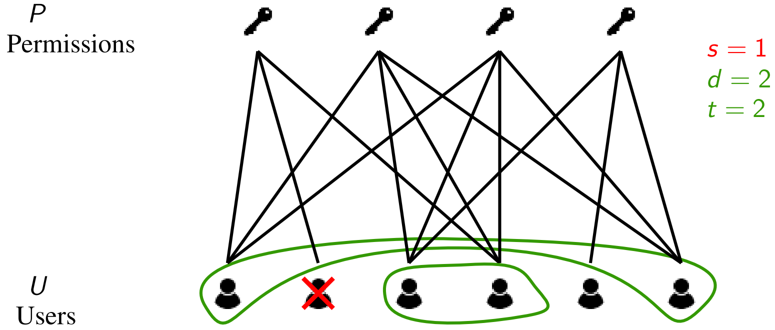
Set of teams: d mutually disjoint sets of $\leq t$ users having collectively all permissions.

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, one can find a set of d teams of size $\leq t$

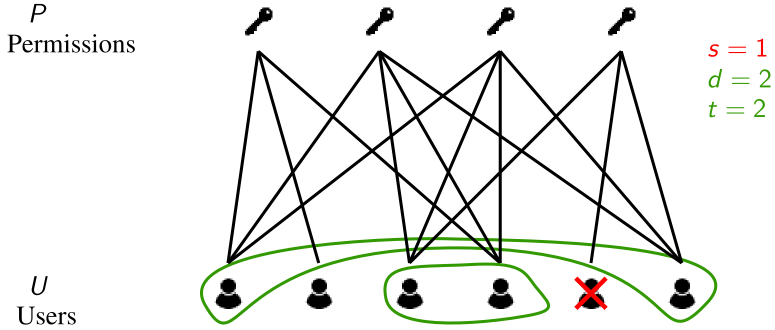


Set of teams: d mutually disjoint sets of $\leq t$ users having collectively all permissions.

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, one can find a set of d teams of size $\leq t$

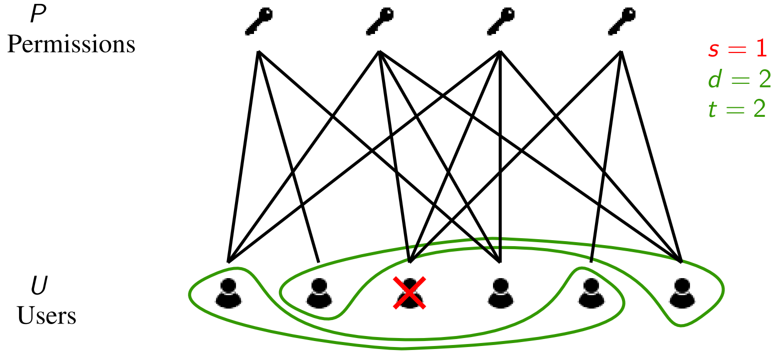


Set of teams: d mutually disjoint sets of $\leq t$ users having collectively all permissions.

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, one can find a set of d teams of size $\leq t$

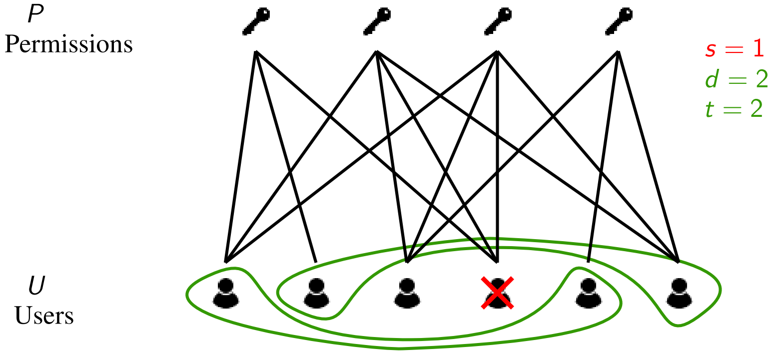


Set of teams: d mutually disjoint sets of $\leq t$ users having collectively all permissions.

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, one can find a set of d teams of size $\leq t$



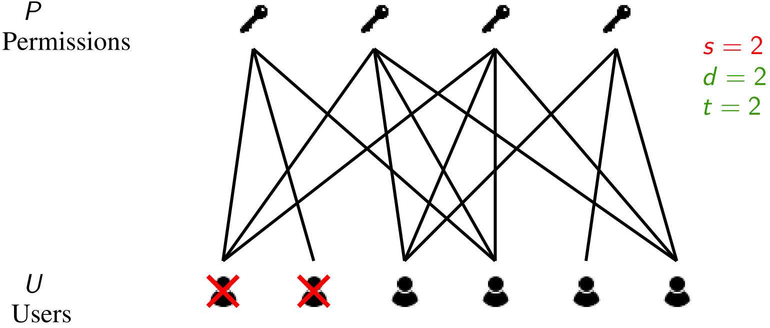
Set of teams: d mutually disjoint sets of $\leq t$ users having collectively all permissions.

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, one can find a set of d teams of size $\leq t$

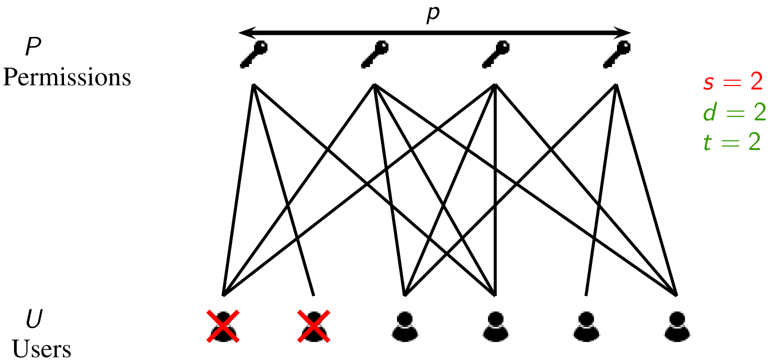


Set of teams: d mutually disjoint sets of $\leq t$ users having collectively all permissions.

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, one can find a set of d teams of size $\leq t$



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

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 $O(g(k).|x|^{f(k)})$

Parameterized algorithms

For a problem instance x coming with its parameter k:

- *XP* if you can solve it in $O(g(k).|x|^{f(k)})$
- *FPT* if you can solve it in $O(f(k).|x|^c)$, c does not depend on $k, |x|$

Parameterized algorithms

For a problem instance x coming with its parameter k:

- XP if you can solve it in $O(g(k).|x|^{f(k)})$
- para-NP-hard: NP-hard when k is fixed to some constant
  $\Longrightarrow$ no XP algorithm unless $P = NP$
- FPT if you can solve it in $O(f(k).|x|^c)$, c does not depend on $k, |x|$

Parameterized algorithms

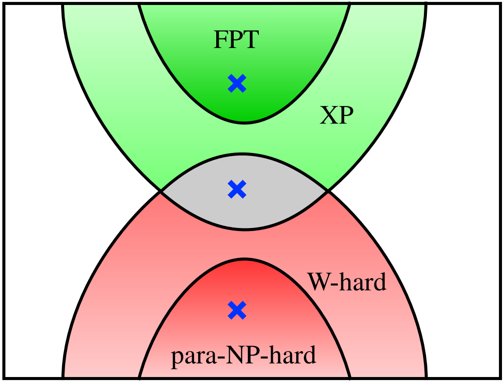
For a problem instance x coming with its parameter k:

- *XP* if you can solve it in $O(g(k).|x|^{f(k)})$
- para-NP-hard: NP-hard when k is fixed to some constant
  $\Longrightarrow$ no *XP* algorithm unless $P = NP$
- *FPT* if you can solve it in $O(f(k).|x|^c)$, c does not depend on $k, |x|$
- $W[1]$-hardness: parameter-preserving reduction from a $W[1]$-hard problem
  $\Longrightarrow$ 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 $O(g(k).|x|^{f(k)})$
- 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 $O(f(k).|x|^c)$, c does not depend on $k, |x|$
- $W[1]$-hardness: parameter-preserving reduction from a $W[1]$-hard problem
  $\implies$ no *FPT* algorithm unless $FPT = W[1]$



Contents

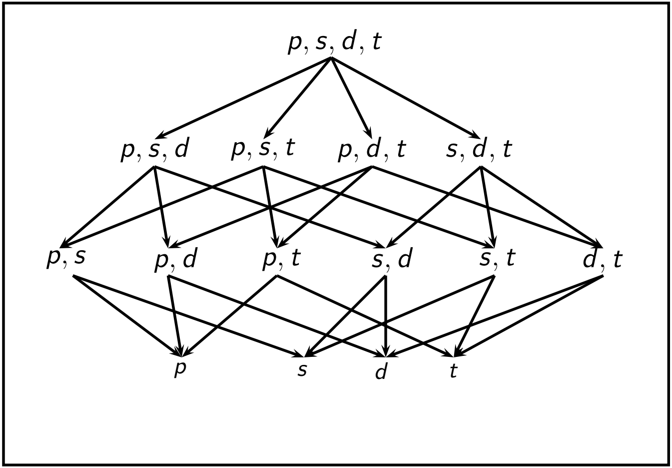
Results

RCP

Input: $UP \subseteq U \times P$, integers s, d, t (recall that $p = |P|$)

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

Parameterized landscape of RCP:

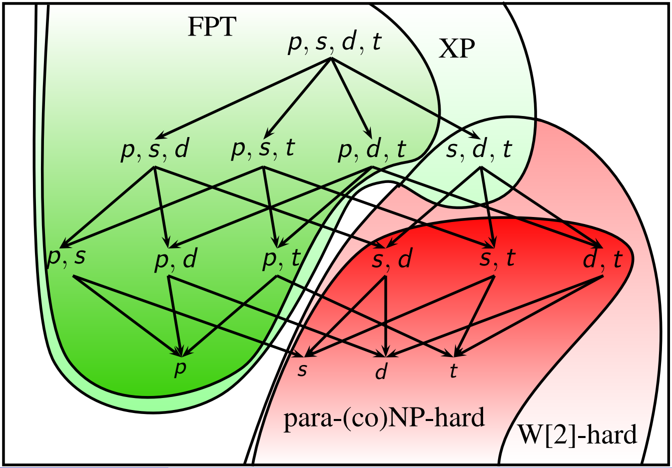


Results

RCP

Input: $UP \subseteq U \times P$, integers s, d, t (recall that $p = |P|$)
Output: upon removal of any set of s users, are there still d teams of size t ?

Parameterized landscape of RCP:



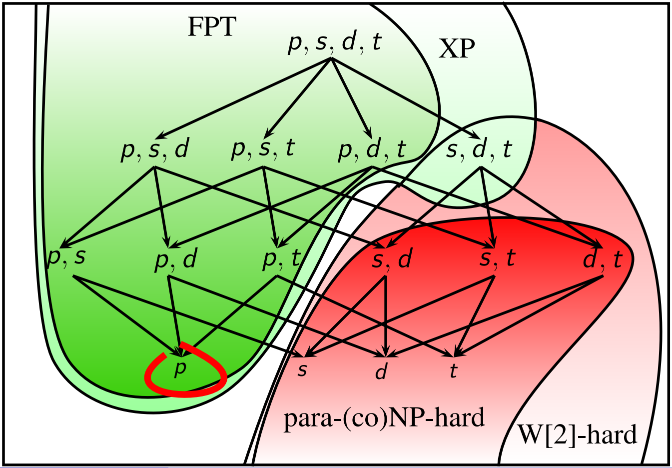
Results

RCP

Input: $UP \subseteq U \times P$, integers s, d, t (recall that $p = |P|$)

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

Parameterized landscape of RCP:



Integer Linear Programs

Example of an ILP:

$$-x_1 + x_2 \leq 1$$

$$3x_1 + 2x_2 \leq 12$$

$$6x_1 - 3x_2 \geq 0$$

$$x_1, x_2 \geq 0$$

$$x_1, x_2 \in \mathbb{N}$$

has a solution (ex: $x_1 = 1$, $x_2 = 2$)

Integer Linear Programs

Example of an ILP:

$$-x_1 + x_2 \leq 1$$

$$3x_1 + 2x_2 \leq 12$$

$$6x_1 - 3x_2 \geq 0$$

$$x_1, x_2 \geq 0$$

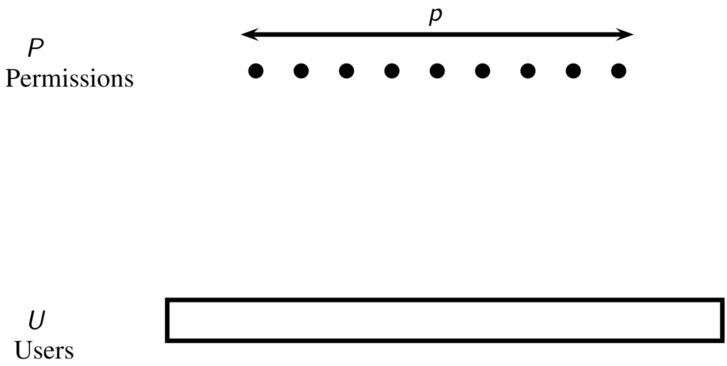
$$x_1, x_2 \in \mathbb{N}$$

has a solution (ex: $x_1 = 1$, $x_2 = 2$)

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

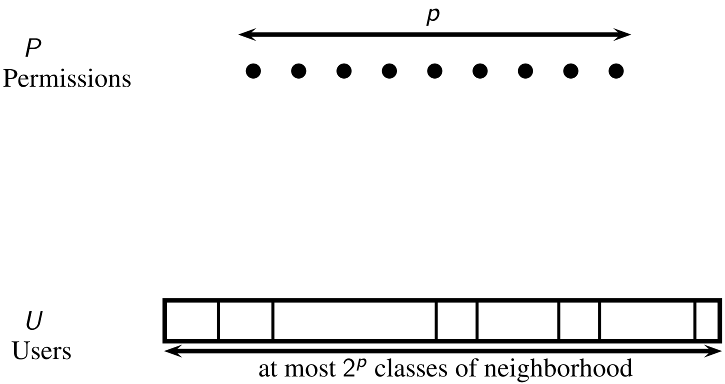
Whether a given Integer Linear Program (ILP) has a non-empty solution set can be decided in $O^*(n^{2.5n+o(n)})$ time and polynomial space, where n is the number of variables.

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



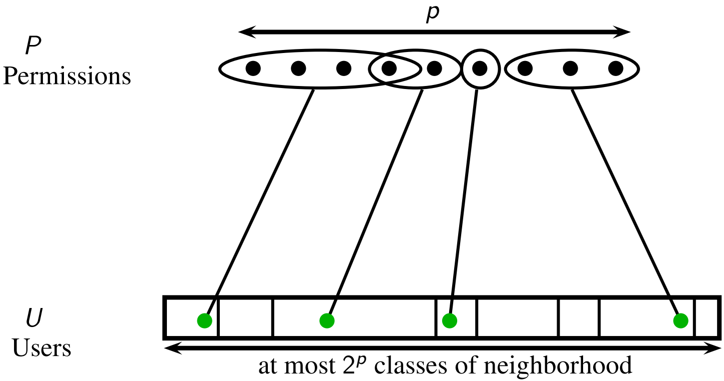
RCP<$s = 0$> is FPT parameterized by p

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



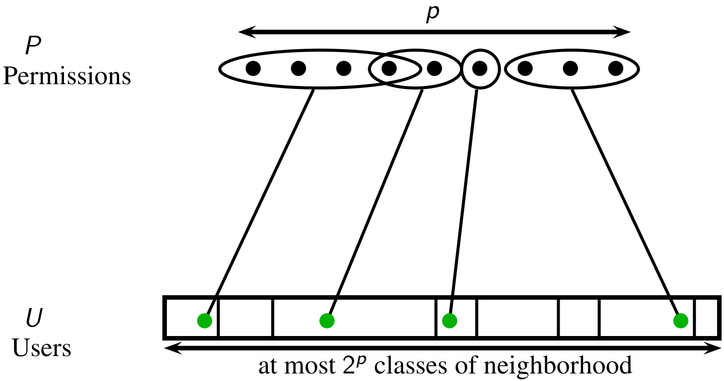
RCP<$s = 0$> is FPT parameterized by p

- 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:



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:

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

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:

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

- 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

From RCP<$s = 0$> to RCP<>

To find a set of teams: solve the following ILP:

$$\sum_{c \in \mathcal{C}} x_c \geq d \tag{1}$$

$$\sum_{c \in \mathcal{C}[N]} x_c \leq |U[N]| \qquad \forall N \subseteq P \tag{2}$$

($U[N]$ = users having neighborhood N)

From RCP<$s = 0$> to RCP<>

To find a set of teams: solve the following ILP:

$$\sum_{c \in \mathcal{C}} x_c \geq d \tag{1}$$

$$\sum_{c \in \mathcal{C}[N]} x_c \leq |U[N]| - z_N \quad \forall N \subseteq P \tag{2}$$

$$\sum_{N \subseteq P} z_N \leq s \tag{3}$$

($U[N]$ = users having neighborhood N)

From RCP<$s = 0$> to RCP<>

To find a set of teams: solve the following ILP:

$$\sum_{c \in \mathcal{C}} x_c \geq d \qquad (1)$$

$$\sum_{c \in \mathcal{C}[N]} x_c \leq |U[N]| - z_N \quad \forall N \subseteq P \qquad (2)$$

$$\sum_{N \subseteq P} z_N \leq s \qquad (3)$$

($U[N]$ = users having neighborhood N)

Unfortunately, the above ILP doesn't solve RCP<> directly

From RCP<$s = 0$> to RCP<>

To find a set of teams: solve the following ILP:

$$\sum_{c \in \mathcal{C}} x_c \geq d \tag{1}$$

$$\sum_{c \in \mathcal{C}[N]} x_c \leq |U[N]| - z_N \quad \forall N \subseteq P \tag{2}$$

$$\sum_{N \subseteq P} z_N \leq s \tag{3}$$

($U[N]$ = users having neighborhood N)

Unfortunately, the above ILP doesn't solve RCP<> directly

What we need to solve: for every assignment of variables z_N, is there an assignment of variables x_c ?

Contents

ILP resiliency

Suppose the set of variables is $X \uplus Z$:

- F_X: conjunction of inequalities involving variables X only
- F_Z: conjunction of inequalities involving variables Z only
- F_{XZ}: conjunction of inequalities involving variables from X and Z

<u>Classical ILP:</u> find an assignment of $X \cup Z$ such that $F_X \wedge F_Z \wedge F_{XZ}$ is satisfied

ILP resiliency

Suppose the set of variables is $X \uplus Z$:

- F_X: conjunction of inequalities involving variables X only
- F_Z: conjunction of inequalities involving variables Z only
- F_{XZ}: conjunction of inequalities involving variables from X and Z

Classical ILP: find an assignment of $X \cup Z$ such that $F_X \wedge F_Z \wedge F_{XZ}$ is satisfied

Z-resiliency: for any assignment of Z satisfying F_Z, does there exist an assignment of X such that both assignments satify $F_X \wedge F_Z \wedge F_{XZ}$?

ILP resiliency

Suppose the set of variables is $X \uplus Z$:

- F_X: conjunction of inequalities involving variables X only
- F_Z: conjunction of inequalities involving variables Z only
- F_{XZ}: conjunction of inequalities involving variables from X and Z

Classical ILP: find an assignment of $X \cup Z$ such that $F_X \wedge F_Z \wedge F_{XZ}$ is satisfied

Z-resiliency: for any assignment of Z satisfying F_Z, does there exist an assignment of X such that both assignments satify $F_X \wedge F_Z \wedge F_{XZ}$?

Theorem [Eisenbrand, Shmonin, 2008]

Parametric-∀∃-ILP is FPT parameterized by the number of variables, constraints, size of encoding of the matrices of the ILPs.

Our result:
There is a reduction from ILP Resiliency to Parametric-∀∃-ILP

ILP resiliency

Suppose the set of variables is $X \uplus Z$:

- F_X: conjunction of inequalities involving variables X only
- F_Z: conjunction of inequalities involving variables Z only
- F_{XZ}: conjunction of inequalities involving variables from X and Z

Classical ILP: find an assignment of $X \cup Z$ such that $F_X \wedge F_Z \wedge F_{XZ}$ is satisfied

Z-resiliency: for any assignment of Z satisfying F_Z, does there exist an assignment of X such that both assignments satify $F_X \wedge F_Z \wedge F_{XZ}$?

Theorem [Eisenbrand, Shmonin, 2008]

Parametric-∀∃-ILP is FPT parameterized by the number of variables, constraints, size of encoding of the matrices of the ILPs.

Our result:
There is a reduction from ILP Resiliency to Parametric-∀∃-ILP

Now: how to solve it in FPT time parameterized by the number of variables, constraints, and unary size of encoding of matrices of the ILPs.

ILP resiliency

Suppose the set of variables is $X \uplus Z$:

- F_X: conjunction of inequalities involving variables X only
- F_Z: conjunction of inequalities involving variables Z only
- F_{XZ}: conjunction of inequalities involving variables from X and Z

Classical ILP: find an assignment of $X \cup Z$ such that $F_X \wedge F_Z \wedge F_{XZ}$ is satisfied

Z-resiliency: for any assignment of Z satisfying F_Z, does there exist an assignment of X such that both assignments satify $F_X \wedge F_Z \wedge F_{XZ}$?

Idea of the algorithm:

- eliminate variables X and obtain an equivalent disjunction of ~~ILPs~~

$$L_1 \vee \cdots \vee L_r$$

- then: test whether there exists an assignment of variables Z such that

$$\neg L_1 \wedge \cdots \wedge \neg L_r$$

  is satisfied

Fourier-Motzkin elimination for Integer Linear Programs

Elimination of a variable $x_1 \in X$ of an ILP:

(assume all coefficients of x_1 are a)

Fourier-Motzkin elimination for Integer Linear Programs

Elimination of a variable $x_1 \in X$ of an ILP:

(assume all coefficients of x_1 are a)

- $L =$ "lower than" inequalities

$$ax_1 \leq b^\ell - \sum_{k=2}^{n} a_k^\ell x_k \qquad \forall \ell \in L \tag{4}$$

- $G =$ "greater than" inequalities

$$b^g - \sum_{k=2}^{n} a_k^g x_k \leq ax_1 \qquad \forall g \in G \tag{5}$$

Fourier-Motzkin elimination for Integer Linear Programs

Elimination of a variable $x_1 \in X$ of an ILP:

(assume all coefficients of x_1 are a)

- $L =$ "lower than" inequalities

$$ax_1 \leq b^\ell - \sum_{k=2}^{n} a_k^\ell x_k \qquad \forall \ell \in L \tag{4}$$

- $G =$ "greater than" inequalities

$$b^g - \sum_{k=2}^{n} a_k^g x_k \leq ax_1 \qquad \forall g \in G \tag{5}$$

Then: replace all inequalities of L and G, by:

$$b^g - \sum_{k=2}^{n} a_k^g x_k \leq ax_1 \leq b^\ell - \sum_{k=2}^{n} a_k^\ell x_k \qquad \forall \ell \in L, \forall g \in G \tag{6}$$

Fourier-Motzkin elimination for Integer Linear Programs

$$b^g - \sum_{k=2}^{n} a_k^g x_k \leq ax_1 \leq b^\ell - \sum_{k=2}^{n} a_k^\ell x_k \quad \forall \ell \in L, \forall g \in G \tag{7}$$

Fourier-Motzkin elimination for Integer Linear Programs

$$b^g - \sum_{k=2}^{n} a_k^g x_k \leq ax_1 \leq b^\ell - \sum_{k=2}^{n} a_k^\ell x_k \quad \forall \ell \in L, \forall g \in G \tag{7}$$

$$b^g - \sum_{k=2}^{n} a_k^g x_k \leq b^\ell - \sum_{k=2}^{n} a_k^\ell x_k \quad \forall \ell \in L, \forall g \in G \tag{8}$$

Fourier-Motzkin elimination for Integer Linear Programs

$$b^g - \sum_{k=2}^{n} a_k^g x_k \leq a x_1 \leq b^\ell - \sum_{k=2}^{n} a_k^\ell x_k \quad \forall \ell \in L, \forall g \in G \tag{7}$$

$$b^g - \sum_{k=2}^{n} a_k^g x_k \leq b^\ell - \sum_{k=2}^{n} a_k^\ell x_k \quad \forall \ell \in L, \forall g \in G \tag{8}$$

a multiple of a must lie between the left-hand and right-hand side !
$10 \leq 6x_1 \leq 11$ has a solution in (8) but not in (7)

Solution [Williams, 1076]:

Fourier-Motzkin elimination for Integer Linear Programs

$$b^g - \sum_{k=2}^{n} a_k^g x_k \leq a x_1 \leq b^\ell - \sum_{k=2}^{n} a_k^\ell x_k \quad \forall \ell \in L, \forall g \in G \tag{7}$$

$$b^g - \sum_{k=2}^{n} a_k^g x_k \leq b^\ell - \sum_{k=2}^{n} a_k^\ell x_k \quad \forall \ell \in L, \forall g \in G \tag{8}$$

a multiple of a must lie between the left-hand and right-hand side !
$10 \leq 6x_1 \leq 11$ has a solution in (8) but not in (7)

Solution [Williams, 1076]:

$$\bigvee_{h \in \{0,\dots,a-1\}} \begin{array}{c} b^g - \sum_{k=2}^{n} a_k^g x_k + h \leq b^\ell - \sum_{k=2}^{n} a_k^\ell x_k \\ \text{and} \\ b^g - \sum_{k=2}^{n} a_k^g x_k + h \equiv 0 \mod a \end{array} \quad \forall \ell \in L, \forall g \in G \tag{9}$$

$\rightarrow$ Back to ILP resiliency

Z-resiliency: for any assignment of Z satisfying F_Z, does there exist an assignment of X such that both assignments satify $F_X \wedge F_Y \wedge F_{XY}$?

Idea of the algorithm:

- eliminate variables X and obtain an equivalent disjunction of systems of linear inequalities and congruences (SLIC)

$$L_1 \vee \cdots \vee L_r$$

- then: test whether there exists an assignment of variables Z such that

$$\neg L_1 \wedge \cdots \wedge \neg L_r$$

  is satisfied

$\rightarrow$ Back to ILP resiliency

Z-resiliency: for any assignment of Z satisfying F_Z, does there exist an assignment of X such that both assignments satify $F_X \wedge F_Y \wedge F_{XY}$?

Idea of the algorithm:

- eliminate variables X and obtain an equivalent disjunction of systems of linear inequalities and congruences (SLIC)

$$L_1 \vee \cdots \vee L_r$$

- then: test whether there exists an assignment of variables Z such that

$$\neg L_1 \wedge \cdots \wedge \neg L_r$$

  is satisfied
- equivalent to a disjunction of some SLIC

$$L'_1 \vee \cdots \vee L'_{r'}$$

$\rightarrow$ Back to ILP resiliency

Z-resiliency: for any assignment of Z satisfying F_Z, does there exist an assignment of X such that both assignments satify $F_X \wedge F_Y \wedge F_{XY}$?

Idea of the algorithm:

- eliminate variables X and obtain an equivalent disjunction of systems of linear inequalities and congruences (SLIC)

$$L_1 \vee \cdots \vee L_r$$

- then: test whether there exists an assignment of variables Z such that

$$\neg L_1 \wedge \cdots \wedge \neg L_r$$

  is satisfied

- equivalent to a disjunction of some SLIC

$$L'_1 \vee \cdots \vee L'_{r'}$$

- and then: how to test satisfiability of a "SLIC" ??

→ Back to ILP resiliency

Z-resiliency: for any assignment of Z satisfying F_Z, does there exist an assignment of X such that both assignments satify $F_X \wedge F_Y \wedge F_{XY}$?

Idea of the algorithm:

- eliminate variables X and obtain an equivalent disjunction of systems of linear inequalities and congruences (SLIC)

$$L_1 \vee \cdots \vee L_r$$

- then: test whether there exists an assignment of variables Z such that

$$\neg L_1 \wedge \cdots \wedge \neg L_r$$

  is satisfied

- equivalent to a disjunction of some SLIC

$$L'_1 \vee \cdots \vee L'_{r'}$$

- and then: how to test satisfiability of a "SLIC" ?? → eliminate all variables !

Contents

Generalization to other domains

Closest String

Input: k strings $s_1, ..., s_k$ of length L, $d \in \mathbb{N}$
Question: is there a string s^* of length L s.t. $d(s^*, s_i) \leq d$ for all $i = 1, ..., k$?
(such a s is called a d-closest string)

Generalization to other domains

Closest String

Input: k strings $s_1, ..., s_k$ of length L, $d \in \mathbb{N}$
Question: is there a string s^* of length L s.t. $d(s^*, s_i) \leq d$ for all $i = 1, ..., k$?
(such a s is called a d-closest string)

Resiliency Closest String

Input: k strings $s_1, ..., s_k$ of length L, $d, M \in \mathbb{N}$
Perturbation: changing at most M symbols in the strings
Question: for every set of strings obtained after a perturbation, does there exists a d-closest string ?

Generalization to other domains

Closest String

Input: k strings $s_1, ..., s_k$ of length L, $d \in \mathbb{N}$
Question: is there a string s^* of length L s.t. $d(s^*, s_i) \leq d$ for all $i = 1, ..., k$?
(such a s is called a d-closest string)

Resiliency Closest String

Input: k strings $s_1, ..., s_k$ of length L, $d, M \in \mathbb{N}$
Perturbation: changing at most M symbols in the strings
Question: for every set of strings obtained after a perturbation, does there exists a d-closest string ?

Result: Resiliency Closest String is FPT parameterized by k.

Generalization to other domains

- Closest String
- Scheduling: Makespan Minimization on Unrelated Machines
- Computational social choice: Swap Bribery
- ...?

Conclusion

- ILP-resiliency provides a very general framework
  Other applications?
- The known algorithm is FPT parameterized by:
  - number of variables
  - number of constraints
  - encoding length of the matrices of the ILPs

  $\Rightarrow$ can we do better?
  - using less parameters?
  - or: prove a lower bound: ILP-resiliency W[.]-hard parameterized by number of variables (and constraints) only ?

Voilà !
Questions ?