Mining closed high utility itemsets based on propositional satisfiability

Amel Hidouri, Said Jabbour, Badran Raddaoui, Boutheina Ben Ben Yaghlane

Version of Record: https://www.sciencedirect.com/science/article/pii/S0169023X21000549
Manuscript_76c254959f9338249cf33955bae4abb6

Mining Closed High Utility Itemsets based on Propositional Satisfiability

Amel Hidouri[1,2], Said Jabbour[2], Badran Raddaoui[3], and Boutheina Ben Yaghlane[1]

[1] LARODEC, University of Tunis, Tunis, Tunisia
boutheina.byaghlane@gmail.com
[2] CRIL - CNRS UMR 8188, University of Artois, France
{hidouri,jabbour}@cril.fr
[3] SAMOVAR, Télécom SudParis, Institut Polytechnique de Paris, France
badran.raddaoui@telecom-sudparis.eu

Abstract. A high utility itemset mining problem is the question of recognizing a set of items that have utility values greater than a given user utility threshold. This generalization of the classical problem of frequent itemset mining is a useful and well-known task in data analysis and data mining, since it is used in a wide range of real applications. In this paper, we first propose to use symbolic Artificial Intelligence for computing the set of all closed high utility itemsets from transaction databases. Our approach is based on reduction to enumeration problems of propositional satisfiability. Then, we enhance the efficiency of our SAT-based approach using the weighted clique cover problem. After that, in order to improve scalability, a decomposition technique is applied to derive smaller and independent sub-problems in order to capture all the closed high utility itemsets. Clearly, our SAT-based encoding can be constantly enhanced by integrating the last improvements in powerful SAT solvers and models enumeration algorithms. Finally, through empirical evaluations on different real-world datasets, we demonstrate that the proposed approach is very competitive with state-of-the-art specialized algorithms for high utility itemsets mining, while being sufficiently flexible to take into account additional constraints to finding closed high utility itemsets.

Keywords: Data Mining · High Utility · Symbolic Artificial Intelligence· Propositional Satisfiabilty

1 Introduction

Data mining is a multi-disciplinary field that employs machine learning, statics and symbolic Artificial Intelligence in order to find novel valid relationships

This work is an extended version of our paper published in the International Conference on Data Warehousing and Knowledge Discovery [37].

among data. A pivotal task in knowledge discovery process concerns pattern extraction w.r.t. various proprieties such as novelty, usefulness and understandability. The problem of selecting a set of itemsets from the original data has recently gained a lot of attention in the two last decades. Mining High Utility itemsets (HUIM, for short) as one of the major keystone in the discovery of patterns generalizes the problem of frequent itemsets (FIM), since traditional frequent patterns cannot fulfill the requirement of finding the most valuable information that contribute to the major part of the total profits in the retail databases. HUIM considers item quantities and weights and refers to recognize the set of items that appear together in a given transaction database and having a high importance to the user, measured by a *utility function*. The utility means the importance/profitability of items to users. Indeed, the utility of items in a transaction depends on the importance of different items, which is called *external* utility, and the importance of the items in the transaction, called *internal* utility. Then, the utility of an itemset is defined as the multiplication of the external and the internal utilities. An itemset is called a *high utility itemset* if its utility is no less than a user specified threshold value; otherwise, the itemset is called a *low utility itemset*. Mining high utility itemsets from transaction databases is an important task and it has a wide range of applications, including website click stream analysis, business promotion in chain hypermarkets, online e-commerce management, mobile commerce environment planning, and biomedical applications [1–3].

Generally speaking, existing approaches, usually called *specialized* approaches, (see related work section) are designed to find particular kinds of itemsets from transaction databases with utilities. In that sense, new additional constraints (i.e., user preference) cannot be easily integrated in the original algorithm, and they require a re-implementation of the whole application. In recent years, several constraint-based languages for modeling and solving data mining problems have been designed where the data mining task is modeled as a constraint network or a propositional formula whose models correspond to the patterns of interest. These methods, usually coined as *declarative* approaches, have then found applications in diverse data mining tasks ranging from frequent itemset mining [9] and its concise representations [35, 36], sequences [27] and association rules [23, 24]. Clearly, this tight connection between constraint-based languages and pattern discovery allows data mining problems to benefit from several powerful symbolic Artificial Intelligence solving techniques. More interestingly, propositional satisfiability based approaches have gained a considerable audience with the advent of a new generation of solvers able to handle large instances encoding data mining problems [6, 7, 9].

A bottleneck of the most declarative methods is the encoding size which affect the algorithm efficiency. Thus, the performance of such approaches decreases when the database size grows or the threshold takes low values. To improve the efficiency of declarative mining algorithms, a decomposition technique could be applied in order to divide the original mining problem to smaller and independent sub-problems.

In this paper, we introduce SATCHUIM, a new algorithm that makes an original use of symbolic Artificial Intelligence technique, i.e., propositional satisfiability, for efficiently enumerating all closed high utility itemsets embedded in a transaction database. Technically, the main contribution consists in developing a method based on propositional satisfiablity to model and solve the task of mining a concise and complete representation of HUIs in terms of constraints. Furthermore, we enhance the efficiency of our proposed approach using the weighted clique cover problem. After that, to cope with the scalability issue, which is one of the most important challenge of declarative frameworks, we use a decomposition technique that allows to derive smaller and independent enumeration sub-problems.

The paper is organized as follows. The next section presents a background on which our work relies. Then, we present the state-of-the-art of high utility itemset mining problem in Section 3. Section 4 is devoted to our novel SAT-based framework as well as its decomposition-based version for discovering closed itemsets with highest utilities. Section 5 discusses the results of our experimental evaluation on different real-world datasets to show the efficiency and usefulness of our approach. Finally, we conclude our work in Section 6 with some further perspectives.

2 Background

In this section, we present the relevant preliminaries and formally define the utility mining and the propositional satisfiability problems. Notice that all the notations used in the rest of paper are summarized in Table 1.

2.1 High Utility Itemset Mining Problem

Let Ω denote a universe of distinct items (or symbols) that occur in the database. A transaction database $D = \{T_1, T_2, \ldots, T_m\}$ is a set of m transactions or records such that each transaction T_i is a set of items, i.e., $T_i \subseteq \Omega$, and T_i has a unique identifier i called its transaction identifier (TID, for short). Each item $a \in \Omega$ is associated with a positive number $w_{ext}(a)$, called its *external utility* (e.g., unit profit). For each transaction T_i, a positive number $w_{int}(a, T_i)$ is called the *internal utility* of the item $a \in T_i$ (e.g., purchase quantity).

Definition 1 (Support of an itemset). *Let D be a transaction database. We define the support of an itemset X as the number of transactions in D that contain X, i.e., $supp(X) = |T_i \ / \ X \subseteq T_i \ s.t. \ (i, T_i) \in D|$.*

Definition 2 (Utility of an item/itemset in a transaction). *Given a transaction database D, the utility of an item a in a transaction $(i, T_i) \in D$, denoted by $u(a, T_i)$, is defined as $u(a, T_i) = w_{ext}(a) \times w_{int}(a, T_i)$. Then, the utility of an itemset $X \subseteq \Omega$ in a transaction $(i, T_i) \in D$, denoted by $u(X, T_i)$, is defined as:*

$$u(X, T_i) = \sum_{a \in X} u(a, T_i) \tag{1}$$

Notation	Meaning
Ω	A set of n items $\{a_1, \ldots, a_n\}$ s.t. each item a_j has a profit value p_a
D	Transaction database, $D = \{T_1, T_2, \ldots, T_m\}$
TID	An identifier of a transaction $(i, T_i) \in D$
X	A k-itemset containing k distinct items $\{a_1, a_2, \ldots, a_k\}$
$w_{int}(a, T_i)$	The purchase quantity of an item a in a transaction T_i
$w_{ext}(a)$	The unit profit of an item a
$u(a, T_j)$	The utility of an item a in a transaction T_i
$u(X, T_i)$	The utility of an itemset X in a transaction T_i
$u(X)$	The utility of an itemset X in the whole database
$TU(T_i)$	The sum of the utilities of items in a transaction T_i
θ	A predefined minimum high utility threshold
$TWU(X, D)$	The transaction-weighted utility of an itemset X in the database D
HUI	A High Utility itemset
CHUI	A Closed High Utility Itemset
Prop	A set of propositional variables
Form	A set of propositional formulas
Δ	A Boolean interpretation

Table 1: Summary of notations

Example 1. Let us consider a transaction database D containing four transactions, given in Table 2, which will serve as a running example throughout the paper. Each line in Table 2 represents a transaction, where each letter represents an item that is associated to a purchase quantity (i.e., internal utility). For instance, the transaction T_2 contains items a, c, and e with an internal utility of 2, 6 and 2, respectively. In addition, Table 3 indicates the external utility of each of these items. In fact, external utility of a, c, and e are respectively 4, 1 and 3. Then, the utility of the item a in T_2 is $u(a, T_2) = 4 \times 2 = 8$. Finally, the utility of the itemset $\{a, c\}$ in T_2 is $u(\{ac\}, T_2) = u(a, T_2) + u(c, T_2) = 4 \times 2 + 1 \times 6 = 14$.

TID	Items				
T_1	$(a, 1)$		$(c, 1)$	$(d, 1)$	
T_2	$(a, 2)$		$(c, 6)$		$(e, 2)$
T_3	$(a, 1)$	$(b, 2)$	$(c, 1)$	$(d, 6)$	$(e, 1)$
T_4		$(b, 4)$	$(c, 3)$	$(d, 3)$	$(e, 1)$

Table 2: Sample Transaction Database

Item	Unit profit
a	4
b	2
c	1
d	2
e	3

Table 3: External Utility

Now, the utility of an itemset X in a transaction database D is defined as the sum of the itemset utilities in all the transactions of D where X appears. More formally:

Definition 3 (Utility of an itemset in a database). *Let D be a transaction database. The utility of an itemset X in D, denoted by $u(X)$, is defined as:*

$$u(X) = \sum_{(i,T_i) \in D \mid X \subseteq T_i} u(X, T_i) \tag{2}$$

Example 2. Let us consider again the transaction database given in Example 1. Then, the utility of the itemset $\{a, c\}$ is $u(\{a, c\}) = u(\{a, c\}, T_1) + u(\{a, c\}, T_2) + u(\{a, c\}, T_3) = 5 + 14 + 5 = 24$.

Definition 4 (Closed itemset). *Let D be a transaction database and X an itemset in D. X is called a closed itemset if there exists no itemset X' such that $X \subset X'$, and $\forall (i, T_i) \in D, X \in T_i \rightarrow X' \in T_i$.*

Problem Definition.
Given a transaction database D and a user-specified minimum utility threshold θ, the goal of mining (closed) high utility itemsets problem is to find the set of all (closed) itemsets in D with a utility no less than θ, i.e.,

$$HUI = \{X : u(X) \mid X \subseteq \Omega, u(X) \geq \theta\} \tag{3}$$

That is, computing closed high utility itemsets is equivalent to finding the itemsets that cover the largest part of the database utility.

Example 3. Let us consider again the transaction database given in Example 1. Given a minimum utility threshold $\theta = 20$, then the high-utility itemsets with their utility in the database are $\{a, c, e\} : 28, \{c, d, e\} : 28, \{b, c, d, e\} : 40$. In this example, the itemsets $\{a, d\}$, $\{c, d\}$, $\{c, e\}$, $\{a, c, d\}$, $\{a, c, e\}$, $\{b, c, d, e\}$ and $\{a, b, c, d, e\}$ are closed HUIs, with utility values 23, 25, 22, 24, 28, 20 and 24, respectively.

Definition 5 (Transaction Utility). *Given a transaction database D, then the transaction utility of a transaction T_i in D, denoted by $TU(T_i)$, is the sum of the utility of all items in T_i, i.e.,*

$$TU(T_i) = \sum_{a \in T_i} u(a, T_i) \tag{4}$$

Definition 6 (Transaction Weighted Utilization). *The transaction weighted utilization of an itemset X in a transaction database D, denoted by $TWU(X, D)$, is defined as the sum of the transaction utility of transactions containing X, i.e.,*

$$TWU(X) = \sum_{(i,T_i)\in D \mid X\subseteq T_i} TU(T_i) \tag{5}$$

Here, the difference between $TWU(X, D)$ and $u(X)$ is that for $TWU(X, D)$, we sum the utilities of the whole transactions containing X, while $u(X)$ computes only the utilities of X in the transactions where X appears.

Example 4. Let us consider again Example 1. We have, $TWU(\{a, c, d\}, D) = TU(T_1) + TU(T_3) = 7 + 24 = 31$.

We conclude this subsection by pointing out that the transaction weighted utilization measure has three important properties that are used to prune the search space.

Overestimation. The TWU of an itemset X is always higher than or equal to the utility of X, i.e., $TWU(X, D) \geq u(X)$.

Anti-monotonicity. Let X and Y be two itemsets. If $X \subseteq Y$, then $TWU(X, D) \geq TWU(Y, D)$.

Pruning. Let X be an itemset. If $TWU(X, D) < \theta$, then X is a low-utility itemset as well as all its supersets.

2.2 Propositional Logic and SAT Problem

In this subsection, we present the syntax and the semantics of classical propositional logic.
Given a countable set of propositional variables $Prop$, we use the letters p, q, r, etc. to range over $Prop$. The set of propositional formulas, denoted $Form$, is defined inductively started from $Prop$, the constant $\top$ denoting *true*, the constant $\bot$ denoting *false*, and using logical connectives $\neg, \wedge, \vee, \rightarrow$. It is worth noticing that we can restrict the language to the connectives $\neg$ and $\wedge$, since we have the following equivalences: A $\vee B \equiv \neg(\neg A \wedge \neg B)$ and A $\rightarrow B \equiv \neg B \vee A$. The equivalence connective $\leftrightarrow$ is defined by $A \leftrightarrow B \equiv (A \rightarrow B) \wedge (B \rightarrow A)$. We also use $P(A)$ to denote the set of propositional variables appearing in the formula A. $\models$ refers to logical inference. A Boolean interpretation Δ of a formula A is defined as a function from $P(A)$ to $(0, 1)$ (0 corresponds to *false* and 1 to *true*). A model of a formula A is a Boolean interpretation Δ that satisfies A, i.e., $\Delta(A) = 1$. A formula A is satisfiable if there exists a model of A. Moreover, the formula A is valid or a theorem, if every Boolean interpretation is a model of A. We use $Mod(A)$ to denote the set of models of A.

Let us now define the conjunctive normal form (CNF, for short) representation of propositional formulas. A CNF formula is a conjunction ($\wedge$) of clauses

where a clause is a disjunction ($\vee$) of literals. A literal is a propositional variable (p) or its negation ($\neg p$). A CNF formula can also be seen as a set of clauses, and a clause as a set of literals. We can point out that any propositional formula can be translated into the corresponding CNF formula using linear Tseitin encoding [20].

SAT is the decision problem that aims to determine the satisfiability of a CNF formula, i.e., whether there exists a model of all clauses in a CNF form. This is known as NP-Complete problem. Interestingly, state-of-the-art SAT solvers have been shown of practical use in solving real-world instances encoding industrial problems up to million of variables and clauses. SAT solving has been exploited in various fields, including planning, bio-informatics, cryptography and more recently data mining and relational databases. In most of these applications, we are mainly interested in identifying the satisfiability of a CNF formula, or in computing an optimal solution in Maximum Satisfiability (Max-SAT, for short). However, in data mining we mainly deal with the computation of all the models of a CNF formula.

3 Related work

Several proposals have been studied to enumerate HUIs (see [4, 5, 32] for a survey on the field). A popular approach to solve this problem is to discover the set of high utility itemsets in two phases. This approach commonly adopts the Transaction-Weighted-Downward Closure model to prune the search space. It first generates a set of candidate high-utility itemsets by overestimating their utility in phase 1. Then, in phase 2, it performs an extra database scan to calculate the exact utility of candidates and filter out low-utility itemsets.

The two phases based approach is adopted by Two-Phase [10], IHUP [13] and Up-Growth [18] algorithms. While two-phases based approaches are well studied, they remain inefficient because they not only generate too many candidates in the first phase, but they also need to scan, in phase 2, multiple times the database, which can be computationally expensive.

To address these issues, numerous studies have been conducted in order to develop efficient methods for mining high utility itemets directly using a single phase (called one phase algorithms). To prune the search space, one phase approaches rely on the concept of remaining utility. Among these algorithms HUI-Miner [12], D2HUP [17], FHM [14], EFIM [15], mHUIMiner [16], and ULB [19]. According to the comparisons in [5] between these various HUIM algorithms, it has been demonstrated that one phase algorithms outperform candidate generation based algorithms such as Two-Phase and Up-Growth, which are impractical for discovering HUIs from transaction databases. Furthermore, the authors in [5] demonstrated that the most efficient algorithms (in terms of memory consumption) are EFIM and D2HUP (in running time). The newest HUIM algorithms, mHUIMiner and ULB-Miner, typically perform between EFIM and d2HUP, but in a few cases, mHUIMiner outperforms both.

Rather than mining the entire set of HUIs, some researchers proposed more concise representations that significantly reduce the number of mined patterns (see [4] for clear definitions of these representations). The first algorithm to find compact representation of HUIs, called CHUD ($Closed^+$ High Utility Itemset Discovery), was introduced by [30]. This method aids in resolving the issue of a large number of candidates being generated. Notice that CHUD is an extension of Eclat [33] and DCI-Closed [34] algorithms. The CHUD algorithm uses a vertical database and computes CHUIs in a depth-first search. In [29], the authors proposed EFIM-Closed a lossless and compact representation for high utility itemset mining that is able to provide complete set of closed high-utility itemsets. Another algorithm, called CHUIMiner, to find CHUIs is proposed [31]. This method computes the utility of itemsets without generating candidates.

4 SAT Encoding of (Closed) High Utility Itemset Mining

In this section, we introduce our SAT-based formulation that enables us to specify in terms of constraints the task of finding (closed) high utility itemsets over transaction databases.

Our main goal is to provide an efficient way to encode and enumerate all closed high utility itemsets with SAT. Without loss of generality, we fix a transaction database $D = \{(1, T_1), \ldots, (m, T_m)\}$ and a minimum utility threshold θ. Our SAT encoding for HUIM that we will consider is based on the use of propositional variables to represent the items and the transaction identifiers in D. Specifically, for each item a (resp. transaction identifier i), we associate a propositional variable, denoted as p_a (resp. q_i). These propositional variables will be used in 0/1 linear inequalities to capture all possible itemsets and their covers.

More formally, given a Boolean interpretation Δ, the candidate itemset and its cover are expressed as $\{a \in \Omega \mid \Delta(p_a) = 1\}$ and $\{i \in \mathbb{N} \mid \Delta(q_i) = 1\}$, respectively. Now, we introduce our SAT-based encoding using the propositional variables described previously. The first propositional formula allows us to obtain the cover of the candidate itemset.

$$\bigwedge_{i=1}^{m} (\neg q_i \leftrightarrow \bigvee_{a \in \Omega \setminus T_i} p_a) \tag{6}$$

Intuitively, this propositional formula expresses that q_i is true if and only if the candidate itemset is supported by the i^{th} transaction. More specifically, the considered itemset is not supported by the i^{th} transaction (i.e., q_i is *false*), when there exists an item a (i.e., p_a is *true*) that does not appear to the transaction ($a \in \Omega \setminus T_i$), i.e., when q_i is *false* that means at least an item not appearing in the transaction i is set to *true*.

Let us now give the formula expressing that the utility of the candidate itemset has to be larger than the specified utility threshold θ:

$$\sum_{i=1}^{m} \sum_{a \in T_i} u(a, T_i) \times (p_a \wedge q_i) \geqslant \theta \tag{7}$$

Using additional variables, Constraint 7 can be rewritten using the following two formulas:

$$\sum_{i=1}^{m} \sum_{a \in T_i} u(a, T_i) \times r_{ai} \geqslant \theta \tag{8}$$

$$\bigwedge_{i=1}^{m} \bigwedge_{a \in T_i} (r_{ai} \leftrightarrow p_a \wedge q_i) \tag{9}$$

In the sequel, we use Φ_{huim} to denote the CNF encoding that corresponds to the conjunction of equations (6), (8), and (9).

Proposition 1. *Given a transaction database $D = \{(1, T_1), \ldots, (m, T_m)\}$ and a minimum utility threshold θ. Then, the CNF formula Φ_{huim} models the problem of mining high utility itemsets from D.*

Proof. It is straightforward to see that there exists a mapping between the set of models of Φ_{huim} and the high utility itemsets.

Example 5. We consider the transaction database given in Example 1. Then, the formula that encodes the problem of enumerating all high utility itemsets in D with $\theta = 20$ can be written as follows:

$\neg q_1 \leftrightarrow (p_b \vee p_e)$ $\quad \neg q_2 \leftrightarrow (p_b \vee p_d)$ $\quad \neg q_3 \leftrightarrow \bot$ $\quad \neg q_4 \leftrightarrow p_a$
$r_{a1} \leftrightarrow p_a \wedge q_1$ $\quad r_{c1} \leftrightarrow p_c \wedge q_1$ $\quad r_{d1} \leftrightarrow p_d \wedge q_1$ $\quad r_{a2} \leftrightarrow p_a \wedge q_2$
$r_{c2} \leftrightarrow p_c \wedge q_2$ $\quad r_{e2} \leftrightarrow p_e \wedge q_2$ $\quad r_{a3} \leftrightarrow p_a \wedge q_3$ $\quad r_{b3} \leftrightarrow p_b \wedge q_3$
$r_{c3} \leftrightarrow p_c \wedge q_3$ $\quad r_{d3} \leftrightarrow p_d \wedge q_3$ $\quad r_{e3} \leftrightarrow p_e \wedge q_3$ $\quad r_{b4} \leftrightarrow p_b \wedge q_4$
$r_{c4} \leftrightarrow p_c \wedge q_4$ $\quad r_{d4} \leftrightarrow p_d \wedge q_4$ $\quad r_{e4} \leftrightarrow p_e \wedge q_4$
$4r_{a1} + r_{c1} + 2r_{d1} + 8r_{a2} + 6r_{c2} + 6r_{e2} + 4r_{a3} + 4r_{b3} + r_{c3} + 12r_{d3} + 3r_{e3} + 8r_{b4}$
$+3r_{c4} + 6r_{d4} + 3r_{e4} \geq 20$

Once the CNF formula Φ_{huim} is constructed, we systematically use a SAT solver to enumerate all its models. Obviously, each found model of Φ_{huim} corresponds to a high utility itemset in the original transaction database. Specifically, our method proceeds by recursively assigning variables corresponding to items and performing unit propagation. Then, Constraint (8) is checked during the search to verify if a conflict occurs. Such checking can be easily performed by considering the value $\sum_{i=1}^{m} \sum_{a \in T_i} u(a, T_i)$ and by subtracting $u(a, T_i)$ each time r_{ai} becomes *false*. A comparison to θ is also performed to continue the search or to backtrack, otherwise.

In the HUIM context, there is two possibilities to define the closure constraint on itemsets. In the first case, we can define a closed itemset as an itemset which has any superset having the same utility [29]. The second choice is to consider the

support constraint used in classical frequent itemset mining to define the closure. However, the first definition does not allow us to obtain a reduced number of itemsets, since in real-world application we can not find many itemsets that have exactly the same utility as well as their supersets. In high utility enumeration, a CHUI is a HUI having no proper supersets that are HUIs and appear in the same number of transactions. So, our idea here is to use the support constraint of traditional FIM task to define the closure constraint in the context of high utility itemset mining. Hence, the following propositional formula allows us to force a candidate itemset to be closed:

$$\Phi_{clos} = \bigwedge_{a \in \Omega} (p_a \vee \bigvee_{a \notin T_i} q_i) \tag{10}$$

Intuitively, Constraint 10 ensures that if the candidate itemset is involved in all transactions containing the item a, then a must be added to the itemset. In other words, when in all the transactions where a does not appear, the candidate itemset is not included, this implies that the candidate itemset belongs only to transactions containing the item a. Consequently, to be closed, the item a must be added to the candidate itemset. We stress here that Constraint 10 is necessary and sufficient to force the itemset to be closed.
Now, the closed HUIM task, denoted as CHUIM, can be encoded as the conjunction of the formulas 6, 8, 9 and 10. More formally, $\Phi_{chuim} = \Phi_{clos} \wedge \Phi_{huim}$.

In a propositional satisfiability problem, if the CNF formula is satisfiable, the SAT solver provides the corresponding model(s). To enumerate the models of our Φ_{chuim} encoding, we extend a backtracking based search algorithm like Davis–Putnam–Logemann–Loveland (DPLL, for short) procedure [28]. In the sequel, we briefly describe the basic component of DPLL (see Algorithm 1) designed to enumerate all models of a given CNF formula.

Typically, the DPLL solver is a tree-based backtrack search procedure. A decision variable is chosen, i.e., by assigning variables corresponding to items (line 17) to be added to Φ, followed by a propagation of unit literal assignments (line 18). If there is a unit clause (all literals are false except one namely p), the literal p is propagated (line 3-4). Otherwise, Constraint (8) is checked (line 9) to verify its consistency. A comparison to θ is then performed to continue the search or to backtrack. Such checking can be easily accomplished by considering the value $\sum_{i=1}^{m} \sum_{a \in T_i} u(a, T_i)$ and by subtracting $u(a, T_i)$ each time r_{ai} becomes *false*. If all literals are assigned without contradiction, then Δ is a model of the CNF formula (line 12). Obviously, each found model of Φ_{chuim} gives rise to a (closed) high utility itemset from the transaction database (line 13) by restricting the model to variables encoding items.

4.1 Pruning Strategy

The basic version of our algorithm performs a backtrack search similarly to the TWU measure in order to prune the search space. Now, a useful extension to

Algorithm 1: *DPLL_Enum*: A backtracking search procedure for models enumeration

Input: Φ: a CNF formula, θ: a minimum utility threshold
Output: S: the set of models of Φ

$\Delta = \emptyset$; /* interpretation */
$S = \emptyset$;
if $\underline{(\Phi \models p)}$ **then**
  return $DPLL\_Enum(\Phi \wedge p, \Delta \cup \{p\}))$; /* unit clause */
end
if $\underline{(\Phi \models \bot)}$ **then**
  return $False$; /* conflict */
end
if $\underline{check\_utility\_candidate(\theta) == false}$ **then**
  return $False$;
end
if $\underline{(\Delta \models \Phi)}$ **then**
  $S \leftarrow S \cup \{\Delta\}$; /* new found model */
  return $False$;
else
end
p = select_variable($Var(\Phi)$);
return $DPLL\_Enum(\Phi \wedge p, \Delta \cup \{p\}) \vee DPLL\_Enum(\Phi \wedge \neg p, \Delta \cup \{\neg p\})$;
return S;

perform better pruning in the search tree consists to add a new constraint to the previous Φ_{chuim} encoding. Notice that this constraint is derived from Inequation (8) using the weighted clique cover problem [21]. Our main idea is to identify the subsets of variables r_{ai} which cannot be *true* simultaneously. Next, we show how this new constraint can be derived in a suitable way in order to make our pruning strategy more efficient. To do this, let us first introduce a graphical representation of the original transaction database as follows.

Definition 7. *Let $D = \{T_1, T_2, \ldots, T_m\}$ be a transaction database. Then, the graph associated to D is an undirected graph $G_D = (V, E)$ such that each item in each transaction represents a vertex in G_D, i.e., v_{ai} is the vertex associated to the item a in the transaction i with $1 \leq i \leq m$. In addition, an edge $(v_{ai}, v_{a'j}) \in E$ iff the transaction i contains a but not a'.*

Example 6. Given the transaction database of Example 1. Then, the graph G_D associated to this database is depicted in Figure 1.

Definition 7 ensures that each edge of G_D connects two items of D that cannot belong simultaneously to the same high utility itemset. For instance, in the database of Example 1, the item a in the transaction T_1 cannot appear with b in T_3. Now, our aim is to partition the graph G_D into overlapping sets. To do this, we will use the notion of *clique cover* defined as follows.

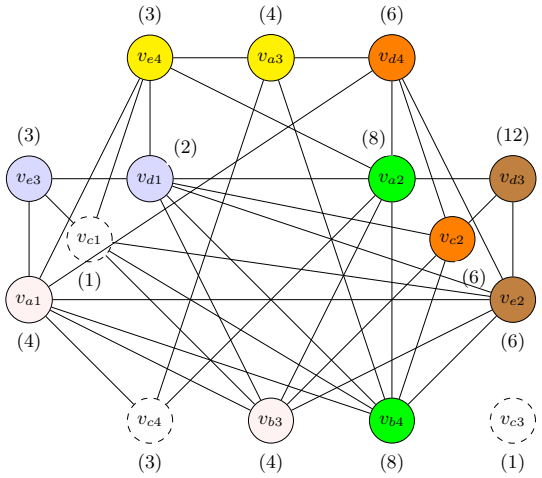

Fig. 1: The graph G_D associated to the transaction database of Example 1

Definition 8 (Clique Cover). *Let $G = (V, E)$ be an undirected graph and $C = \{C_1, \ldots, C_k\}$ where $C_i \subseteq V$ for $1 \leq i \leq k$. Then, S is a clique cover of G iff $\bigcup_{1 \leq i \leq k} C_i = V$ s.t. each sub-graph $G_i = (C_i, E_i)$ where $E_i = \{(a, b) \in E \mid a, b \in C_i\}$ is a clique*[4].

The clique cover is a fundamental problem in graph theory and it has numerous applications in several areas such as social network analysis and bioinformatics. Notice that the problem of clique cover has extensively studied in the literature [25, 26].

Example 7. Let us consider the graph in Example 6. Clearly, the set of sets $S = \{\{v_{d1}, v_{e3}\}, \{v_{a1}, v_{b3}\}, \{v_{a2}, v_{b4}\}, \{v_{c2}, v_{d4}\}, \{v_{e2}, v_{d3}\}, \{v_{a3}, v_{e4}\}, \{v_{c1}\}, \{v_{c3}\}, \{v_{c4}\}\}$ is a clique cover of the graph G_D.

Given the graph G_D, the cliques of G_D are a convenient way of conceptualizing the required constraint that we need to consider in our SAT encoding. In fact, a clique of G_D corresponds to a subset of variables r_{ai} that among them at most one can be assigned to *true*. This allows us to introduce a new constraint that can be used to prune effectively the search space compared to the TWU measure. Consequently, our new constraint can be derived where the sum of weights of each subset is replaced with the maximum weight as stated in the following proposition.

Proposition 2. *Let $D = \{T_1, T_2, \ldots, T_m\}$ be a transaction database and G_D the graph associated to D. If $C = \{C_1, \ldots, C_k\}$ is a clique cover of G_D, then the following constraint holds :*

[4] A clique is a graph whose nodes are all pairwise adjacent.

$$\sum_{1 \leq i \leq k} \max_{v_{aj} \in C_i} u(a, T_j)(\bigvee_{v_{aj} \in C_i} r_{aj}) \geq \theta \tag{11}$$

Additional variables x_i $(1 \leq i \leq k)$ can be used to simplify Constraint (11) in the following way:

$$x_i \leftrightarrow \bigvee_{v_{aj} \in C_i} r_{aj} \quad \forall\, 1 \leq i \leq k$$

Note that the weighted clique cover problem is NP-hard and the number of solutions can be very large. Our aim here is to minimize $\sum_{i=1}^{k} w_i$ with $w_i = max_{v_{aj} \in C_i} u(a, T_j)$. The goal is then to obtain large cliques with maximum weight. To avoid the NP-Hardness of the related problem, we next consider a greedy approach to find a possible cover. To do this, we proceed by growing the current clique one vertex at a time by looping through the remaining adjacent vertices with high weight values in the current graph.

Example 8. Let us consider again Example 6. For the clique cover $S = \{\{v_{d1}, v_{e3}\}, \{v_{a1}, v_{b3}\}, \{v_{a2}, v_{b4}\}, \{v_{c2}, v_{d4}\}, \{v_{e2}, v_{d3}\}, \{v_{a3}, v_{e4}\}, \{v_{c1}\}, \{v_{c3}\}, \{v_{c4}\}\}$, we can deduce the following constraint:

$$3x_1 + 4x_2 + r_{c_1} + 8x_3 + 6x_4 + 12x_5 + 4x_6 + r_{c3} + 3r_{c4} \geq 20$$

where

$$\begin{array}{ll} x_1 = (r_{d1} \vee r_{e3}) & x_2 = (r_{a1} \vee r_{b3}) \\ x_3 = (r_{a2} \vee r_{b4}) & x_4 = (r_{c2} \vee r_{d4}) \\ x_5 = (r_{e2} \vee r_{d3}) & x_6 = (r_{a3} \vee r_{e4}) \end{array}$$

Using such new constraint, it is clear that if the minimum threshold exceeds 20, then we can trivially check that the set of high utility itemsets is empty which is not the case when taking into account only Constraint (7). More generally, by considering both Constraint (7) and the new derived one allows us to prune the search space more efficiently.

4.2 A Decomposition-based Encoding for mining CHUIs

In this subsection, we present a decomposition-based paradigm that splits the original transaction database into smaller and independent subsets in order to avoid encoding the whole base. Our decomposition is motivated by the fact that encoding the whole database into propositional logic can lead to very large formulas, that is the solving can be unfeasible. In fact, given the set of items $\Omega = \{a_1, \ldots, a_n\}$ of D, the set of high utility itemsets can be partitioned into $E_1, \ldots, E_n$ where E_1 is the susbet of itemsets containing a_1, E_2 of those not containing a_1 but a_2, and so on until E_n of those do not involving $a_1, \ldots, a_{n-1}$ but a_n. From the encoding point of view, the subset E_i is obtained by enumerating the models of Φ_i resulting from Φ by propagating p_{a_k} to *false* for all $1 \leq k < i$ and p_{a_i} to *true*. Formally, this yields to the next formula:

$$\Phi_i = \Phi \wedge p_{a_i} \wedge \bigwedge_{1 \leq k < i} \neg p_{a_k}$$

Intuitively, Φ_i is based on a recursive split of the formula Φ w.r.t. its positive and negative literals. As expressed, the formula Φ_i enforces p_{a_i} to be *true*. Consequently, the encoding can be restricted to transactions containing the item a_i. Also, the literal p_{a_k} for all $1 \leq k < i$ assigned *false* allows to exclude the item a_k to be in the candidate itemset. Thus, this allows to avoid encoding the entire original database and without causing too large formulas as well as the associated memory problems. Clearly, the splitting of Φ generates a set of independent sub-formulas that encode subsets of a specific set of transactions in the original database.

Example 9. The partitioning tree of the transaction database of Example 1 is depicted in Figure 2. Here, we consider at the beginning all transactions containing the item a. Then, we pick all those not containing the item a but the item b, and so on.

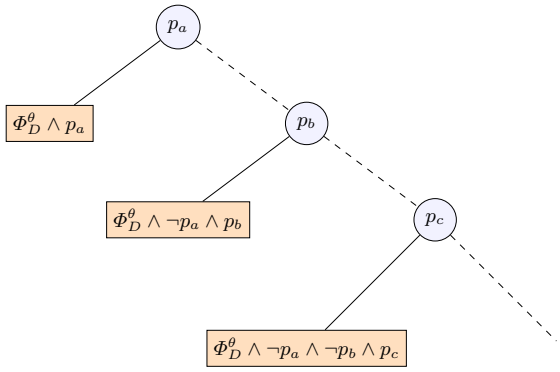

Fig. 2: Item based partitioning tree of the database D

Next, the pseudo-code of our algorithm using a decomposition-based method for enumerating all CHUIs is summarized in Algorithm 2. The algorithm, coined as SATCHUIM (**SAT** based **C**losed **H**igh **U**tility **I**temset **M**ining), takes a transaction database D and a minimum utility threshold θ, and it outputs all the closed high utility itemsets in D. Notice that the decomposition technique applied to the database D provides some significant advantages. First, splitting the whole database into independent sub-bases can reduce significantly the size of the original problem. Also, the algorithm does not need to generate candidate sets since the reduced database does not contain any low utility itemset, i.e., each itemset that has utility lower than TWU is discarded.

The algorithm follows an order over the set of items appeared in D by counting items occurrence, i.e, items are sorted in increasing order of their support. Obviously, the splitting strategy is performed before the solving process. At each iteration, each item a_i is fixed to be in the itemset and the encoding is restricted only to transactions containing a_i, denoted D_{a_i}. Then, if the TWU of the item

a_i is less than the θ threshold, a_i is discarded and it will not belong to the set of HUIs. In the next iteration, we ignore in the current sub-database all items having TWU less than θ. Then, the function *encode_huim_cnf* is called over D_{a_i} in order to encode the problem into CNF. Finally, the enumeration of the models of the CNF formula is performed using the function *DPLL_Enum* (Algorithm 1).

Algorithm 2: SAT based Closed High Utility Itemset Mining (`SATCHUIM`)

Input: D: a transaction database, θ: a user-specified utility threshold
Output: S: the set of all closed high-utility itemsets of D

$\Omega = \langle a_1, \ldots, a_n \rangle \leftarrow items(D)$;
$S \leftarrow \emptyset$;
for $i \in 1..n$ **do**
  if $TWU(a_i) < \theta$ **then**
    continue;
  end
  $D_{a_i} \leftarrow \{(i_k, T_k) \in D \mid a_i \in T_k\}$;
  $\Gamma \leftarrow \emptyset$;
  for $b \in items(D_{a_i})$ **do**
    if $TWU(b, D_{a_i}) < \theta$ **then**
      $\Gamma \leftarrow \Gamma \cup \{b\}$;
    end
  end
  $\Phi \leftarrow encode\_huim\_cnf(D_{a_i}, \theta) \wedge p_{a_i} \wedge \bigwedge_{1 \leq j < i} \neg p_{a_j} \wedge \bigwedge_{b \in \Gamma} \neg p_b$;
  $S \leftarrow S \cup DPLL\_Enum(\Phi, \theta)$;
end
return S;

Proposition 3 (Correctness). *Let D be a transaction database.* `SATCHUIM` *returns all closed high utility itemsets of D.*

Again, we use the decomposition strategy for a scalability reason. In fact, we have shown in our previous work [37] that a SAT-based approach cannot scale to large datasets, e.g., *Chainstore* and *Kosarak*, since the CNF encoding of such transaction databases is huge and cannot be resolved by the SAT solvers. Hence, in this paper we applied the decomposition-based technique described previously to achieve efficiency by reducing the size of the CNF encoding without losing completeness.

5 Experimental Results

In this section, we carried out an experimental evaluation of our SAT-based formulation for mining profitable itemsets, and then compare the performance

of our method against the most efficient algorithms for discovering (closed) high utility itemsets using real-world transaction databases.

5.1 Experimental Setup

Experiments are performed on a computer with an Intel Xeon quad-core machine with 32GB of RAM running at 2.66 Ghz. To evaluate the practical performance of our approach, experiments were carried on seven real-life datasets commonly used in the HUIM literature: *Chess, Mushroom, Connect, Kosarak, Foodmart, Accidents* and *Chainstore* [22]. These datasets have various characteristics and represent data taken from real-life scenarios. For each benchmark, we report in Table 4 the number of transactions (#Trans), the number of items (#Items), the number of items per transaction or average transaction length (AvgTransLen), and the density[5]. The density factor has a direct impact on the computation time of mining algorithms. In our experiments, both sparse and dense datasets were used for performance evaluation. For each dataset, we fix a timeout of 2 hours.

We conducted two experiments to evaluate the performance of our proposed approach. In the first experiment, we compared our `SATHUIM` algorithm to the two most efficient specialized approaches for enumerating HUIs [5]: `EFIM` [15], and `D2HUP` [17]. Moreover, in the second set of experiments, we compare our `SATCHUIM` method against three baselines for mining closed HUIs, namely `EFIM-Closed` [29], `CHUD` [30], and `CHUI-Miner` [31]. For these baselines, we used the Sequential Pattern Mining Framework (SPMF, for short) open-source data mining library [22] written in Java. Our algorithms are implemented in C++ and we used the MiniSAT solver [38] to enumerate all models of the CNF encoding. In our experiments, for the minimum utility threshold values, we follow the work of Zida et al. [15].

Table 4: Datasets Characteristics

Instance	**#Trans**	**#Items**	**AvgTransLen**	**Density(%)**
Chess	3196	75	37	49.33
Foodmart	4141	1559	4.42	0.28
Mushroom	8124	119	23	19.33
Connect	67557	129	43	33.33
Accidents	340183	468	33.8	7.22
Kosarak	990002	41270	8.1	0.02
Chainstore	1112949	46086	7.23	0.02

[5] The density of a database D defines the ratio between the average length of transactions in D and the number of distinct items in D.

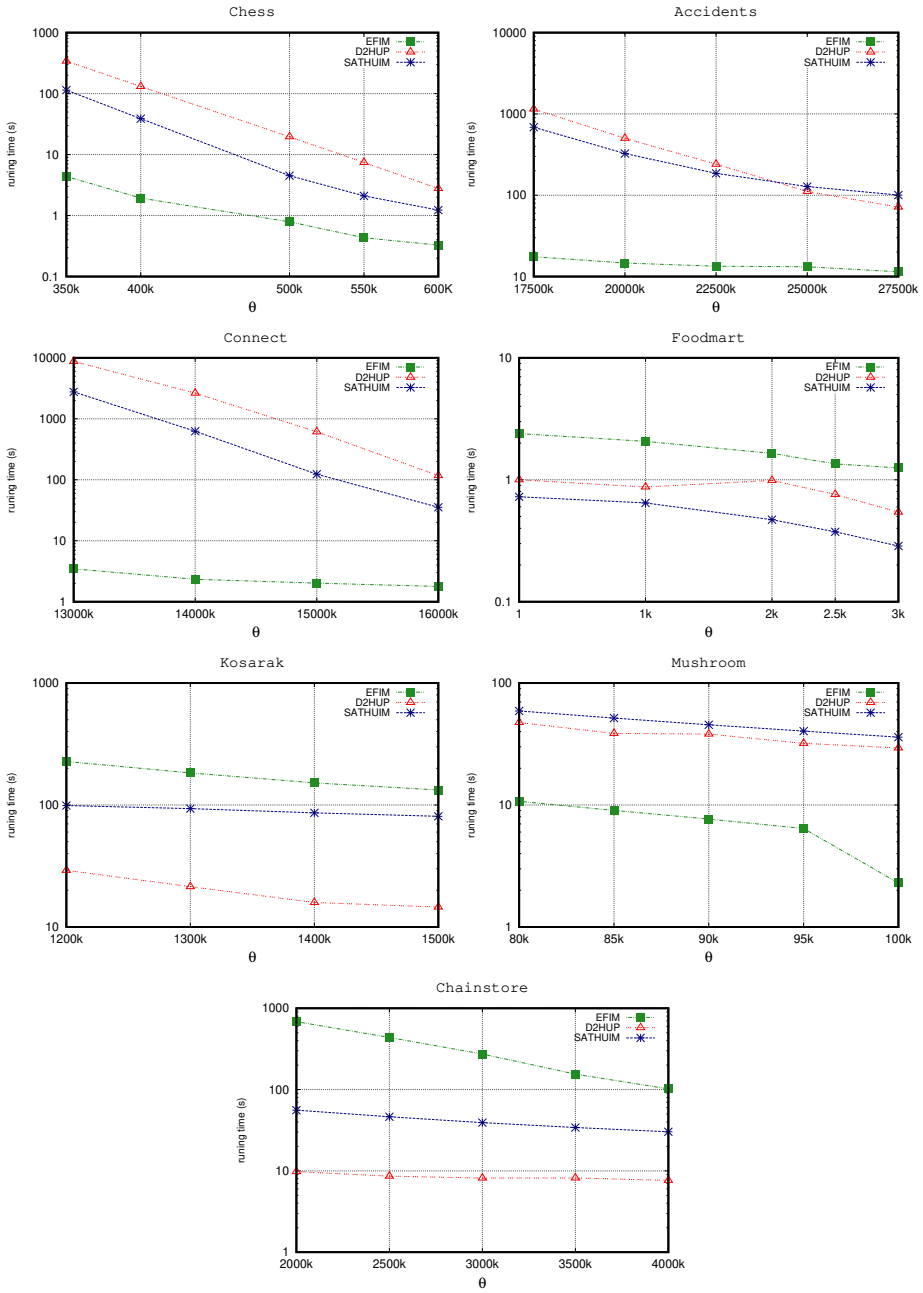

Fig. 3: Running times of SATHUIM against baselines on different datasets

5.2 Results for HUIs Enumeration

In this experiment, we run each method on each database while comparing the algorithms' running time for different minimum utility thresholds. Figure 3 reported the comparative results, i.e. the CPU time, of our method `SATHUIM` against the two specialized ones. Note that for our algorithm, the computation time includes the time for generating the CNF formula and that for enumerating all models (i.e., the HUIs). The empirical results show the feasibility of our SAT-based approach. As illustrated in Figure 3, it is clear that the performance of all algorithms depends on the dataset characteristics. In fact, all approaches need more time to discover the set of HUIs from large databases. In addition, the minimum support threshold θ has a strong influence on the performance of algorithms. We can also observe that `EFIM` is generally more efficient than our `SATHUIM` method. Moreover, our algorithm is the second-fastest method and it falls between `EFIM` and `D2HUP` in nearly all cases, except *Foodmart* and *Mushroom*. As a summary, our SAT-based method is competitive and it achieves interesting performance in enumerating HUIs, compared with the two baselines `EFIM` and `D2HUP`.

5.3 Results for CHUIs Enumeration

In this subsection, we turn to the empirical evaluation of our `SATCHUIM` algorithm and compare it to three baselines, namely, `EFIM-Closed` [29], `CHUD` [30], and `CHUI-Miner` [31] for enumerating closed high utility itemsets from transaction databases.

Figure 4 shows the execution times of the different algorithms. According to these experimental results, our proposal achieves a good performance in 5 out of 7 databases for different minimum utility threshold values. As we can also see from the results, our algorithm remains competitive with the best baselines for *Mushroom* and *Chainstore* databases. Notably, the required time for our `SATCHUIM` algorithm to find all CHUIs increases when the θ value decreases on *Chess*, *accidents*, *connect* and *foodmart*. In contrast, the running time remains almost constant when θ threshold varies for the datasets *Chainstore*, *kosarak* and *Mushroom*. In terms of average CPU time and for low values of θ, `SATCHUIM` surpasses `CHUD` by about 163 and 9 times on *Chess* and *Kosarak* datasets , respectively. Likewise, `SATCHUIM` surpasses `CHUI-Miner` by about 147 and 3 times on *Chess*, *kosarak*, respectively. In addition, our algorithm is up to 60 times faster than `CHUI-Miner`, while `CHUD` took too long time to terminate on the dataset *Connect*.

From the scalability point of view, we observe that our proposed SAT-encoding is able to scale for all the minimum support threshold values under the time limit for large datasets; while `CHUD` and `CHUI-Miner` algorithms not able to scale on the datasets *accidents*, *Connect* and *Chainstore* under the time limit. Overall, the empirical evaluation confirms that our `SATCHUIM` algorithm is very promising.

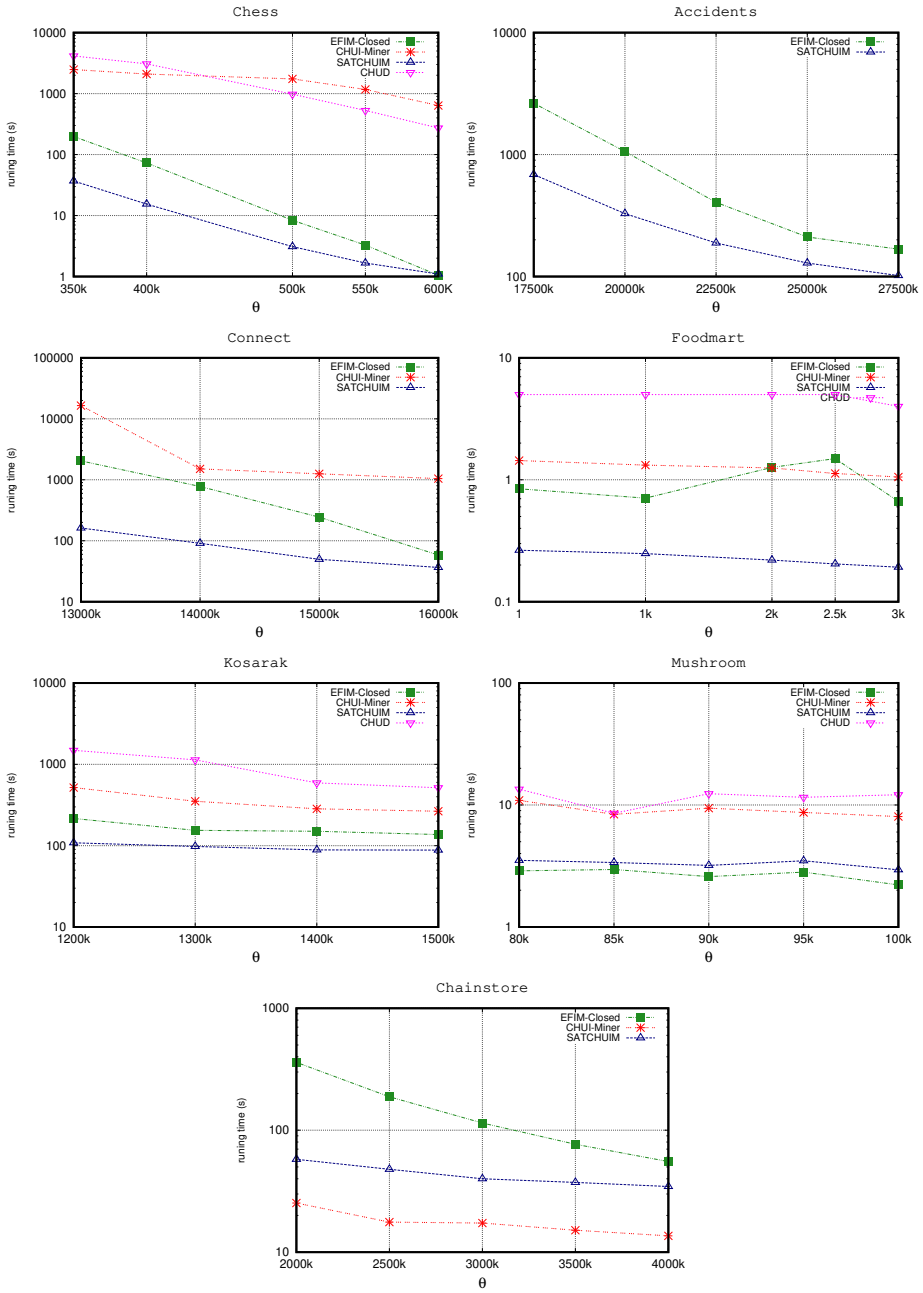

Fig. 4: Running times of SATCHUIM against baselines on different datasets

Finally, Table 5 provides the variation of the number of generated patterns as well as the number of propositional variables and clauses used to encode the problem for both classical HUIs and closed HUIs. The main observation is that the number of found itemsets highly depends on the selected threshold values: it decreases when the utility threshold increases and vice versa. Furthermore, the number of patterns can be limited when the minimum utility threshold is large. Let us also mention that the CHUIs is smaller compared to the set of HUIs. For instance, for `Chess` dataset the number of HUIs is equal to 428023, while the number of CHUIs is about 114660. According to the experiment results shown in Table 5, we can observe that the number of clauses can exceed 400 millions (i.e., Kosarak dataset). Note that this number corresponds to the sum of the number of clauses of the different sub-problems generated by our decomposition technique. Despite this large number of clauses and variables, our approach remains efficient and scale on all datasets in a reasonable time.

6 Conclusion

In this paper, we have introduced a novel SAT-based approach for mining (closed) high utility itemsets. The proposed method exploits a number of well-known established techniques for SAT-based problem solving. Technically, the main idea is to represent a (closed) high utility itemset mining task as a CNF propositional formula such that each of its models corresponds to a (closed) high utility itemset of interest. In contrast with existing specialized algorithms, we have shown a flexible formulation in terms of constraints of the task of discovering (closed) HUIs. Experimental results have also shown that our SAT-based approach is very competitive with the state-of-the-art techniques.

Despite our promising results, we intend to develop a parallel version to even improve the performance of our SAT-based approach for enumerating (closed) HUIs. Moreover, setting the appropriate minimum utility threshold is a hard question, so we plan to extend our declarative SAT approach to enumerate the Top-k (closed) high utility itemsets mining.

References

1. C.F. Ahmed, S.K. Tanbeer, B.-S. Jeong, and Y.-K. Lee, "Efficient Tree Structures for High Utility Pattern Mining in Incremental Databases," IEEE Trans. Knowledge and Data Eng., vol. 21, no. 12, pp. 1708-1721, Dec. 2009.
2. B.-E. Shie, H.-F. Hsiao, V., S. Tseng, and P.S. Yu, "Mining High Utility Mobile Sequential Patterns in Mobile Commerce Environments," Proc. 16th Int'l Conf. DAtabase Systems for Advanced Applications (DASFAA '11), vol. 6587/2011, pp. 224-238, 2011.
3. S.J. Yen and Y.S. Lee, "Mining High Utility Quantitative Association Rules." Proc. Ninth Int'l Conf. Data Warehousing and Knowledge Discovery (DaWaK), pp. 283-292, Sept. 2007.

			HUIM		CHUIM	
Dataset	θ	**#Var**	**#Clauses**	**#HUIs**	**#Clauses**	**#CHUIs**
Chess	350k	3271	4428501	1520768	4429609	348633
	400k		4372199	428023	4373242	114660
	500k		4181330	24979	4182148	10888
	550k		4149748	4214	4150431	2439
	600k		4026018	583	4026669	394
Foodmart	1	5700	493550	233231	568824	6680
	1k		476507	219012	539443	6454
	2k		399040	154670	448748	5273
	2.5k		350465	117592	395121	4552
	3k		299205	85034	338374	3804
Mushroom	80k	8243	5404598	2042385	5407235	8017
	85k		5365314	1700923	5367873	7267
	90k		5332928	1426503	5335436	6569
	95k		5306318	1221641	5308730	5957
	100k		5280456	1045780	5282798	5337
Connect	1300k	67686	101535054	452108	101535737	19336
	1400k		99721925	87602	99722512	7334
	1500k		92501566	11317	92502128	1831
	1600k		87973803	613	87974308	214
Accidents	17500k	340651	352955542	23824	352956530	23824
	20000k		328418715	5991	328419469	5991
	22500k		310834204	1341	310834834	1341
	25000k		299295508	266	299296039	266
	27500k		294620791	46	294621239	46
Kosarak	1100k	1031272	412482672	102	412482836	102
	1200k		392671175	87	392671301	87
	1300k		367975399	78	367975491	78
	1400k		343918121	67	343918197	67
	1500k		318686329	61	318686395	61
Chainstore	2000k	1159035	83494966	127	83495116	127
	2500k		72354830	87	72354869	87
	3000k		62883758	66	62883775	66
	3500k		55819252	57	55819266	57
	4000k		49970086	46	49970098	46

Table 5: Comparative results using different minimum utility threshold values

4. Fournier-Viger, P., Lin, J. C.-W., Vo, B, Chi, T.T., Zhang, J., Le, H. B. (2017). A Survey of Itemset Mining. WIREs Interdisciplinary reviews - Data Mining and Knowledge Discovery, Wiley.
5. Zhang Chonsheng et al. An empirical evaluation of high utility itemset mining algorithms. In: 101(2018). pp.91-115.
6. T. Guns, S. Nijssen, and L. D. Raedt, "Itemset mining: A constraint programming perspective," Artificial Intelligence, vol. 175, no. 12-13, pp. 1951–1983, 2011..
7. L. D. Raedt, T. Guns, and S. Nijssen, "Constraint programming for itemset mining," in ACM SIGKDD, 2008, pp. 204–212.

8. E. Coquery, S. Jabbour, L. Sais, and Y. Salhi, "A sat-based approach for discovering frequent, closed and maximal patterns in a sequence," in Proceedings of the 20th European Conference on Artificial Intelligence (ECAI'12), 2012, pp. 258–263.
9. S. Jabbour, L. Sais, and Y. Salhi, "The top-k frequent closed itemset mining using top-k sat problem," in Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases (ECML/PKDD'03), 2013, pp. 403–418.
10. Liu, Y., Liao, W., Choudhary, A.: A two-phase algorithm for fast discovery of high utility itemsets. In: Proc. 9th Pacic-Asia Conf. on Knowl. Discovery and Data Mining, pp. 689695 (2005)
11. Tseng, V. S., Shie, B.-E., Wu, C.-W., Yu., P. S.: Efficient algorithms for mining high utility itemsets from transactional databases. IEEE Trans. Knowl. Data Eng. 25(8), 1772-1786 (2013)
12. Liu, M., Qu, J.: Mining high utility itemsets without candidate generation. In: Proc. 22nd ACM Intern. Conf. Info. and Know. Management, pp. 5564 (2012).
13. Krishnamoorthy, S.: Pruning strategies for mining high utility itemsets. Expert Systems with Applications, 42(5), 2371-2381 (2015)
14. Fournier-Viger, P., Wu, C.-W., Zida, S., Tseng, V. S.: FHM: Faster high-utility itemset mining using estimated utility co-occurrence pruning. In: Proc. 21st Intern. Symp. on Methodologies for Intell. Syst., pp. 8392 (2014)
15. Zida, Souleymane, Fournier Viger, Philippe, Lin, Chun-Wei, Wu, Cheng-Wei, Tseng, Vincent. (2017). EFIM: a fast and memory efficient algorithm for high-utility itemset mining. Knowledge and Information Systems. 51. 10.1007/s10115-016-0986-0.
16. Alex Yuxuan Peng, Yun Sing Koh, and Patricia Riddle. "mHUIMiner: A Fast High Utility Itemset Mining Algorithm for Sparse Datasets". en. In: Advances in Knowledge Discovery and Data Mining. Ed. by Jinho Kim et al. Vol. 10235. Cham: Springer International Publishing, 2017, pp. 196–207.
17. Liu, Junqiang Wang, ke Fung, Benjamin. (2012). Direct Discovery of High Utility Itemsets without Candidate Generation. Proceedings - IEEE International Conference on Data Mining, ICDM. 984-989. 10.1109/ICDM.2012.20.
18. Vincent S. Tseng, Cheng-Wei Wu, Bai-En Shie, and Philip S. Yu. 2010. UP-Growth: an efficient algorithm for high utility itemset mining. In Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '10). Association for Computing Machinery, New York, NY, USA, 253–262. DOI:https://doi.org/10.1145/1835804.1835839.
19. Quang-Huy Duong et al. "Efficient High Utility Itemset Mining Using Buffered Utility-lists". In: Applied Intelligence 48.7 (July 2018), pp. 1859–1877.
20. Tseitin, G.: On the complexity of derivations in the propositional calculus. In: Structures in Constructives Mathematics and Mathematical Logic, Part II. pp. 115-125 (1968).
21. Wen-Lian Hsu, George L. Nemhauser. A polynomial algorithm for the minimum weighted clique cover problem on claw-free perfect graphs, Discrete Mathematics, 1982,Pages 65-71.
22. P. Fournier-Viger. SPMF: A Java Open-Source Data Mining Library. URL: www.philippe-fournier-viger.com/spmf/ (visited on 08/15/2018).
23. A. Boudane, S. Jabbour, L. Sais, Y. Salhi, "A SAT-Based Approach for Mining Association Rules," in Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence (IJCAI'16), 2016, pp. 2472-2478.

24. A. Boudane, S. Jabbour, L. Sais, Y. Salhi, "Enumerating Non-redundant Association Rules Using Satisfiability," in Proceedings of the Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining (PAKDD'17), 2017, pp. 824-836.
25. James Cheng, Yiping Ke, Ada Wai-Chee Fu, Jeffrey Xu Yu, and Linhong Zhu. 2011. Finding maximal cliques in massive networks. ACM Trans. Database Syst. 36, 4.
26. Eblen, J.D., Phillips, C.A., Rogers, G.L. et al. The maximum clique enumeration problem: algorithms, applications, and implementations. BMC Bioinformatics 13, S5 (2012).
27. S. Jabbour et al., Boolean satisfiability for sequence mining, in proceedings of CIKM'13, pages 649-658.
28. Martin Davis, George Logemann, and Donald Loveland. 1962. A machine program for theorem-proving. Commun. ACM 5, 7 (July 1962), 394–397. DOI:https://doi.org/10.1145/368273.368557.
29. Fournier-Viger P., Zida S., Lin J.CW., Wu CW., Tseng V.S. (2016) EFIM-Closed: Fast and Memory Efficient Discovery of Closed High-Utility Itemsets. In: Perner P. (eds) Machine Learning and Data Mining in Pattern Recognition. MLDM 2016. Lecture Notes in Computer Science, vol 9729. Springer, Cham. https://doi.org/10.1007/978-3-319-41920-6-15.
30. V. S. Tseng, C. Wu, P. Fournier-Viger and P. S. Yu, "Efficient Algorithms for Mining the Concise and Lossless Representation of High Utility Itemsets," in IEEE Transactions on Knowledge and Data Engineering, vol. 27, no. 3, pp. 726-739, 1 March 2015, doi: 10.1109/TKDE.2014.2345377.
31. C. Wu, P. Fournier-Viger, J. Gu and V. S. Tseng, "Mining closed+ high utility itemsets without candidate generation," 2015 Conference on Technologies and Applications of Artificial Intelligence (TAAI), 2015, pp. 187-194.
32. Rahmati, Bahareh & Sohrabi, Mohammad. (2019). A Systematic Survey on High Utility Itemset Mining. International Journal of Information Technology & Decision Making. 18. 10.1142/S0219622019300027.
33. S. Selvan and R. V. Nataraj, "Efficient Mining of Large Maximal Bicliques from 3D Symmetric Adjacency Matrix," in IEEE Transactions on Knowledge and Data Engineering, vol. 22, no. 12, pp. 1797-1802, Dec. 2010, doi: 10.1109/TKDE.2010.97.
34. C. Lucchese, S. Orlando and R. Perego, "Fast and memory efficient mining of frequent closed itemsets," in IEEE Transactions on Knowledge and Data Engineering, vol. 18, no. 1, pp. 21-36, Jan. 2006, doi: 10.1109/TKDE.2006.10.
35. Dlala, Imen & Jabbour, Said & Badran, Raddaoui & Sais, Lakhdar. (2018). A Parallel SAT-Based Framework for Closed Frequent Itemsets Mining: 24th International Conference, CP 2018, Lille, France, August 27-31, 2018, Proceedings. 10.1007/978-3-319-98334-9_37.
36. Dlala, Imen & Jabbour, Said & Badran, Raddaoui & Sais, Lakhdar. (2018). A Parallel SAT-Based Framework for Closed Frequent Itemsets Mining: 24th International Conference, CP 2018, Lille, France, August 27-31, 2018, Proceedings. 10.1007/978-3-319-98334-9_37.
37. Hidouri A., Jabbour S., Raddaoui B., Yaghlane B.B. (2020) A SAT-Based Approach for Mining High Utility Itemsets from Transaction Databases. In Data Analytics and Knowledge Discovery. DAWAK 2020.
38. Niklas Eént Niklas Sörensson "An Extensible SAT-solver". In Proceedings of SAT, 2003, p. 502–518.