

On Coupling FCA and MDL in Pattern Mining

Tatiana Makhalova, Sergei O. Kuznetsov, Amedeo Napoli

▶ **To cite this version:**

Tatiana Makhalova, Sergei O. Kuznetsov, Amedeo Napoli. On Coupling FCA and MDL in Pattern Mining. ICFCA 2019 - 15th International Conference on Formal Concept Analysis, Jun 2019, Frankfurt, Germany. pp.332-340. hal-02162928

HAL Id: hal-02162928
https://hal.science/hal-02162928

Submitted on 23 Jun 2019

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

On Coupling FCA and MDL in Pattern Mining

Tatiana Makhalova[1,2], Sergei O. Kuznetsov[1], and Amedeo Napoli[2]

[1] National Research University Higher School of Economics, Moscow, Russia
[2] Université de Lorraine, CNRS, Inria, LORIA, F-54000 Nancy, France
tpmakhalova@hse.ru, skuznetsov@hse.ru, amedeo.napoli@loria.fr

Abstract. Pattern Mining is a well-studied field in Data Mining and Machine Learning. The modern methods are based on dynamically updating models, among which MDL-based ones ensure high-quality pattern sets. Formal concepts also characterize patterns in a condensed form. In this paper we study MDL-based algorithm called Krimp in FCA settings and propose a modified version that benefits from FCA and relies on probabilistic assumptions that underlie MDL. We provide an experimental proof that the proposed approach improves quality of pattern sets generated by Krimp.

1 Introduction

Pattern Mining (PM) has an important place in Data Mining (DM) and Knowledge Discovery (KD). One main concern of PM is to discover something interesting in a pattern space that is exponentially large w.r.t. the size of the dataset. "Interestingness" can be defined as generalities or specificities underlying the data, (dis)accordance to user hypotheses, etc. One distinguishes *static* and *dynamic* approaches for PM [1].

The static approaches envelop a large number of interestingness measures [9]. The patterns are mined under non-changeable assumptions about interestingness. For example, in frequent PM, one assumes that all the patterns having support greater than a user-specified threshold are interesting. Other popular measures work under independence [2], randomization models [8] or other constraints [4, 13]. The main drawbacks of the static approaches are (i) pattern redundancy, i.e., the discovered pattern set contains a lot of similar patterns, and (ii) subjectiveness, i.e., it is often not easy to provide explanation or justification about using one measure rather than some others.

Mining with interestingness measures addresses a traditional question of PM, i.e., finding *all the patterns that satisfy some given constraints*. By contrast, in [1] (Chapter 8), it is argued that one should ask for a small (easily interpretable) and non-redundant (with high diversity) set of interesting patterns. This is precisely what dynamic approaches to PM are aimed at.

A dynamic approach implies taking into account initial assumptions, e.g., background knowledge, and building progressively a suitable "model", i.e., a pattern set. Following this way, the mining protocol starts with a general model,

possibly very simple, which is then iteratively enriched with new extracted information. A lot of existing dynamic approaches are based on Minimum Description Length (MDL) principle which allows one to select patterns compressing the dataset at the most [11, 12, 14]. MDL has important properties: (i) it ensures lossless compression, i.e., the encoded and decoded data are guaranteed to be the same; (ii) it does not concern itself with materialized codes, the compression is means for model selection rather than compressors for reducing the amount of bits. In PM, a model is a set of patterns. The best pattern set H among sets of patterns $\mathcal{H}$ is one that minimizes the description length $L(H, D) = L(H) + L(D|H)$, where $L(H)$ is the length, in bits, of the model $H \in \mathcal{H}$ and $L(D|H)$ is the description length, in bits, of the data D encoded by the model. This version is called two-part MDL and is most appropriate for model selection [6].

In this paper we consider transactional (binary) datasets and propose an MDL-based approach for mining a small set of closed itemsets. This approach relies on Krimp [14], where both the pattern candidate set and code length function are revised to allow for a more natural reasoning on the description length from a probabilistic perspective. By contrast to the existing MDL-based approaches to PM [10, 11, 15], our method does not involve any sampling techniques and iterative adjustments of probability estimates. It relies on "likeliness" of co-occurrence of attributes under the independence model. We give an experimental proof that the proposed method returns a small set of non-redundant patterns that describe a big portion of data.

The paper is organized as follows. Section 2 introduces Krimp, the seminal algorithm for PM. In Section 3 we discuss Krimp from a probabilistic perspective and propose its modified version where patterns are formal concepts. Section 4 provides an experimental justification that the proposed approach improves quality of pattern sets. In Section 5 we conclude and give directions of future work.

2 Krimp: An Example of Pattern Mining with MDL

In Krimp [14], model H is a two-column *code table* CT, itemsets and their associated prefix codes are arranged in the left- and right-hand columns, respectively. The order of itemsets is important. The initial code table contains only singleton patterns and is called *standard code table*.

A cover function $cover(CT, I_g)$ encodes a set of items I_g of object g and returns a set of mutually disjoint itemsets $\mathcal{S} \subseteq CT$, such that $\cup_{I \in \mathcal{S}} I = I_g$. In Krimp *cover* implemented under a greedy strategy. Items I_g are initialized as uncovered and denoted by I_u. Starting from the top-itemset I_i of CT the cover function tries to cover uncovered items I_u. If $I_i \subseteq I_u$ then I_i is appended to $\mathcal{S}$ and $I_u = I_u \setminus I_i$. Once I_u is empty, *cover* returns $\mathcal{S}$. The probability distribution for $I \in CT$ is given by

$$P(I) = \frac{usage(I)}{\sum_{J \in CT} usage(J)}, \tag{1}$$

where $usage(I) = |\{g \in G \mid I \in cover(CT, I_g)\}|$. We call the probability estimates given in Formula 1 *usage-based* estimates. The length of $I \in CT$ is computed with the Shannon codes, i.e., $length(I) = -\log P(I)$.

The patterns being added to CT are chosen from a *candidate set*. The candidate set is a subset of frequent (closed) itemsets that are ordered w.r.t. their frequencies, lengths and lexicographically (this order is called *standard*). At each iteration a non-singleton itemset is considered as a candidate to CT. It is added to CT if it allows for a smaller encoding length, otherwise it is removed from the code table and candidate set.

The two-part description length of dataset D encoded with CT is given by

$$L(D, CT) = L(CT|D) + L(D|CT), \tag{2}$$

where $L(CT|D)$ is the length of the code table CT computed on dataset D and $L(D \mid CT)$ is the length of D encoded with CT. These values are given by

$$L(D|CT) = \sum_{g \in D} \sum_{I \in cover(CT, I_g)} length(I) = -\sum_{I \in CT} usage(I) \log P(I) \tag{3}$$

$$L(CT|D) = \sum_{I \in CT} length(I) = -\sum_{I \in CT} \log P(I).$$

$L(D|CT)$ is the negative log-likelihood of data and $L(CT|D)$ is the negative log-likelihood of a model CT. The code table is filled up in a greedy manner. Initially, all items in D are set as uncovered.

Example. An example of Krimp-based PM is given in Figure 1. The candidate set (CS) is closed itemsets with frequency threshold 0.25 in the standard order. At *Step 1* top-ranked pattern ac is used to cover objects g_1, g_4 and g_5, the usage of single attributes a and c decreases by 3 (to 0 and 1, respectively). ac is accepted for CT, since adding it to CT allows for a shorter description length (Formula 2). At *Step 2* de is also added to CT. At *Step 3* the last candidate bc is examined. It can cover only object g_2, that does not minimize $L(D, CT)$, thus, bc is discarded. The subset of MDL-optimal patterns is $\{ac, de\}$.

Dataset		CS
g_1	abc	ac
g_2	$bcde$	de
g_3	de	bc
g_4	$acde$	
g_5	ac	

CT	P(X)	Covering
c	$4/15$	$(a)(b)(c)$
a	$3/15$	$(b)(c)(d)(e)$
d	$3/15$	$(d)(e)$
e	$3/15$	$(a)(c)(d)(e)$
b	$2/15$	$(a)(c)$

Initial state.

CT	P(X)	Covering
ac	$3/12$	**(ac)**(b)
d, e	$3/12$	$(b)(c)(d)(e)$
b	$2/12$	$(d)(e)$
c	$1/12$	**(ac)**$(d)(e)$
a	0	**(ac)**

Step 1. ac is added.

CT	P(X)	Covering
ac	$3/9$	**(ac)**(b)
de	$3/9$	$(b)(c)$**(de)**
b	$2/9$	**(de)**
c	$1/9$	**(ac)(de)**
a, d, e	0	**(ac)**

Step 3. bc is discarded.

Fig. 1. Some stages of the Krimp algorithm. "Covering" tables show the dataset with covering by itemsets from the corresponding code table. For CTs we show left-hand columns. Probability $P(X)$ is used to compute code lengths.

3 Krimp in FCA

3.1 Motivation

In [14], it was noticed that *"for datasets with little noise the closed frequent pattern set can be much smaller and faster to mine and process"*, i.e., closed itemsets in Krimp provide better compression than arbitrary ones. However in practice, datasets are usually noisy and the number of formal concepts increases exponentially with the size of the dataset and noise rate. In this study propose to use Krimp in FCA settings to benefit better compression by concepts (see basic notions of FCA in [5]). We consider Krimp from a probabilistic perspective to make it more suitable for compression with concepts even for real-world noisy data. We illustrate weaknesses of the original Krimp in FCA settings by means of a small example.

Example. The best pattern set covers a big portion of data by a small number of patterns having a small overlapping area. A dataset composed of two patterns is given in Figure 2. Top-ranked frequent itemsets with the threshold 0.4 in the standard order are **cd**, **acd**, **cdf**, *ac*, *ad*, *cf*, *df*, **abcd**, *abc*, *abd*, *ab*, *bc*, *bd*, **cdef**, *cde*, etc. The Kripm-based covering with the given list of candidates contains an extra pattern *cd*, see Figure 2, (b). The set of closed candidates (they are highlighted in bold) is much smaller, but since only disjoint patterns are permitted the major portion of data remains uncovered, see Figure 2, (c). For closed itemsets with frequency in range $[0.2, 0.4]$ we get "true" patterns, see Figure 2, (d). However, according to the model, pattern *abcd* is more important (interesting) than *cdef*, since it has higher usage. It is a side effect of the greedy strategy rather than capture of the regularity underlying the data.

The example shows that closed itemsets provide more condensed data representation, however, PM with Krimp is affected by heuristics. The latter hampers revealing true patterns underlying the data. In this section we address the following questions: (i) Can we make Krimp less affected by heuristics? (ii) Can we further reduce the number of closed patterns and what is the most suitable way to do that? We study these questions in Sections 3.2 and 3.3, respectively.

3.2 Probability Estimates

Replacing frequency by usage in Formula 1 allows for cutting the sum in the denominator and increases the probability of the top-ranked patterns in CT. Put it differently, usage-based estimates favor the frequent patterns that bring "new information" to CT. However, the estimates are affected by the greedy cover strategy and do not always capture true data structure, e.g., an "artificial" splitting of an itemset into smaller patterns is given in Figure 2, (b) and assignment of different lengths to itemsets caused by heuristics and the lexicographical order is given in Figure 2, (d).

Replacing the usage in Formula 1 by frequency allows us to get rid of the bias introduced by the greedy strategy, but at the same time that increases the total

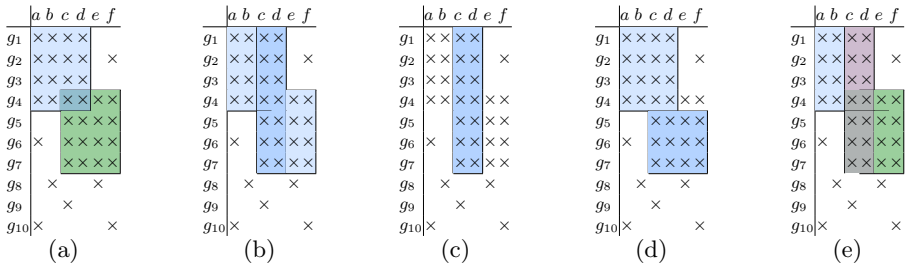


Fig. 2. Formal context and its coverings. The best data covering w.r.t. the number of patterns and covering rate is given in (a). The Krimp-based coverings are shown in (b)-(d): covering by frequent itemsets, $fr \geq 0.4$ (b), by closed frequent itemsets $fr \geq 0.4$ (c), by closed frequent itemsets, $fr \in [0.4, 0.6]$ (d). Subfigure (e) shows the covering by frequent itemsets that overlap.

length of the model and makes frequent patterns less probable. Nevertheless, this estimates more intuitive and resistant to side effects caused by heuristics. We call these estimates *frequency-based* estimates. Using these estimates is equivalent to covering with overlaps. Permitting patterns to overlap has a positive impact since it does not entail an artificial splitting of patterns.

Example. Let us demonstrate the benefits of frequency-based estimates by means of the running example. We use the same approach as before, but now we permit patterns to overlap and replace *usage* with $frequency$ in Formula 1. Covering with closed itemsets of frequency between $[0.3, 0.6]$ gives us a pattern set shown in Figure 2 (a) with correct lengths (compare with (d)). Covering by frequent closed itemsets with frequency threshold $fr \geq 0.4$ is given in Figure 2, (e). Being overlapped with *cd*, "true" itemsets *abcd* and $cdef$ are in the code table, but the pattern set is redundant (it contains an extra pattern *cd*). Thus, the proposed estimates are less affected by heuristics and are able to identify "true" patterns.

3.3 New Approach for Computing Candidates to Code Table

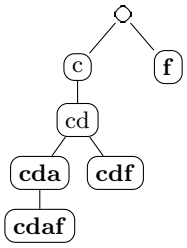
In this section we propose a new approach for candidate computing that complies with the independent pattern model (analogously to the independent attribute model).

The cover function $cover(CT, I_g)$ returns a set of patterns $I_{i_1} \cup \ldots \cup I_{i_k} = I_g$. The length of the object g is given by $length(g) = \sum_{j=i_1,\ldots,i_k} length(I_j) = -\sum_{j=i_1,\ldots,i_k} \log P(I_j)$. It means that under the given model object g is observed with probability $2^{-length(g)} = \prod_{j=i_1,\ldots,i_k} P(I_j)$. The probability is computed under the independent pattern model. We call an itemset I *likely-occurring* (LO) if $\exists I_r \subset I$, such that $P(I \setminus I_r)P(I_r) < P(I)$, i.e., itemsets I_r and $I \setminus I_r$ are more likely occur together in I than separately. Consistent with this reasoning we introduce a tree-based depth-first algorithm that sequentially grows closed

itemsets (concepts). It is closely related to FP-trees [7], however it grows trees in an attribute-wise manner (while in the FP-Growth algorithm transactions are added sequentially to grow a tree). Another important difference is an additional "likely-occurring" constraint on adding / updating nodes instead of a frequency threshold.

The basic idea is the following. Input of the algorithm is a dataset, output is a subset of closed likely-occurring itemsets. At the beginning, the attributes are ordered in the frequency-descending order. At each iteration an attribute is added to a tree. An attribute may extend the label of the current node, be added as a new node to the tree, or be ignored. An attribute m extends label I of node n, if each object that shares attributes of the node n has attribute m. Otherwise we check if itemset I and m are likely to appear together, i.e., $P(I)P(\{m\}) < P(I \cup \{m\})$, and we add a node labeled by $I \cup \{m\}$ if it is the case. The computed tree consists of likely-occurring closed itemsets.

Example. Let us consider how a tree is growing using the dataset from Figure 2. Attributes are ordered in the frequency-descending order and lexicographically, i.e., c, d, a, f, b, e. The iteration when a is added to the tree is given in Figure 3. We start from the root and add a new child node labeled by $I \cup \{m\}$ to the node labeled by I, if the new node satisfies the LO-condition. We stop traversal towards to leafs if the parent node labeled I and attribute m does not satisfy the LO-condition.



The sequence of the tree updates:

adding attribute a
$P(cd)P(a) = 0.42 < 0.50 = P(cda) \rightarrow$ create node cda

adding attribute f
$P(cda)P(f) = 0.35 < 0.40 = P(cdaf) \rightarrow$ create node $cdaf$
$P(cd)P(f) = 0.47 < 0.50 = P(cdf) \rightarrow$ create node cdf
create node f

Fig. 3. An intermediate step of tree building. Attributes a and f are added to a tree built on attributes c and d.

4 Experiments

We use the discretized datasets from LUCS-KDD repository [3], their parameters are given in Table 1. We compare pattern sets computed by Krimp, where candidate sets are frequent closed patterns with usage-based estimates, and the proposed approach, where the candidate set is likely-occurring concepts with frequency-based estimates (LOF). Pattern sets are compared following the overlapping strategy, i.e., we take the patterns and cover objects enabling the patterns to overlap. For pattern sets we study the following characteristics.

The size of a candidate set, i.e., the number of closed ($\#FC$) and LO ($\#LO$) closed itemsets for Krimp and the proposed approach, respectively. The candidate set size for the proposed approach is much smaller. More than that it does not require any thresholds.

The code table size, i.e., the number of non-singleton/singleton patterns. Small code tables are preferable. Non-singleton patterns (NS) are added from the candidate set, singleton patterns (S) are used to cover data fragments uncovered by NS-patterns. A small number of S-patterns refers to good "descriptiveness" of NS-patterns. The code tables computed using the proposed approach are smaller, 79 vs 51 for NS-patterns and 35 vs 30 for S-patterns, on average.

Pattern shape. Under shape of a pattern we mean its length, i.e., the number of attributes, and its average frequency. LOF patterns are shorter on average (5,16 vs 7,09) and more frequent (0,15 vs 0,10) than Krimp-selected ones.

Overlapping rate and rate of uncovered cells. Overlapping rate is the average number of NS-patterns that cover an item (cell) in a dataset. Non-redundant pattern sets have overlapping rate close to 1. The rate of uncovered cells, i.e., the rate of cells that are not covered by NS-patterns characterizes how well patterns describe the dataset, the uncovered cell rate close to 0 is preferable. It is important to check these values in a pair, since overlapping rate about 1 does not justify high quality of a pattern set. Krimp- and LOF-generated pattern sets on average have almost the same rate of uncovered cells, however, the overlapping rate of LOF is smaller, that allows us to conclude that LOF-pattern sets have better quality in terms of "descriptiveness". More than that, LOF pattern sets cover the same portion of data with a smaller number of patterns (see $\#NS$).

Table 1. The average values of characteristics of pattern sets following the overlapping covering strategy.

dataset	size	den.	#FC	#LO	CT size.		$\#NS/\#S$		Overlapping		Uncovered		Avg NS freq.		Avg pat.len.	
					Krimp	LOF	Krimp	LOF	Krimp	LOF	Krimp	LOF	Krimp	LOF	Krimp	LOF
anneal	898×71	0.20	9 611	1 204	83/58	52/38	2.39	2.90	0.12	0.05	0.03	0.14	11.83	6.67
breast	699×16	0.64	641	74	24/10	7/9	1.56	2.14	0.03	0.10	0.07	0.48	9.04	5.71
carEv	1728×25	0.29	12 638	1 030	94/5	36/4	1.47	1.04	0.01	0.04	0.04	0.08	3.47	2.00
ecoli	327×29	0.29	694	77	25/24	16/15	2.88	1.66	0.10	0.09	0.14	0.19	6.08	3.75
heartDis	303×50	0.29	36 738	1 683	54/45	48/37	3.27	2.29	0.13	0.11	0.2	0.13	5.09	5.38
hepat.	155×52	0.36	199 981	7 716	44/48	71/35	2.40	3.43	0.13	0.10	0.20	0.12	5.59	8.03
horseCol	368×83	0.21	228 477	30 626	101/78	140/72	2.58	2.41	0.19	0.12	0.11	0.07	3.92	5.46
iris	150×19	0.25	162	43	13/9	11/15	1.39	1.10	0.17	0.26	0.11	0.11	3.92	2.82
led7	3200×24	0.50	7 037	150	152/16	11/14	2.33	1.48	0.01	0.17	0.03	0.32	6.80	2.55
mush	8124×90	0.25	186 331	8 046	211/47	103/55	1.00	3.89	0.14	0.05	0.00	0.10	19.53	10.84
pageBl	5473×44	0.26	994	68	45/30	28/15	2.49	1.38	0.00	0.11	0.10	0.06	10.27	6.79
pima	768×38	0.22	3 203	202	50/34	32/26	5.53	1.76	0.04	0.13	0.20	0.08	5.86	4.28
ticTacToe	958×29	0.33	59 503	1 298	160/21	47/27	2.89	1.43	0.05	0.13	0.07	0.11	4.02	2.55
wine	178×68	0.20	14 554	4 757	52/65	115/55	1.91	2.21	0.29	0.12	0.10	0.05	3.90	5.43
avg	**1 666×46**	**0.31**	**54 326**	**4 070**	**79/35**	**51/30**	**2.43**	**2.08**	**0.10**	**0.11**	**0.10**	**0.15**	**7.09**	**5.16**

5 Conclusion

In this paper we consider how MDL principle can be applied in FCA settings. We consider Krimp algorithm, that is an MDL-based approach for Pattern Mining,

and study it from the probabilistic point of view. Relying on the probabilistic interpretation of MDL we propose an algorithm for generating a subset of closed itemsets (that compose a pattern search space) and new probability estimates for patterns that are used to compute pattern code length. The experiments show that the proposed approach allows for a smaller set of patterns that are less redundant than Krimp-based generated ones. The modified estimates of probability permit patterns to overlap and are less affected by the greedy cover strategy used to build a pattern set. The encoding function can be further improved by introducing an error code function for singleton patterns with low usage and more complex constraints on generating candidates.

Acknowledgment

The work was supported by the Russian Science Foundation under grant 17-11-01294 and performed at National Research University Higher School of Economics, Moscow, Russia.

References

1. Aggarwal, C.C., Han, J.: Frequent pattern mining. Springer (2014)
2. Brin, S., Motwani, R., Silverstein, C.: Beyond market baskets: Generalizing association rules to correlations. In: ACM Sigmod Record. vol. 26, pp. 265–276 (1997)
3. Coenen, F.: The lucs-kdd discretised/normalised arm and carm data library. Department of Computer Science, The University of Liverpool, UK (2003), http://www.csc.liv.ac.uk/ frans/KDD/Software/LUCS_KDD_DN
4. Gallo, A., De Bie, T., Cristianini, N.: Mini: Mining informative non-redundant itemsets. In: PKDD. pp. 438–445. Springer (2007)
5. Ganter, B., Wille, R.: In: Formal Concept Analysis: Logical Foundations. Springer Verlag Berlin, RFA (1999)
6. Grünwald, P.D.: The minimum description length principle. MIT press (2007)
7. Han, J., Pei, J., Yin, Y.: Mining frequent patterns without candidate generation. In: ACM sigmod record. vol. 29, pp. 1–12. ACM (2000)
8. Hanhijärvi, S., Ojala, M., Vuokko, N., Puolamäki, K., Tatti, N., Mannila, H.: Tell me something i don't know: randomization strategies for iterative data mining. In: Proceedings of the 15th ACM SIGKDD. pp. 379–388. ACM (2009)
9. Kuznetsov, S.O., Makhalova, T.: On interestingness measures of formal concepts. Information Sciences **442-443**, 202 – 219 (2018)
10. Mampaey, M., Vreeken, J., Tatti, N.: Summarizing data succinctly with the most informative itemsets. TKDD **6**(4), 16 (2012)
11. Siebes, A., Kersten, R.: A structure function for transaction data. In: Proceedings of SDM. pp. 558–569. SIAM (2011)
12. Smets, K., Vreeken, J.: Slim: Directly mining descriptive patterns. In: Proceedings of SDM. pp. 236–247. SIAM (2012)
13. Tatti, N.: Maximum entropy based significance of itemsets. Knowledge and Information Systems **17**(1), 57–77 (2008)
14. Vreeken, J., Van Leeuwen, M., Siebes, A.: Krimp: mining itemsets that compress. Data Mining and Knowledge Discovery **23**(1), 169–214 (2011)
15. Wang, C., Parthasarathy, S.: Summarizing itemset patterns using probabilistic models. In: Proceedings of the 12th ACM SIGKDD. pp. 730–735. ACM (2006)