TPLP **24** (2): 313–343, 2024.

doi:10.1017/S1471068423000066 First published online 19 April 2023

Extended High-Utility Pattern Mining: An Answer Set Programming-Based Framework and Applications

FRANCESCO CAUTERUCCIO
DII, Polytechnic University of Marche, Ancona (Italy)
(*e-mail:* f.cauteruccio@univpm.it)

GIORGIO TERRACINA
DEMACS, University of Calabria, Rende (Italy)
(*e-mail:* terracina@mat.unical.it)

submitted 25 February 2022; revised 23 March 2023; accepted 28 March 2023

Abstract

Detecting sets of relevant patterns from a given dataset is an important challenge in data mining. The relevance of a pattern, also called utility in the literature, is a subjective measure and can be actually assessed from very different points of view. Rule-based languages like Answer Set Programming (ASP) seem well suited for specifying user-provided criteria to assess pattern utility in a form of constraints; moreover, declarativity of ASP allows for a very easy switch between several criteria in order to analyze the dataset from different points of view. In this paper, we make steps toward extending the notion of High-Utility Pattern Mining; in particular, we introduce a new framework that allows for new classes of utility criteria not considered in the previous literature. We also show how recent extensions of ASP with external functions can support a fast and effective encoding and testing of the new framework. To demonstrate the potential of the proposed framework, we exploit it as a building block for the definition of an innovative method for predicting ICU admission for COVID-19 patients. Finally, an extensive experimental activity demonstrates both from a quantitative and a qualitative point of view the effectiveness of the proposed approach.

KEYWORDS: high-utility pattern mining, answer set programming, facets, advanced utility functions

1 Introduction

Pattern mining is one of the data mining branches that attracted vast attention in the literature. Pattern mining algorithms extract human-understandable knowledge from databases and have fostered a growing interest in the development of scalable and flexible methods for data analysis. In this context, flexibility is intended as the ability to incorporate users' prior knowledge and multiple criteria to measure the relevance of a pattern in the analysis process. These criteria can be modeled in the form of constraints to validate a set of candidate patterns. The first and most common studied constraint is the frequency threshold, where a pattern is validated only if it appears sufficiently often (Agarwal and Srikant 1994). Frequent pattern mining is a fairly well-studied problem,

and effective solutions are available also in the logic programming arena (Järvisalo 2011; Gebser *et al.* 2016; Guyet *et al.* 2014).

However, frequency alone may be of little interest in many cases. As an example, consider a sales database and the pattern {*windshield washer fluid*, *new windshield wipers*}; this pattern can be very frequent but uninteresting, as it represents a common purchase. But what if we also consider the price of the products and a constraint on the minimum profit given by the corresponding purchase pattern? In this case, the pattern {*car*, *car alarm*} might be more interesting than the previous one, even if less frequent.

In light of this last consideration, the academic community has begun to emphasize that pattern validity can be assessed according to *utility functions* (Fournier-Viger *et al.* 2019; Gan *et al.* 2019); the introduction of the notion of the utility of an item and, more generally, of a pattern made it possible to develop a new generation of pattern mining approaches called High-Utility Pattern Mining – HUPM (Fournier-Viger *et al.* 2019; Gan *et al.* 2019; Yao *et al.* 2006).

However, the basic assumption of HUPM is that each item is associated with *one*, *static*, external utility; this limits the flexibility expected from modern data analysis processes. For instance, continuing the example on the sales database, it would not be possible to devise validation constraints that dynamically combine price and minimum packaging size of the various products composing the pattern in the utility calculation; this could be very valuable, for example, in logistics optimization applications. Generalizing, it would be really important to be able to combine, in a flexible way, different aspects of the items, which we call *facets* in the following, to compute patterns' utility.

Another limitation of HUPM lies in the fact that the expected input is a *flat* representation of transactions; this allows only local utility notions to be defined. On the contrary, a multi-layered representation of the data, coupled with the possibility of combining different facets in utility functions, may allow more advanced pattern constraints to be defined. As an example, by grouping purchases by customer and considering the degree of each customer satisfaction as a facet, one can identify patterns with a good correlation between customer purchases and their satisfaction. This is not possible with classical HUPM methods.

Finally, it can be useful to incorporate users' prior knowledge into pattern constraints, in the form of *pattern masks*, to assess pattern utility. For instance, it may be useful to state that only certain combinations of product categories are relevant for the analysis. Again, this is not currently possible in the classical HUPM setting.

A first contribution of the present paper is the definition of a framework, which we call e-HUPM, that generalizes the classical HUPM in the following directions:

- *Dataset representation.* A multi-layered representation of the input is defined, where aggregation levels of transactions, objects, and containers are conventionally identified.
- *Facets.* We introduce the notion of facet, which can be associated with an item, a transaction, an object or a container; each of these elements may be characterized by more than one facet.
- *Utility functions.* In order to make the most of facets and layers, we introduce a taxonomy of utility functions classes, most of which are not allowed in classical HUPM.

https://doi.org/10.1017/S1471068423000066 Published online by Cambridge University Press

- *Pattern masks*. We introduce the notion of pattern mask, in order to specify structural or semantic constraints patterns must comply with.

To solve the pattern mining problem introduced above, a guess-and-check resolution scheme, with the support of rule-based languages such as Answer Set Programming (ASP), seems to be quite intuitive and effective. In particular, we have seen that solutions for frequent pattern mining are already available in ASP (Järvisalo 2011; Gebser *et al.* 2016; Guyet *et al.* 2014); however, the constraints we are concerned with in this work to validate patterns come in the form of potentially complex functions, which are difficult, or even impossible, to express in ASP. In order to solve this issue, we resort to recent extensions of ASP systems, such as DLVHEX (Eiter *et al.* 2016; 2018), WASP (Dodaro and Ricca 2020), clingo (Gebser *et al.* 2019), etc., which allow external calculation functions, usually written in Python, to be integrated into ASP programs.

A second contribution of the present work is a modular ASP encoding of the proposed framework, so that, given a pool of alternative encodings for each module, even those unfamiliar with ASP can set up their own variants in a way that provides a high degree of flexibility for data analysis.

Finally, as a third contribution, we exploit the proposed framework and the corresponding ASP-based solution as a building block for the definition of an innovative method for predicting the ICU admission for COVID-19 patients, based on patients' blood results and vital signs.

The remainder of the paper is organized as follows. In Section 2, the general framework is proposed, along with all of its components and the problem definition. Section 3 is devoted to the design of the ASP solution. The application of our framework to a biomedical context is presented in Section 4, whereas an extensive experimental evaluation is presented in Section 5. In Section 6, a broad overview of related work is presented. Finally, in Section 7 we draw our conclusions and highlight future research directions.

2 A general framework for extending high-utility pattern mining (e-HUPM)

In this section, we present our framework. First, we briefly recall the classical background definitions related to the HUPM problem. Then, we show how we extend the classical problem in several ways. In particular, we first extend the concept of *Transaction database* with *Containers* and *Objects*; then, we extend the concept of utility with the notion of *facet*. After this, we introduce a new classification of *pattern utility* functions and the notion of *pattern mask*. Finally, we provide a formal definition of the problem addressed in this work.

2.1 Background

We now briefly summarize the classical definitions and notations for the HUPM problem based on the ones presented in Fournier-Viger *et al.* (2019). It is worth pointing out that several variants of this problem have been proposed in the literature; since it is out of the scope of the paper covering all of them here, we focus on the most classical one.

A quantitative transaction database D is composed of a set of transactions and denoted as $D = \{T_1, T_2, \ldots, T_n\}$. Each transaction T_p is uniquely identified by a transaction

https://doi.org/10.1017/S1471068423000066 Published online by Cambridge University Press

identifier (tid_p) and contains a set of items I_p where each $I_p \subseteq I = \{i_1, i_2, \ldots, i_m\}$. Each item $i \in I$ is associated with an *external utility* $eu(i)$, and every occurrence of i in a transaction T_p is associated with an *internal utility* $iu(i, T_p)$ which generally represents the quantity of i in T_p. In the classical definition of HUPM, both external and internal utilities are positive numbers. The objective of HUPM is the identification of sets of items (patterns) that present a high utility, that is, a utility higher than a certain threshold th_u.

The utility of an item i in a transaction T_p is obtained as $eu(i) \times iu(i, T_p)$. Given a pattern P appearing in a transaction T_p, the utility of P in T_p is denoted as $tu(P, T_p)$ and is computed as $\sum_{i \in P} eu(i) \times iu(i, T_p)$.

Given the set T_P of transactions containing occurrences of the pattern P, the utility of P in the database D is denoted by $u(P)$ and computed as $u(P) = \sum_{T_p \in T_P} tu(P, T_p)$.

Problem definition. Given a pattern P, P is a high-utility pattern if its utility $u(P)$ computed over its occurrences in D is greater than a utility threshold th_u. The problem of high-utility pattern mining is to discover all high-utility patterns in a database D.

2.2 Databases, containers and objects

The first extension we propose with respect to the classical HUPM relates to the database representation. In the classical context, a database is simply composed of a flat set of transactions. Here we assume that a database of transactions is organized in different abstraction layers. This allows us to include more sophisticated utility functions in order to analyze, as an example, correlations among different transactions at different abstraction levels. Here we propose a possible hierarchy for the database, which fixes some boundaries for the definition of the problem; clearly, more general structures and more layers can be defined in different contexts. The proposed reference database structure is as follows:

$$\textit{Database} \rightarrow \textit{Container} \rightarrow \textit{Object} \rightarrow \textit{Transaction}$$

In particular, given a database D and a set of transactions $\{T_1, T_2, \ldots, T_n\}$, D is organized as a set of *containers* $C = \{C_1, C_2, \ldots, C_r\}$ where each container C_s can be associated with a set of *objects* $O = \{O_1, O_2, \ldots, O_t\}$ and each object O_u contains a set of transactions $\{T_1, T_2, \ldots, T_v\}$. Clearly, each transaction is composed of a (possibly ordered) set of items. Each transaction belongs to precisely one object, and each object is associated with precisely one container. Observe that this representation allows covering several interesting application scenarios.

Example 1. ***(Running example)*** In order to better describe our framework, let us introduce a running example concerning scientific reviews, which will be explored further in the experiments. This context presents several peculiarities captured by our framework. In particular, we build upon the paper reviews use case presented in Chakraborty *et al.* (2020); it deals with papers, reviews, and annotated sentences. Each paper is a container in our model; for each paper, several reviews are available: each review is an object in our model. Each review contains a list of sentences: each sentence is a transaction in our model.

□

https://doi.org/10.1017/S1471068423000066 Published online by Cambridge University Press

Table 1. *Terminology and facets for the paper reviews use case running example introduced in Example 1*

e-HUPM	Use case	Facets	Domain of the facet
Database	Set of reviews	–	
Container	Paper	(Decision)	({0 (Reject), 1 (Accept)})
Object	Review	(Rating, Confidence)	({1–10}, {1–5})
Transaction	Sentence	(Appropriateness, Clarity, Originality, Soundness, Comparison, Substance, Impact, Recommendation)	{−1 (Negative), 1 (Positive), 0 (Neutral or Absent) }
Item	Word	–	

2.3 Utilities and facets

The second main generalization deals with utility representation. As a matter of fact, a great limit of the classical definition of HUPM, and its variants, is that each item is associated with a unique, external, and fixed value of utility. We next extend the notion of utility with the concept of *facet*. In fact, in several contexts, the utility of an item can be defined from different perspectives, which we call facets. Then, each item can be associated with a list of values defining its utility from different perspectives. Moreover, given the new organization of the database, facets can be defined also for transactions, objects, and containers. Formally:

Item utility vector. Given an item i, the utility of i is defined by an *item utility vector* $IU_i = [iu_1, iu_2, \ldots, iu_l]$, where each iu_k describes a certain facet of i.

Transaction utility vector. Given a transaction T_p, the utility of T_p is defined by a *transaction utility vector* $TU_{T_p} = [tu_1, tu_2, \ldots, tu_m]$, where each tu_k describes a facet of T_p. Observe that these facets for the transaction describe properties of the transaction as a whole and represent a different information than the standard internal utility of an item i in the transaction T_p. In order to keep the compatibility with the classical problem, we assume that the internal utility of i in T_p is available and represented as $q(i, T_p)$.

Object utility vector. Given an object O_u, the utility of O_u is defined by an *object utility vector* $OU_{O_u} = [ou_1, ou_2, \ldots, ou_n]$, where each ou_k describes a facet of O_u.

Container utility vector. Given a container C_s, the utility of C_s is defined by a vector $CU_{C_s} = [cu_1, cu_2, \ldots, cu_o]$, where each cu_k describes a facet of C_s.

It is worth observing that the length of any of the above utility vectors could be 0 if no interesting facet can be defined for that vector. The list of facets is fixed at problem modeling stage; however, we assume that all utility vectors of a certain type have the same number of facets. All utilities introduced above are not constrained to be numeric values. The interpretation and combination of utilities is left to the pattern utility computation function, introduced in the next section.

Example 2. ***(Running example)*** Table 1 illustrates the correspondence between our e-HUPM framework, and the paper reviews use case adopted from Chakraborty *et al.* (2020). The table depicts the selected facets for each domain element and its representation in the database, for example, facets for containers, objects, and transactions; the database and items have no associated facets. Each container represents a paper, and for a container C_s, we have one single facet, that is the decision. Reviews information

https://doi.org/10.1017/S1471068423000066 Published online by Cambridge University Press

Table 2. *Examples of container, object, and transaction utility vectors for the paper reviews use case running example*

Database elements	Utility vectors	Database elements	Utility vectors
Containers (papers)		Transaction (reviews' sentence)	
C_1	$CU_{C_1} = [0]$	S_1	$TU_{S_1} = [-1, -1, 0, 1, 0, 0, 0, -1]$
C_2	$CU_{C_2} = [1]$	S_2	$TU_{S_2} = [1, 0, 1, 0, -1, -1, 0, 0]$
Object (reviews)		S_3	$TU_{S_3} = [1, 1, 1, 0, 0, 0, 0, 1]$
R_1	$OU_{R_1} = [2, 4]$	S_4	$TU_{S_4} = [0, 1, -1, 1, 1, 0, 1, 1]$
R_2	$OU_{R_2} = [4, 3]$		
R_3	$OU_{R_3} = [9, 3]$		

Table 3. *An excerpt of the transactions contained in the paper reviews use case, along with some objects and containers, used in the running example*

Container	Object	TID	Transaction
C_1	R_1	S_1	{(paper, 2), (hard, 1), (narrow, 1)}
	R_2	S_2	{(problem, 1), (paper, 1), (concern, 1), (reproducibility, 1)}
C_2	R_3	S_3	{(paper, 1), (readable, 1)}
		S_4	{(paper, 1), (good, 1), (experiment, 1), (reproducibility, 1)}

are represented by objects and for an object O_u we have two facets, namely rating and confidence. Each review's sentence is a transaction, and for a transaction T_p, we have the facets corresponding to the eight annotated aspects defined in Chakraborty *et al.* (2020) and reported in Table 1. Finally, each sentence's word is an item. Table 2 depicts the utility vectors for container, object, and transaction respectively, relative to a simple instance excerpt adapted from the dataset; this excerpt is reported in Table 3 for completeness of presentation. In this table, for a better readability, we report the word corresponding to each item. We point out that a preprocessing pipeline is applied to the sentences with usual stemming and lemmatization procedures. Note that each transaction T_p is a set of pairs $(i, q(i, T_p))$ where i is an item and $q(i, T_p)$ its quantity in the transaction T_p. □

2.4 Pattern utility computation

Recall that a pattern is a (possibly ordered) set of items, and it may occur in a certain number of transactions. The generalization of utilities in facets and the structuring of the database at different abstraction levels pave the way to more advanced and diversified computations of utilities.

Intra-pattern utility. First of all, given a pattern P, composed of a set of r items and occurring in a transaction T_p, all the item utility vectors of items $i \in P$ must be merged into a unique item utility vector IU. In this process, internal utilities of items for T_p can be taken into account. Formally, given the set $IUS = \{IU_1, \ldots IU_r\}$ of item utility vectors associated with each item $i \in P$, let's define the *intra-pattern* utility function ip

https://doi.org/10.1017/S1471068423000066 Published online by Cambridge University Press

which takes as input the pattern P, the transaction T_p and the associated set of item utility vectors IUS, and generates a unique item utility vector for the pattern occurrence:

$$IU_{T_p} = ip(P, T_p, IUS).$$

Example 3. An example of *intra-pattern* utility function is the following:

$$IU_{T_p} = ip(P, T_p, \{IU_1, \dots IU_r\}) =$$

$$[\sum_{i \in [1..r]} (IU_i[1] \times q(i, T_p)), \sum_{i \in [1..r]} (IU_i[2] \times q(i, T_p)), \dots, \sum_{i \in [1..r]} (IU_i[l] \times q(i, T_p))].$$

Depending on the context of interest, the combination of the utilities across the single facets can be carried out with functions different from the SUM. As an example, MAX, MIN, or AVG operators can be applied across the same facet of the different items in the pattern. Interestingly, if $r = 1$ it corresponds to the classical definition. □

Pattern utility functions. Now, given a pattern P occurring in a transaction T_p, the corresponding *occurrence utility vector* $OccU_{T_p}$ is obtained by juxtaposing the item, transaction, object, and container utility vectors:

$$OccU_{T_p} = [IU_{T_p}, TU_{T_p}, OU_{T_p}, CU_{T_p}] =$$

$$[iu_1, \dots, iu_l, tu_1, \dots, tu_m, ou_1, \dots, ou_n, cu_1, \dots, cu_o].$$

Here, for the sake of simplicity, we refer to OU_{T_p} (resp., CU_{T_p}) as the object (resp., container) utility vector of the object (resp., container) containing transaction T_p.

Given the set T_P of transactions containing occurrences of the pattern P, the *pattern utility vector* U_P is obtained as the collection of all the occurrence utility vectors of P:

$$U_P = \bigcup_{T_p \in T_P} OccU_{T_p}.$$

Example 4. ***(Running example)*** Let us continue the running example. Consider the pattern $P = \{\mathsf{paper}, \mathsf{reproducibility}\}$ and the set of transactions $T_P = \{S_2, S_4\}$ in which it occurs. To proceed with the pattern utility computation, we first should compute its intra-pattern utility by merging the item utility vectors into a unique item utility vector. However, in the paper reviews use case, items have no facets. Hence, a simple intra-pattern utility function which returns an empty list can be exploited, that is $IU_{S_2} = IU_{S_4} = []$. The corresponding occurrence utility vector $OccU_{S_2}$ and $OccU_{S_4}$ can now be derived as the juxtaposition of unique transaction, object and container utility vectors:

$$OccU_{S_2} = [IU_{S_2}, TU_{S_2}, OU_{R_2}, CU_{C_1}] = [1, 0, 1, 0, -1, -1, 0, 0, 4, 3, 0],$$
$$OccU_{S_4} = [IU_{S_4}, TU_{S_4}, OU_{R_3}, CU_{C_2}] = [0, 1, -1, 1, 1, 0, 1, 1, 9, 3, 1].$$

Then, the pattern utility vector U_P consists of the occurrence utility vectors $OccU_{S_2}$ and $OccU_{S_4}$, that is:

$$U_P = \{OccU_{S_2}, OccU_{S_4}\} = \{[1, 0, 1, 0, -1, -1, 0, 0, 4, 3, 0], [0, 1, -1, 1, 1, 0, 1, 1, 9, 3, 1]\}.$$

□

https://doi.org/10.1017/S1471068423000066 Published online by Cambridge University Press

Now, from U_P we can virtually build a matrix where each row represents a utility vector associated with an occurrence of P and each column represents a facet of P. The utility u of the pattern P can be then obtained as an arbitrary combination of the values in U_P, using a function $u(P)$.

In order to formalize function $u(P)$, we distinguish between formulas that operate *by row* (we call them *horizontal first* and we refer to them as f_h), formulas that operate *by column* (we call them *vertical first* and we refer them as f_v), and formulas that operate on the whole data at once (we call them *mixed* and we refer them as f).

Formally, utility of P can be classified in:

- *Horizontal first*; it first combines utilities of the various facets in each occurrence (by row) and then combines the values across all occurrences (by column): $u(P) = f_v(f_h(U_P))$
- *Vertical first*; it first evaluates utilities on a facet basis across the occurrences (by column) and then combines the obtained values across the facets (by row): $u(P) = f_h(f_v(U_P))$
- *Mixed*; it combines the values in U_P at once: $u(P) = f(U_P)$

Observe that $u(P)$ is a single number, whereas intermediate computations may provide sets of values.

Both *Horizontal first*, *Vertical first*, and *Mixed* utility functions may be further classified in:

- *inter-transaction* utility: functions that combine item and transaction utilities;
- *pattern-vs-object* utility: functions that compute the utility of the pattern by correlating one or more item or transaction utility facets with one or more object utility facets;
- *pattern-vs-container* utility: functions that correlate one or more item or transaction utility facets with one or more container utility facets.

The list of potentially interesting functions is virtually infinite. A non-exhaustive set of functions for some of the classes introduced above is provided next. Their specific semantics is strongly dependent on the application context.

- *Horizontal first – inter-transaction*:
  - — Filter & Sum: Filters one single facet first and then sums the obtained values[1].
  - — Filter & Times: Filters one single facet first and then multiplies the obtained values[2].
  - — Coherence degree: returns a predefined value if (a subset of) facets show coherent values and then computes the percentage of pattern occurrences containing this value; an index of coherence can be the fact that all facets are positive or all facets are negative.
- *Horizontal first – pattern-vs-object/pattern-vs-container*:

[1] Observe that this is equivalent to the classical HUPM scenario.
[2] Observe that, if the facet represents a probability, this can be used to compute a combined probability.

https://doi.org/10.1017/S1471068423000066 Published online by Cambridge University Press

— Disagreement degree: returns a predefined value if the value of an item/-transaction facet disagrees with that of an object/container facet and then computes the percentage of pattern occurrences containing this value.

- *Vertical first – inter-transaction*:
  - — Max & Sum: computes the maximum for each facet across transactions first and then sums the obtained values. To understand the applicability of this function, consider a context in which each facet is a rating of a certain aspect of a transaction (e.g. shipping speed, reliability, etc.); this function makes it possible to find the patterns with the highest overall ratings on all facets.
  - — Std & Max: computes the standard deviation of each facet across transactions first and then takes the maximum value among facets.
- *Vertical first – pattern-vs-object*:
  - — Mixed Coherence degree: for each facet, computes the fraction of transactions with positive (resp., negative) values, then filters on an item/transaction facet and on an object facet and multiplies the corresponding fractions.
- *Mixed pattern-vs-object / pattern-vs-container*:
  - — Pearson's Correlation (Pearson 1895): computes the correlation between one of the item/transaction facets and one of the object/container facets. It can be used to measure the agreement/disagreement about information at pattern/transaction level and object/container level.
  - — Multiple Correlation (Lewis-Beck *et al.* 2003): computes the correlation between two or more item/transaction facets and one of the object/container facets. It can be used to measure how well a given object/container facet can be predicted using a linear function of a set of item/transaction facets.

Example 5. ***(Running example)*** Let us consider the pattern utility vector U_P for the pattern $P = \{\mathsf{paper}, \mathsf{reproducibility}\}$ shown in Example 4. We now give examples for some of the classes of utility functions introduced above.

We start by giving a simple example for Horizontal and Vertical first functions. We consider an inter-transaction function involving Filter & Max. More in detail, let $f_h = \text{filter}(\cdot)$ and $f_v = \max(\cdot)$ be the Filter and Max functions, respectively. Then, we have $u(P) = f_v(f_h(U_P)))$. Assume we want to focus on the rating facet of each object utility vector; therefore, $u(P)$ consists in filtering the rating facet (by means of f_h) and then selecting the maximum value of such facet (by means of f_v). In detail, $f_h(U_P) = \text{filter}(U_P) = \{[4], [9]\}$, where [4] (resp., [9]) indicates the rating facet of $OccU_{S_2}$ (resp., $OccU_{S_4}$) represented as a singleton. Then, by applying f_v across each occurrence, we obtain $u(P) = f_v(f_h(U_P)) = \max(\text{filter}(U_P)) = \max(\{[4], [9]\}) = 9$. In this case, the same result is obtained by applying these functions in a vertical first fashion, for example, $u(P) = f_h(f_v(U_P))$.

Instead, if we consider a mixed pattern-vs-object utility function, an interesting example is the computation of the Pearson correlation coefficient. Let us define $f = r_{xy}$, where r_{xy} is the classic Pearson correlation coefficient. Also, suppose that x indicates the Clarity aspect value present in the sentence and y indicates the rating facet of the review, across

https://doi.org/10.1017/S1471068423000066 Published online by Cambridge University Press

all occurrence utility vectors in U_P. According to U_P, we have $x = [0, 1]$ and $y = [4, 9]$, and by applying the pattern-vs-object utility function f we obtain $f(U_P) = r_{xy} = 1.0$, indicating a perfect positive correlation between Clarity aspect and the rating facets for the pattern $P = \{\mathsf{paper}, \mathsf{reproducibility}\}$. □

2.5 Pattern masks

As a final extension, we introduce *pattern masks*, that is, properties that patterns (resp., pattern occurrences) must satisfy in order to be considered valid patterns (resp., occurrences). Simple examples of pattern masks are given below. Clearly, other kinds of masks can be defined, depending on the context of interest.

- Mask on pattern size: it constrains the number of items in a pattern into a given range.
- Mask on pattern (external) properties: it constrains items in the pattern to satisfy certain properties. As an example, if items are words from review sentences, each word can be associated with a Part-of-Speech (POS) tag, and it can be interesting to constrain each pattern to contain at least a *noun*, a *verb*, and an *adjective*.

2.6 Definition of extended high-utility patterns

In the classical problem, a high-utility pattern is detected disregarding its support on the database focusing on its utility only. In the new setting we are proposing in this work, we must consider that there are utility functions which may have low significance in presence of few transactions supporting the pattern. As an example, in order to properly compute the Pearson correlation at least two, but preferably more, data points are necessary. As a consequence, in order to provide a framework as general as possible, our problem formulation includes the definition of a minimum support for the pattern, in order to detect high-utility patterns; obviously, this threshold can be set to 1, meaning that a pattern should appear at least once, whenever a minimum support is not relevant.

Problem definition. Given a pattern P containing a set of items, and a pattern mask M, P is an extended high-utility pattern if P and its occurrences satisfy M, its utility $u(P)$ is greater than a utility threshold th_u, and it occurs in at least th_f transactions. The problem of extended high-utility pattern mining is to discover all extended high-utility patterns in a database D.

3 Design of the ASP approach

As previously pointed out, one of the main objectives of this work is to provide as much flexibility as possible in the definition of what is a valid pattern. In what follows, we provide an encoding as general as possible for the framework introduced in Section 2; this can be specialized for specific settings, and, indeed, we also show how it can be specialized for the paper reviews use case introduced as running example in Section 2.

It is important to point out again that coding complex formulae for pattern utility, such as some of those described in Section 2, may in general not be easy (or even impossible)

https://doi.org/10.1017/S1471068423000066 Published online by Cambridge University Press

to achieve with a purely declarative approach; think, for instance, about the computation of the standard deviation or the multiple correlation coefficient. Similar difficulties may arise when it is necessary to handle real or negative numbers. To overcome this issue, we resort to recent extensions of ASP systems, such as DLVHEX (Eiter *et al.* 2016; 2018), clingo (Gebser *et al.* 2019), and WASP (Dodaro and Ricca 2020), which allow the integration of external computation sources, usually written in Python, within the ASP program. The ASP standardization group has not yet released standard language directives for such features; in order to present our ASP formalization in this section, we refer to syntax and semantics of DLVHEX (Eiter *et al.* 2016), which is more intuitive. In Section 4, we also show a complete implementation in WASP (Dodaro and Ricca 2020) that exploits constraint propagators.

The general encoding is presented in Listing 1; it is structured in separate parts, in such a way that the various aspects of the problem can be modeled, and changed, with local modifications.

The first part (lines 1–9) recalls the expected schema for input facts to simplify the reading of the code. Observe that attributes `Position` and `Q` for predicate `item` will be needed only if item position within the transaction, and its internal utility are actually needed for the problem at hand (see Section 2.3). Similarly, the number of facets for item, transaction, object, and container utility vectors will be specific to the application scenario; the corresponding predicates may be omitted if no facets are available for them. Thresholds for pattern frequency (th_f) and utility (th_u) must be provided as input as well with facts `occurrencesThreshold` and `utilityThreshold` (line 11).

Rule prototype in line 13 allows items to be pre-filtered, according to users' background knowledge. Think for instance to items with a price lower than a certain threshold which are certainly not relevant for the analysis. Such filters can be very application-specific, and we therefore leave this part open to user specification. Predicate `usefulItem` is true only for unfiltered items.

The second part of the formulation considers the generation of candidate patterns and the verification of their minimum frequency. The chosen encoding generates one answer set for each pattern; this simplifies both pattern representation and the application of user-defined constraints to candidate patterns. As pointed out in Section 2.6, in the classical HUPM context minimum support is not considered at all as a constraint on pattern validity (only utility is actually considered); in this setting, all combinations of items would be candidate patterns. However, we decided to keep the frequency constraint in order to cope with specific application scenarios. To identify the set of candidate patterns with a minimum support, we build upon the effective and elegant solution already presented in Järvisalo (2011), which has been adapted to our context (lines 15-18). Here, the predicate `inCandidatePattern` is true for an item i only if i is a useful item and belongs to a frequent pattern. The predicate `inSupport` is true for a transaction t if it supports all the items in the candidate pattern. The lack of support for some items is determined by the `conflictAt` predicate. We also use the support predicate `contains` to project the first two variables of `item`. Observe that minimum frequency is checked directly in the choice rule (line 15).

The third part of the formulation (lines 20–21) generates the unique item utility vector for the pattern occurrence; in particular, predicate `patternItemUtilityVectors` is true for each item i in the candidate pattern supported by a transaction t, and collects the

https://doi.org/10.1017/S1471068423000066 Published online by Cambridge University Press

corresponding facets and internal utility. Predicate `intraPatternUtilityVector` then instantiates the unique item utility vector for the corresponding transaction by the application of the external function `computeIntraPatternUtility`. Observe that, generally, this computation involves simple sum-product operations; in this case, it might be more convenient to express the formula directly within the rule. Moreover, we point out that, when items have no facets and internal utility, this part of the encoding can be omitted.

The fourth part of the formulation (lines 23–25) builds the occurrence utility vector (line 23) and checks utility criteria on candidate patterns (line 25). Here, the different database layers introduced by the framework are taken into account and the full power of external functions is exploited. Observe that when using WASP constraint propagators, the constraint expressed in line 25 is replaced by a propagator.

Finally, the last part of the formulation (lines 27–29) is devoted to express pattern masks; this part is very application-specific, and we show a mask on pattern size just as a simple example. Here, users' background knowledge may play a relevant part. As an example, if items are words and we know for each word its POS tag, a more elaborated mask can be defined requiring that a pattern must contain at least a noun, a verb, and an adjective.

Listing 1. *A general ASP encoding for the e-HUPM problem.*

```
1  %%% Input schema:
2  % container(ContainerId)
3  % object(ObjectId, ContainerId)
4  % transaction(Tid, ObjectId)
5  % item(Item, Tid, Position, Q)
6  % itemUtilityVector(Item, I1, ..., Il)
7  % transactionUtilityVector(Tid, T1, ..., Tm)
8  % objectUtilityVector(ObjectId, O1, ..., On)
9  % containerUtilityVector(ContainerId, C1, ..., Co)
10 %%% Parameters
11 occurrencesThreshold(...). utilityTreshold(...).
12 %%% Item pre-filtering
13 usefulItem(I):- item(I,_,_,_),....any condition on the items.
14 %%% Candidate pattern generation and check on minimum support
15 {inCandidatePattern(I)}:- usefulItem(I), #count{Tid : inSupport(Tid), contains(I,Tid)}=N, N >= Tho,
       occurrencesThreshold(Tho).
16 inSupport(Tid):- transaction(Tid,_), #count{I : usefulItem(I), conflictAt(Tid,I)}=0.
17 conflictAt(Tid,I):- inCandidatePattern(I), transaction(Tid,_), not contains(I,Tid).
18 contains(I,Tid):- item(I,Tid,_,_).
19 %%% Intra-pattern Utility computation (needed only if items have utilities)
20 patternItemUtilityVectors(Tid,Item,I1,...,Il,Q):- inCandidatePattern(Item), itemUtilityVector(Item,I1
       ,...,Il), inSupport(Tid), item(Item,Tid,Position,Q).
21 intraPatternUtilityVector(Tid,I1,...,Il):- &computeIntraPatternUtility[patternItemUtilityVectors](Tid,
       I1,...,Il).
22 %%% Pattern Utility computation
23 occurrenceUtilityVector(Tid,I1,...,Il,T1,...Tm,O1,...On,C1,...,Co):- inSupport(Tid),
       intraPatternUtilityVector(Tid,I1,...,Il), transactionUtilityVector(Tid,T1,...,Tm), transaction(
       Tid, ObjectId), objectUtilityVector(ObjectId,O1,...,On), object(ObjectId, ContainerId),
       containerUtilityVector(ContainerId,C1,...,Co).
24 %%% The following constraint is implemented as a constraint propagator in WASP
25 :- &computeUtility[occurrenceUtilityVector](U), U < Thu, utilityTreshold(Thu).
26 %%% Pattern mask (e.g.) on size
27 minLength(...). maxLength(...).
28 :- #count{T : inCandidatePattern(T)} < L, minLength(L).
29 :- #count{T : inCandidatePattern(T)} > L, maxLength(L).
```

The general ASP encoding presented in Listing 1 can be specialized in different ways depending on the application context and on the focus of the analysis. In what follows, we show a complete formulation of the proposed approach for the paper reviews use case

https://doi.org/10.1017/S1471068423000066 Published online by Cambridge University Press

introduced as running example in Section 2; in particular, we show how the various parts of the general program described above, which we call *Blocks* in the following to simplify the presentation, can be modeled by different ASP code portions that can be combined in many different ways. In our opinion, this makes evident the flexibility provided by our hybrid solution combining ASP and Python, where the main ability of declarative programming, which allows programmers to define the what and not the how, is fully exploited in the definition and combination of the various Blocks, whereas the potential of Python to switch between different complex utility functions is exploited by external functions.

Figure 1 shows the proposed formulations, where the diverse Blocks are numbered in order to simplify the presentation. In particular, Block 1 simply defines the schema of the input dataset, along with expected input parameters. Observe that, for the paper reviews use case, no utility is associated with items; consequently, there is no `itemUtilityVector` and, thus, no intra-pattern utility computation is needed. For this reason, the proposed formulation does not include a Block for intra-pattern utility computation.

As far as the pre-filtering part is concerned, Block 2a and Block 2b show two different alternatives. In the code of Block 2a, only words associated with at least one positive or negative sentiment value for the sentences they appear in are considered useful for the analysis; Block 2b shows a pre-filtering that includes only words for which a disagreement between Appropriateness and Decision exists. Obviously, many other pre-filtering rules can be defined for the problem at hand.

Block 3a and Block 3b refer to the candidate pattern generation and occurrences check part. In particular, Block 3a reproduces the code already described in the general encoding, whereas Block 3b shows how easy it can be with ASP to switch from the classical scenario to a very different problem, in this case sequence pattern mining. In particular, in this formulation, the classical code is slightly modified in such a way that the order of items within sentences becomes relevant, and an occurrence of an item in a sentence is validated only if all items occur precisely in the same order. It is interesting to observe that the switch of this Block alone makes it possible to tackle very different pattern mining problems and that switching from a pattern mining problem to another one with an imperative programming solution would require to rewrite long code portions.

Pattern utility computation, coded in Block 4, is standard; in this case, the different potential utility functions are coded in Python by the external function `computeUtility`.

Finally, Block 5a and Block 5b express two different ways of defining pattern masks. Block 5a defines a simple mask on pattern size, whereas Block 5b implements a more elaborated mask where, if we assume to know the POS tag of each word, it states that if a pattern contains at least three words, these must be at least a noun, a verb, and an adjective. Of course, again, one can imagine many other types of masks.

As a final consideration, it is noteworthy that even with a limited number of alternatives for each block, a wide variety of final encodings can be generated. As an example, the combination of Blocks 1-2a-3a-4-5a represents a full encoding; other possibilities are Blocks 1-2b-3a-4-5b or Blocks 1-2a-3b-4-5a, and many more. Given a pool of alternative encodings for each Block, even those unfamiliar with ASP could put up their own variants to provide maximum flexibility in data analysis.

https://doi.org/10.1017/S1471068423000066 Published online by Cambridge University Press

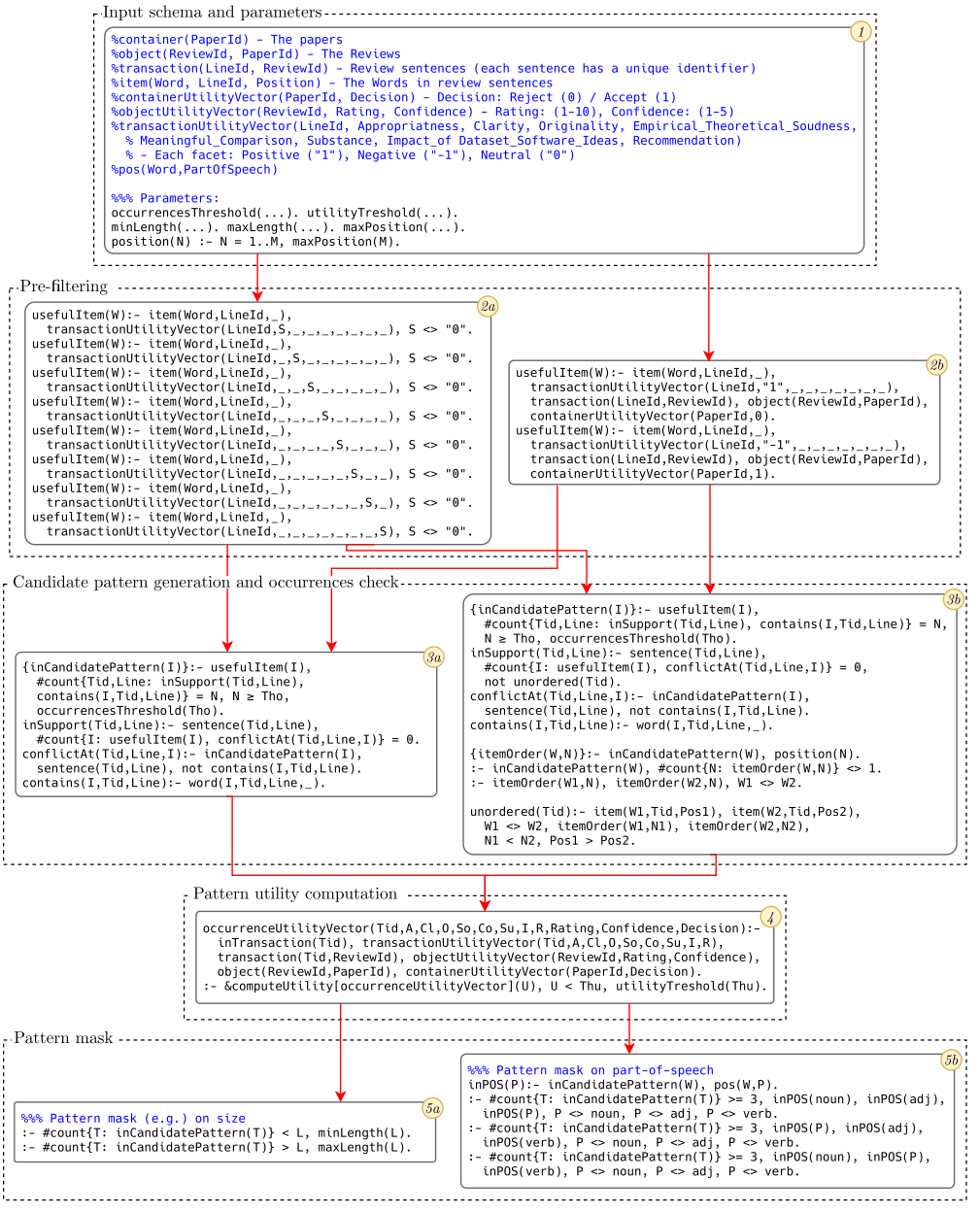

Fig. 1. Modular composition of ASP subprograms for the paper reviews use case introduced as running example.

4 Application to ICU admission prediction for COVID-19 patients

In this section, we present a possible application of the proposed approach in a biomedical context; in particular, we show how high-utility patterns derived from a structured database can be exploited to compute correlations, and possibly make predictions, for ICU admission of confirmed COVID-19 cases, based on patients' blood results and vital signs. It is worth pointing out that the main objective of this section is not to provide

https://doi.org/10.1017/S1471068423000066 Published online by Cambridge University Press

a final solution to the problem, but to show with a proof-of-concept that our e-HUPM framework can actually help in relevant practical problems.

To this purpose, we exploit the COVID-19 dataset publicly available at https://www.kaggle.com/S%C3%ADrio-Libanes/covid19. The dataset contains almost 2000 anonymized data for confirmed COVID-19 patients; each patient in the database underwent several encounters. The dataset contains the following information: *(i)* patient demographic information, *(ii)* patient previous grouped diseases, *(iii)* if the patient was admitted to the ICU during the observation period (ICU = 1) or not (ICU=0), and, for each encounter, *(iv)* 36 parameters for blood results, and *(v)* 6 parameters for vital signs. With respect to our e-HUPM model, a patient is an object, having a single facet, namely ICU; each visit is a transaction whose items are categorical data from *(i)* and *(ii)* and whose 42 facets are numerical data from *(iv)* and *(v)*. No facets and internal utility are defined for items; there is no container layer for the dataset.

Listing 2. *A full example of the ASP encoding for the ICU admission prediction for COVID-19 patients.*

```
1  %%% Input schema:
2  % object(PatientId)
3  % objectUtilityVector(PatientId, ICU)
4  % transaction(VisitId, PatientId)
5  % transactionUtilityVector(VisitId, ALBUMIN_MEAN, BE_ARTERIAL_MEAN, BE_VENOUS_MEAN, BIC_ARTERIAL_MEAN,
       BIC_VENOUS_MEAN, BILLIRUBIN_MEAN, BLAST_MEAN, CALCIUM_MEAN, CREATININ_MEAN, FFA_MEAN, GGT_MEAN,
       GLUCOSE_MEAN, HEMATOCRITE_MEAN, HEMOGLOBIN_MEAN, INR_MEAN, LACTATE_MEAN, LEUKOCYTES_MEAN,
       LINFOCITOS_MEAN, NEUTROPHILES_MEAN, P02_ARTERIAL_MEAN, P02_VENOUS_MEAN, PC02_ARTERIAL_MEAN,
       PCO2_VENOUS_MEAN, PCR_MEAN, PH_ARTERIAL_MEAN, PH_VENOUS_MEAN, PLATELETS_MEAN, POTASSIUM_MEAN,
       SAT02_ARTERIAL_MEAN, SAT02_VENOUS_MEAN, SODIUM_MEAN, TGO_MEAN, TGP_MEAN, TTPA_MEAN, UREA_MEAN,
       DIMER_MEAN, BLOODPRESSURE_DIASTOLIC_MEAN, BLOODPRESSURE_SISTOLIC_MEAN, HEART_RATE_MEAN,
       RESPIRATORY_RATE_MEAN, TEMPERATURE_MEAN, OXYGEN_SATURATION_MEAN)
6  % item(VisitId, Value)
7  %%% Parameters
8  occurrencesThreshold(...). utilityThreshold(...). maxCardItemset(...).
9  %%% Pre-filtering: all items are useful
10 usefulItem(I):- item(_,I).
11 %%% We compute the Pearson correlation between each target facet and ICU. Each run differs only for
12 %%% the projected facet and will detect patterns with a high correlation between that facet and ICU.
13 transactionUtilityVectorP(Tid,T):- transactionUtilityVector(Tid, _, _, _, _, _, T, _, _, _, _, _, _, _,
       _, _, _, _, _, _, _, _, _, _, _, _, _, _, _, _, _, _, _, _, _, _, _, _, _, _, _, _, _, _, _, _, _, _).
14 %% Candidate pattern generation and check on minimum support
15 {inCandidatePattern(I)}:- usefulItem(I), #count{Tid: inSupport(Tid), contains(I,Tid)}=N, N>=Tho,
       occurrencesThreshold(Tho).
16 inSupport(Tid):- transaction(Tid,_), #count{I: usefulItem(I), conflictAt(Tid,I)}=0.
17 conflictAt(Tid,I):- inCandidatePattern(I), transaction(Tid,_), not item(Tid,I).
18 %% The utility of the whole occurrence is the juxstaposition of all the required values
19 occurrenceUtilityVector(Tid,ICU,Target):- inSupport(Tid), transaction(Tid,PatientId),
20     objectUtilityVector(PatientId,ICU), transactionUtilityVectorP(Tid,Target).
21 %% The following is implemented as a constraint propagator in WASP and must be commented here
22 %:- &computeUtility[occurrenceUtilityVector](U), U < Thu, utilityThreshold(Thu).
23 %% size of each pattern is at most M
24 cardItemset(N):- #count{I : inCandidatePattern(I)} = N.
25 :- cardItemset(N), maxCardItemset(M), N > M.
26 :- cardItemset(N), N < 1.
```

To give a full example of application of our framework, Listing 2 reports the ASP program specialized to this context, where WASP propagators are used to implement the constraint in line 22. The Listing 3 shows an excerpt of the propagator implementation; here, the most important thing to note is the simplicity with which the Pearson correlation can be calculated (see line 13 of Listing 3). It is beyond the scope of this paper to explain all the details of the implementation of propagators. The interested reader is

https://doi.org/10.1017/S1471068423000066 Published online by Cambridge University Press

Listing 3. *The core Python function of the constraint propagator exploited in WASP.*

```
1  def compute(answer_set):
2      global aspvars, threshold, icu_values, target_values
3      icu_values, target_values = [], []
4      for x in answer_set:
5          if x < 0: continue
6          if x not in aspvars: continue
7
8          (_, icu, target) = aspvars[x]
9          icu_values.append(icu)
10         target_values.append(target)
11
12     # Compute Pearson via scipy library
13     pearson_value = stats.pearsonr(icu_values, target_values)[0]
14
15     # Check if computed value is valid
16     sat = abs(pearson_value) >= threshold
17     return (pearson_value, not sat)
```

referred to (Dodaro and Ricca 2020) for the literature on propagators and to https://www.mat.unical.it/~cauteruccio/tplp-ehupm/ for the complete implementation of propagators for this problem.

The first objective of this analysis was to find patterns, derived from patient demographic information and patient disease groups, showing a significant Pearson's correlation between each of the 42 facets and ICU outcome. As an example, for the pattern: (Gender:Male, AgePercentile:60, DiseaseGroup5:YES) we found a Pearson correlation of -0.75 between Oxygen Saturation values and ICU; similarly, for the pattern (Gender:Female, Immunocompromised:NO, DiseaseGroup5:YES, DiseaseGroup6:NO, Hypertension:YES) we found a Pearson correlation of -0.82 between Hemoglobin values and ICU. More generally, we found 590 (resp., 1422) patterns showing an absolute value of the Pearson correlation greater than 0.5 between Oxygen Saturation (resp., Hemoglobin) values and ICU occurring in at least 10 transactions in the dataset. Similar number of patterns have been derived also for the other facets. Note that, to switch from one facet to another, it was sufficient to change the attribute to be projected in line 13 of Listing 2.

As a preliminary analysis of obtained results, in light of exploiting them for ICU prediction, we analyzed the fraction of transactions in the database supported by at least one valid pattern, and the percentage of combinations of patients attributes covered by at least one valid pattern. This last has been measured as the fraction of combinations that can be extracted from available data which are also supported by at least one valid pattern. Observe that, having a high fraction of covered transactions (resp., combinations) might be important in order to improve the chances to have a personalized predictive model for each upcoming unknown patient. However, setting very low minimum thresholds just to increase the number of valid patterns may, in principle, be counterproductive for their subsequent use in prediction tasks. It is then important to study these properties for a better understanding of the next steps of analysis. Results obtained from this analysis are shown in Figure 2 for several values of minimum occurrence and minimum correlation thresholds. From the analysis of this figure, we can observe that even a slight increase of the minimum correlation has a strong impact on the fraction of transactions/-combinations covered by valid patterns; an increase of the minimum occurrence, instead, has a lower impact. Interestingly, too stringent thresholds end up to completely miss all transactions/combinations in the dataset and, clearly, these should be avoided. As

https://doi.org/10.1017/S1471068423000066 Published online by Cambridge University Press

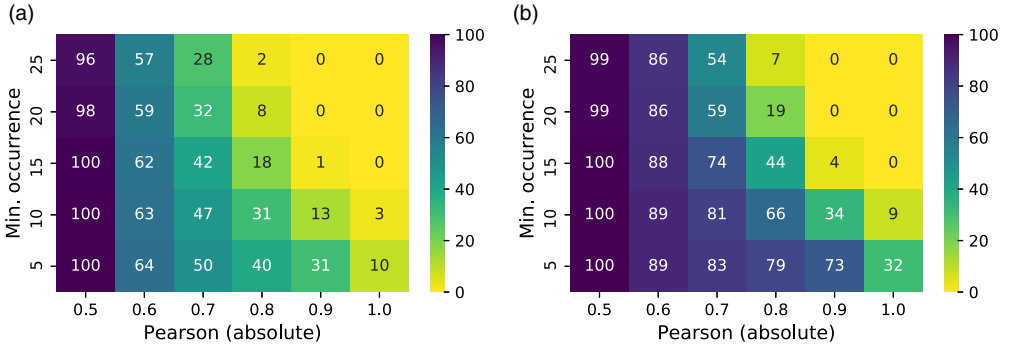

Fig. 2. Percentage of transactions (a) and Percentage of combinations of patient attributes (b) covered by valid patterns.

pointed out in Schober *et al.* (2018) a Pearson between 0.40 and 0.69 represents moderate correlations and, as such, a correlation of 0.5 should not be neglected. Similarly, correlations above 0.7 represent strong or very strong relationships; these might be considered very reliable but, as observed from this experiment, exploiting these values may end up in discarding too much data.

At the end of the first phase, we have a set $\mathcal{P} = \{\langle p_i, \pi_j \rangle\}$ of patterns p_i for each facet π_j corresponding to a target parameter (as an example, Oxygen saturation, Hemoglobin, etc.). Now, each $\langle p_i, \pi_j \rangle$ is characterized by a number ν_{ij} of transactions supporting it and a Pearson correlation γ_{ij}. It is then possible to derive a linear regression model $\mu_{ij}(\cdot)$ between π_j and ICU for p_i. Given the value v_{jk} for the facet π_j of a generic (possibly unseen) transaction t_k supported by $\langle p_i, \pi_j \rangle$, we can then apply $\mu_{ij}(v_{jk})$ to estimate the value of ICU. $\mu_{ij}(v_{jk})$ is a value in the real interval $[0, 1]$ and constitutes a building block for the overall prediction process which is described next.

Given a generic unseen transaction t_k, let first identify the set $\mathcal{P}_{t_k} \subseteq \mathcal{P}$ of patterns in $\mathcal{P}$ contained also in t_k. Observe that, in principle, $\mathcal{P}_{t_k}$ might be void; in this case, no prediction can be carried out. For each $\langle p_i, \pi_j \rangle \in \mathcal{P}_{t_k}$ and the corresponding value v_{jk} in t_k let apply the regression model $\mu_{ij}(v_{jk})$ previously constructed, in order to get an estimation of ICU for the patient corresponding to t_k based on v_{jk}. Now, since there can be more patterns in $\mathcal{P}_{t_k}$ for t_k, it is possible to produce a more accurate estimation by combining the predictions obtained from all the patterns.

There are several ways in which we can combine the various predictions. Since each $\langle p_i, \pi_j \rangle$ is characterized by both a Pearson correlation γ_{ij} and a number of transactions ν_{ij} supporting it, we can use both these values as weights in an average mean of all obtained predictions. In fact, a very high correlation indicates a very good predictive capability; however, if this correlation is not supported by many transactions, it might be unreliable for unseen transactions. On the contrary, a high number of transactions supporting the pattern indicates a good support for the prediction, but obtaining a high correlation from several transactions is, in general, more complicated. By combining these two factors, we can weigh each prediction in a more accurate way. In particular, the formula exploited to estimate ICU for the patient corresponding to the transaction t_k is:

$$ICU_k = \frac{\sum_{ij} \mu_{ij}(v_{jk}) \cdot \gamma_{ij} \cdot \nu_{ij}}{\sum_{ij} \gamma_{ij} \cdot \nu_{ij}},$$

https://doi.org/10.1017/S1471068423000066 Published online by Cambridge University Press

Table 4. *Accuracy (left) and missing rate (right)*

Min. Occ.	*Pearson (absolute)*						*Min. Occ.*	*Pearson (absolute)*					
	0.5	0.6	0.7	0.8	0.9	1.0		0.5	0.6	0.7	0.8	0.9	1.0
5	0.73	0.75	0.76	0.74	0.78	0.77	5	0.15	0.39	0.48	0.55	0.61	0.74
10	0.71	0.73	0.73	0.72	0.79	0.78	10	0.15	0.39	0.5	0.6	0.72	0.78
15	0.72	0.71	0.7	0.71	0.83	–	15	0.16	0.4	0.53	0.68	0.79	1.0
20	0.71	0.71	0.71	0.67	–	–	20	0.17	0.41	0.6	0.75	1.0	1.0
25	0.71	0.7	0.71	0.7	–	–	25	0.18	0.43	0.62	0.79	1.0	1.0

Observe that ICU_k is a value in the real interval $[0, 1]$; in order to finalize the estimation, we carry out a rounding up of its value.

In order to test the validity of the approach, we carried out a 5-fold cross-validation where, in each run, 80% of tuples are used for extracting patterns and building the prediction model, and 20% of tuples to compute predictions. A prediction is considered exact if its value corresponds to the ICU value present in the database for the corresponding patient. This allows us to measure the Accuracy of the approach as the fraction of exact results vs the total ones. Table 4 (left) shows the average accuracy obtained among all the runs varying the minimum allowed frequency and the Pearson correlation. Since a transaction might not be supported by valid patterns, Table 4 (right) shows also the missing rate, that is, the percentage of transactions with no predictions. We also computed the variance for the obtained results on Accuracy, which ranges from a minimum of 0.0015 to a maximum of 0.057.

From the analysis of this table, we can observe that the Accuracy is quite stable and above 70% across the various combinations of parameters, with a slight increase as the Pearson increases. However, looking at the missing rate, we can observe that increasing values of the correlation significantly increases also the missing rate. If we observe Figure 2, there are actually threshold values for which the missing rate is very high (even 1, when no predictions are available); the Accuracy obtained for these configurations, although higher than that of the others, is of less significance since these configurations only allow us to obtain predictions on a small set of data. Reasonable values for the missing rate are obtained for a minimum absolute Pearson equal to 0.5, which corresponds also to a satisfying Accuracy. The quite stable behavior of the Accuracy can be motivated by the weighed formula we adopted for computing the predictions, which is able to adapt either to a high number of supported patterns and transactions or to a high Pearson with less support. Summarizing, it turns out that it is better to have a good coverage with a moderate correlation than a very strong correlation with a low coverage.

5 Experimental evaluation

In this section, we show and discuss the results of several experiments we carried out in order to assess the effectiveness of the proposed approach. We focus on the paper reviews use case introduced as running example and we consider a *quantitative* analysis,

https://doi.org/10.1017/S1471068423000066 Published online by Cambridge University Press

(a)

		Pattern size				
	th_f	1	2	3	4	5
200	4	27.2	25.8	31.5	32.4	27.8
	6	11.8	6.6	4.2	5.2	2.8
	8	5.2	3.9	2.2	1.7	1.7
	10	3.7	1.7	1.3	1.3	1.3
400	4	732.0	288.8	300.6	331.4	338.7
	6	364.0	50.4	35.1	29.5	42.7
	8	200.7	21.7	17.0	16.3	10.1
	10	46.4	14.0	10.7	11.3	7.6
600	4	2597.2	1028.9	1091.7	1225.9	1262.5
	6	1162.7	240.1	129.3	89.6	116.0
	8	890.0	88.9	57.6	57.1	29.9
	10	622.6	52.3	32.3	33.9	20.0
800	4	5243.0	2882.8	3420.4	3556.5	3820.6
	6	3802.0	775.8	525.0	284.4	230.3
	8	2529.7	288.3	160.1	132.4	68.7
	10	1916.9	164.0	101.2	98.9	50.3

(b)

		Pattern size				
	th_f	1	2	3	4	5
200	4	0.74	0.78	0.75	0.66	0.68
	6	0.51	0.52	0.46	0.48	0.41
	8	0.42	0.44	0.40	0.36	0.35
	10	0.36	0.37	0.35	0.33	0.32
400	4	4.24	4.94	4.42	3.92	3.60
	6	3.04	3.01	2.75	2.63	2.32
	8	2.27	2.32	2.18	2.11	2.01
	10	1.76	1.83	1.73	1.60	1.54
600	4	12.66	14.19	13.39	11.62	11.12
	6	9.62	10.11	8.97	8.78	7.65
	8	7.26	7.17	6.70	6.44	5.72
	10	5.68	5.61	5.39	4.88	4.68
800	4	34.37	47.15	36.23	31.55	26.89
	6	25.92	32.87	25.85	22.68	20.13
	8	21.53	22.08	20.01	18.43	17.49
	10	15.40	17.41	15.87	14.18	13.65

Fig. 3. Average running time (in seconds) for utility functions sum (a) and disagreement degree (b).

aiming to characterize the applicability of the approach in terms of performances, and a *qualitative* analysis, aiming to assess the quality of results.

The full dataset exploited in these experiments includes 814 papers, 1148 reviews, and 2230 annotated sentences, containing overall 15,124 distinct words. The complete dataset, expressed in ASP format, is available at https://www.mat.unical.it/~cauteruccio/tplp-ehupm/. Other properties are shown in Table 1. All the experiments have been performed on a 2.3GHz MacBook computer (Intel Core i9) with 16 GB of memory. The data preprocessing pipeline was implemented in Python 3.9 and exploited the spaCy (https://spacy.io/) library. In order to run our ASP programs, we used clingo version 5.4.0 with the flag `--mode=gringo` as the grounder and WASP version 2.0 compiled via clang (version 11.0.3) with Python 3.9 support as the solver.

5.1 Quantitative analysis

In a first series of experiments, we computed the average running times for two different settings. The first one is a sort of a stress test: it involves all the facets provided by the use case and computes their sum via an external function call; thus, all input items and values are relevant for the computatio, and, consequently, a large number of patterns are expected. The second one is more realistic and computes the disagreement degree between one of the annotated sentiments and the decision about the corresponding paper, namely the percentage of pattern occurrences showing a positive sentiment on originality and a reject decision; in this case, given its simplicity, the utility function is directly encoded with ASP rules. Results for running times are shown in Figure 3 for increasing number of papers, pattern size, and occurrence threshold (th_f); each data point represents the average running time computed for utility thresholds equal to $[1, 5, 10, 15, 20, 25, 30]$ for the sum and $[15, 50, 75, 100]$ for the disagreement degree. To rescale the dataset, we randomly selected the set of papers to be removed at each downsizing step, so that each instance is a strict subset of its larger versions. This selection process has been carried out once, and the same datasets have been used throughout the experiments.

First of all, from the analysis of the results, it is possible to observe that the choice of the setting can significantly impact the performances. This is due to a combination of factors, including the amount of input data relevant to the calculation, the number

https://doi.org/10.1017/S1471068423000066 Published online by Cambridge University Press

of patterns satisfying the thresholds, and the complexity of the utility function itself. In more detail, looking at Figure 3(a), we can observe that increasing the number of papers significantly increases execution time but, increasing the occurrence threshold significantly reduces required time. Moreover, especially for the bigger datasets, a higher occurrence threshold combined with a higher pattern size significantly reduces execution time. This behavior can be mainly motivated by the portion of encoding for identifying frequent patterns; in particular, checking the number of occurrences within the choice rule, instead of using a standard constraint, makes rules more complex in general, but allows the evaluator to cut the search space more easily for bigger occurrence thresholds and, also, for larger patterns. In fact, it is worth pointing out that ground programs resulted to be smaller for increasing occurrence thresholds. We also observed that (results not shown due to space constraints) the utility threshold does not significantly impact performances. Similar considerations can be drawn for the tests on disagreement degree (Figure 3b). For these tests, it is worth observing the following differences with respect the previous ones using the sum: *(i)* input relevant facts are significantly lower (only words in sentences annotated with originality are useful); *(ii)* required time and memory (see below) are significantly lower; *(iii)* the number of valid patterns is significantly lower and, in this case, decreases with increasing utility thresholds. In particular, the maximum number of extracted patterns is 2480 with the sum, and 95 with the disagreement degree.

In a second series of experiments we considered the impact of external function calls in ASP. In particular, we consider two scenarios in which the utility function can be both expressed directly in ASP and in Python. Before going on with the analysis, let us emphasize once again that the constraints we are dealing with in this work to validate the patterns may come in the form of potentially complex functions, which are difficult, or even impossible, to express in ASP, and, as a result, external functions significantly broaden the scope of applicability of the proposed approach. Moreover, we observe that a deep analysis of when it is better to implement functions directly in ASP or in Python is out of the scope of this paper. The interested reader can find an extensive discussion on this aspect in Cuteri *et al.* (2017).

In the first experiment, we considered the sum function, as introduced above, with the following parameters: pattern size 3, occurrence threshold 6, utility threshold 5. This is the worst-case scenario for evaluating external functions, since ASP includes built-in functions for computing the sum, and, in fact, results presented in Figure 4(a) show that the impact of the external function calls may become consistent, especially when the number of papers increases.

The second experiment considered the product function; in particular, for the purposes of this experiment, we added a facet to each review representing a (fictitious) probability, and we considered the combined probability as the utility function, that is, the product of all the probabilities associated with pattern occurrences. Also in this case, we implemented a version of the program using external functions and a version implementing the product computation directly in ASP; it is worth observing that, in this case, no built-in function like the sum is available in ASP for the product, and, thus, it must be computed with the support of recursive rules (probabilities have been obviously scaled to integers between 0 and 100). Parameters used in these tests are as follows: pattern size 3, occurrence threshold 6, utility threshold 5. Results presented in Figure 4b clearly show that, in this case, the version with external functions significantly outperforms the

https://doi.org/10.1017/S1471068423000066 Published online by Cambridge University Press

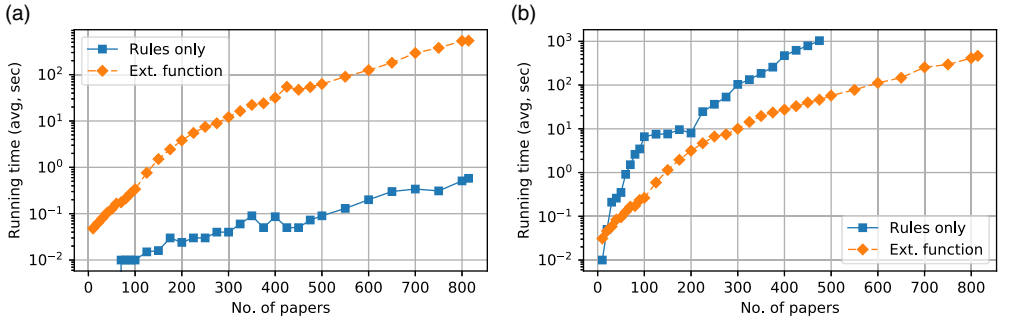

Fig. 4. Average running time (log scale) for utility functions sum (a) and product (b) using pure ASP or external functions.

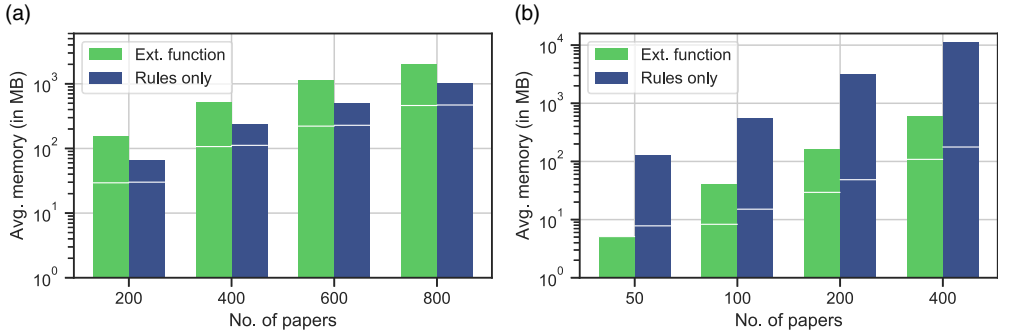

Fig. 5. Average memory usage (log scale) for utility functions sum (a) and product (b). Horizontal white lines denote the boundary for the memory required by the grounder (bottom) and solver (top).

one using ASP only, thus motivating the adoption of external sources of computation for utility functions which are not straightforward to be expressed in ASP.

In order to show the impact of external functions on memory usage, we report in Figure 5 memory consumption for functions sum and product expressed directly in ASP or in Python. From the analysis of this figure, it is possible to observe, again, that for the sum function, the impact of external functions may become consistent. However, when we turn to the product function, the amount of memory required by the version using external functions is extremely lower. In both cases, the main difference is on the solver part, whereas the grounder part requires similar amounts of memory.

5.2 Qualitative analysis

In this section, we describe the results of some experiments carried out to show if, and how much, facets and utility functions introduced in Section 2 can help users in analyzing input data from different points of view and at different abstraction levels.

In particular, we concentrate on the computation of coherence and disagreement degrees, introduced in Section 2.4, between the decision on a paper (Accept/Reject) and one of the eight aspects used to label review sentences (Chakraborty *et al.* 2020). In particular, the objective is to look for patterns (sets of words) characterizing with good accuracy the relationship of interest and to qualitatively check if they are meaningful.

https://doi.org/10.1017/S1471068423000066 Published online by Cambridge University Press

Before going into the details of the analysis, in order to better show the properties of our approach, consider that the paper review dataset contains 686 words with contrasting values between a "sentiment" (on one of the aspects among Appropriateness, Clarity, Originality, Empirical/Theoretical Soundness, Meaningful Comparison, Substance, Impact of Dataset/Software/Ideas and Recommendation) and the Decision. Among them, 481 different words are in sentences with an "Originality" sentiment; this situation leads to 929 patterns of size between 2 and 4 with at least 4 occurrences just related to "Originality." As a consequence, any classical approach based only on pattern frequency would be overwhelmed by a large amount of patterns.

Table 5 shows some of the obtained results. Here, given the sentiment label on a sentence aspect X, the utility function measures the percentage of occurrences of the pattern showing coherence/disagreement between the sentiment on X and the decision on the paper, as specified in the first column. The second column of Table 5 reports the total number of patterns obtained by the approach for each test, whereas the third and fourth columns show an excerpt of the most relevant patterns along with their utilities. Parameters exploited for these tests are as follows: Minimum frequency=4, Minimum utility=70,[3] pattern size between 2 and 4.[4] It is worth pointing out that each run completed the computation in just few seconds, and switching between one test and the other involved few clicks and minor modifications to the ASP program.

From the analysis of Table 5, we can draw the following observations: *(i)* The number of valid patterns characterizing each setting is very low; this allows the user to concentrate on the most relevant ones. *(ii)* Patterns extracted in the different settings are quite different; this indicates a good specificity of derived patterns. *(iii)* For some settings, the number of derived patterns is 0; this means that the corresponding situation cannot be characterized. As far as this last observation is concerned, consider the very interesting situation analyzing Recommendation and Reject: in this case, no patterns characterize a positive recommendation (which we recall is a derived sentiment from the sentences) associated with a reject decision, whereas 84 patterns characterize the opposite situation, namely negative recommendation and reject; this provides an interesting insight on the appropriateness of derived patterns. We also carried out a similar analysis relating the Rating facet, associated with the Review (which is an object in our framework), and the Decision facet, associated with the Paper (which is a container in our framework). Results are shown in the last two rows of Table 5; these confirm the analysis relating recommendation and decision and show how easy is to switch the analysis also between different levels of abstraction.

6 Related work

Considering its importance as a research topic in the data mining field, HUPM received a huge interest both from academic community and industrial practitioners. HUPM finds application both in classical contexts where (normal) itemset mining was born, such as

[3] Observe that in this case, the utility value expresses a percentage, as a consequence, 70 should be read as 70% of the occurrences of the pattern satisfy the condition.

[4] Only for some specific cases we extended the size up to 6 in order to show more interesting patterns; these are annotated with a "*."

https://doi.org/10.1017/S1471068423000066 Published online by Cambridge University Press

Table 5. *Qualitative analysis on coherence/disagreement degrees*

Setting	# tot	Sample patterns	Utility
Originality(positive)-Reject	7	Paper interesting approach Paper interesting write Simple nice idea Intriguing idea Idea proposed interesting	100 80 75 75 71
Originality(positive)-Accept	1	Original approach	75
Originality (negative)-Accept	6	Incremental paper Easy conference paper Easy follow interesting Paper write easy read	80 75 75 75
Disagreement between Clarity and Decision: Clarity(positive)-Reject or Clarity(negative)-Accept	71	Easy paper overall Easy conference paper Easy follow interesting Paper write easy read Presentation generally easy Follow interesting result*	100 100 85 85 80
Appropriateness(positive)-Accept	0		
Appropriateness(negative)-Accept	2	Niche audience Paper accessible	75 75
Impact of Dataset/Software/Ideas(positive)-Accept	20	Paper write clearly present Dataset significant Proposed method datasets baselines Overall contribution	100 100 75 74
Recommendation(negative)-Reject	84	Paper reject Accept conference need significant Acceptance work need significant Recommend acceptance conference Need significant work*	100 75 75 75
Recommendation(positive)-Reject	0		
Rating low (1–4)-Accept	0		
Rating high (5–10)-Accept	173	Paper write clearly easy Paper follow write clearly Dataset new Theoretical approach Present clearly	100 100 100 100 75

https://doi.org/10.1017/S1471068423000066 Published online by Cambridge University Press

market analysis, and in novel ones such as biomedicine, mobile computing, etc. Suvarna and Srinivas (2019). Thanks to its exhibited versatility, a huge volume of research studies has been produced regarding HUPM and its variants. According to Semantic Scholar[5], in the last twenty years more than 8000 studies related to HUPM have been published.

In order to better characterize, the work related to our approach and to discern among the plethora of presented studies, we divide the analysis of the related literature in two parts. The former provides a bird's-eye view of the general literature about HUPM, focusing on variations of the utility measure; the latter focuses on some declarative-based approaches to pattern mining.

It is worth pointing out that, to the best of our knowledge, no previous work addressed in a comprehensive way the various extensions introduced in the present paper; additionally, our work is the first ASP-based solution to the e-HUPM problem.

6.1 Approaches for HUPM and its variants

As previously pointed out, HUPM is a general container for several high-utility-based mining approaches (Fournier-Viger *et al.* 2019; Gan *et al.* 2019). The corresponding research studies can be divided in several groups, depending on the aspect one is interested to analyze, such as employed algorithms or pruning strategies (Krishnamoorthy 2017; Yun *et al.* 2017; Wu *et al.* 2019; Yun *et al.* 2019; Sumalatha and Subramanyam 2020). In our context, we mainly focus on research that studies utility measures and their variants (Yao *et al.* 2006; Shen *et al.* 2002; Fournier-Viger *et al.* 2014; Cagliero *et al.* 2014; Fournier-Viger *et al.* 2020; Deng 2020).

One of the first works on unifying utility-based measures is presented in Yao *et al.* (2006), where the authors introduce a unified utility framework in which a user-defined function $f(x, y)$ is exploited to define utilities. Here, the significance of an item is measured by two parts: one is the statistical significance of the item, measured by x, and is an objective measure; the other is the semantic significance of the item, measured by y, which is a subjective measure dependent on the user perception of utility. Moreover, a weight to each transaction, called vertical weight, can be assigned. The authors show that this generalization can assign semantic significance at different levels of granularity (item, transaction, and cell level). The approach proposed in Yao *et al.* (2006) and our own share some similarities, but the present work further extends the pattern mining capabilities, as specified next. *(i)* First of all, Yao *et al.* (2006) attempts to provide different levels of abstraction for the identification of pattern utility; the present paper has a similar goal but extends it not only to transactions but also to generic abstraction layers, enabling the introduction of more advanced utility functions. *(ii)* Allowing f to be a user-defined function extends the scope of application; however, only one property at a time can be used to state the semantic significance of an item; in our approach, the introduction of facets allows not only to consider several properties at the same time, but also to combine them dynamically. *(iii)* The introduction of facets also on transactions, objects, and containers allows not only analyses at different levels of granularity, as done in Yao *et al.* (2006) but also the identification of patterns expressing correlations (and possibly causality) between different properties, at different levels. *(iv)* Finally, we have

[5] https://www.semanticscholar.org

https://doi.org/10.1017/S1471068423000066 Published online by Cambridge University Press

shown that beside classical sum and product operations on utility values, exploited in Yao *et al.* (2006), more complex functions like Pearson's correlation or Multiple correlation, allowed by the availability of facets, broaden the scope of potential analyses.

An effort to define a general model, called Objective-Oriented Utility-Based Association mining, in which mined association patterns are both statistically and semantically related to a user's given objective, is explained in Shen *et al.* (2002). Here, the database is composed of a set of records structured in a predefined set of attributes, and an association rule is enhanced with an objective and utility value. Thus, the approach aims at deriving patterns from the records that both statistically and semantically support a given objective, by taking into account minimum support, confidence, and utility. While the scope of Shen *et al.* (2002) and our own are not completely overlapping, namely association rule mining vs high-utility pattern mining, our approach can embrace the concept of objective-driven mining. Thus, with the adoption of suitable pattern masks, facets, and utility functions, our proposal can be considered to some extent as an extension of the concepts presented in Shen *et al.* (2002).

The authors in Deng (2020) define a novel measure called occupancy and formulate the problem of mining high occupancy itemsets. The occupancy is defined as the sum of the ratio between the cardinality of an itemset p and the cardinality of the transactions that contain p. Then, a high occupancy itemset is an itemset whose occupancy is not below a given threshold. Interestingly, the approach described in Deng (2020) can be seen as a special case of our approach, where a *horizontal first* utility function class is applied.

It is also worth pointing out that different extensions of the HUPM problem are presented in the literature. Different aspects are considered, and each aspect aims at addressing the limitations of the classical problem (Fournier-Viger *et al.* 2019). As an example, an HUPM algorithm may show a large number of patterns to the user if the minimum utility threshold given in input is too low, thus it can be very hard for a user to analyze and describe retrieved patterns. To this end, several concise representations for high-utility patterns have been studied. Four main representations are closed high-utility patterns, maximal patterns, generators of high-utility patterns, and maximal high-utility patterns (Fournier-Viger *et al.* 2019); moreover, in Gan *et al.* (2019) the Kulc measure has been adopted to extract non-redundant strongly correlated and profitable patterns. In our approach, the number of retrieved patterns can be reduced by defining more stringent utility functions; however, we can see our approach and the ones mentioned above as orthogonal, since they do not focus on the definition of utility, but rather on the representation and filtering of obtained results.

Other different aspects are studied in the literature, such as HUPM with negative utility and HUPM with discount strategies (Fournier-Viger *et al.* 2019). These are all covered by our framework, which can be seen as an extension of these special cases with further potentialities.

In addition to the aforementioned approaches, it is worth noting that HUPM has been employed in several different contexts, such as topic detection (Choi and Park 2019), relevant information extraction (Belhadi *et al.* 2020), aspect-based sentiment analysis (Demir *et al.* 2019), and mining in noisy databases (Baek *et al.* 2020). This variety of application contexts for HUPM further motivates our framework which, thanks to its generality, can accommodate very different settings, as it has been shown through the paper.

https://doi.org/10.1017/S1471068423000066 Published online by Cambridge University Press

Finally, there are also different pattern extraction methods, not related to HUPM, which could be indirectly encompassed by our framework. An example is that of subgroup discovery (Herrera *et al.* 2011; Atzmueller 2015). Briefly, subgroup discovery is a method for knowledge discovery in databases whose aim is to identify relations between a target variable and many explaining and independent variables, determined by a quality function that can be flexibly defined (Atzmueller 2015). Intuitively, it means to discover the subgroups of the population that are statistically most interesting and, therefore, can be identified as a sort of pattern extraction technique. This task has been addressed with several approaches, including those exploiting evolutionary algorithms (del Jesus *et al.* 2007), complex network aspects (Škrlj *et al.* 2018), and inductive logic programming (Černoch and Železný 2012). Although our notion of utility could encompass similar quality measures as those used in subgroup discovery, the aim of our framework is different from the task of subgroup discovery. Indeed, this last task could be further investigated in a future work by considering the multi-level structure offered by our approach. The application of our framework we presented in Section 4 differs from subgroup discovery in terms of the purpose of the knowledge discovery task. The goal of classification in our application is to generate a model for each class that contains rules representing class characteristics regarding the descriptions of training examples. This model is used to predict the class of unknown objects. In contrast, subgroup discovery attempts to discover interesting relations with respect to the property of interest (Helal 2016).

6.2 Declarative-based approaches for pattern mining

The usage of declarative systems to tackle combinatorial problems is a consolidated approach which attracted a peculiar interest from researchers. The possibility of coupling the power of a high-level declaration with an optimized solver allows users to specify how patterns should be and not how they should be computed. There are different existing approaches that blend together the expressiveness and readiness to use of a declarative system within the problem of pattern mining (Järvisalo 2011; Gebser *et al.* 2016; Samet *et al.* 2017; Paramonov *et al.* 2019; Guyet *et al.* 2014; 2016). The main objective of this section is to show the growing interest in declarative-based approaches for pattern mining as an alternative to dedicated algorithms, especially when the main focus is to add expressiveness to the standard problem at hand. To the best of our knowledge there is no ASP-based approach tackling the HUPM problem, even in its basic form; then, with respect to this aspect, our approach moves a step forward in the application of ASP with external functions in interesting data mining contexts.

The seminal work in Järvisalo (2011) exploits ASP in the task of finding all frequent itemsets: each itemset is an answer set; thus, the enumeration of all answer sets corresponds to the enumeration of all frequent itemsets. Although both our approach and the one in Järvisalo (2011) exploit ASP, the latter addresses only the standard frequent itemset setting; in our extended HUPM setting, we also exploit the recent ASP language extensions for the application of complex functions on sets of data inferred during the evaluation of the program.

In Gebser *et al.* (2016), the context of sequence mining is addressed, and an ASP-based mining approach is presented. Here, the focus is to express and incorporate knowledge in the mining process. The authors consider frequent (closed or maximal) sequential

https://doi.org/10.1017/S1471068423000066 Published online by Cambridge University Press

patterns and complex preferences on patterns. Frequent patterns are mined following similar encoding principles of Järvisalo (2011), whereas preference-based mining is accomplished by exploiting the *asprin* system (Brewka *et al.* 2015) which allows expressing combinations of qualitative and quantitative preferences among answer sets. Rare sequential pattern mining is considered in (Samet *et al.* 2017).

A general and hybrid approach to pattern mining is presented in Paramonov *et al.* (2019), where different dedicated mining systems are employed to detect frequent patterns and ASP is used to filter the results. Here, the tasks considered are itemset, sequence, and graph mining, where local (frequency, size, and cost) and global (condensed representations such as maximal, closed, and skyline) constraints are combined in a generic way. The declarative approach here is devoted to post-process the patterns in order to select the valid ones.

Mining serial patterns, that is, frequent sequential patterns from a unique sequence of itemsets, is introduced in Guyet *et al.* (2014), whereas sequential pattern mining with two representations of embeddings (fill-gaps and skip-gaps) and several kinds of patterns are considered in Guyet *et al.* (2016).

Other declarative approaches, not constrained to ASP and based, for example, on constraint programming or SAT, are also discussed in the literature (Négrevergne *et al.* 2013; Guns *et al.* 2017; 2016). Itemset mining is considered in Négrevergne *et al.* (2013), in which a novel algebra for programming pattern mining problems is introduced. It allows for the generic combination of constraints on individual patterns with dominance relations between patterns. Dominance relations are pairwise preferences between patterns, used to express the idea that a pattern p_1 is preferred over another pattern p_2. Various settings are described with combinations of constraints and dominance relations, including maximal and closed itemset mining, sky patterns, etc. Dominance programming offers a great extensibility, although it is not as general as our approach. Furthermore, the context considered here only aims at itemset mining. Itemset, sequence, and graph mining tasks are also considered in a framework introduced in Guns *et al.* (2016), in which these pattern types are studied under the lens of constraint-based mining (Mannila and Toivonen 1997). A declarative framework for constraint-based mining is presented in Guns *et al.* (2017), where the objective is to bridging the methodological gap between data mining and constraint programming by providing a framework to support constraint-based pattern mining. In particular, the authors offer several contributions, spanning from a novel library of functions and constraints in the MiniZinc language to support modeling itemset mining tasks, to the automatic composition of execution strategies involving different solving methods. Indeed, the work in Guns *et al.* (2017) shares some common similarities with our proposed approach. In particular, both the approaches provide a general framework for high-utility mining, and they are based on declarative paradigms. Our approach supports the various extensions to HUPM introduced in Section 2, which are not supported by Guns *et al.* (2017). Nevertheless, Guns *et al.* (2017) also deals with multiple databases, which is a feature not currently supported by our approach.

An interesting task studied in the literature is the discovery of skyline patterns, or "skypatterns" (Soulet *et al.* 2011; Ugarte *et al.* 2017). Skypatterns draw their definition from the economics and multi-criteria decision-making worlds, in which they are commonly called Pareto efficiency or optimality queries. Intuitively, given a multidimensional space where a preference is defined for each dimension, a point a dominates another point

https://doi.org/10.1017/S1471068423000066 Published online by Cambridge University Press

b if a is better than b in at least one dimension, and a is not worse than b on every other dimension. Given a set of patterns, the skyline set contains patterns that are not dominated by any other pattern (Ugarte *et al.* 2017). Skypattern mining provides an easy expression of preferences over patterns by combining several measures on patterns. An approach for skypattern mining is presented in Ugarte *et al.* (2017). Here, the authors address the discovery of skypatterns by proposing a unified methodology based on the following contributions, namely *(i)* a condensed representation of patterns, *(ii)* a dynamic method based on constraint programming that allows a continuous refinement of the skyline constraints, and *(iii)* the notion of particular local extrema in the search space of the problem. The work in Ugarte *et al.* (2017) shares few similarities with ours; actually, the two approaches can be considered complementary.

7 Conclusion

In this paper, we introduced a general framework for HUPM with several extensions that significantly broaden the applicability of HUPM even in non-classical contexts. We provided an ASP-based computation method, which exploits the most recent extensions of ASP, and we have shown that this solution allows for a reasonable and versatile implementation. A real use case on paper reviews has been employed as a fil-rouge to analyze the different aspects of the proposed framework; in particular, we have shown that the introduction of facets and suitable advanced utility functions can both reduce the amount of relevant patterns and provide deep insights on the data, thus advancing the state-of-the-art on the interesting topic of high-utility pattern mining. An application to a biomedical context has also been presented and tested, which showed how the novel pattern mining framework can be suitably exploited to design effective prediction models on biomedical data.

The presented work is just a first step in this new direction of utility-based analyses and several future research directions are now open. First of all, it will be interesting to apply the new features of the framework to a wide spectrum of contexts, such as biomedical data analysis, IoT data analysis, and fraud detection. A classification of the computational properties of the various extended utility functions will be also useful to devise ad-hoc algorithms. In particular, we plan to develop dedicated and efficient algorithms for specific settings of interest, in order to provide fast computational methods for extended high-utility patterns. Another interesting line of future research regards a deeper study of the application of our e-HUPM to prediction tasks in various contexts.

Competing interests

The authors declare none.

References

Agarwal, R. and Srikant, R. Fast algorithms for mining association rules. In *Proc. of the 20th VLDB Conference* 1994. Morgan Kaufmann, 487–499.

Atzmueller, M. 2015. Subgroup discovery. *WIREs Data Mining and Knowledge Discovery*, *5*, 1, 35–49.

https://doi.org/10.1017/S1471068423000066 Published online by Cambridge University Press

Baek, Y., Yun, U., Kim, H., Kim, J., Vo, B., Truong, T. and Deng, Z.-H. 2020. Approximate high utility itemset mining in noisy environments. *Knowledge-Based Systems*, *212*, 106596.

Belhadi, A., Djenouri, Y., Lin, J. C., Zhang, C. and Cano, A. 2020. Exploring pattern mining algorithms for hashtag retrieval problem. *IEEE Access*, *8*, 10569–10583.

Brewka, G., Delgrande, J. P., Romero, J. and Schaub, T. asprin: Customizing answer set preferences without a headache. In *Proc. of the 29th AAAI Conference on Artificial Intelligence, (AAAI 2015)*. AAAI Press, 1467–1474.

Cagliero, L., Cerquitelli, T., Garza, P. and Grimaudo, L. 2014. Misleading generalized itemset discovery. *Expert Systems with Applications*, *41*, 4, Part 1, 1400–1410.

Černoch, R. and Železný, F. Subgroup discovery using bump hunting on multi-relational histograms. In *Proc. of Inductive Logic Programming (ILP 2012)*, Windsor Great Park, UK. Springer Berlin Heidelberg, 76–90.

Chakraborty, S., Goyal, P. and Mukherjee, A. Aspect-based sentiment analysis of scientific reviews. In *Proc. of the ACM/IEEE Joint Conference on Digital Libraries (JCDL 2020)* 2020. ACM, 207–216.

Choi, H.-J. and Park, C. H. 2019. Emerging topic detection in twitter stream based on high utility pattern mining. *Expert Systems with Applications*, *115*, 27–36.

Cuteri, B., Dodaro, C., Ricca, F. and Schüller, P. 2017. Constraints, lazy constraints, or propagators in ASP solving: An empirical analysis. *Theory and Practice of Logic Programming*, *17*, 5–6, 780-799.

del Jesus, M. J., Gonzalez, P., Herrera, F. and Mesonero, M. 2007. Evolutionary fuzzy rule induction process for subgroup discovery: A case study in marketing. *IEEE Transactions on Fuzzy Systems*, *15*, 4, 578–592.

Demir, S., Alkan, O., Cekinel, F. and Karagoz, P. 2019. *Extracting Potentially High Profit Product Feature Groups by Using High Utility Pattern Mining and Aspect Based Sentiment Analysis*. Springer International Publishing, Cham, 233–260.

Deng, Z.-H. 2020. Mining high occupancy itemsets. *Future Generation Computer Systems*, *102*, 222–229.

Dodaro, C. and Ricca, F. 2020. The external interface for extending wasp. *Theory and Practice of Logic Programming*, *20*, 2, 225-248.

Eiter, T., Fink, M., Ianni, G., Krennwallner, T., Redl, C. and Schüller, P. 2016. A model building framework for answer set programming with external computations. *Theory and Practice of Logic Programming*, *16*, 4, 418–464.

Eiter, T., Germano, S., Ianni, G., Kaminski, T., Redl, C., Schüller, P. and Weinzierl, A. 2018. The DLVHEX system. *KI - Künstliche Intelligenz*, *32*, 2–3, 187–189.

Fournier-Viger, P., Gomariz, A., Campos, M. and Thomas, R. Fast vertical mining of sequential patterns using co-occurrence information. In *Proc. of the 18th Pacific-Asia Conference, (PAKDD 2014)* 2014, vol. 8443. *LNCS*. Springer, 40–52.

Fournier-Viger, P., Lin, J.-W., Nkambou, R., Vo, B. and Tseng, V. 2019. *High-Utility Pattern Mining*. Springer.

Fournier-Viger, P., Wang, Y., Lin, J. C., Luna, J. M. and Ventura, S. Mining cross-level high utility itemsets. In *Proc. of the 33rd International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems, (IEA/AIE 2020)* 2020, vol. 12144. *LNCS*. Springer, 858–871.

Gan, W., Lin, C., Fournier-Viger, P., Chao, H., Tseng, V. and Yu, P. 2019. A survey of utility-oriented pattern mining. *IEEE Transactions on Knowledge and Data Engineering*, *33*, 4, 1306–1327.

Gan, W., Lin, J. C.-W., Chao, H.-C., Fujita, H. and Yu, P. 2019. Correlated utility-based pattern mining. *Information Science*, *504*, 470–486.

https://doi.org/10.1017/S1471068423000066 Published online by Cambridge University Press

Gebser, M., Guyet, T., Quiniou, R., Romero, J., and Schaub, T. Knowledge-based sequence mining with ASP. In *Proc. of the 25th International Joint Conference on Artificial Intelligence, (IJCAI 2016)*. IJCAI/AAAI Press, 1497–1504.

Gebser, M., Kaminski, R., Kaufmann, B. and Schaub, T. 2019. Multi-shot ASP solving with clingo. *Theory and Practice of Logic Programming*, *19*, 1, 27–82.

Guns, T., Dries, A., Nijssen, S., Tack, G. and Raedt, L. D. 2017. Miningzinc: A declarative framework for constraint-based mining. *Artificial Intelligence*, *244*, 6–29.

Guns, T., Paramonov, S. and Négrevergne, B. On declarative modeling of structured pattern mining. In *Proc. of the 2016 AAAI Workshop Declarative Learning Based Programming* 2016, volume WS-16-07 of *AAAI Workshops*. AAAI Press.

Guyet, T., Moinard, Y. and Quiniou, R. 2014. Using answer set programming for pattern mining. *CoRR*, *abs/1409.7777*.

Guyet, T., Moinard, Y., Quiniou, R. and Schaub, T. Efficiency analysis of ASP encodings for sequential pattern mining tasks. In *Advances in Knowledge Discovery and Management - Volume 7 [Best of EGC 2016,Reims, France]* 2016, vol. 732. *Studies in Computational Intelligence*. Springer, 41–81.

Helal, S. 2016. Subgroup Discovery Algorithms: A Survey and Empirical Evaluation. *Journal of Computer Science and Technology*, *31*, 3, 561–576.

Herrera, F., Carmona, C. J., González, P. and del Jesus, M. J. 2011. An overview on subgroup discovery: foundations and applications. *Knowledge and Information Systems*, *29*, 3, 495–525.

Järvisalo, M. Itemset mining as a challenge application for answer set enumeration. In *Proc. of the 11th International Conference on Logic Programming and Nonmonotonic Reasoning, (LPNMR 2011)* 2011, vol. 6645. *LNCS*. Springer, 304–310.

Krishnamoorthy, S. 2017. Hminer: Efficiently mining high utility itemsets. *Expert Systems with Applications*, *90*, 168–183.

Lewis-Beck, M., Bryman, A. and Liao, T. 2003. *The Sage Encyclopedia of Social Science Research Methods*. Sage Publications.

Mannila, H. and Toivonen, H. 1997. Levelwise search and borders of theories in knowledge discovery. *Data Mining and Knowledge Discovery*, *1*, 3, 241–258.

Négrevergne, B., Dries, A., Guns, T. and Nijssen, S. Dominance programming for itemset mining. In *Proc. of the 13th International Conference on Data Mining (ICDM 2013)*. IEEE Computer Society, 557–566.

Paramonov, S., Stepanova, D. and Miettinen, P. 2019. Hybrid asp-based approach to pattern mining. *Theory and Practice of Logic Programming*, *19*, 4, 505–535.

Pearson, K. 1895. Note on regression and inheritance in the case of two parents. *Proceedings of the Royal Society of London*, *58*, 240–242.

Samet, A., Guyet, T. and Négrevergne, B. Mining rare sequential patterns with ASP. In *Late Breaking Papers of the 27th International Conference on Inductive Logic Programming, Orléans, France, September 4–6, 2017* 2017, vol. 2085. *CEUR Workshop Proceedings*. CEUR-WS.org, 51–60.

Schober, P., Boer, C. and Schwarte, L. 2018. Correlation coefficients: appropriate use and interpretation. *Anesthesia & Analgesia*, *126*, 5, 1763–1768.

Shen, Y., Zhang, Z. and Yang, Q. Objective-oriented utility-based association mining. In *Proc. of the 2002 IEEE International Conference on Data Mining (ICDM 2002)*. IEEE Computer Society, 426–433.

Škrlj, B., Kralj, J., Vavpetič, A. and Lavrač, N. Community-based semantic subgroup discovery. In *Proc. of the 6th International Workshop on New Frontiers in Mining Complex Patterns (NFMCP)*, Skopje, Macedonia. Springer International Publishing, 182–196.

Soulet, A., Raïssi, C., Plantevit, M. and Cremilleux, B. Mining dominant patterns in the sky. In *Proc. of IEEE 11th International Conference on Data Mining (ICDM'11)*, Vancouver, Canada. IEEE, 655–664.

Sumalatha, S. and Subramanyam, R. 2020. Distributed mining of high utility time interval sequential patterns using mapreduce approach. *Expert Systems with Applications*, *141*, 112967.

Suvarna, U. and Srinivas, Y. Efficient high-utility itemset mining over variety of databases: A survey. In *Soft Computing in Data Analytics*. Springer Singapore, 803–816.

Ugarte, W., Boizumault, P., Crémilleux, B., Lepailleur, A., Loudni, S., Plantevit, M., Raïssi, C. and Soulet, A. 2017. Skypattern mining: From pattern condensed representations to dynamic constraint satisfaction problems. *Artificial Intelligence*, *244*, 48–69.

Wu, J. M.-T., Lin, J. C.-W. and Tamrakar, A. 2019. High-utility itemset mining with effective pruning strategies. *ACM Transactions on Knowledge Discovery from Data*, *13*, 6, 1–22.

Yao, H., Hamilton, H. and Geng, L. A unified framework for utility-based measures for mining itemsets. In *Proc. of ACM SIGKDD 2nd Workshop on Utility-Based Data Mining*. ACM, 28–37.

Yun, U., Nam, H., Lee, G. and Yoon, E. 2019. Efficient approach for incremental high utility pattern mining with indexed list structure. *Future Generation Computer Systems*, *95*, 221–239.

Yun, U., Ryang, H., Lee, G. and Fujita, H. 2017. An efficient algorithm for mining high utility patterns from incremental databases with one database scan. *Knowledge-Based Systems*, *124*, 188–206.

https://doi.org/10.1017/S1471068423000066 Published online by Cambridge University Press