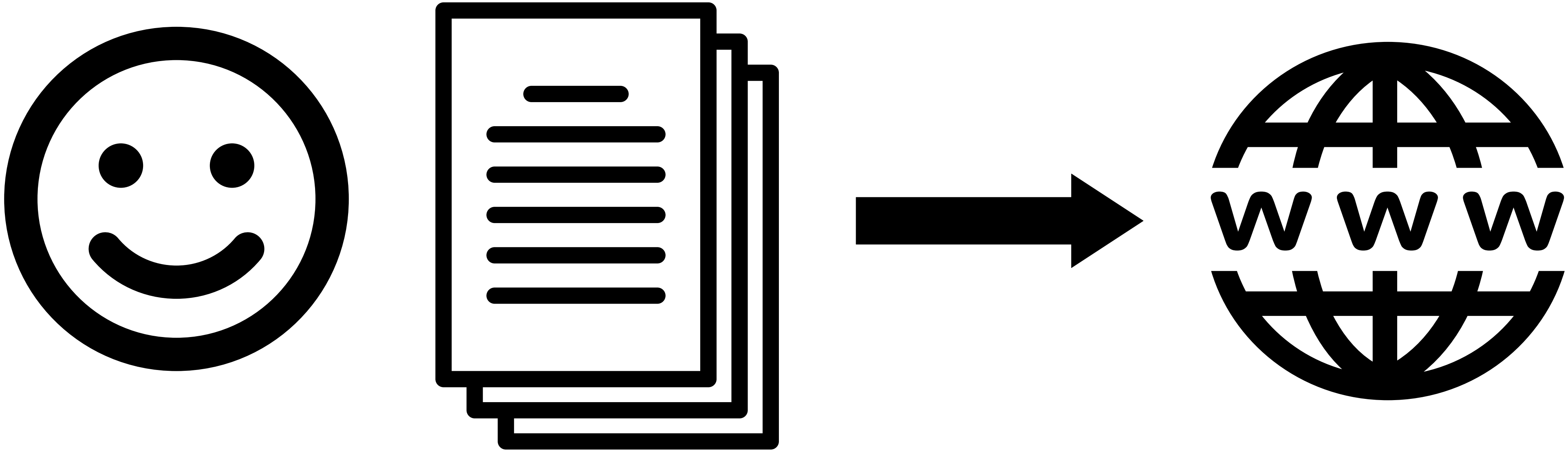
Hide and Mine in Strings: Hardness and Algorithms

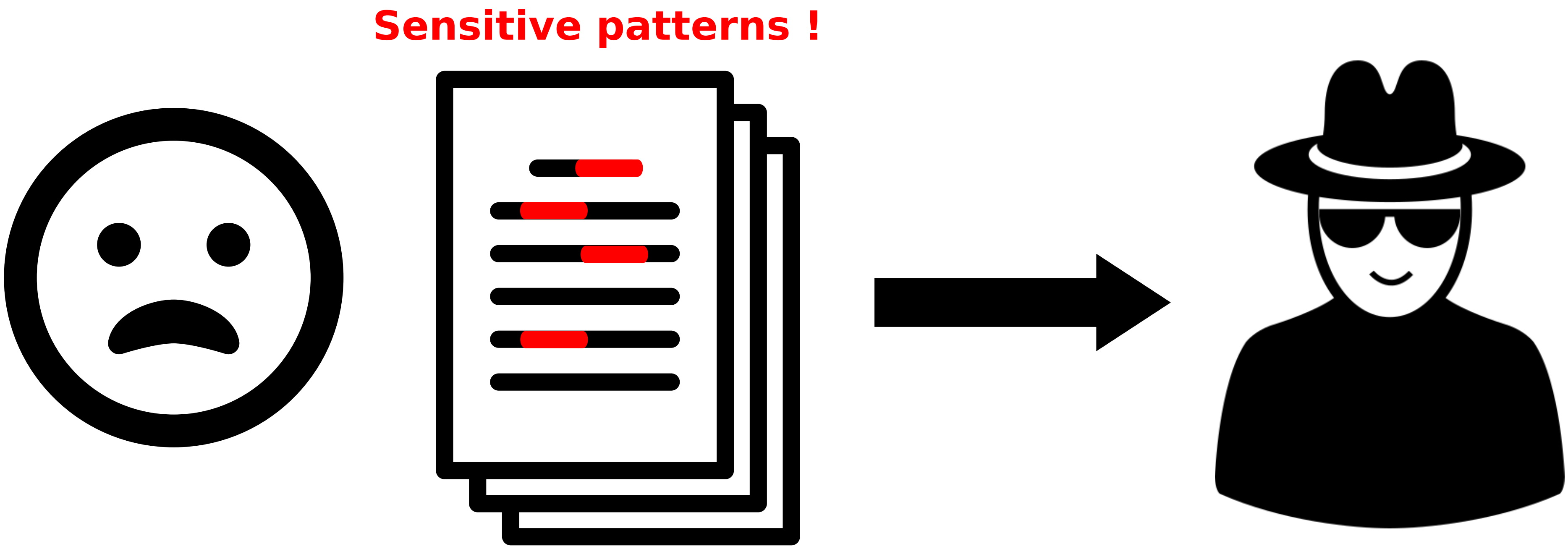
Published in ICDM 2020

Giulia Bernardini, Alessio Conte, **Garance Gourdel**, Roberto Grossi, Grigorios Loukides, Nadia Pisanti, Solon P. Pissis, Giulia Punzi, Leen Stougie, Michelle Sweering

Problem & Motivation

Release to the public for pattern mining




Sensitive patterns !

Hide the sensitive patterns !

For a given integer k,

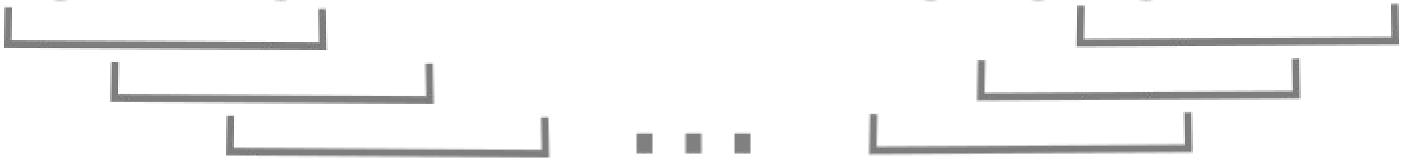
We describe the text by its k-mers.

$k = 3$

Applications in:

- Route planning [Chen et al, ‘15]
- Marketing [Agrawal et al. ‘95]
- Clinical diagnosis [Koboldt et al ‘13]

$$W = \mathrm{GACAAAAAACCCAT}$$



The Hide and Mine problem

String sanitization

For a given integer k,

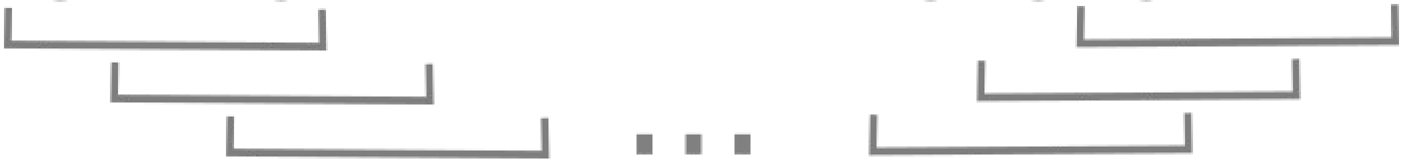
We describe the text by its k-mers.

$$k = 3$$

Given a set S of forbidden k-mers to hide.

$$S = \{ACA, CAA, AAA, AAC, CCA\}$$

$$W = \text{GACAAAAACCCAT}$$



The Hide and Mine problem

String sanitization

For a given integer k,

We describe the text by its k-mers.

$k = 3$

Given a set S of forbidden k-mers to hide.

$S = \{ACA, CAA, AAA, AAC, CCA\}$

GACAAAAACCCAT

ACA AAC

CAA CCA

AAA

Forbidden k-mers

For a given integer k ,

We describe the text by its k-mers.

$k = 3$

Given a set S of forbidden k-mers to hide.

$S = \{ACA, CAA, AAA, AAC, CCA\}$

GACAAAAAACCCAT

CCC

CAT

Non-forbidden k-mers

$k = 3 \quad S = \{ACA, CAA, AAA, AAC, CCA\}$

GACAAAAACCCAT

CCC

CAT

Non-forbidden k-mers

How can we reconstruct a text with those k-mers?

The Hide and Mine problem

String sanitization

$k = 3 \quad S = \{ACA, CAA, AAA, AAC, CCA\}$

GACAAAAACCCAT

CCC

CAT

Non-forbidden k-mers

How can we reconstruct a text with those k-mers?

- We preserve the non-forbidden k-mers, and their order.

The Hide and Mine problem

String sanitization

$k = 3 \quad S = \{ACA, CAA, AAA, AAC, CCA\}$

GACAAAAACCCAT
CCC
CAT

Non-forbidden k-mers

How can we reconstruct a text with those k-mers?

- We preserve the non-forbidden k-mers, and their order.

GACCC
CAT

The Hide and Mine problem

String sanitization

$k = 3 \quad S = \{ACA, CAA, AAA, AAC, CCA\}$

GACAAAAACCCAT

CCC

CAT

Non-forbidden k-mers

How can we reconstruct a text with those k-mers?

- We preserve the non-forbidden k-mers, and their order.

GACCC

CAT

We might not have a full text!

The Hide and Mine problem

String sanitization

$k = 3 \quad S = \{ACA, CAA, AAA, AAC, CCA\}$

GACAAAAACCCAT

CCC

CAT

Non-forbidden k-mers

How can we reconstruct a text with those k-mers?

- We preserve the non-forbidden k-mers, and their order.
- We can use a special character “#” as a separator.

GACCC

CAT

We might not have a full text!

$k = 3 \quad S = \{ACA, CAA, AAA, AAC, CCA\}$

GACAAAAACCCAT
CCC
CAT

Non-forbidden k-mers

How can we reconstruct a text with those k-mers?

- We preserve the non-forbidden k-mers, and their order.
- We can use a special character “#” as a separator.

X=GACCC#CAT

The Hide and Mine problem

String sanitization

$$k = 3 \quad S = \{ACA, CAA, AAA, AAC, CCA\} \quad W = GACAAAAACCCAT$$

- We preserve the non-forbidden k-mers, and their order.
- We can use a special character “#” as a separator.

Plenty of possibilities!

The Hide and Mine problem *String sanitization*

$$k = 3 \quad S = \{ACA, CAA, AAA, AAC, CCA\} \quad W = GACAAAAACCCAT$$

- We preserve the non-forbidden k-mers, and their order.
- We can use a special character “#” as a separator.

Plenty of possibilities!

Separate all non-forbidden k-mers by # X_tr = GAC#ACC#CCC#CAT

The Hide and Mine problem *String sanitization*

$$k = 3 \quad S = \{ACA, CAA, AAA, AAC, CCA\} \quad W = GACAAAAACCCAT$$

- We preserve the non-forbidden k-mers, and their order.
- We can use a special character “#” as a separator.

Plenty of possibilities!

Separate all non-forbidden k-mers by # X_tr = GAC#ACC#CCC#CAT

Closest w.r.t. edit distance X_ed = GAC#AA#ACCC#CAT [Bernardini et al. CPM’20]

The Hide and Mine problem

String sanitization

$$k = 3 \quad S = \{ACA, CAA, AAA, AAC, CCA\} \quad W = GACAAAAACCCAT$$

- We preserve the non-forbidden k-mers, and their order.
- We can use a special character “#” as a separator.

Plenty of possibilities!

Separate all non-forbidden k-mers by # X_tr = GAC#ACC#CCC#CAT

Closest w.r.t. edit distance X_ed = GAC#AA#ACCC#CAT [Bernardini et al. CPM’20]

Shortest solution X_min = GACCC#CAT [Bernardini et al. PKDD’19]

$k = 3 \quad S = \{ACA, CAA, AAA, AAC, CCA\} \quad W = GACAAAAACCCAT$

X=GACCC#CAT

Problem: we cannot leave the #, they indicate the former positions of forbidden k-mers

$k = 3 \quad S = \{ACA, CAA, AAA, AAC, CCA\} \quad W = GACAAAAACCCAT$

X=GACCC#CAT

Problem: we cannot leave the #, they indicate the former positions of forbidden k-mers

=> We **replace** them by letters in Σ.

$k = 3 \quad S = \{ACA, CAA, AAA, AAC, CCA\} \quad W = GACAAAAACCCAT$

X=GACCC#CAT

Problem: we cannot leave the #, they indicate the former positions of forbidden k-mers

=> We **replace** them by letters in Σ.

X=GACCC#CAT => Z=GACCCGCAT

$k = 3 \quad S = \{ACA, CAA, AAA, AAC, CCA\} \quad W = GACAAAAACCCAT$

X=GACCC#CAT

Problem: we cannot leave the #, they indicate the former positions of forbidden k-mers

=> We **replace** them by letters in Σ.

X=GACCC#CAT => Z=GACCCGCAT

We add k-mers : CCG, CGC, and GCA.

$k = 3 \quad S = \{ACA, CAA, AAA, AAC, CCA\} \quad W = GACAAAAACCCAT$

X=GACCC#CAT

Problem: we cannot leave the #, they indicate the former positions of forbidden k-mers

=> We **replace** them by letters in Σ.

X=GACCC#CAT => Z=GACCCGCAT

We add k-mers : CCG, CGC, and GCA.

Can we still do frequent pattern mining?

The Hide and Mine problem

String sanitization

$$k = 3 \quad S = \{ACA, CAA, AAA, AAC, CCA\} \quad W = GACAAAAACCCAT$$

For frequent pattern mining, we want to minimize the number of τ **-ghosts**.

The Hide and Mine problem

String sanitization

$k = 3 \quad S = \{ACA, CAA, AAA, AAC, CCA\} \quad W = GACAAAAACCCAT$

For frequent pattern mining, we want to minimize the number of τ **-ghosts**.

Definition:
Given $\tau > 0$,
a k-mer U is a τ-ghost if :

$$\text{Freq}_X(U) < \tau$$

and

$$\text{Freq}_Z(U) \geq \tau$$

$k = 3 \quad S = \{ACA, CAA, AAA, AAC, CCA\} \quad W = GACAAAAACCCAT$

For frequent pattern mining, we want to minimize the number of τ **-ghosts**.

Definition:
Given $\tau > 0$,
a k-mer U is a τ-ghost if :

$$\mathrm{Freq}_X(U) < \tau$$

and

$$\mathrm{Freq}_Z(U) \geq \tau$$

Example:

$k = 3 \quad S = \{ACA, CAA, AAA, AAC, CCA\} \quad W = GACAAAAACCCAT$

For frequent pattern mining, we want to minimize the number of τ **-ghosts**.

Definition:
Given $\tau > 0$,
a k-mer U is a τ-ghost if :

$$\mathrm{Freq}_X(U) < \tau$$

and

$$\mathrm{Freq}_Z(U) \geq \tau$$

Example:

$$\tau = 2$$

$$k = 3 \quad S = \{ACA, CAA, AAA, AAC, CCA\} \quad W = GACAAAAACCCAT$$

For frequent pattern mining, we want to minimize the number of τ **-ghosts**.

Definition:
Given $\tau > 0$,
a k-mer U is a τ-ghost if :

$$\text{Freq}_X(U) < \tau$$

and

$$\text{Freq}_Z(U) \geq \tau$$

Example:

$$\tau = 2$$

X= **GAC**#ACC#CCC#CAT

$k = 3 \quad S = \{ACA, CAA, AAA, AAC, CCA\} \quad W = GACAAAAACCCAT$

For frequent pattern mining, we want to minimize the number of τ **-ghosts**.

Definition:
Given $\tau > 0$,
a k-mer U is a τ-ghost if :

$$\text{Freq}_X(U) < \tau$$

and

$$\text{Freq}_Z(U) \geq \tau$$

Example:

$$\tau = 2$$

X= **GAC**#ACC#CCC#CAT

Z= **GACGAC**CGCCCGCAT

The Hide and Mine problem

String sanitization

$$k = 3 \quad S = \{ACA, CAA, AAA, AAC, CCA\} \quad W = GACAAAAACCCAT$$

For frequent pattern mining, we want to minimize the number of τ **-ghosts**.

Definition:

Given $\tau > 0$,

a k-mer U is a τ-ghost if :

$$\text{Freq}_X(U) < \tau$$

and

$$\text{Freq}_Z(U) \geq \tau$$

Example:

$$\tau = 2$$

X= **GAC**#ACC#CCC#CAT

Z= **GACGAC**CGCCCGCAT

=> GAC is a τ-ghost !

Hide and Mine problem

Input and Parameters

- Threshold on frequency τ
- Length of forbidden pattern k
- Set of forbidden patterns S
- String with # to replace

$$X = X_1 \# X_2 \# ... \# X_\delta$$

s.t. $\forall i \in [1, \delta], |X_i| \geq k - 1$

Goal:

Replace all #s in X so that the number of **τ-ghosts is minimum** and there is **no forbidden pattern**.

Contributions

Hide and Mine Decision: Decision variant

Can we replace the #s in X without introducing **any τ-ghosts ?**

Hide and Mine Decision: Decision variant

Can we replace the #s in X without introducing **any τ-ghosts ?**

Hide and Mine Decision is **NP-complete** by reduction from Bin Packing.

Hide and Mine is **NP-hard** by reduction from the decision variant.

Hide and Mine is **NP-hard** by reduction from the decision variant.

Hide and mine has no α-approximation for any $\alpha \geq 1$, unless $P = NP$.

- Hide and Mine Minimum Threshold :
  Find the minimum $\tau_1 \geq \tau$,
  such that there are no τ_1-ghosts.

τ_1 s.t.

- Hide and Mine Minimum Threshold :
  Find the minimum $\tau_1 \geq \tau$,
  such that there are no τ_1-ghosts.

τ_1 s.t. 

Hide and Mine Minimum Threshold is **NP-hard,**
by reduction from the decision variant.

Hide and Mine Minimum Threshold

- Hide and Mine Minimum Threshold :
  Find the minimum $\tau_1 \geq \tau$,
  such that there are no τ_1-ghosts.

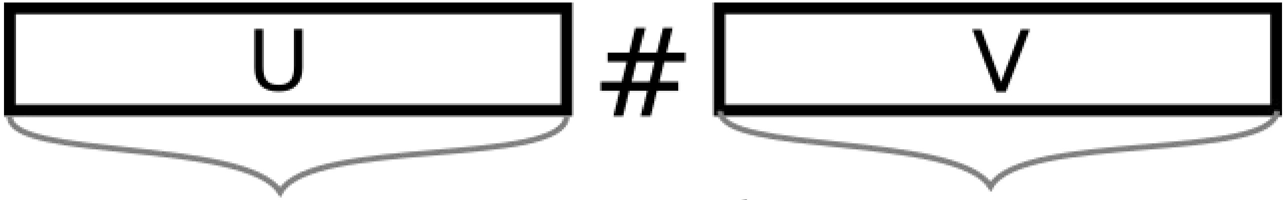
τ_1 s.t.

Hide and Mine Minimum Threshold is **NP-hard,** by reduction from the decision variant.

Hide and Mine Minimum Threshold has no α-approximation for any $\alpha \geq 1$, unless $P = NP$.

Algorithms:
Integer Linear Programming

Context of a #



$k-1$ characters on the left-side $k-1$ characters on the right-side

Definition: the context of this # is (U,V).

Property: The context is enough to know what k-mers will be added by a replacement. If we replace the #by $j \in \Sigma$ we add all k-mers in UjV .

1) Regroup the #s by their contexts: γ number of different contexts

1) Regroup the #s by their contexts: γ number of different contexts

2) Number the contexts and rename the #s:

$\#_i$ has context $C_i = (U_i, V_i)$. δ_i number of $\#_i$.

Idea

1) Regroup the #s by their contexts: γ number of different contexts

2) Number the contexts and rename the #s:

$\#_i$ has context $C_i = (U_i, V_i)$. δ_i number of $\#_i$.

3) Determine the set of critical k-mers: $\{N_\ell\}_{\ell \in [\lambda]}$

k-mers that may become ghost because of the replacement

Idea

1) Regroup the #s by their contexts: γ number of different contexts
2) Number the contexts and rename the #s:
$\#_i$ has context $C_i = (U_i, V_i)$. δ_i number of $\#_i$.
3) Determine the set of critical k-mers: $\{N_\ell\}_{\ell \in [\lambda]}$
k-mers that may become ghost because of the replacement
4) We want to find all $x_{i,j}$: this represents the number of time we replaced a $\#_i$ by $j \in \Sigma$.

1) Regroup the #s by their contexts: γ number of different contexts
2) Number the contexts and rename the #s:
$\#_i$ has context $C_i = (U_i, V_i)$. δ_i number of $\#_i$.
3) Determine the set of critical k-mers: $\{N_\ell\}_{\ell \in [\lambda]}$
k-mers that may become ghost because of the replacement
4) We want to find all $x_{i,j}$: this represents the number of time we replaced a $\#_i$ by $j \in \Sigma$.
5) We compute for each replacement what critical k-mers it would add.

Hide and Mine Decision variant

$\boldsymbol{\gamma}$ number of distinct contexts present in X;
$\boldsymbol{\delta_i}$ number of occurrences of letter $\#_i$ in X, for $i \in [\gamma]$;
$\boldsymbol{\lambda}$ number of distinct critical length-k strings;
$\boldsymbol{\alpha^i_{\ell,j}}$ additional number of occurrences of N_ℓ introduced by replacing a $\#_i$ with a letter $j \in \Sigma$, for $\ell \in [\lambda]$;
$\boldsymbol{e_\ell}$ difference $(\tau - 1) - \mathrm{Freq}_X(N_\ell)$, for $\ell \in [\lambda]$.

Find a solution $x \in \mathbb{Z}^{\gamma \times |\Sigma|}$

$$\begin{cases} x_{i,j} \geq 0 & \forall (i,j) \in [\gamma] \times \Sigma \\ x_{i,j} = 0 & \forall (i,j) \in \mathcal{F} \\ \sum_{i \in [\gamma], j \in \Sigma} \alpha^i_{\ell,j} x_{i,j} \leq e_\ell & \forall \ell \in [\lambda] \\ \sum_{j \in \Sigma} x_{i,j} = \delta_i & \forall i \in [\gamma] \end{cases}$$

(set of forbidden replacements: replacements that create a forbidden pattern)

$\boldsymbol{\gamma}$	number of distinct contexts present in X;
$\boldsymbol{\delta_i}$	number of occurrences of letter $\#_i$ in X, for $i \in [\gamma]$;
$\boldsymbol{\lambda}$	number of distinct critical length-k strings;
$\boldsymbol{\alpha^i_{\ell,j}}$	additional number of occurrences of N_ℓ introduced by replacing a $\#_i$ with a letter $j \in \Sigma$, for $\ell \in [\lambda]$;
$\boldsymbol{e_\ell}$	difference $(\tau - 1) - \mathrm{Freq}_X(N_\ell)$, for $\ell \in [\lambda]$.

z_ℓ 0 if N_ℓ does not become a ghost
1 if N_ℓ does become a ghost

Goal: Find $x \in \mathbb{Z}^{\gamma \times |\Sigma|}$

That minimize

$$\sum_{\ell=1}^{\lambda} z_\ell$$

$$\begin{cases} x_{i,j} \geq 0 & \forall (i,j) \in [\gamma] \times \Sigma \\ x_{i,j} = 0 & \forall (i,j) \in \mathcal{F} \\ z_\ell \geq 0 & \forall \ell \in [\lambda] \\ \sum_{i \in [\gamma], j \in \Sigma} \alpha^i_{\ell,j} x_{i,j} - k\delta z_\ell \leq e_\ell & \forall \ell \in [\lambda] \\ \sum_{j \in \Sigma} x_{i,j} = \delta_i & \forall i \in [\gamma] \end{cases}$$

ILP: polynomial algorithms

Integer Linear Programming runs in linear time in the number of constraints when the number of variables is a constant.

ILP: polynomial algorithms

Integer Linear Programming runs in linear time in the number of constraints when the number of variables is a constant.

Hide and Mine decision variant has a **polynomial time algorithm** if either:

a) The *size of the alphabet* and the *number of contexts* of the #s are constants.

b) The *size of the alphabet* and *k are constants.*

c) The *number of critical k-mers* and *k* are constants.

Hide and Mine has a **polynomial time algorithm** if either:

1) The following are constants:
   a) the *size of the alphabet,*
   b) the *number of contexts* of the #s,
   c) the number of critical k-mers.

2) The following are constants:
   a) k,
   b) the number of critical k-mers.

Algorithms - Heuristic

To be publish soon in the journal version

1) Compute statistics on the number of k-mer without # in X.

2) For the i-th # in the string :

- Let Z_i be the string with all previous # replaced.
- For $j \in \Sigma$, consider the string U j V (U,V the context of #i).
  - If it contains a forbidden pattern, $S_j = \varnothing$ and $S_j^{<\tau}$ is undefined.
  - If not, S_j is the set of all k-mers in U j V and $S_j^{<\tau}$ the set of all k-mers Y in S_j s.t. $\mathrm{Freq}_{Z_i}(Y) < \tau$.

3) Choose the j (if there is one) that minimizes : $\sum_{Y \in S_j^{<\tau}} [\tau - \mathrm{Freq}_{Z_i}(Y)]^{-1}$

Experiments

To be publish soon in the journal version

- Implemented in C++ available on GitHub (soon).
- Gurobi solver used to solve the ILP.
- 5 datasets :
  - OLD: Oldenburg
  - TRU: Trucks
  - MSN: MSNBC
  - DNA: Escherichia coli genome
  - SYN: Uniformly random strings
- Comparison of ILP and Heuristic with TPM: part III of [Bernardini et al. PKDD'19]

TABLE I: (a) Dataset characteristics. (b) Default values used.

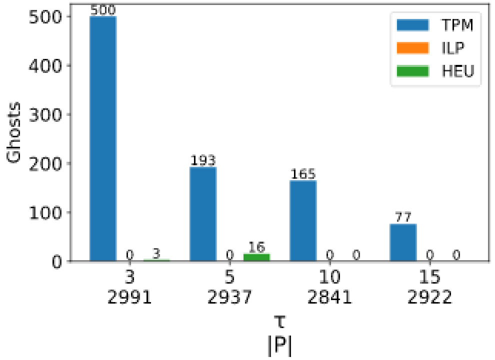
Dataset	length n	alphabet size $\|\Sigma\|$	no sens patterns $\|\mathcal{S}\|$	no sens positions $\|\mathcal{P}\|$	pattern length k	threshold τ
OLD	85,563	100	$[60, 320]$	$[926, 5673]$	$[3, 6]$	$[3, 15]$
TRU	5,763	100	$[10, 70]$	$[363, 3813]$	$[2, 5]$	$[5, 30]$
MSN	4,698,764	17	$[60, 480]$	$[16792, 133590]$	$[3, 8]$	$[100, 300]$
DNA	4,641,652	4	$[30, 60]$	$[715, 1617]$	$[9, 15]$	$[5, 30]$
SYN	20,000,000	10	$[10, 1000]$	$[1967, 2001226]$	$[3, 6]$	$[5, 20]$

(a)

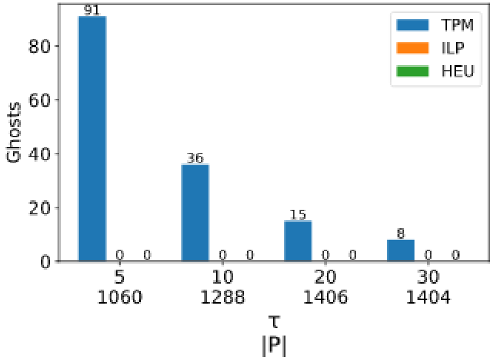
Dataset	no sens patterns $\|\mathcal{S}\|$	pattern length k	threshold τ
OLD	120	6	10
TRU	30	3	20
MSN	240	8	200
DNA	50	11	20
SYN	100	5	10

(b)

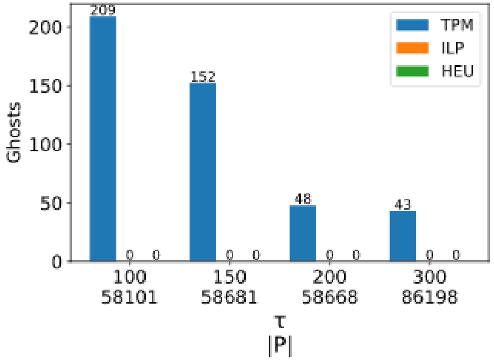
Varying τ



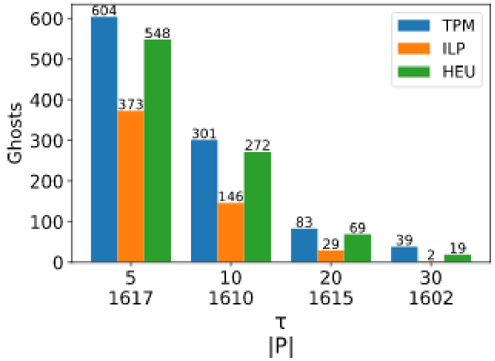
(a) OLD



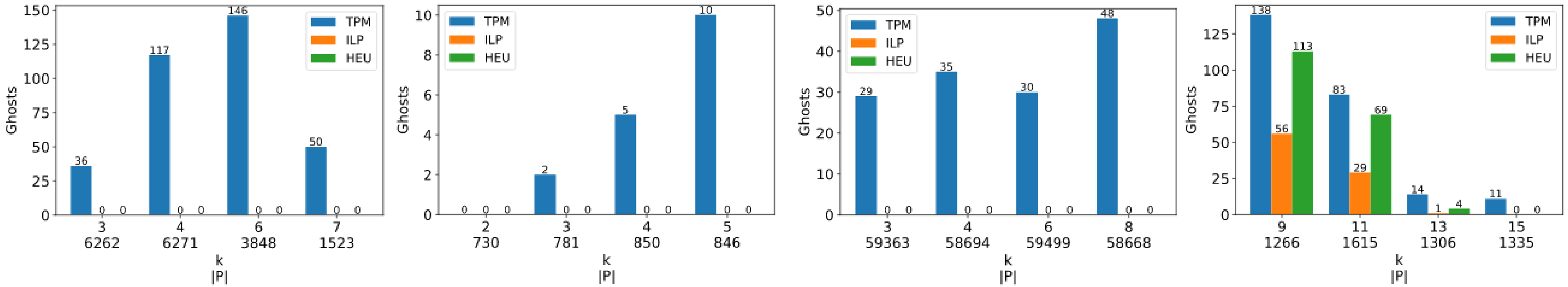
(b) TRU



(c) MSN

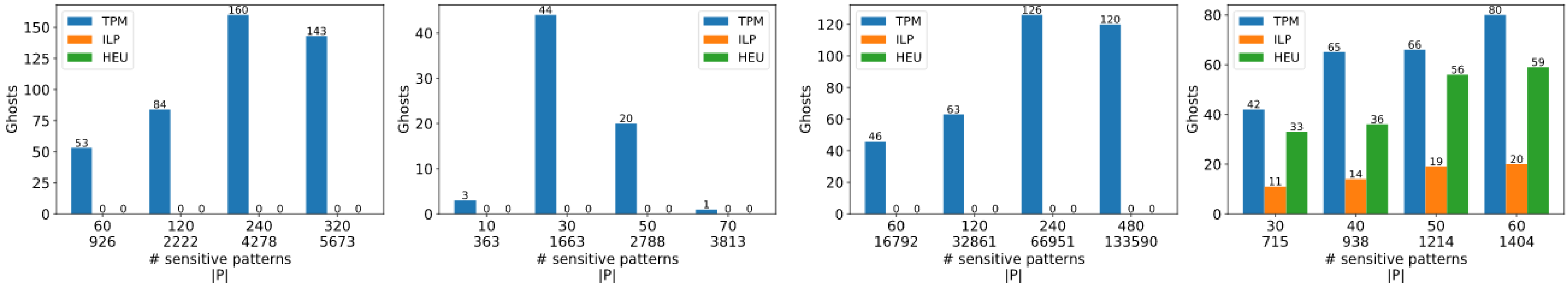


(d) DNA



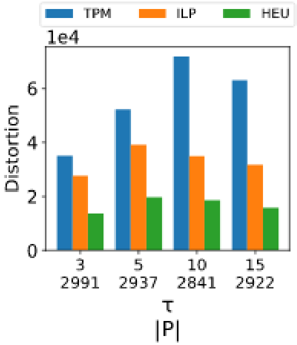
(a) OLD (b) TRU (c) MSN (d) DNA

Varying |S|

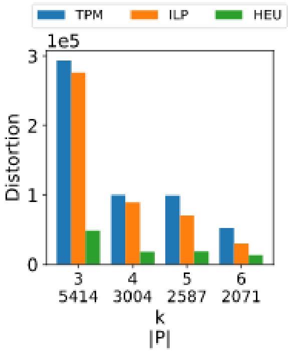


(a) OLD (b) TRU (c) MSN (d) DNA

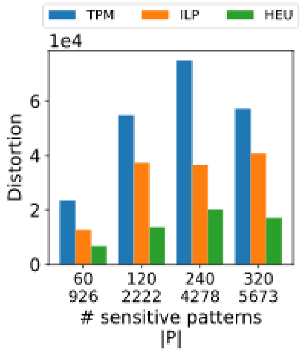
$$\sum_{U} (\mathrm{Freq}_X(U) - \mathrm{Freq}_Z(U))^2$$ where $U \in \Sigma^k$ is a non-forbidden pattern.



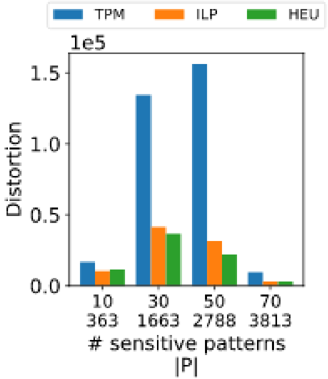
(a) OLD



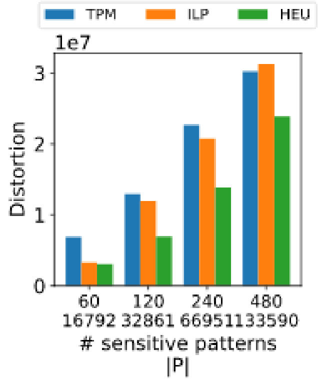
(b) OLD



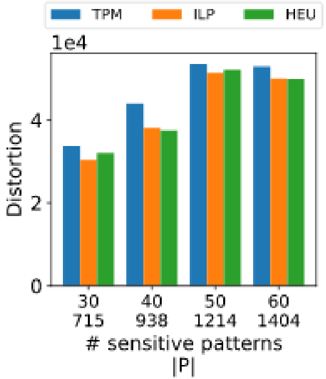
(c) OLD



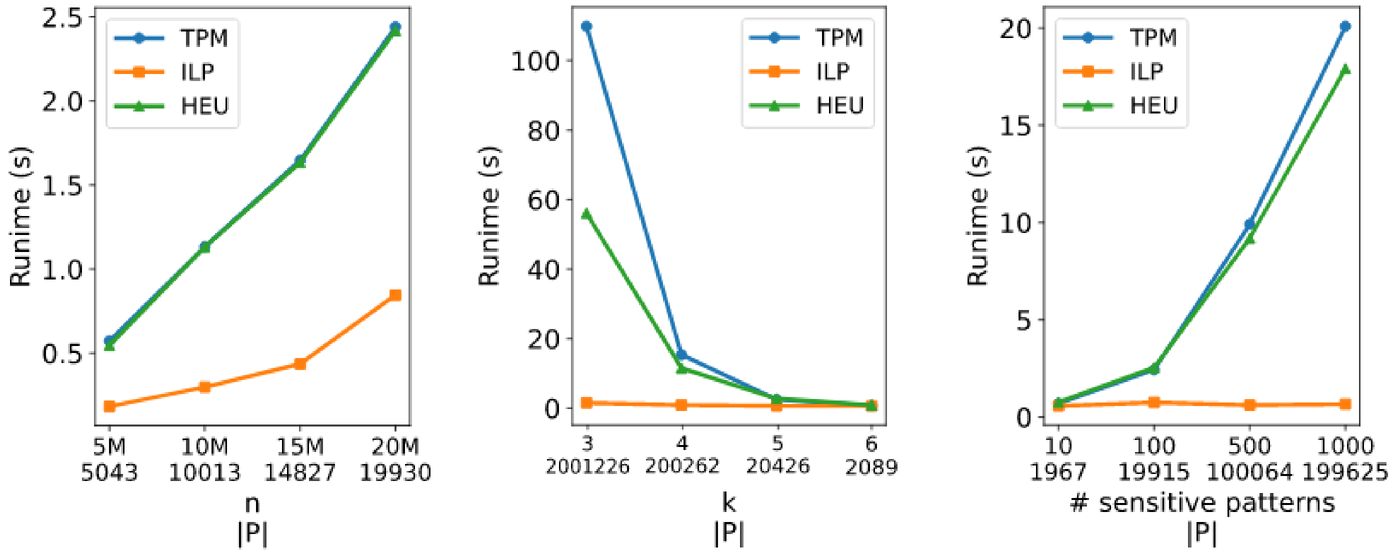
(d) TRU



(e) MSN



(f) DNA



(a) Substr. of SYN (b) SYN (c) SYN

CPM Advertisement !

Take home message

Take home message

- **Hide and Mine** and its variants are all **NP-hard** and **hard to approximate**.

Take home message

- **Hide and Mine** and its variants are all **NP-hard** and **hard to approximate**.

- **Hide and Mine** and its **decision variant** can be solved via **ILP**, which works in **polynomial time under realistic assumptions** on the input parameters.

Take home message

- **Hide and Mine** and its variants are all **NP-hard** and **hard to approximate**.

- **Hide and Mine** and its **decision variant** can be solved via **ILP**, which works in **polynomial time under realistic assumptions** on the input parameters.

- **Experiments** on both synthetic and real world datasets that **confirm the theoretical findings**.

Take home message

- **Hide and Mine** and its variants are all **NP-hard** and **hard to approximate**.

- **Hide and Mine** and its **decision variant** can be solved via **ILP**, which works in **polynomial time under realistic assumptions** on the input parameters.

- **Experiments** on both synthetic and real world datasets that **confirm the theoretical findings**.

Thank you for your attention !