Parameterized algorithmics in Access Control: linking theory and practice

Gregory Gutin
Royal Holloway University of London,UK

3rd PCCR, 1st Aug 2022, Haifa, Israel

Outline

Outline

My conferences: FPT theory and practice

1st PCCR Brno, 2010 (FPT theory: permutation CSPs above average)

My conferences: FPT theory and practice

1st PCCR Brno, 2010 (FPT theory: permutation CSPs above average)

NII Shonan Meeting (Japan, 2013) on Parameterized Complexity and the Understanding, Design and Analysis of Heuristics

My conferences: FPT theory and practice

1st PCCR Brno, 2010 (FPT theory: permutation CSPs above average)

NII Shonan Meeting (Japan, 2013) on Parameterized Complexity and the Understanding, Design and Analysis of Heuristics

IEEE Symp. ai4i/HCC/GC/Trans-AI 2019 Sep, CA, USA

Workflow Satisfiability Problem (WSP)

WSP is of interest in Access Control (BPA in DBSec and SACMAT 2015 were on WSP). WSP has many variations.

Workflow Satisfiability Problem (WSP)

WSP is of interest in Access Control (BPA in DBSec and SACMAT 2015 were on WSP). WSP has many variations.

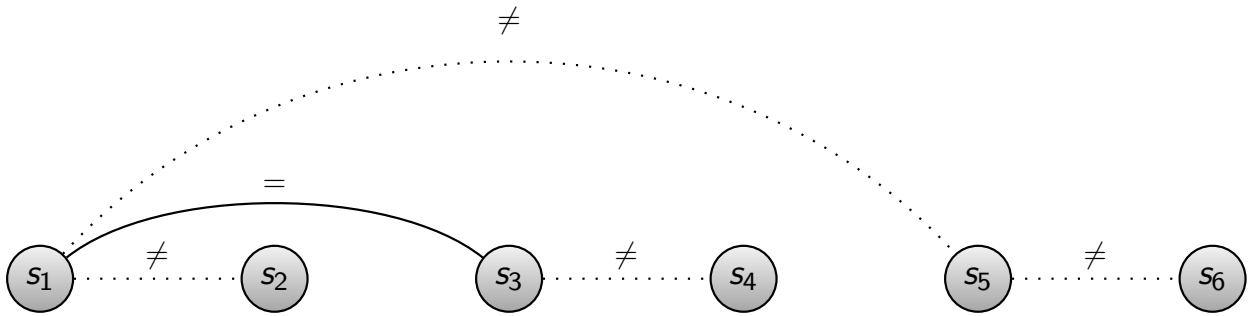
(basic) WSP instance: a set $U = \{u_1, u_2, u_3, u_4\}$ of users and a set S of six steps as follows.

s_1 create purchase order	s_2 approve purchase order
s_3 sign goods received note	s_4 countersign goods received note
s_5 create payment	s_6 approve payment

Workflow Satisfiability Problem (WSP), cont'd

s_1 create purchase order	s_2 approve purchase order
s_3 sign goods received note	s_4 countersign goods received note
s_5 create payment	s_6 approve payment

Constraints and authorizations:



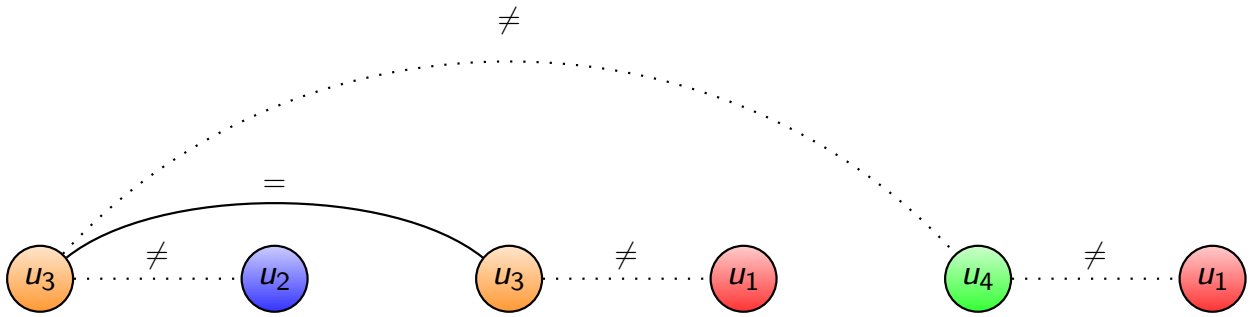
Authorized Users					
u_1 u_2 u_3	u_2 u_3	u_3 u_4	u_1 u_3	u_1 u_3 u_4	u_1 u_4

Is there a solution?

Workflow Satisfiability Problem (WSP), cont'd

s_1 create purchase order	s_2 approve purchase order
s_3 sign goods received note	s_4 countersign goods received note
s_5 create payment	s_6 approve payment

Constraints and authorizations:



Authorized Users					
u_1	u_2	u_3	u_1	u_1	u_1
u_2	u_3	u_4	u_3	u_3	u_4
u_3				u_4	

Is there a solution? Yes.

Definition of basic WSP

- Instance $W = (S, U, C \cup A)$, where S and U are sets of steps and users, C is a family of non-unary constraints with variables S and domain U. Unary constraints (authorizations): $A(s_i) = U'_i \subseteq U$.

Definition of basic WSP

- Instance $W = (S, U, C \cup A)$, where S and U are sets of steps and users, C is a family of non-unary constraints with variables S and domain U. Unary constraints (authorizations): $A(s_i) = U'_i \subseteq U$.
- $\pi : S \to U$ is a plan; π is valid if π satisfies all constraints $C \cup A$. W is satisfiable if there is a valid plan π.

Definition of basic WSP

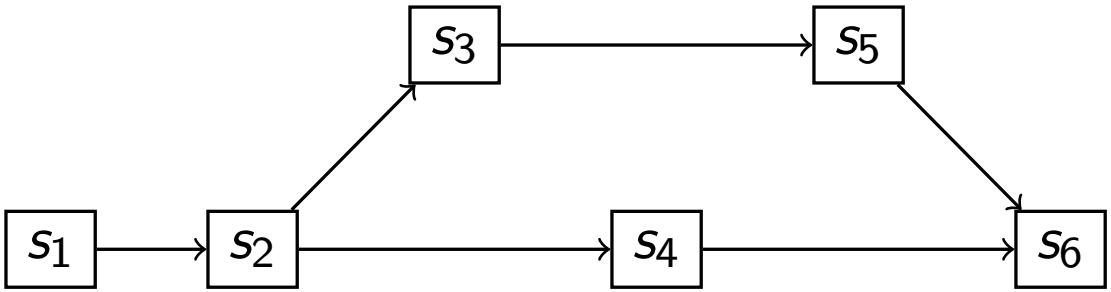
- Instance $W = (S, U, C \cup A)$, where S and U are sets of steps and users, C is a family of non-unary constraints with variables S and domain U. Unary constraints (authorizations): $A(s_i) = U'_i \subseteq U$.
- $\pi : S \to U$ is a plan; π is valid if π satisfies all constraints $C \cup A$. W is satisfiable if there is a valid plan π.
- Assumption: we can check, in poly time, whether a plan satisfies a constraint.

Definition of basic WSP

- Instance $W = (S, U, C \cup A)$, where S and U are sets of steps and users, C is a family of non-unary constraints with variables S and domain U. Unary constraints (authorizations): $A(s_i) = U'_i \subseteq U$.
- $\pi : S \to U$ is a plan; π is valid if π satisfies all constraints $C \cup A$. W is satisfiable if there is a valid plan π.
- Assumption: we can check, in poly time, whether a plan satisfies a constraint.
- basic WSP = CSP, where CSP variables = WSP steps, CSP values = WSP users. We'll talk about WSP.

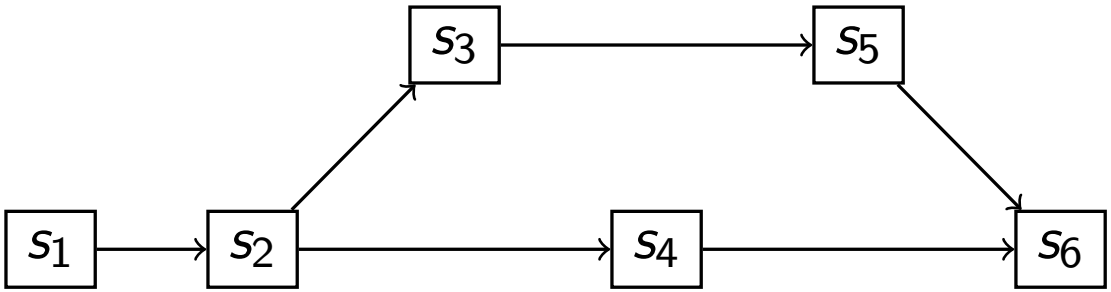
Less Basic WSP

Parallel branches, release points, etc. Crampton, GG, Watrigant (SACMAT 2017), Crampton, GG, Majumdar (SACMAT 2019)



Less Basic WSP

Parallel branches, release points, etc. Crampton, GG, Watrigant (SACMAT 2017), Crampton, GG, Majumdar (SACMAT 2019)



The rest of the talk: Basic WSP. Less Basic WSPs can often be reduced to Basic WSP.

Parameterized WSP

- In practice, $k = |S| \ll n = |U|$ and k is relatively small.

Parameterized WSP

- In practice, $k = |S| \ll n = |U|$ and k is relatively small.
- Wang and Li (ACM Trans. Inf. Syst. Secur., 2010): WSP parameterized by k (k-WSP). Is it fixed-parameter tractable (FPT)? Very likely not: k-WSP is W[1]-hard.

Parameterized WSP

- In practice, $k = |S| \ll n = |U|$ and k is relatively small.
- Wang and Li (ACM Trans. Inf. Syst. Secur., 2010): WSP parameterized by k (k-WSP). Is it fixed-parameter tractable (FPT)? Very likely not: k-WSP is W[1]-hard.
- Wang and Li (2010) considered constraints $(s_i, T, =)$ and $(s_i, T, \neq)$, where $T \subseteq S$. Restricted to these non-unary constraints, k-WSP is FPT. Unary constraints are arbitrary.

Parameterized WSP

- In practice, $k = |S| \ll n = |U|$ and k is relatively small.
- Wang and Li (ACM Trans. Inf. Syst. Secur., 2010): WSP parameterized by k (k-WSP). Is it fixed-parameter tractable (FPT)? Very likely not: k-WSP is W[1]-hard.
- Wang and Li (2010) considered constraints $(s_i, T, =)$ and $(s_i, T, \neq)$, where $T \subseteq S$. Restricted to these non-unary constraints, k-WSP is FPT. Unary constraints are arbitrary.
- Improved and generalized by Crampton, GG and Yeo (ACM Trans. Inf. Syst. Secur., 2013) to regular (undefined here) non-unary constraints with $O^*(2^k)$-algorithm. (Incl.Excl.)

Outline

User-Independent Constraints

- Cohen, Crampton, Gagarin, GG and Jones (JAIR, 2014): constraint c is user-independent (UI) if for every plan $\pi : S \to U$ satisfying c and permutation θ of U, plan $\pi' : S \to U$ also satisfies c, where for each $s \in S$, $\pi'(s) = \theta(\pi(s))$.

User-Independent Constraints

- Cohen, Crampton, Gagarin, GG and Jones (JAIR, 2014): constraint c is user-independent (UI) if for every plan $\pi : S \to U$ satisfying c and permutation θ of U, plan $\pi' : S \to U$ also satisfies c, where for each $s \in S$, $\pi'(s) = \theta(\pi(s))$.
- Counting constraints $(r, Q, \leq)$ and $(r, Q, \geq)$ are UI. $(r, Q, \leq)$ is satisfied by π if $|\pi(Q)| \leq r$ (confidentiality) and $(r, Q, \geq)$ is satisfied by π if $|\pi(Q)| \geq r$ (diversity). Example: AllDiff(Q)=$(|Q|, Q, \geq)$.

User-Independent Constraints

- Cohen, Crampton, Gagarin, GG and Jones (JAIR, 2014): constraint c is user-independent (UI) if for every plan $\pi : S \to U$ satisfying c and permutation θ of U, plan $\pi' : S \to U$ also satisfies c, where for each $s \in S$, $\pi'(s) = \theta(\pi(s))$.
- Counting constraints $(r, Q, \leq)$ and $(r, Q, \geq)$ are UI. $(r, Q, \leq)$ is satisfied by π if $|\pi(Q)| \leq r$ (confidentiality) and $(r, Q, \geq)$ is satisfied by π if $|\pi(Q)| \geq r$ (diversity). Example: AllDiff(Q)=$(|Q|, Q, \geq)$.
- All the non-unary constraints defined in the 2004 American National Standards Institute Role Based Access Control standard are UI.

Results for Non-unary User-Independent Constraints (Unary Constraints are Arbitrary)

Cohen, Crampton, Gagarin, GG and Jones (JAIR 2014):

- k-WSP with UI non-unary constraints can be solved in time $O^*(2^{k \log k})$.

Results for Non-unary User-Independent Constraints (Unary Constraints are Arbitrary)

Cohen, Crampton, Gagarin, GG and Jones (JAIR 2014):

- k-WSP with UI non-unary constraints can be solved in time $O^*(2^{k \log k})$.
- Unless Exponential Time Hypothesis (ETH) fails, no $O^*(2^{o(k \log k)})$-time algorithm.

Results for Non-unary User-Independent Constraints (Unary Constraints are Arbitrary)

Cohen, Crampton, Gagarin, GG and Jones (JAIR 2014):

- k-WSP with UI non-unary constraints can be solved in time $O^*(2^{k \log k})$.
- Unless Exponential Time Hypothesis (ETH) fails, no $O^*(2^{o(k \log k)})$-time algorithm.
- BFS-like user-iterative algorithm with exponential space.

Results for Non-unary User-Independent Constraints (Unary Constraints are Arbitrary), con'd

- Karapetyan, Gagarin and GG (FAW 2015): a backtracking $O^*(2^{k \log k})$-time poly-space algorithm for k-WSP with UI non-unary constraints, much faster in practice. Further improved by Karapetyan, Parks, GG and Gagarin (JAIR 2019).

Results for Non-unary User-Independent Constraints (Unary Constraints are Arbitrary), con'd

- Karapetyan, Gagarin and GG (FAW 2015): a backtracking $O^*(2^{k \log k})$-time poly-space algorithm for k-WSP with UI non-unary constraints, much faster in practice. Further improved by Karapetyan, Parks, GG and Gagarin (JAIR 2019).
- Gutin and Wahlström (IPL 2016): k-WSP with UI non-unary constraints cannot be solved in time $O^*(c^{k \log k})$ for any $c < 2$ unless Strong ETH fails.

Outline

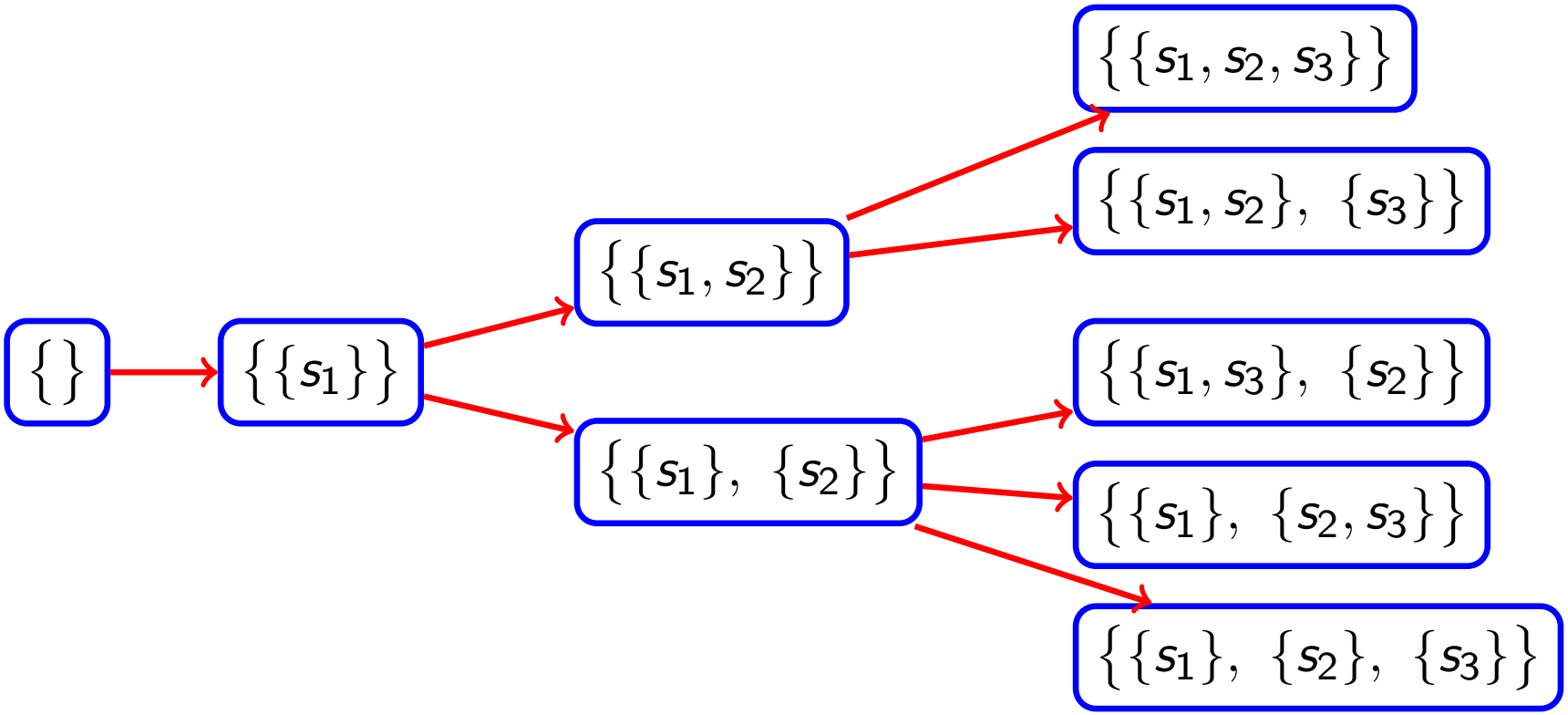
Basic Algorithm

- Generate all (unordered) partitions (called patterns) of S into non-empty subsets. [Runtime $O^*(B_k) = O^*(2^{k \log k})$, where B_k is k'th Bell number, $B_k \leq k!$.]
- Remove all of the patterns, which do not satisfy at least one non-unary constraint. [Runtime $O^*(B_k)$ provided that checking each non-unary constraint satisfaction takes $O^*(1)$ time.]

Basic Algorithm

- Generate all (unordered) partitions (called patterns) of S into non-empty subsets. [Runtime $O^*(B_k) = O^*(2^{k \log k})$, where B_k is k'th Bell number, $B_k \leq k!$.]
- Remove all of the patterns, which do not satisfy at least one non-unary constraint. [Runtime $O^*(B_k)$ provided that checking each non-unary constraint satisfaction takes $O^*(1)$ time.]
- For each pattern $S_1 \uplus \cdots \uplus S_t = S$ construct a bipartite graph B with bipartition $(\{S_1, \ldots, S_t\}, U)$ s.t. $S_i u_j \in E(B)$ if u_j is authorised for all steps in S_i. If B has a matching M of size t, then return π with $\pi(S_i) = u_j$ for every $S_i u_j \in M$. [Runtime $O^*(B_k)$.]

Generating Patterns for Backtracking



Outline

Instance Generation

- $n = 10k$, $10 \leq k \leq 60$.

Instance Generation

- $n = 10k$, $10 \leq k \leq 60$.
- Unary constraints: for each $u \in U$, first choose $b_u = |A(u)|$ randomly and uniformly from $\{1, \ldots, \lfloor k/2 \rfloor\}$, then $A(u)$ from all subsets of S of size b_u.

Instance Generation

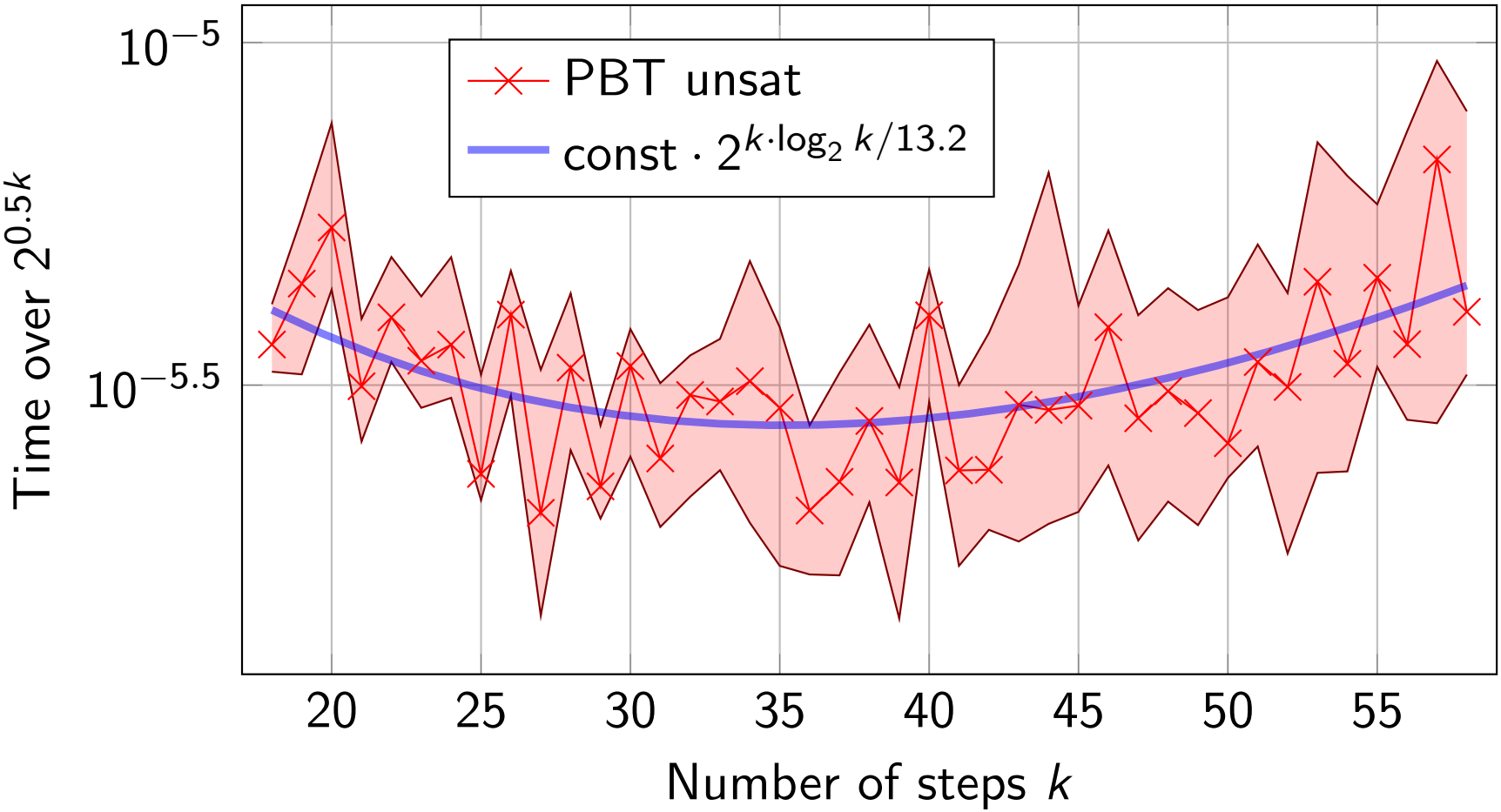
- $n = 10k$, $10 \leq k \leq 60$.
- Unary constraints: for each $u \in U$, first choose $b_u = |A(u)|$ randomly and uniformly from $\{1, \ldots, \lfloor k/2 \rfloor\}$, then $A(u)$ from all subsets of S of size b_u.
- Non-unary constraints:
  - k at-least-3-out-of-5 $c = (T, \geq)$, where $T \subseteq S$ of size 5. A plan π satisfies c if $|\pi(T)| \geq 3$.

Instance Generation

- $n = 10k$, $10 \leq k \leq 60$.
- Unary constraints: for each $u \in U$, first choose $b_u = |A(u)|$ randomly and uniformly from $\{1, \ldots, \lfloor k/2 \rfloor\}$, then $A(u)$ from all subsets of S of size b_u.
- Non-unary constraints:
  - k at-least-3-out-of-5 $c = (T, \geq)$, where $T \subseteq S$ of size 5. A plan π satisfies c if $|\pi(T)| \geq 3$.
  - k at-most-3-out-of-5 $c = (T, \leq)$, where $T \subseteq S$ of size 5. A plan π satisfies c if $|\pi(T)| \leq 3$.

Instance Generation

- $n = 10k$, $10 \leq k \leq 60$.
- Unary constraints: for each $u \in U$, first choose $b_u = |A(u)|$ randomly and uniformly from $\{1, \ldots, \lfloor k/2 \rfloor\}$, then $A(u)$ from all subsets of S of size b_u.
- Non-unary constraints:
  - k at-least-3-out-of-5 $c = (T, \geq)$, where $T \subseteq S$ of size 5. A plan π satisfies c if $|\pi(T)| \geq 3$.
  - k at-most-3-out-of-5 $c = (T, \leq)$, where $T \subseteq S$ of size 5. A plan π satisfies c if $|\pi(T)| \leq 3$.
  - e non-equals $c = (s_i, s_j, \neq)$: plan π satisfies c if $\pi(s_i) \neq \pi(s_j)$. e is chosen s.t. $\mathbb{P}[(S, U, C)$ being satisfiable$]$ is approx 50%. They are hardest instances.


PBT unsat
const · $2^{k \cdot \log_2 k/13.2}$
Time over $2^{0.5k}$
10^{-5}
$10^{-5.5}$
20
25
30
35
40
45
50
55
Number of steps k

Contestants

PBT Improved Algorithm aka Pattern Back Tracking (in C#)

Contestants

PBT Improved Algorithm aka Pattern Back Tracking (in C#)

PUI the initial FPT algorithm by Cohen et al. (in C++)

Contestants

PBT Improved Algorithm aka Pattern Back Tracking (in C#)

PUI the initial FPT algorithm by Cohen et al. (in C++)

UDBP (Res) SAT4J using 'ordinary' pseudo-Boolean formulation of WSP in the resolution proof system mode (cutting plane mode was much worse).

Contestants

PBT Improved Algorithm aka Pattern Back Tracking (in C#)

PUI the initial FPT algorithm by Cohen et al. (in C++)

UDBP (Res) SAT4J using 'ordinary' pseudo-Boolean formulation of WSP in the resolution proof system mode (cutting plane mode was much worse).

PBPB (Res) SAT4J using 'FPT' formulation pseudo-Boolean formulation of WSP in the resolution proof system mode.

Contestants

PBT Improved Algorithm aka Pattern Back Tracking (in C#)

PUI the initial FPT algorithm by Cohen et al. (in C++)

UDBP (Res) SAT4J using 'ordinary' pseudo-Boolean formulation of WSP in the resolution proof system mode (cutting plane mode was much worse).

PBPB (Res) SAT4J using 'FPT' formulation pseudo-Boolean formulation of WSP in the resolution proof system mode.

PBPB (CutP) SAT4J using 'FPT' formulation pseudo-Boolean formulation of WSP in the cutting plane mode.

Contestants

PBT Improved Algorithm aka Pattern Back Tracking (in C#)

PUI the initial FPT algorithm by Cohen et al. (in C++)

UDBP (Res) SAT4J using 'ordinary' pseudo-Boolean formulation of WSP in the resolution proof system mode (cutting plane mode was much worse).

PBPB (Res) SAT4J using 'FPT' formulation pseudo-Boolean formulation of WSP in the resolution proof system mode.

PBPB (CutP) SAT4J using 'FPT' formulation pseudo-Boolean formulation of WSP in the cutting plane mode.

CP-SAT CP-SAT from OR-tools (Google) using 'ordinary' CSP formulation of WSP

Ordinary Pseudo-Boolean formulations of WSP

- $x_{s,u} = 1$ iff u is assigned to s.

Ordinary Pseudo-Boolean formulations of WSP

- $x_{s,u} = 1$ iff u is assigned to s.
- 'Ordinary' formulation (only not-equals):

$$\begin{aligned} \textstyle\sum_{u \in U} x_{s,u} = 1 \quad & \forall s \in S, \\ x_{s,u} = 0 \quad & \forall s \in S \text{ and } \forall u \in U \setminus A(s), \\ x_{s_1,u} + x_{s_2,u} \leq 1 \quad & \forall \text{ not-equals with scope } \{s_1, s_2\} \text{ and } \forall u \in U. \end{aligned}$$

FPT Formulation

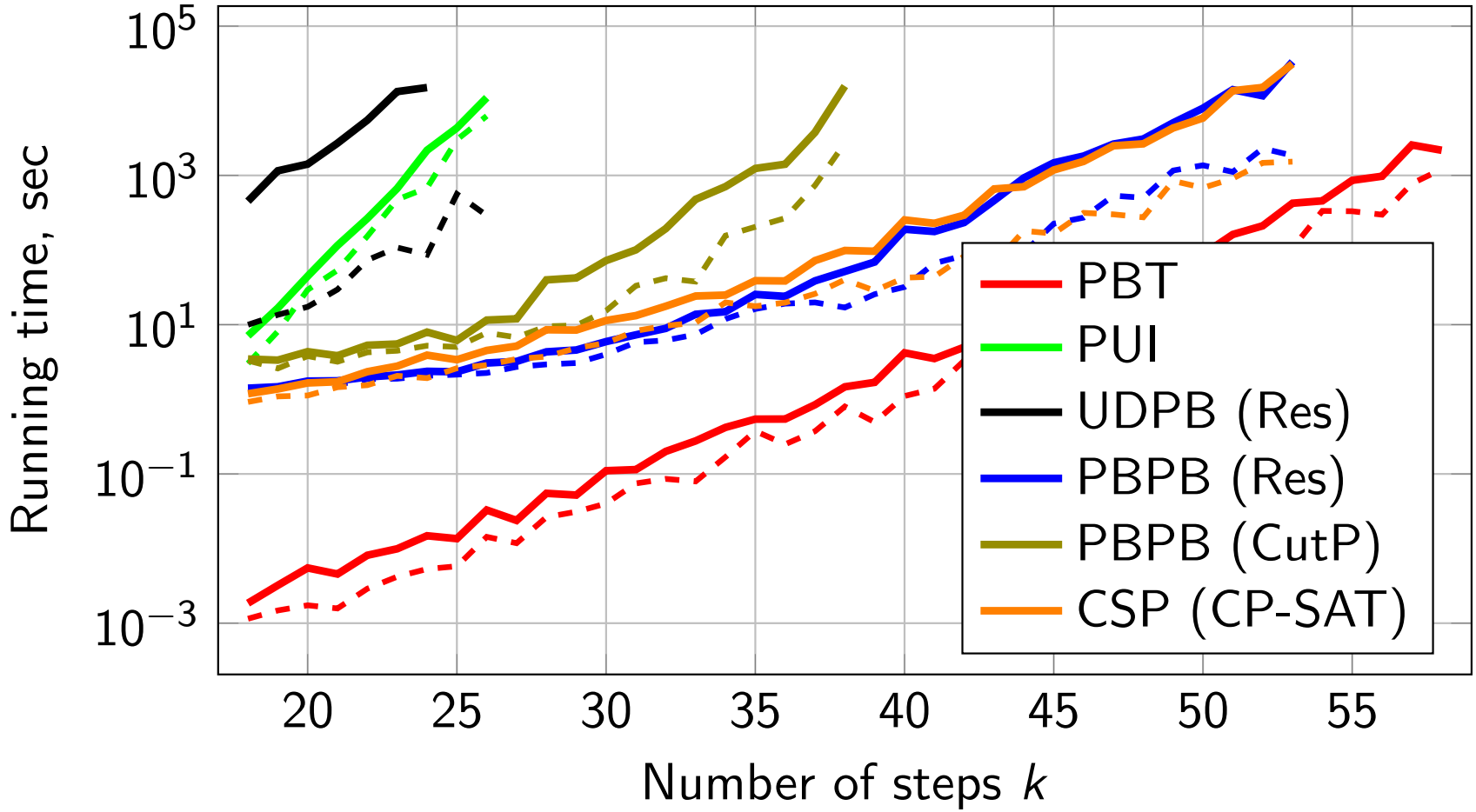
'FPT' formulation uses $x_{s,u}$'s and 'FPT variables' $M_{s,s'}$. $M_{s,s'} = 1$ if s, s' are assigned the same user and $M_{s,s'} = 0$, otherwise. Much longer formulation.

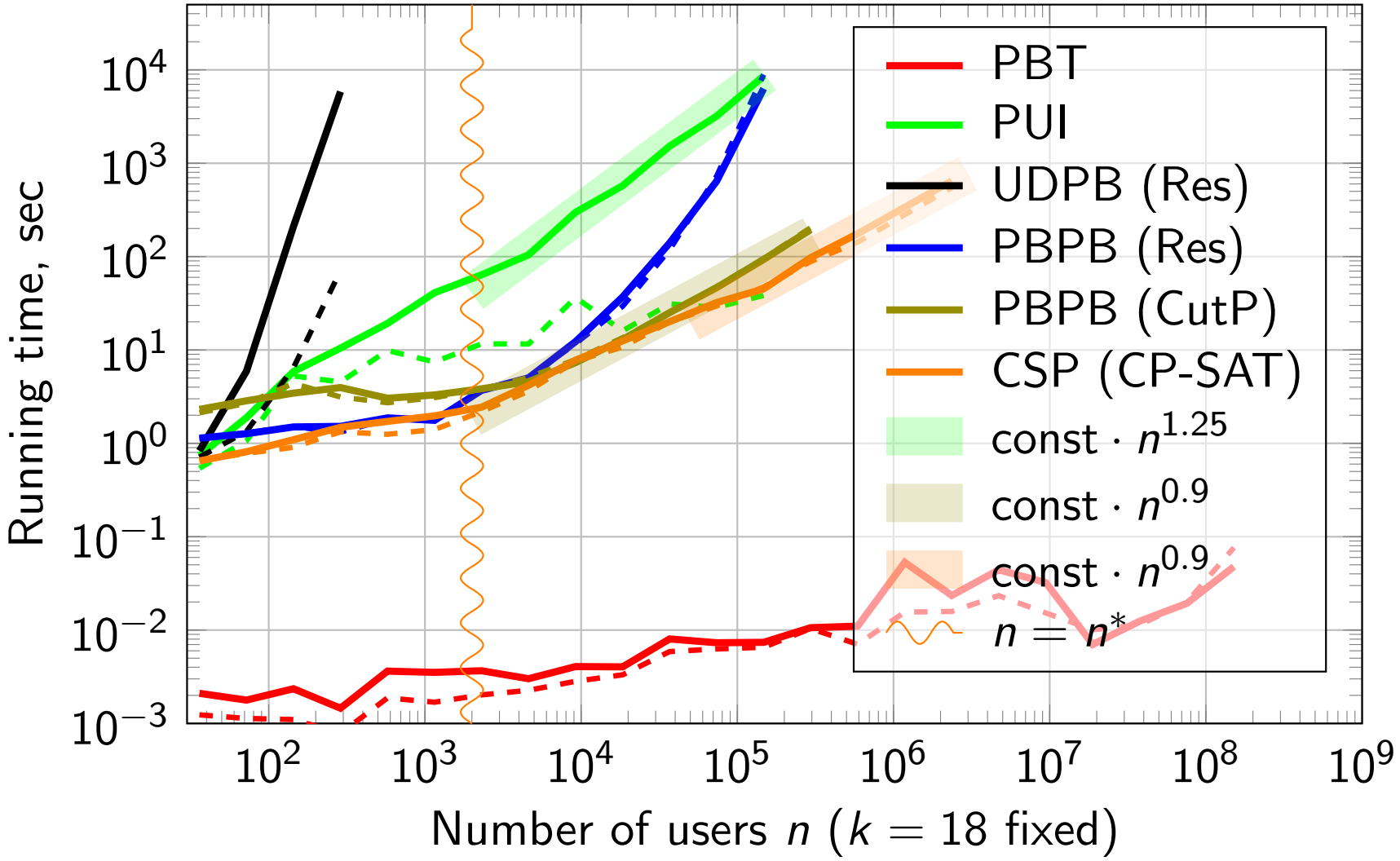
FPT Formulation

'FPT' formulation uses $x_{s,u}$'s and 'FPT variables' $M_{s,s'}$. $M_{s,s'} = 1$ if s, s' are assigned the same user and $M_{s,s'} = 0$, otherwise. Much longer formulation.

Theorem

Every non-binary UI constraint can be encoded using $M_{s,s'}$'s.


Running time, sec
10^5
10^3
10^1
10^{-1}
10^{-3}
20
25
30
35
40
45
50
55
Number of steps k
PBT
PUI
UDPB (Res)
PBPB (Res)
PBPB (CutP)
CSP (CP-SAT)


Running time, sec
10^4
10^3
10^2
10^1
10^0
10^{-1}
10^{-2}
10^{-3}
10^2
10^3
10^4
10^5
10^6
10^7
10^8
10^9
Number of users n ($k = 18$ fixed)
PBT
PUI
UDPB (Res)
PBPB (Res)
PBPB (CutP)
CSP (CP-SAT)
const $\cdot\, n^{1.25}$
const $\cdot\, n^{0.9}$
const $\cdot\, n^{0.9}$
$n = n^*$

Report of Experimental Results

- Winner: PBT, Runners-up: CP-SAT and the best of PBPB (Res) and PBPB (CutP).

Report of Experimental Results

- Winner: PBT, Runners-up: CP-SAT and the best of PBPB (Res) and PBPB (CutP).
- Comparison: For $k = 50$, PBT takes median time 100 sec. for UNSAT instances. CP-SAT is 2-3 orders of magnitude slower than PBT and uses much more memory.

Report of Experimental Results

- Winner: PBT, Runners-up: CP-SAT and the best of PBPB (Res) and PBPB (CutP).
- Comparison: For $k = 50$, PBT takes median time 100 sec. for UNSAT instances. CP-SAT is 2-3 orders of magnitude slower than PBT and uses much more memory.
- Remarkable: CP-SAT uses 'ordinary' CSP formulation of WSP, not 'FPT' (we did not know how to get one), nevertheless displays an FPT-like behaviour.

Report of Experimental Results

- Winner: PBT, Runners-up: CP-SAT and the best of PBPB (Res) and PBPB (CutP).
- Comparison: For $k = 50$, PBT takes median time 100 sec. for UNSAT instances. CP-SAT is 2-3 orders of magnitude slower than PBT and uses much more memory.
- Remarkable: CP-SAT uses 'ordinary' CSP formulation of WSP, not 'FPT' (we did not know how to get one), nevertheless displays an FPT-like behaviour.
- Likely explanation: CP-SAT formulation is reduced to a SAT formulation. 'FPT variables' appear in the SAT formulation.

Outline

Valued WSP

- Crampton, GG, Karapetyan (SACMAT 2015, BPA).

Valued WSP

- Crampton, GG, Karapetyan (SACMAT 2015, BPA).
- Sometimes WSP instance cannot be satisfied, but it will be OK to falsify some 'soft' constraints especially if they are not falsified 'too much', e.g. for at-most-3-out-of-5 "only" 4 users, not 5, are assigned.

Valued WSP

- Crampton, GG, Karapetyan (SACMAT 2015, BPA).
- Sometimes WSP instance cannot be satisfied, but it will be OK to falsify some 'soft' constraints especially if they are not falsified 'too much', e.g. for at-most-3-out-of-5 "only" 4 users, not 5, are assigned.
- Formalisation: For each constraint c and plan π, if π satisfies c then $w_c(\pi) = 0$, otherwise $w_c(\pi) > 0$. We wish to find $\text{argmin}_\pi \sum_{c \in C} w_c(\pi)$.

Valued WSP

- Crampton, GG, Karapetyan (SACMAT 2015, BPA).
- Sometimes WSP instance cannot be satisfied, but it will be OK to falsify some 'soft' constraints especially if they are not falsified 'too much', e.g. for at-most-3-out-of-5 "only" 4 users, not 5, are assigned.
- Formalisation: For each constraint c and plan π, if π satisfies c then $w_c(\pi) = 0$, otherwise $w_c(\pi) > 0$. We wish to find $\text{argmin}_\pi \sum_{c \in C} w_c(\pi)$.
- A constraint c is UI (for Valued WSP) if $w_c(\pi) = w_c(\pi')$, where π and π' are equivalent, i.e. the same up to permutation of users.

Valued WSP

- Crampton, GG, Karapetyan (SACMAT 2015, BPA).
- Sometimes WSP instance cannot be satisfied, but it will be OK to falsify some 'soft' constraints especially if they are not falsified 'too much', e.g. for at-most-3-out-of-5 "only" 4 users, not 5, are assigned.
- Formalisation: For each constraint c and plan π, if π satisfies c then $w_c(\pi) = 0$, otherwise $w_c(\pi) > 0$. We wish to find $\text{argmin}_\pi \sum_{c \in C} w_c(\pi)$.
- A constraint c is UI (for Valued WSP) if $w_c(\pi) = w_c(\pi')$, where π and π' are equivalent, i.e. the same up to permutation of users.
- If all non-unary constraints are UI, $O^*(2^{k \log k})$-algorithm.

Bi-Objective WSP

- Crampton, GG, Karapetyan (J. Comput. Sec. 2017).

Bi-Objective WSP

- Crampton, GG, Karapetyan (J. Comput. Sec. 2017).
- This approach was used by Curry et al. (SACMAT 2020), all authors with HashiCorp. Inc., SF, USA. On partial Pareto front for Multi-Objective problem in Role-Based Access Control (RBAC).

Bi-Objective WSP

- Crampton, GG, Karapetyan (J. Comput. Sec. 2017).
- This approach was used by Curry et al. (SACMAT 2020), all authors with HashiCorp. Inc., SF, USA. On partial Pareto front for Multi-Objective problem in Role-Based Access Control (RBAC).
- C_1 unary constraints, $w_1(\pi) = \sum_{c \in C_1} w_c(\pi)$, C_2 non-unary constraints, $w_2(\pi) = \sum_{c \in C_2} w_c(\pi)$.

Bi-Objective WSP

- Crampton, GG, Karapetyan (J. Comput. Sec. 2017).
- This approach was used by Curry et al. (SACMAT 2020), all authors with HashiCorp. Inc., SF, USA. On partial Pareto front for Multi-Objective problem in Role-Based Access Control (RBAC).
- C_1 unary constraints, $w_1(\pi) = \sum_{c \in C_1} w_c(\pi)$, C_2 non-unary constraints, $w_2(\pi) = \sum_{c \in C_2} w_c(\pi)$.
- Find Pareto front (Pf) of $\text{argmin}_\pi(w_1(\pi), w_2(\pi))$. $\pi \in$ Pf iff $\nexists\, \pi'$ with $w_1(\pi') + w_2(\pi') < w_1(\pi) + w_2(\pi)$, $w_1(\pi') \leq w_1(\pi)$, and $w_2(\pi') \leq w_2(\pi)$.

Bi-Objective WSP

- Crampton, GG, Karapetyan (J. Comput. Sec. 2017).
- This approach was used by Curry et al. (SACMAT 2020), all authors with HashiCorp. Inc., SF, USA. On partial Pareto front for Multi-Objective problem in Role-Based Access Control (RBAC).
- C_1 unary constraints, $w_1(\pi) = \sum_{c \in C_1} w_c(\pi)$, C_2 non-unary constraints, $w_2(\pi) = \sum_{c \in C_2} w_c(\pi)$.
- Find Pareto front (Pf) of $\text{argmin}_\pi(w_1(\pi), w_2(\pi))$. $\pi \in$ Pf iff $\nexists\, \pi'$ with $w_1(\pi') + w_2(\pi') < w_1(\pi) + w_2(\pi)$, $w_1(\pi') \leq w_1(\pi)$, and $w_2(\pi') \leq w_2(\pi)$.
- If all non-unary constraints are UI, $O^*(2^{k \log k})$-algorithm.

Experimental Results for Bi-Objective WSP

- Contestants: PBnB vs CPLEX 12.6.

Experimental Results for Bi-Objective WSP

- Contestants: PBnB vs CPLEX 12.6.
- $k = 20, 25, 30, 35$, $n = 10k$. 1 hour per instance.

Experimental Results for Bi-Objective WSP

- Contestants: PBnB vs CPLEX 12.6.
- $k = 20, 25, 30, 35$, $n = 10k$. 1 hour per instance.
- PBnB solved every instance for $k \leq 30$ and a large majority for $k = 35$.

Experimental Results for Bi-Objective WSP

- Contestants: PBnB vs CPLEX 12.6.
- $k = 20, 25, 30, 35$, $n = 10k$. 1 hour per instance.
- PBnB solved every instance for $k \leq 30$ and a large majority for $k = 35$.
- CPLEX 12.6 solved all instances only for $k = 20$, for $k = 35$ less than 43%.

Outline

Role Mining Problem

- Role-Based Access Control (RBAC), etc.

Role Mining Problem

- Role-Based Access Control (RBAC), etc.
- **Input:** users $U = \{u_1, \ldots, u_m\}$, permissions $P = \{p_1, \ldots, p_n\}$, Boolean matrix **UPA** with $\mathbf{UPA}_{ij} = 1$ iff u_i allowed p_j and $r \in \mathbb{N}_{>0}$ (normally, $r \ll \min\{m, n\}$).

Role Mining Problem

- Role-Based Access Control (RBAC), etc.
- **Input:** users $U = \{u_1, \ldots, u_m\}$, permissions $P = \{p_1, \ldots, p_n\}$, Boolean matrix **UPA** with $\mathbf{UPA}_{ij} = 1$ iff u_i allowed p_j and $r \in \mathbb{N}_{>0}$ (normally, $r \ll \min\{m, n\}$).
- **Output:** (If they exist) **UA**, **PA** s.t. $\mathbf{UA} \wedge \mathbf{PA} = \mathbf{UPA}$. Here $\wedge$ instead of $\times$ and $\vee$ instead of $+$.

Role Mining Problem

- Role-Based Access Control (RBAC), etc.
- **Input:** users $U = \{u_1, \ldots, u_m\}$, permissions $P = \{p_1, \ldots, p_n\}$, Boolean matrix **UPA** with $\mathbf{UPA}_{ij} = 1$ iff u_i allowed p_j and $r \in \mathbb{N}_{>0}$ (normally, $r \ll \min\{m, n\}$).
- **Output:** (If they exist) **UA**, **PA** s.t. $\mathbf{UA} \wedge \mathbf{PA} = \mathbf{UPA}$. Here $\wedge$ instead of $\times$ and $\vee$ instead of $+$.
- What if the output is NO?

Noise Role Mining Problem

- **Input:** the same as ROLE MINING and k.

Noise Role Mining Problem

- **Input:** the same as Role Mining and k.
- **Output:** (If they exist) **UA**, **PA** s.t. $\text{dist}_{\text{Hamming}}(\textbf{UA} \wedge \textbf{PA}, \textbf{UPA}) \leq k$.

Noise Role Mining Problem

- **Input:** the same as Role Mining and k.
- **Output:** (If they exist) **UA**, **PA** s.t. $\text{dist}_{\text{Hamming}}(\textbf{UA} \wedge \textbf{PA}, \textbf{UPA}) \leq k$.
- Introduced by Lu, Vaidya and Atluri (Int. Conf. Data Eng., 2008). Studied by many incl. Fomin, Golovach and Panolan (Data Mining Know. Disc., 2020) who used PC.

Shortcomings of Noise Role Mining Problem

- For Security-preserving Role Mining, we should also require **UPA** $\geq$ **UA** $\wedge$ **PA**.

Shortcomings of Noise Role Mining Problem

- For Security-preserving Role Mining, we should also require **UPA** $\geq$ **UA** $\wedge$ **PA**.
- For Availability-preserving Role Mining, we should require **UPA** $\leq$ **UA** $\wedge$ **PA**.

Shortcomings of Noise Role Mining Problem

- For Security-preserving Role Mining, we should also require **UPA** $\geq$ **UA** $\wedge$ **PA**.
- For Availability-preserving Role Mining, we should require **UPA** $\leq$ **UA** $\wedge$ **PA**.
- So, Noise Role Mining should be generalized.

F-distances

label $m \times n$-matrix $\mathbf{F}$ with $\mathbf{f}_{ij} \in \{\top, \bot\}$, $\mathbf{A}$ and $\mathbf{B}$ are Boolean $m \times n$-matrices.

F-distances

label $m \times n$-matrix $\mathbf{F}$ with $\mathbf{f}_{ij} \in \{\top, \bot\}$, $\mathbf{A}$ and $\mathbf{B}$ are Boolean $m \times n$-matrices.

$$\mathrm{dist}_{\mathbf{F}}(\mathbf{a}_{ij}, \mathbf{b}_{ij}) = \begin{cases} \infty & \mathbf{f}_{ij} = \bot \text{ and } \mathbf{a}_{ij} \neq \mathbf{b}_{ij}, \\ |\mathbf{a}_{ij} - \mathbf{b}_{ij}| & \text{otherwise} \end{cases}$$

F-distances

label $m \times n$-matrix $\mathbf{F}$ with $\mathbf{f}_{ij} \in \{\top, \bot\}$, $\mathbf{A}$ and $\mathbf{B}$ are Boolean $m \times n$-matrices.

$$\mathrm{dist}_{\mathbf{F}}(\mathbf{a}_{ij}, \mathbf{b}_{ij}) = \begin{cases} \infty & \mathbf{f}_{ij} = \bot \text{ and } \mathbf{a}_{ij} \neq \mathbf{b}_{ij}, \\ |\mathbf{a}_{ij} - \mathbf{b}_{ij}| & \text{otherwise} \end{cases}$$

$$\mathrm{dist}_{\mathbf{F}}(\mathbf{A}, \mathbf{B}) = \sum_{i=1}^{m} \sum_{j=1}^{n} \mathrm{dist}_{\mathbf{F}}(\mathbf{a}_{ij}, \mathbf{b}_{ij}).$$

F-distances

label $m \times n$-matrix $\mathbf{F}$ with $\mathbf{f}_{ij} \in \{\top, \bot\}$, $\mathbf{A}$ and $\mathbf{B}$ are Boolean $m \times n$-matrices.

$$\mathrm{dist}_{\mathbf{F}}(\mathbf{a}_{ij}, \mathbf{b}_{ij}) = \begin{cases} \infty & \mathbf{f}_{ij} = \bot \text{ and } \mathbf{a}_{ij} \neq \mathbf{b}_{ij}, \\ |\mathbf{a}_{ij} - \mathbf{b}_{ij}| & \text{otherwise} \end{cases}$$

$\mathrm{dist}_{\mathbf{F}}(\mathbf{A}, \mathbf{B}) = \sum_{i=1}^{m} \sum_{j=1}^{n} \mathrm{dist}_{\mathbf{F}}(\mathbf{a}_{ij}, \mathbf{b}_{ij})$.

For Noise Role Mining, set $\mathbf{f}_{ij} = \top$ for all i, j. For Security-preserving Role Mining, set $\mathbf{f}_{ij} = \bot$ iff $\mathbf{UPA}_{ij} = 0$.

Generalized Noise Role Mining Problem (GNRM)

- **Input:** the same as Noise Role Mining and **F**.
  Parameter: $r + k$.

Generalized Noise Role Mining Problem (GNRM)

- **Input:** the same as Noise Role Mining and **F**.
  Parameter: $r + k$.
- **Output:** (If they exist) **UA**, **PA** s.t.
  $\text{dist}_{\mathbf{F}}(\mathbf{UA} \wedge \mathbf{PA}, \mathbf{UPA}) \leq k$.

Generalized Noise Role Mining Problem (GNRM)

- **Input:** the same as Noise Role Mining and **F**.
  Parameter: $r + k$.
- **Output:** (If they exist) **UA**, **PA** s.t.
  $\text{dist}_{\mathbf{F}}(\mathbf{UA} \wedge \mathbf{PA}, \mathbf{UPA}) \leq k$.
- **Theorem** (Crampton et al., SACMAT 2022, BPA) GNRM admits an $O^*(2^{O(r2^r+rk)})$-time algorithm.

Generalized Noise Role Mining Problem (GNRM)

- **Input:** the same as Noise Role Mining and **F**.
  Parameter: $r + k$.
- **Output:** (If they exist) **UA**, **PA** s.t.
  $\text{dist}_{\mathbf{F}}(\mathbf{UA} \wedge \mathbf{PA}, \mathbf{UPA}) \leq k$.
- **Theorem** (Crampton et al., SACMAT 2022, BPA) GNRM admits an $O^*(2^{O(r2^r+rk)})$-time algorithm.
- Used the approach of Fomin et al. (Data Mining Know. Disc., 2020)

Generalized Noise Cover Role Mining Problem (GNCMP)

- Instances of GNRM may not have a solution for acceptable values of k and r (as in our experiments).

Generalized Noise Cover Role Mining Problem (GNCMP)

- Instances of GNRM may not have a solution for acceptable values of k and r (as in our experiments).
- New parameter: $r + \mathrm{cov}_U + \mathrm{cov}_P$. Here cov_U and cov_P are the number of users and permissions, where changes are allowed.

Generalized Noise Cover Role Mining Problem (GNCMP)

- Instances of GNRM may not have a solution for acceptable values of k and r (as in our experiments).
- New parameter: $r + \text{cov}_U + \text{cov}_P$. Here cov_U and cov_P are the number of users and permissions, where changes are allowed.
- Generalized Noise Cover Role Mining: GNRM, but changes are allowed in at most cov_U rows and cov_P columns of **UPA**.

Generalized Noise Cover Role Mining Problem (GNCMP)

- Instances of GNRM may not have a solution for acceptable values of k and r (as in our experiments).
- New parameter: $r + \mathrm{cov}_U + \mathrm{cov}_P$. Here cov_U and cov_P are the number of users and permissions, where changes are allowed.
- Generalized Noise Cover Role Mining: GNRM, but changes are allowed in at most cov_U rows and cov_P columns of **UPA**.
- **Theorem** (Crampton et al., SACMAT 2022) Noise/Availability-preserving/Security-preserving Cover Role Mining parameterized by $r + \mathrm{cov}_U + \mathrm{cov}_P$ admits an FPT algorithm.

Experiments for GNRM

- Used pseudo-random instance generator of Vaidya et al. (2010), $n = m = 20$. CP-SAT solver from Google's OR-Tools, version 9.2.9972.

Experiments for GNRM

- Used pseudo-random instance generator of Vaidya et al. (2010), $n = m = 20$. CP-SAT solver from Google's OR-Tools, version 9.2.9972.
- Interesting outcome for $k_{\min}$: SECURITY and NOISE versions it's almost the same, but AVAILABILITY it's often much larger.

Experiments for GNRM

- Used pseudo-random instance generator of Vaidya et al. (2010), $n = m = 20$. CP-SAT solver from Google's OR-Tools, version 9.2.9972.
- Interesting outcome for $k_{\min}$: SECURITY and NOISE versions it's almost the same, but AVAILABILITY it's often much larger.
- Interesting outcome for CP-SAT solver: FPT-like runtime for all three types of instances.

Experiments for GNRM

- Used pseudo-random instance generator of Vaidya et al. (2010), $n = m = 20$. CP-SAT solver from Google's OR-Tools, version 9.2.9972.
- Interesting outcome for $k_{\min}$: SECURITY and NOISE versions it's almost the same, but AVAILABILITY it's often much larger.
- Interesting outcome for CP-SAT solver: FPT-like runtime for all three types of instances.
- Usually, it's much faster to solve SECURITY instances than NOISE and AVAILABILITY ones.

Outline

Some Take Aways for Practical Computing

- FPT algorithms may be quite useful in practice.

Some Take Aways for Practical Computing

- FPT algorithms may be quite useful in practice.
- If an NP-hard problem is FPT, off-the-shelf solvers may be made more efficient if some variables used in the constraints correspond to the parameter(s).

Some Take Aways for Practical Computing

- FPT algorithms may be quite useful in practice.
- If an NP-hard problem is FPT, off-the-shelf solvers may be made more efficient if some variables used in the constraints correspond to the parameter(s).
- Some solvers may take advantage of NP-hard problems being FPT without any help from the users.