# Permutation-based Sequential Pattern Hiding

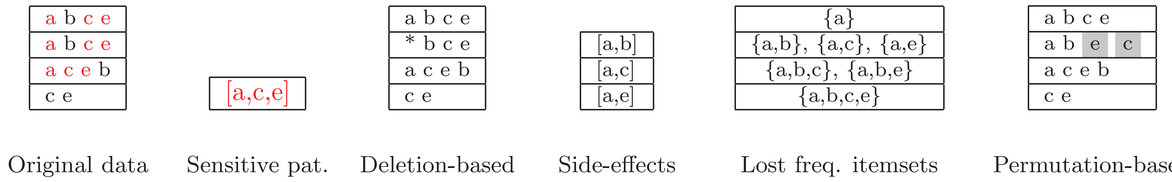
Robert Gwadera, Aris Gkoulalas-Divanis, and Grigorios Loukides

robert.gwadera@epfl.ch, arisdiva@ie.ibm.com, g.loukides@cs.cf.ac.uk

## Motivation and Contribution

**Motivation**

- Sequence data are shared to enable mining (e.g., in marketing and healthcare).
- *Sensitive* patterns, which lead to intrusive inferences about individuals or leak confidential information about organizations, may be exposed.
- Existing methods use symbol deletion, which reduces symbol support.
- High utility loss in sequence mining and tasks based on itemset properties.




**Contribution**: The first, permutation-based approach to hiding sensitive sequential patterns.

- Study of the problem of avoiding side-effects
- *PDPG*, for generating permutations that avoid side-effects and have minimum distortion
- *PH*, for sanitizing transactions

## Avoiding side-effects

**Conditions for identifiying candidates for lost and ghost, based on**

- the shared symbols between a sensitive and a nonsensitive pattern.

| Swapping symbols in | $s$ | $s'$ | $s''$ | $t$ | $\pi(s)$ | $t$ after $\pi(s)$ |
|---|---|---|---|---|---|---|
| $s''$ | $[a,b,c]$ | $[a,d,c]$ | $[a,c]$ | $[a,b,d,c]$ | $[c,b,a]$ | $[c,b,d,a]$ |
| $s \setminus s''$, $s''$ | $[a,b,c]$ | $[d,b,c]$ | $[b,c]$ | $[a,d,b,c]$ | $[c,b,a]$ | $[c,d,b,a]$ |

- the support of the nonsensitive pattern $s'$:
  - $s'$ becomes *lost* if it satisfies symbol-based conditions and $sup_{\mathcal{S}}(s') < minSup + \Delta_{supp}(s|\pi(s))$
  - $s'$ becomes *ghost* if it satisfies symbol-based conditions and $sup_{\mathcal{S}}(s') \geq minSup - \Delta_{supp}(s|\pi(s))$

**NP-completeness**

- Finding a permutation that prevents lost and avoids ghost patterns (*pattern-constrained*) is NP-complete (proof by reduction to SAT)

**Identifiying a satisfiable prefix of a permutation w.r.t. a candidate pattern**

- Conditions based on distortion (Cayley distance)
- Swapping the remaining symbols of $s$ must lead to preserving a candidate $d^-$ for lost
- At least one swap of symbols of $s$ must lead to preserving a $d^+$ for ghost

| $t$ | $s$ | $d^-$ | $d^+$ | $\pi(s\|2)$ | $t'$ | $t''$ |
|---|---|---|---|---|---|---|
| $[0,1,2,3,4]$ | $[1,2,3]$ | $[0,1,2]$ | $[2,1,4]$ | $[2,1]$ | $[0,2,1,3,4]$ | $[0,3,1,2,4]$ |

## Permutation generation

***PDPG***

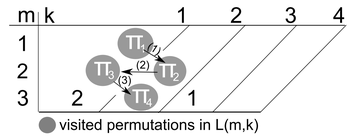
Aims at generating a *pattern-constrained* $\pi(s|m)$ with min. Cayley distance (max. # cycles $k$).

- Random search over all possible solutions
- Extends $\pi(1|m=1)=[(0)]$ to $\pi(s|m=|s|)$, iteratively
- Randomly selects the (satisfiable) permutations to extend, from both types of permutations w.r.t. their cycle structure
- The selection is made from the most likely type to produce a satisfiable $p(s|m+1)$

**Example**

| $t$ | $s$ | $d^-$ | $d^+$ | $t'$ |
|---|---|---|---|---|
| $[0,1,2,3,4]$ | $[1,2,3]$ | $[2,3,4]$ | $[3,2,4]$ | $[0,2,1,3,4]$ |

| $m$ | $k$ | $\pi(s\|m)$ | $d^-=[2,3,4]$ | | $d^+=[3,2,4]$ | |
|---|---|---|---|---|---|---|
| | | | Sat. | $\mathcal{C}(d^-(s),\mathcal{I})$ | Sat. | $\mathcal{C}(d^{+)}(s),\mathcal{I})$ |
| 1 | 1 | $\pi_1=[(0)]$ | true | 0 | true | 1 |
| 2 | 2 | $\pi_2=[(0),(1)]$ | true | 0 | true | 1 |
| 2 | 1 | $\pi_3=[(0,1)]$ | true | 0 | true | 1 |
| 3 | 2 | $\pi_4=[(0,1),2]$ | true | 0 | true | 1 |



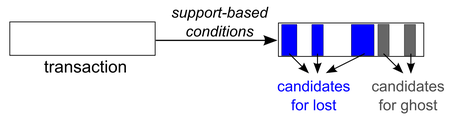

***RR-PDPG***

Aims at generating a permutation that satisfies *most* candidates, when all candidates cannot be satisfied

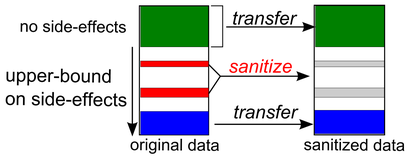
- Halves the candidates and applies *PDPG*, until the generated permutation satisfies them
- Short candidates for ghost are the easiest to satisfy
- Long candidates for lost are the hardest

## *PH* algorithm

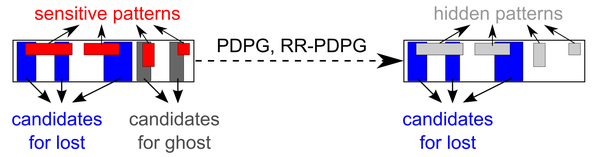
**Identification of candidate patterns**




**Selection of transactions to sanitize**




**Sanitization of each selected transaction**




## Backgr. knowledge attacks

Attackers can discover a sensitive pattern $s$ among nonsensitive patterns with the same support and/or do not consider certain permutations of $s$.

To prevent such attacks:

- Use all available permutations
- Use at least $c$ different permutations, all having support at least $\lambda$

## Experimental evaluation