SMIM framework to generalize high-utility itemset mining

Siddharth Dawar, Vikram Goyal and *Debajyoti Bera(dbera@iiitd.ac.in)
Department of Computer Science, Indraprastha Institute of Information Technology (IIIT-Delhi)

SMIM: Generalization of High-Utility Itemset Mining (HUIM)

Table 1: Example of a transaction database and utilities of two itemsets $\{A, C\}, \{G, H\}$ for two different utility functions $u()$ (as used in HUIM) and $ucov()$ (which is subadditive and monotone). $ucov()$ also takes a relationship graph as input as shown in Fig. 1.

TID	Transaction	$w(A)$	$w(C)$	$w(G)$	$w(H)$	$u(AC)$	$u(GH)$	$ucov(AC)$	$ucov(GH)$
T_1	$(A:5)(C:10)(D:2)$	5	10	0	0	15	0	35	0
T_2	$(A:10)(C:6)(E:6)(G:5)$	10	6	5	0	16	0	40	0
T_3	$(A:10)(B:4)(D:12)(E:6)(F:5)$	10	0	0	0	0	0	0	0
T_4	$(A:5)(B:2)(C:3)(D:2)(G:1)(H:41)$	5	3	1	41	8	42	20	83
T_5	$(B:8)(C:13)(D:6)(E:3)$	0	13	0	0	0	0	0	0
T_6	$(F:1)(G:2)$	0	0	2	0	0	0	0	0
T_7	$(F:4)(G:3)$	0	0	3	0	0	0	0	0

$u(AC) = 39$ and $u(GH) = 42$, so $u(GH) > u(AC)$; however, $ucov(AC) = 95$ and $ucov(GH) = 83$, so $ucov(GH) < ucov(AC)$. High-utility itemsets may change when the underlying utility function is changed.

High-utility itemset mining (HUIM)

Given a threshold θ, identify itemsets X with utility $u(X) \geq \theta$, where the utility of an itemset is defined as

$$u(X) = \sum_{T \in \mathcal{D}, X \subseteq T} u(X, T),$$

and the utility of X in a transaction is defined as

$$u(X, T) = \sum_{y \in X} w(y, T),$$

where $w(y, T)$ denotes the weight/individual-utility of the item y in T.

SMIM: HUIM using a subadditive & monotone utility function

Given any arbitrary subadditive and monotone (SM) utility function $u'()$ over weighted itemsets, and threshold θ, identify itemsets X with utility $u'(X) \geq \theta$, where the utility of an itemset is defined as

$$u(X) = \sum_{T \in \mathcal{D}, X \subseteq T} u(X, T),$$

and the utility of X in a transaction, denoted $u'(X, T)$, is defined in terms of the items in X along with their weights/individual utilities.

SM utility functions

- A function $f : \mathcal{U} \to \mathbb{R}+$ is defined as subadditive if $\forall X, Y \subseteq \mathcal{U}$, $f(X \cup Y) \leq f(X) + f(Y)$; $SUM(U) = \sum_{y \in U} f(y)$ is subadditive, but $PROD(U) = \prod_{y \in U} f(y)$ is not.
- A function $f : \mathcal{U} \to \mathbb{R}+$ is defined as monotone if $\forall X \subseteq Y \subseteq \mathcal{U}$, $f(X) \leq f(Y)$; $SUM(U) = \sum_{y \in U} f(y)$ is monotone, but $MIN(U) = \min_{y \in U} f(y)$ is not.

Examples of SM functions

Let $w(y)$ denote the weight of an item y in a set X.

- $SUM(X) = \Sigma_{y \in X}\, w(y)$ – this is used in HUIM
- Discounted profit $DP(X)$ = total profit X under the scheme "Buy 1 pencil, get 1 eraser free"
- $Co(X)$ = number of nodes that are either in X or neighbor of some node in X
- (order X in increasing order of weights) $ucov(X, T) = w(y_1) \times Co(X) + \Sigma_{j=2}^{k}(w(y_j) - w(y_{j-1})) \times Co(\{y_j \cdots y_k\})$

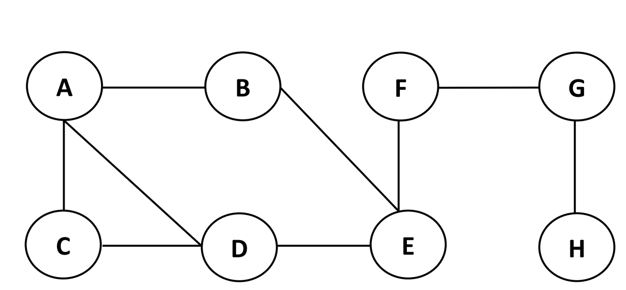

Figure 1: External graph used to compute $Co()$ and $ucov()$

Acknowledgements

This work was supported in part by Infosys Centre for Artificial Intelligence, IIIT-Delhi, and Visvesvaraya Ph.D. scheme for Electronics and IT.

Contributions

1. SMIM framework for mining high-utility itemsets using any subadditive and monotone utility function defined on weighted itemsets.
2. SM utility functions have mathematically helpful properties, and can model traditional HUIM, HUIM in the presence of multi-item discounts, and several variations of HUIM as considered by [Yao,Hamilton,Geng 2006].
3. Going beyond the HUIM framework, SMIM can incorporate extraneous interactions among the items in an itemset in their utility, e.g., to identify influential users in a Twitter dataset.
4. We prove a few interesting utility functions to be subadditive and monotone, e.g., $DP()$, $Co()$, and $ucov()$.
5. A novel inverted-list data structure called SMI-List and an algorithm called SM-Miner to mine high-utility itemsets for SM functions.
6. We also show how to adapt the existing HUIM algorithms for SMIM, but empirically show that SM-Miner delivers better performance.

Adapting HUIM algorithms

- Transaction merging should be disabled when adapting projection-based algorithms to SMIM.
- Ex.: $ucov(\{F, G\}, T_6) = 7$ and $ucov(\{F, G\}, T_7) = 15$. If we merge them to a single transaction $M = \{(F:5), (G:5)\}$, then we get $ucov(\{F, G\}, M) = 20$.
- Tree-based algorithms for HUIM may be adapted towards SMIM if unpromising items are retained during local tree creation since removing them may yield incorrect estimates of some utilities.
- Ex.: Let $T = \{(A_1 : q_1), \dots (A_n : q_n)\}$ be some transaction, itemset $X = \{A_1\}$ and itemset $Y = \{A_2, \dots, A_n\}$. Suppose that A_1 is unpromising. For HUIM, $u(X, T) + u(Y, T) = u(X \cup Y, T)$; therefore, $u(X \cup Y, T) - u(X, T)$ correctly estimates $u(Y, T)$. However, this may not hold for other utility functions where $f(X, T) + f(Y, T) > f(X \cup Y, T)$.

SM-Miner

A single efficient list-based algorithm for mining high SM-utility itemsets given blackbox access to any SM-utility function.

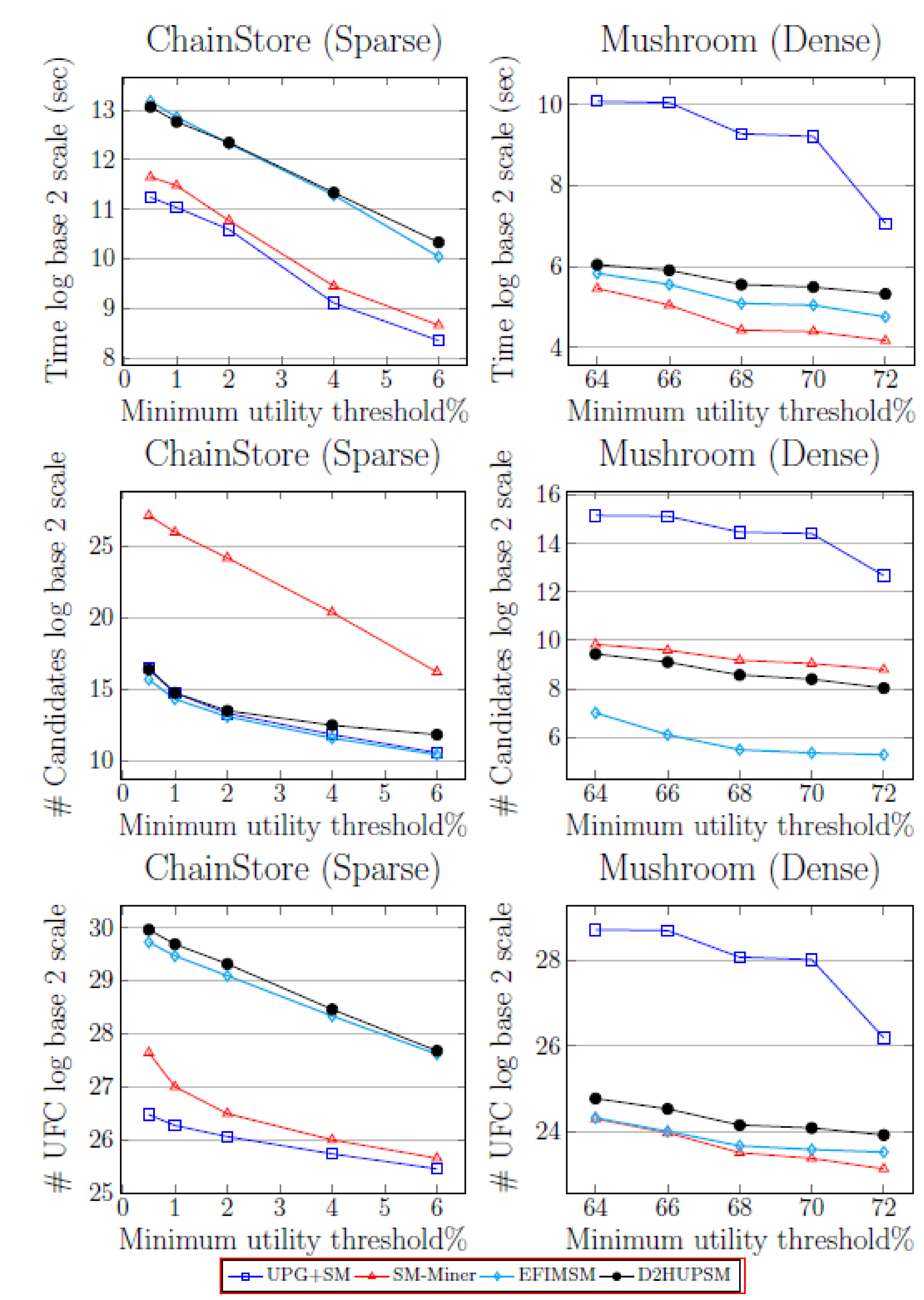

Figure 2: Performance evaluation of SM-Miner (our), and SMIM implementations of EFIM, D2HUP, UP-Growth+ (using $ucov$).

- For HUIM (using SUM utility function), D2HUP and EFIMSM were observed to perform the best on sparse and dense datasets, respectively.
- Tree-based algorithm UPG+-SM and list-based algorithm SM-Miner performs better than projection-based algorithms on sparse datasets. SM-Miner competes with EFIMSM on dense datasets.
- The total execution time of the algorithms appear to be more correlated with the number of utility function calls than the number of candidates generated, unlike for HUIM.

Full version at https://www.iiitd.edu.in/~dbera/docs/2021-smim.pdf