Partial enumeration of minimal transversals of a hypergraph

Lhouari Nourine, Alain Quilliot and Hélène Toussaint

Clermont-Université, Université Blaise Pascal, LIMOS, CNRS, France
{nourine, quilliot, helene.toussaint}@isima.fr

Abstract. In this paper, we propose the first approach to deal with enumeration problems with huge number of solutions, when interestingness measures are not known. The idea developed in the following is to partially enumerate the solutions, i.e. to enumerate only a representative sample of the set of all solutions. Clearly many works are done in data sampling, where a data set is given and the objective is to compute a representative sample. But, to our knowledge, we are the first to deal with sampling when data is given implicitly, i.e. data is obtained using an algorithm. The experiments show that the proposed approach gives good results according to several criteria (size, frequency, lexicographical order).

1 Introduction

Most of problems in data mining ask for the enumeration of all solutions that satisfy some given property [1, 10]. This is a natural process in many applications, e.g. marked basket analysis [1] and biology [2] where experts have to choose between those solutions. An enumeration problem asks to design an output-polynomial algorithm for listing without duplications the set of all solutions. An output-polynomial algorithm is an algorithm whose running time is bounded by a polynomial depending on the sum of the sizes of the input and output.

There are several approachs to enumerate all solutions to a given enumeration problem. Johnson *et al.* [13] have given a polynomial-time algorithm to enumerate all maximal cliques or stables of a given graph. Fredman and Khachiyan [7] have proposed a quasi-polynomial-time algorithm to enumerate all minimal transversal of an hypergraph. For enumeration problems the size of the output may be exponential in the size of the input, which in general is different from optimization problems where the size of the output is polynomially related to the size of the input. The drawback of the enumeration algorithms is that the number of solutions may be exponential in the size of the input, which is infeasible in practice. In data mining, some interestingness measures or constraints are used to bound the size of the output, e.g. these measures can be explicitly specified by the user [8]. In operation research, we use quality criteria in order to consider appropriate decision [21].

In this paper, we deal with enumeration problems with huge number of solutions, *when interestingness measures are not known.* This case happens when the expert has no idea about data and knowledges that are looking for. The objective is to enumerate only a *representative sample of the set of all solutions*. Clearly many works are done in *data sampling*, where are given a data set and the objective is to compute a representative sample. To our knowledges, this idea is new for sampling when data is given implicitly, i.e. data is obtained using an algorithm. One can use the naive approach which first enumerates all the solutions and then applies sampling methods, which is not possible for huge number of solutions.

To evaluate our approach, we consider a challenging enumeration problem, which is related to mining maximal frequent item sets [1, 10], dualization of monotone boolean functions [5] and other problems [10]. We applied our approach to several instances of transversal hypergraphs [17, 20], and obtain good results.

2 Related works

Golovach *et al.* [9] have proposed an algorithm to enumerate all minimal dominating sets of a graph. First they generate maximal independent sets and then apply a flipping operation to them to generate new minimal dominating sets, where the enumeration of maximal independent sets is polynomial. Clearly, a relaxation of the flipping operation leads to a partial enumeration since the number of minimal dominating sets can be exponential in the number of minimal independent sets, e.g. cobipartie graphs. Jelassi *et al.*[12] and Raynaud *et al.*[19] have considered some kind of redundancy in hypergraphs like twin elements to obtain a concise representation. Their ideas can avoid the enumeration of similar minimal transversals of an hypergraph.

3 Transversal hypergraph enumeration

A *hypergraph* $\mathcal{H} = (V, \mathcal{E})$ consists of a finite collection $\mathcal{E}$ of sets over a finite set V. The elements of $\mathcal{E}$ are called *hyperedges*, or simply *edges*. An hypergraph is said simple if for any $E, E' \in \mathcal{E}$ $E \not\subseteq E'$. A *transversal* (or *hitting set*) of $\mathcal{H}$ is a set $T \subseteq V$ that intersects every edge of $\mathcal{E}$. A vertex x in a transversal T is said to be redundant if $T \setminus \{x\}$ is still a transversal. A transversal is *minimal* if it does not contain any redundant vertex. The set $\mathcal{T}$ of all minimal transversal of $\mathcal{H} = (V, \mathcal{E})$ constitutes together with V also a hypergraph $Tr(\mathcal{H}) = (V, \mathcal{T})$, which is called the *transversal hypergraph* of $\mathcal{H}$. We denote by $k = \sum_{E\in\mathcal{E}} \mid E \mid$.

Example 1. Consider the hypergraph $\mathcal{H} = (V, \mathcal{E})$: $V = \{1, 2, 3, 4, 5\}$ and $\mathcal{E} = \{E_1, E_2, E_3\}$ with $E_1 = \{1, 3, 4\}$, $E_2 = \{1, 3, 5\}$ and $E_3 = \{1, 2\}$. The set of all minimal transversals is $\mathcal{T} = \{\{1\}, \{2, 3\}, \{2, 4, 5\}\}$ and $k = 3 + 3 + 2 = 8$

Given a simple hypergraph $\mathcal{H} = (V, \mathcal{E})$, the transversal hypergraph enumeration problem concerns the enumeration without repetitions of $Tr(\mathcal{H})$. This problem has been intensively studied due to its link with several problems isuch as data mining and learning [3, 4, 11, 15, 18]. Recently, Kante *et al.*[14] have shown that the enumeration of all minimal transversals of an hypergraph is polynomially equivalent to the enumeration of all minimal domination sets of a graph. It is known that the corresponding decision problem belongs to coNP, but still open whether there exists an output-polynomial-time algorithm.

4 Partial transversal hypergraph enumeration

We introduce the partial (or incomplete) search algorithm for enumerating minimal transversals of an hypergraph $\mathcal{H}$. The search space is the set of all transversals which is very large. The strategy is divided into two steps:

– The initialization procedure considers a transversal T of $\mathcal{H}$ and then applies a reduction (at random) algorithm to T in order to obtain a minimal transversal T_m of $\mathcal{H}$. This step is detailed in section 4.1.
– The local search algorithm considers a minimal transversal T_m and then applies local changes to T_m in which we add and delete vertices according to some ordering of the vertices. This step is detailed in section 4.2

These steps are repeated for at most k transversals depending on the input hypergraph $\mathcal{H}$. Figure 1 illustrates the proposed approach.

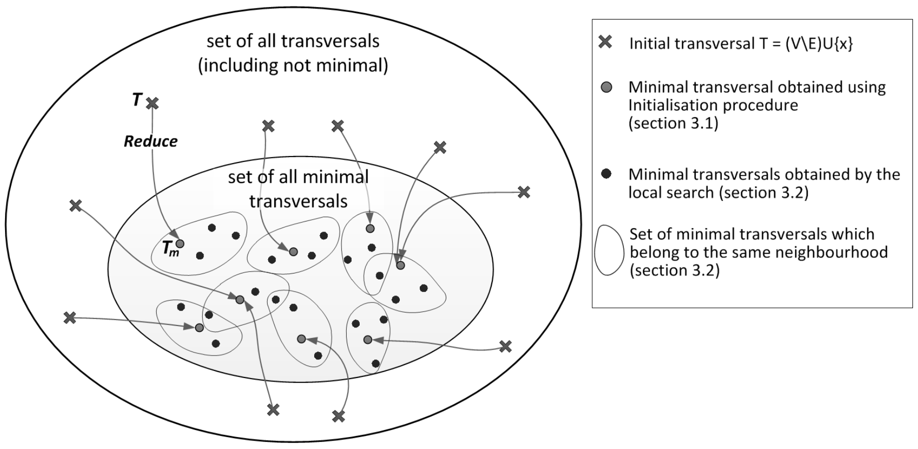

set of all transversals
(including not minimal)
T
Reduce
set of all minimal transversals
T_m
Initial transversal T = (V\E)U{x}
Minimal transversal obtained using Initialisation procedure (section 3.1)
Minimal transversals obtained by the local search (section 3.2)
Set of minimal transversals which belong to the same neighbourhood (section 3.2)

Fig. 1. Approach to partial enumeration of minimal transversals

4.1 Initialization

Let $\mathcal{H}(V, \mathcal{E})$ be the input hypergraph, $E \in \mathcal{E}$ and $x \in E$. The initialization step starts with the transversal $(V \setminus E) \cup \{x\}$ and then applies a reduction algorithm to obtain a minimal transversal. The following property shows that the set $(V \setminus E) \cup \{x\}$ is a transversal and any minimal transversal contained in $(V \setminus E) \cup \{x\}$ contains the vertex x.

Property 1. Let $\mathcal{H} = (V, \mathcal{E})$ be a simple hypergraph, $E \in \mathcal{E}$ and $x \in E$. Then $(V \setminus E) \cup \{x\}$ a transversal of $\mathcal{H}$. Moreover, any minimal transversal $T \subseteq (V \setminus E) \cup \{x\}$ will contain x.

Proof. Let $\mathcal{H} = (V, \mathcal{E})$ be a simple hypergraph and $E' \in \mathcal{E}$ with $E \neq E'$. Since $\mathcal{H}$ is simple then there exists at least one element $y \in E'$ such that $y \notin E$. So $y \in (V \setminus E)$ and thus $E' \cap (V \setminus E) \neq \emptyset$. We conclude that $(V \setminus E) \cup \{x\}$ is a transversal since $x \in E$.

Now let $T \subseteq (V \setminus E) \cup \{x\}$ be a minimal transversal. Then $E \cap (V \setminus E) \cup \{x\} = \{x\}$ and thus x must belong to T otherwise T does not intersect E. □

According to property 1, we can apply the initialization procedure to any pair (x, E) where $E \in \mathcal{E}$ and $x \in E$. In other words, the initialization is applied to at most k transversals of $\mathcal{H}$ as shown in Algorithm 1.

Algorithm 1: Initialization

```
Input  : A hypergraph $\mathcal{H}(V,\mathcal{E})$ and $\sigma$ an ordering of $V$
Output: A sample of minimal transversals
begin
    $STRANS = \emptyset$;
    for $E \in \mathcal{E}$ do
        for $x \in E$ do
            $T = (V \setminus E) \cup \{x\}$;{Initial transversal}
            $T_m = Reduce(T, \sigma)$;
            $STRANS = STRANS \cup \{T_m\}$;
    return($STRANS$);
```

Now we describe the reduction process, which takes a transversal T and a random ordering σ of V and returns a minimal transversal T_m. Indeed, we delete vertices from T according to the ordering σ until we obtain a minimal transversal.

Algorithm 2: Reduce(T, σ)

```
Input  : A transversal $T$ and an ordering $\sigma = \sigma_1 ... \sigma_{|V|}$ of the vertices of
         $\mathcal{H}$.
Output: A minimal transversal
for $i = 1$ to $|V|$ do
    if $T \setminus \{\sigma_i\}$ is a transversal then
        $T \leftarrow T \setminus \{\sigma_i\}$ ;
Return($T$);
```

Example 2 (continued). Suppose we are given the hypergraph in example 1 and $\sigma = (1,2,3,4,5)$ for the input to Algorithm 1. First, it takes the hyperedge $E = \{1,3,4\}$ and for $x = 1$ we obtain the minimal transversal $\{1\}$, for $x = 3$ we obtain $\{2,3\}$ and for $x = 4$ we obtain $\{2,4,5\}$. Then the algorithm continue with the hyper edges $\{1,3,5\}$ and $\{1,2\}$. Finally the algorithm returns $STRANS = \{\{1\},\{2,3\},\{2,4,5\}\}$, i.e. the other iterations do not add new minimal transversals.

Theorem 1. *Algorithm 1 computes at most k minimal transversals of an input hypergraph $\mathcal{H} = (V,\mathcal{E})$.*

Proof. The initialization procedure considers at most k minimal transversals of $\mathcal{H}$. Since a minimal transversal can be obtained several times, the result follows. $\square$

The following proposition shows that any minimal transversal of the hypergraph $\mathcal{H} = (V,\mathcal{E})$ can be obtained using the initialization procedure. Indeed, the choice of the ordering σ is important in the proposed strategy.

Proposition 1. *Let $\mathcal{H} = (V,\mathcal{E})$ be an hypergraph and T be a minimal transversal of $\mathcal{H}$. Then, there exists a total order σ, $E \in \mathcal{E}$ and $x \in E$ such that $T = Reduce((V \setminus E) \cup \{x\}, \sigma)$.*

Proof. Let T be a minimal transversal of $\mathcal{H} = (V,\mathcal{E})$. Then there exists at least one hyperedge $E \in \mathcal{E}$ such that $T \cap E = \{x\}$, $x \in V$, otherwise T is not minimal. Thus $T \subseteq (V \setminus E) \cup \{x\}$. Now, if we take the elements that are not in T before the elements in T in σ, the algorithm $Reduce((V \setminus E) \cup \{x\}, \sigma)$ returns T. $\square$

The initialization procedure guaranties that for any vertex $x \in V$ at least one minimal transversal containing x is listed. The experiments in section 5, shows the sample of minimal transversals obtained by the initialization procedure is a representative sample of the set of all minimal transversals.

4.2 Local search algorithms

The local search algorithm takes each minimal transversal found in the initialization step and searches for new minimal transversals to improve the initial solution. The search of neighbors is based on vertices orderings.

Let $\mathcal{H} = (V,\mathcal{E})$ be an hypergraph and $x \in V$. We define the frequency of x as the number of minimal transversals of $\mathcal{H}$ that contain x. The algorithm takes a minimal transversal T and a bound $Nmax$ which bounds the number of iterations and the number of neighboors generated by T. Each iteration of the while loop, starts with a minimal transversal T and computes two orderings as follows:

- σ^c is an ordering according to the increasing order of frequency of vertices in $V \setminus T$ in minimal transversals already obtained by the current call. This ordering has a better coverage of the solution set, i.e. by keeping the rarest vertices in the transversals.

– σ is a random ordering of the vertices in T.

```
Algorithm 3: Neighboor($T, Nmax$)
Input  : A minimal transversal $T$ of $\mathcal{H} = (V, \mathcal{E})$ and an integer $Nmax$
Output: A set of minimal transversals
$Q = T$;
$i = 1$;
while $i \leq Nmax$ do
    $\sigma^c \leftarrow$ the set $V \setminus T$ sorted in increasing order of frequency of vertices
    in minimal transversals in $Q$;
    $\sigma \leftarrow$ sort $T$ at random;
    Add elements the elements in $\sigma^c$ to $T$ until a vertex $x \in T \setminus \sigma^c$
    becomes redundant in $T$;
    $T =$Reduce($T, \sigma$);
    $Q = Q \cup \{T\}$;
    $i = i + 1$;
return($Q$);
```

Now we give the global procedure of the proposed approach.

```
Algorithm 4: Global procedure for partial enumeration of minimal transversals
Input  : An hypergraph $\mathcal{H} = (V, \mathcal{E})$ and an integer $Nmax$
Output: A sample of minimal transversals of $\mathcal{H}$
$\sigma =$choose a random ordering of $V$;
$STRAN = Q = Initialization(\mathcal{H}, \sigma)$;
while $Q \neq \emptyset$ do
    $T =$ choose a minimal transversal $T$ in $Q$;
    $STRANS = STRANS \cup Neighboor(T, Nmax)$;
Return($STRANS$);
```

In the following, we describe experiments to evaluate the results that have been obtained.

5 Experimentation

The purpose of the experiments is to see if the proposed approach allow us to generate a *representative* set of solutions. For this reason, we have conducted the experiments on two different classes of hypergraphs (see [20]) for which the number of minimal transversals is huge compared to the size of the input. We use Uno's Algorithm SHD (Sparsity-based Hypergraph Dualization, ver. 3.1) [20], to enumerate all minimal transversals. The experiments are done using linux CentOS cpu Intel Xeon 3.6 GHz and C++ language.

In the following, we denote $\mathcal{T}_{partial}$ the set of minimal transversals generated by Algorithm 4, and $\mathcal{T}_{exact}$ the set of all minimal transversals. First, we analyze the percentage $\frac{\mathcal{T}_{partial}}{\mathcal{T}_{exact}}$ and then we evaluate the representativeness of the sample $\mathcal{T}_{partial}$.

5.1 The size of $\mathcal{T}_{partial}$

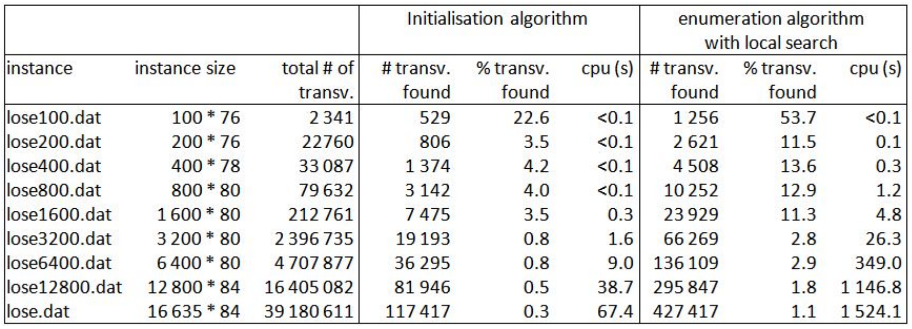
We will distinguish between minimal transversals that are obtained using Algorithm 1 (or the initialization procedure) and those that are generated using the local search. For these tests we set the maximal number of neighboors $Nmax$ to 3.
Tables 1 and 2 show the results for the two classes of hypergraph instances, namely "lose" and random "p8".

The first three columns have the following meaning:

- *instance*: instance name.
- *instance size*: the size of the instance (number of edges $\times$ number of vertices).
- *total # of transv.*: the exact number of minimal transversals $\mid \mathcal{T}_{exact} \mid$.

The second (resp. last) three columns give the results for the initialization procedure (resp. Global algorithm):

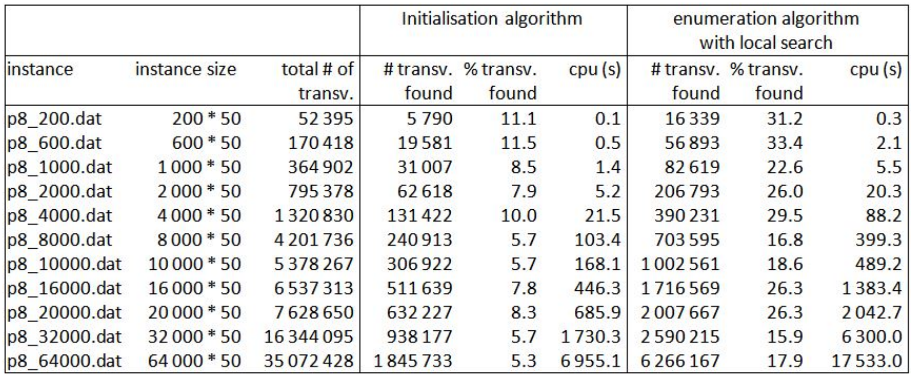
- *# transv. found*: the number of minimal transversals found.
- *% transv. found*: the percentage of minimal transversals found.
- *cpu (s)*: the run time in seconds



			Initialisation algorithm			enumeration algorithm with local search		
instance	instance size	total # of transv.	# transv. found	% transv. found	cpu (s)	# transv. found	% transv. found	cpu (s)
lose100.dat	100 * 76	2 341	529	22.6	<0.1	1 256	53.7	<0.1
lose200.dat	200 * 76	22760	806	3.5	<0.1	2 621	11.5	0.1
lose400.dat	400 * 78	33 087	1 374	4.2	<0.1	4 508	13.6	0.3
lose800.dat	800 * 80	79 632	3 142	4.0	<0.1	10 252	12.9	1.2
lose1600.dat	1 600 * 80	212 761	7 475	3.5	0.3	23 929	11.3	4.8
lose3200.dat	3 200 * 80	2 396 735	19 193	0.8	1.6	66 269	2.8	26.3
lose6400.dat	6 400 * 80	4 707 877	36 295	0.8	9.0	136 109	2.9	349.0
lose12800.dat	12 800 * 84	16 405 082	81 946	0.5	38.7	295 847	1.8	1 146.8
lose.dat	16 635 * 84	39 180 611	117 417	0.3	67.4	427 417	1.1	1 524.1

Table 1. Results for all "*lose*" instances

According to these tests, we can see that the percentage of minimal transversals found using the initialization procedure is very low, but it decreases as far as the size of $\mathcal{T}_{exact}$ increases. Clearly, this percentage is strongly related to the input. Indeed, the number k (entropy) of the hypergraph increases according to the size of the input hypergraph. We can also see that the local search increases significantly the number of solutions found by a factor 2 to 2.5 approximatively.



			Initialisation algorithm			enumeration algorithm with local search		
instance	instance size	total # of transv.	# transv. found	% transv. found	cpu (s)	# transv. found	% transv. found	cpu (s)
p8_200.dat	200 * 50	52 395	5 790	11.1	0.1	16 339	31.2	0.3
p8_600.dat	600 * 50	170 418	19 581	11.5	0.5	56 893	33.4	2.1
p8_1000.dat	1 000 * 50	364 902	31 007	8.5	1.4	82 619	22.6	5.5
p8_2000.dat	2 000 * 50	795 378	62 618	7.9	5.2	206 793	26.0	20.3
p8_4000.dat	4 000 * 50	1 320 830	131 422	10.0	21.5	390 231	29.5	88.2
p8_8000.dat	8 000 * 50	4 201 736	240 913	5.7	103.4	703 595	16.8	399.3
p8_10000.dat	10 000 * 50	5 378 267	306 922	5.7	168.1	1 002 561	18.6	489.2
p8_16000.dat	16 000 * 50	6 537 313	511 639	7.8	446.3	1 716 569	26.3	1 383.4
p8_20000.dat	20 000 * 50	7 628 650	632 227	8.3	685.9	2 007 667	26.3	2 042.7
p8_32000.dat	32 000 * 50	16 344 095	938 177	5.7	1 730.3	2 590 215	15.9	6 300.0
p8_64000.dat	64 000 * 50	35 072 428	1 845 733	5.3	6 955.1	6 266 167	17.9	17 533.0

Table 2. Results for all "*p8*" instances

But it remains coherent with the chosen value $Nmax = 3$. It argues that the local search finds other transversals that are not found either by the initialization procedure nor previous local search. Notice that the parameter $Nmax$ can be increased whenever the size of $\mathcal{T}_{partial}$ is not sufficient.

5.2 The representativeness of $\mathcal{T}_{partial}$

To evaluate the representativeness of $\mathcal{T}_{partial}$, we consider three criteria:

- Size of the minimal transversals in $\mathcal{T}_{partial}$.
- Frequency of vertices in $\mathcal{T}_{partial}$.
- Lexicographical rank of the minimal transversals in $\mathcal{T}_{partial}$.

Each criteria is illustrated using a bar graph for two instances from different classes. The bar graphs in figures 2 and 3 are surprising. Indeed the bar graphs vary nearly in the same manner with respect to the initialization and the local search algorithm for all the considered criteria.

Figures 2(a) 3(a) show that the percentage of minimal transversals of each size (found either by the initialization procedure and local search) fits the percentage of all minimal transversals that are found.

Figures 2(b) and 3(b) show that the same analysis holds when ordering minimal transversals lexicographically (e.g. based on a total ordering of vertices). Clearly, the lexicographical rank of a minimal transversal belongs to the interval $[1..2^{|V|}]$. For visualization aspect, we divide this interval into $|V|$ subintervals, where the subinterval i contains the number of minimal transversals with a rank $r \in [\frac{2^{|V|}}{|V|}i; \frac{2^{|V|}}{|V|}(i+1)[$, $(i = 0, \dots, |V| - 1)$.

Figures 2(c) and 3(c) confirm this behavior when considering frequency of vertices. Indeed, frequency of vertices in minimal transversals in $\mathcal{T}_{partial}$ is the same when considering all minimal transversals, i.e.. the set $\mathcal{T}_{exact}$.

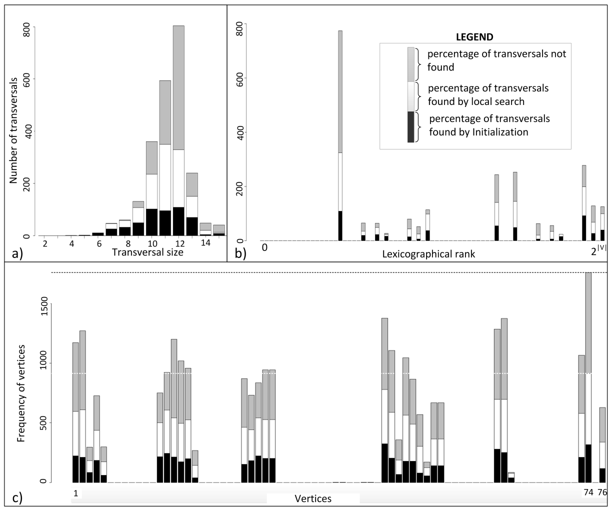


Fig. 2. The bar graph for "lose100": a) number of minimal transversals per size b) number of minimal transversals according the lexicographical rank; c) Frequency of vertices in minimal transversals.

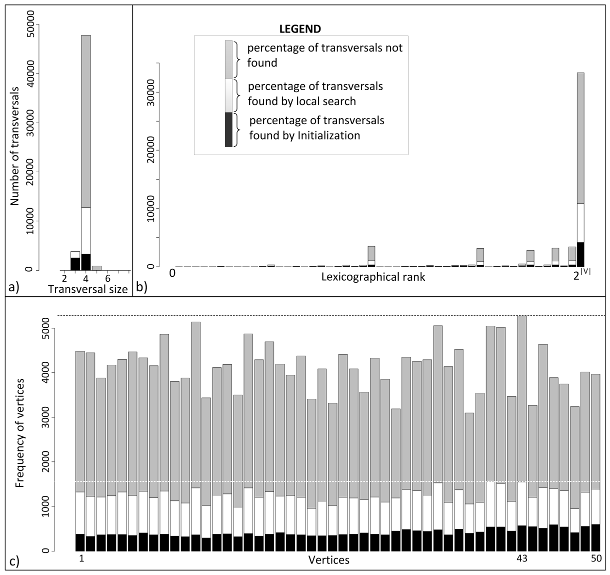


Fig. 3. The bar graph for "p8_200"; a) number of minimal transversals per size; b) number of minimal transversals according the lexicographical rank; c) Frequency of vertices in minimal transversals.

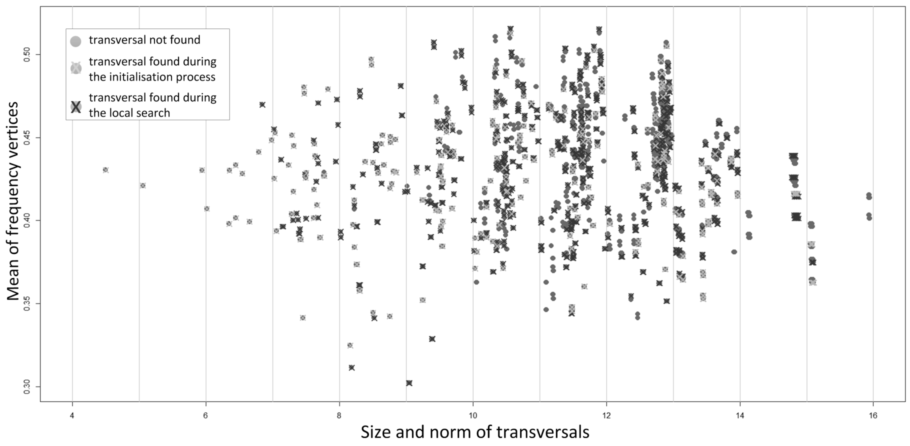


Fig. 4. Visualizing the solutions space of "lose100". The abscissa is given by the size of the transversal (transversals of the same size are spread out using a norm) and the ordinate corresponds to the frequency of its vertices.

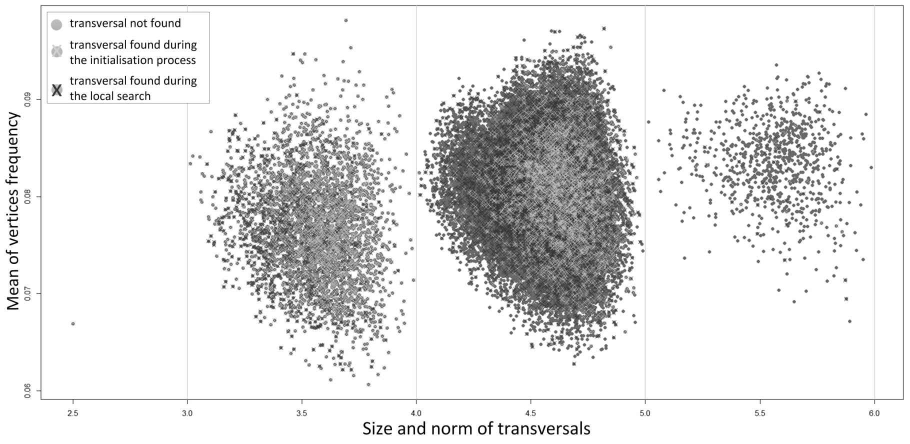


Fig. 5. Visualizing the solutions space of "p8_200". The abscissa is given by the size of the transversal (transversals of the same size are spread out using a norm) and the ordinate corresponds to the frequency of its vertices.

Figures 4 and 5 show that the set $\mathcal{T}_{partial}$ is also representative even when considering minimal transversals with the same size. Indeed, minimal transversals having the same size are spread out using a norm. We notice that the points corresponding to minimal transversals in $\mathcal{T}_{partial}$ are scattered in the image.

This experiment allows us to conclude that the sample $\mathcal{T}_{partial}$ produced by Algorithm 4 is representative relatively to the criteria under consideration. Other results can be found in http://www2.isima.fr/~toussain/

6 Conclusion and discussions

We are convinced that the initialization procedure is the most important in this approach. Indeed, the set of minimal transversals obtained using this procedure is a representative sample, since it garantee that for any vertex of the hypergraph there is at least one minimal transversal which contains it (see property 1). Moreover the local search procedure can be used to increase the number of solutions, and as we have seen in the experiments, it keeps the same properties as the initialization procedure.

We hope that this approach improves enumeration in big data and will be of interests to the readers to investigate heuristics methods [6] for enumeration problems.

This paper opens new challenges related to partial and approximate enumeration problems. For example, given an hypergraph $\mathcal{H} = (V, \mathcal{E})$, is there an algorithm that for any given ε, it enumerates a set $\mathcal{T}_{partial} \subseteq Tr(\mathcal{H})$ such that $(1-\varepsilon)|Tr(\mathcal{H})| \leq |\mathcal{T}_{partial}| \leq |Tr(\mathcal{H})|$? We also require that the algorithm is output-polynomial in the sizes of $\mathcal{H}$, $\mathcal{T}_{partial}$ and $\frac{1}{\varepsilon}$. To our knowledge, there is no work on approximate algorithms for enumeration problems, but results on approximate counting problems may be applied [16].

Acknowledgment: This work has been funded by the french national research agency (ANR DAG project, 2009-2013) and CNRS (Mastodons PETASKY project, 2012-2015).

References

1. R. Agrawal, T. Imielinski, and A. Swami. Mining associations between sets of items in massive databases. In *ACM SIGMOD 1993, Washington D.C.*, pages 207–216, 1993.
2. J. Y. Chen and S. Lonardi. *Biological Data Mining.* Chapman and Hall/CRC, 2009.
3. T. Eiter and G. Gottlob. Identifying the minimal transversals of a hypergraph and related problems. *SIAM J. Comput.*, 24(6):1278–1304, 1995.
4. T. Eiter, G. Gottlob, and K. Makino. New results on monotone dualization and generating hypergraph transversals. *SIAM J. Comput.*, 32(2):514–537, 2003.
5. T. Eiter, K. Makino, and G. Gottlob. Computational aspects of monotone dualization: A brief survey. *Discrete Applied Mathematics*, 156(11):2035–2049, 2008.

6. T. Feo and M. Resende. Greedy randomized adaptive search procedures. *Journal of Global Optimization*, 6(2):109–133, 1995.
7. M. Fredman and L. Khachiyan. On the complexity of dualization of monotone disjunctive normal forms. *Journal of Algorithms*, 21:618–628, 1996.
8. L. Geng and H. J. Hamilton. Interestingness measures for data mining: A survey. *ACM Comput. Surv.*, 38(3), Sept. 2006.
9. P. Golovach, P. Heggernes, D. Kratsch, and Y. Villanger. An incremental polynomial time algorithm to enumerate all minimal edge dominating sets. *Algorithmica*, 72(3):836–859, 2015.
10. D. Gunopulos, R. Khardon, H. Mannila, S. Saluja, H. Toivonen, and R. S. Sharm. Discovering all most specific sentences. *ACM Trans. Database Syst.*, 28(2):140–174, 2003.
11. D. Gunopulos, R. Khardon, H. Mannila, and H. Toivonen. Data mining, hypergraph transversals, and machine learning. In *PODS*, pages 209–216, 1997.
12. M. Jelassi, C. Largeron, and S. Ben Yahia. Concise representation of hypergraph minimal transversals: Approach and application on the dependency inference problem. In *Research Challenges in Information Science (RCIS), 2015 IEEE 9th International Conference on*, pages 434–444, May 2015.
13. D. S. Johnson, C. H. Papadimitriou, and M. Yannakakis. On generating all maximal independent sets. *Inf. Process. Lett.*, 27(3):119–123, 1988.
14. M. M. Kanté, V. Limouzy, A. Mary, and L. Nourine. On the enumeration of minimal dominating sets and related notions. *SIAM Journal on Discrete Mathematics*, 28(4):1916–1929, 2014.
15. L. Khachiyan, E. Boros, K. M. Elbassioni, and V. Gurvich. An efficient implementation of a quasi-polynomial algorithm for generating hypergraph transversals and its application in joint generation. *Discrete Applied Mathematics*, 154(16):2350–2372, 2006.
16. J. Liu and P. Lu. Fptas for counting monotone cnf. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '15, pages 1531–1548. SIAM, 2015.
17. K. Murakami and T. Uno. Efficient algorithms for dualizing large-scale hypergraphs. *Discrete Applied Mathematics*, 170:83–94, 2014.
18. L. Nourine and J.-M. Petit. Extending set-based dualization: Application to pattern mining. In *ECAI*, pages IOS Press ed, Montpellier, France, 2012.
19. O. Raynaud, R. Medina, and C. Noyer. Twin vertices in hypergraphs. *Electronic Notes in Discrete Mathematics*, 27:87–89, 2006.
20. T. Uno. http://research.nii.ac.jp/ uno/dualization.html.
21. C. A. Weber, J. R. Current, and W. Benton. Vendor selection criteria and methods. *European Journal of Operational Research*, 50(1):2 – 18, 1991.