ISSN 2395-8618

Frequent Patterns Mining for the Satisfiability Problem

Celia HIRECHE, Habiba DRIAS, Neyla Cherifa BENHAMOUDA
USTHB, Department of Computer Science, LRIA, Algiers, Algeria
Email: chireche@usthb.dz, hdrias@usthb.dz, benhamoudaneyla@gmail.com
www.lria.usthb.dz

***Abstract*—This paper presents a novel approach for solving the Satisfiability problem by reducing its complexity. First, an improved, 'divide and conquer'version of the Apriori algorithm is introduced. It consists in dividing the problem instance into two or more if necessary, sub-instances and then in executing an ameliorated version of the Apriori algorithm for extracting the frequent variables appearing in the sub-instances.**
These most frequent variables are grouped into clusters and the corresponding problem are considered for resolution. Once done, the clusters can be shown as new smaller instances that are solvable separately using either the DPLL procedure or the BSO algorithm according to the number of variables to be solved.

***Index Terms*—Mining Frequent Patterns, Apriori, Meta-Apriori, Clustering, NP-Complete problems, Problem Solving, Satisfiability Problem, Complexity**

I. Introduction

ONE of the most important tasks of Data Mining[1] is the reducing complexity of data while keeping the integrity of the later.
Three kinds of treatment are used for this purpose, Classification and clustering which consist in dividing the data into small groups according to a certain training data for the classification, and according to the similarities between the elements for the clustering. The third process being the frequent patterns mining, which consists in extracting the most frequent items repeated together.
In this work, we aim at reducing the complexity of the satisfiability problem[2], the most known NP-Complete problem that arouses the most interest of the computational complexity community. The issue consists in finding an assignment to the variables to satisfy an instance represented as a CNF (Conjunctive Normal Form) Boolean formula.
There are two categories of solving approaches, the complete and the incomplete methods[3]. The first guarantees to find the optimal solution if it exists or proves that the problem cannot have a solution if appropriate. These methods cannot cope with large problem instances and would generate a combinatorial explosion and timeout calculation whatever the machine performance.
To get around these problems, the scientific community developed new methods based on approximation. These methods do not guarantee to find a solution even if it exists.
These constrains motivate us to think about a preprocessing -pretreatment- to execute before the resolution. This preprocessing step consists in using a frequent mining patterns to reduce the problem complexity by dividing it into sub problems (clusters) that can be solved separately in a second step, using a complete algorithm and an incomplete one.
The remainder of this document is organized as follows. The next section presents some interesting works related to the satisfiability problem and the Apriori algorithm. The satisfiability problem is then introduced in the third section. The fourth section is dedicated to the presentation of Bees Swarm Optimization Algorithm. The fifth section is consecrated to the Meta-Apriori algorithm, prior to presenting the Apriori-Clustering resolution in the sixth section. The conducted experiments and the obtained results are presented in the last section. Conclusions are finally summarized and some perspectives are suggested.

II. Related Works

Nowadays, several algorithms and solvers exist to get over the satisfiability problem, namely SAT.
The first category of SAT solvers deals with complete algorithms that are able to yield either a satisfying solution or a proof that such a solution does not exist. One of the most known and studied complete solver is the Davis-Putnam-Logemann-Loveland(DPLL)[4]. This backtracking algorithm recursively assigns a truth value to a variable and eliminate all clauses that contains it until being able to check whether the formula is satisfied. Several existing solvers are based on the DPLL algorithm.
An extension of the DPLL is introduced in the Conflict Driven Clauses Learning (CDCL)[3], in which a new clause is learnt when a conflict occurs while assigning values to the variables. Tens of CDCL based solvers exist nowadays[3].
In[5], the authors introduced the first parallel portfolio[6] SAT solver using a multicore architecture, allowing a communication between the four used cores (CDCL solvers) through lockless queues. These solvers are configured differently according to:

- the restart policy, using either a dynamic restart policy depending on the average size of the two last back-jumps, or an arithmetic one,
- the selecting heuristic, where they increase the random noise of the Variable State Independent Decaying Sum (VSIDS) heuristic to diversify the selecting process,
- the polarity using a progress saving politic, saving the polarity of variables between conflict and back-jump

ISSN 2395-8618

level, and a statistic polarity according to the occurrences of each literal (variable and its negation),
- the learning process, using the basic CDCL's implication graph[7] and introducing a novel arc, called inverse arc, that takes into consideration the satisfied clause,
- and finally, the clause sharing, which allows the communication between the cores.

All of these cores will deal with the whole base of clauses and try to solve it using different manner (configuration), which can be very time consuming. It would be a better choice to divide the problem so that the different solvers can cooperate by solving the different parts separately and save time.

The second category of solvers are based on incomplete algorithms. Their principle is to learn the problem's characteristics in order to guide the search without covering the whole search area.

One of the first and most studied incomplete algorithms is the Stochastic Local Search (SLS)[8], such as the famous GSAT. Firstly introduced in[9], the algorithm starts by assigning a truth value to all the variables, then generates the neighbourhood of the current solution by flipping the variables one by one. The choice of the variable to be flipped is made by selecting randomly an unsatisfied clause, and picking the variable that maximizes the number of newly satisfied clauses and minimize the number of newly unsatisfied clauses.

An extension of the GSAT was proposed in [10], named WALKSAT, in which the choice of the variable to be flipped is made by selecting from a random unsatisfied clause, the variable satisfying the GSAT condition with a probability p, and with a probability 1-p picks a variable randomly.Since then a family of WALKSAT solvers were created[11]. As other solvers based on SLS algorithm.

In[12], S. Cai et al. introduced a new two-mode SLS solver that combines be- tween two flip strategy. The first one being the CCA (Configuration Checking with Aspiration) heuristic, which does not allow flipping a variable if its configuration (neighbours) does not change since its last flip. And allow the flipping of those whose score is significant (their flip decreases the number of unsatisfied clauses significantly). If these kinds of variables do not exist, the flip strategy used is switched to the focused local search mode which selects a variable from a random unsatisfied clause. This solver has been combined with other solvers like glucose CCAnr+Glucose[13], which was presented in the SAT's competition 2014, and so others.

However, visiting the neighbourhood of each variable and counting the score of each variable at each step with the probability to switch to a random mode after this process is very time consuming.

In [14], the authors, being inspired by the Frankenstein's novel, introduced a solver which consists on a combination of existing high performance SLS SAT's techniques (solvers) with some mechanism that they introduced, using an automated construction process. The solver includes five parts or blocks, where the first is used for diversification -initializing selecting policy-. The three next parts are consecrated to the resolution itself; WALKSAT's based solvers for the second part, dynamic local search (penalties associated to the clauses) solvers for the third and GWSAT (joining GSAT-WALKSAT) for the fourth part. The fifth block is used to up to date data structures.

Even on this solver, selecting the solver to be used for resolution, can be very time consuming because of the diversity of problems instances.

Data Mining techniques were for the first time used for solving SAT in[15][16], where the authors used clustering[1] methods to reduce the problem instance into many groups using an intuitive method[15], creating a cluster for every new variable. A Genetic-K-Means where the clusters centres are generated using the genetic algorithm[1], and the classification is made using the K-means algorithm[1].

Many other Data Mining techniques exist, including Frequent Patterns Mining which consists on finding the patterns (items, variables, ...) that are repeated together. One of the most known algorithms used for mining frequent patterns is the Apriori Algorithm[1][17].

Introduced in[17], Apriori consists in finding the set of k-Itemsets that occur the most. A set of itemsets candidates is extracted to be validated by scanning the whole base which is very time consuming.

In[18], the authors considered the item with the minimum support, minimizing then the database scans and reducing the runtime. They also used the FP-growth algorithm in order to reduce the memory space.

III. THE SATISFIABILITY PROBLEM

Being one of the most studied NP-Complete problems, all eyes are turned to the Satisfiability problem, for its complexity and its impact on the whole NP- Completeness.

The Boolean Satisfiability problem[2], SAT, is to decide whether or not there is a satisfying assignment to the set of variables x making a Boolean formula (x) true. This formula being in conjunctive normal form (CNF) that is a conjunction of clauses, where each clause is a disjunction of literals, a literal being either a variable or its negation. In other words, find an assignment (true value to each variable) that satisfies all the clauses in the same time. Reminding that a clause is said to be satisfied (true) if and only if at least one of its variables is satisfied (true for a positive literal and false for a negative one).

The formal definition of the problem is shown in the following instance and question pair:

- Instance: m clauses over n literals
- Question: Is there any assignment of variables that satisfies all the clauses?

Example:

Let consider the following set of variables V = {X1, X2, X3} and the following set of clauses C = {C1, C2, C3, C4} defined as follow:

- C1 = X1, X2
- C2 = X2, -X3

ISSN 2395-8618

- C3 = -X1, -X3
- C4=X1, X2, X3

Note that the '-'means the negative form of the variable.

IV. BEES SWARM OPTIMIZATION FOR SOLVING SAT

The Bees Swarm Optimization Algorithm (BSO)[19] is a population-based search algorithm simulating the behaviour of bees when looking for food[20]. In fact, Karl Von Fris - 1946- observed that it is through a specified dance that the bees communicate the distance and the direction of the food source. The richest the source, the vigorous the dance so that when two sources of equal distance are found, the bees exploit the most potential area.

By analogy to the animals (reign), the BSO algorithm works as follow: First, an initial bee, named BeeInit generates a random solution named Sref, from which a search space namely SearchArea is determined using a diversification strategy. Each bee considers a solution of this SearchArea as a starting point to its local search and communicates the best solution found in a table called Dance. The best solution from this table is taken as the references solution (Sref) and the cycle restarts until no better solution to be found.

Algorithm 1 Bees Swarm Optimization Algorithm for SAT

```
Bees : table of bees
Solution : Variables ; Solution ; Evaluation
Sref ←Random Boolean Solution
while non stagnation do
    TL ←Sref
    SearchArea (Sref) : Random generation
    for ( i = 0 ←Bees count) : assign a solution of SearchArea to each bee do
        Local search
        Dance ←Best local search
    end for
    Sref ←Best solution from Dance using fitness procedure(attribute a point to the evaluation for each satisfied clause)
end while
```

V. META APRIORI

The Apriori[17][1] Algorithm is the most popular algorithm in data mining for extracting the frequent itemsets. It detects from a set of transactions, the items that are repeated the most together. It starts by extracting the singles frequent items to then recursively self-join the resulting itemsets until no longer itemset to be extracted (k-itemsets).

The Meta-Apriori algorithm[21] includes three steps; partitioning step, Apriori step, and fusion step. The partitioning step consists in dividing the database into two clusters or more if necessary. This partitioning is made by classifying the transactions according to the frequency of their items, having as result, almost the same items in both of the clusters with the same frequency. The Apriori step, as its name indicated, is the application of an improved Apriori algorithm on previous clusters. These improvements were introduced to reduce the Aprioris time consuming, and consist in:

- A vertical representation for a better representation of the database and the set of candidate itemsets, so that the entry of the structure is the item (vari- able) and the contents is the set of transactions (clauses) where it appears.
- Validation of a candidate when its frequency is equal to the support (the condition is satisfied).
- Elimination of the items that appear less than the minimum support, and the transactions containing a lesser number of item than the current itemset size.

To end with the fusion step, where the itemsets of both of the clusters (of all clusters if more than two) are joined.

Algorithm 2 Meta-Apriori Algorithm

```
Variables :
CS1, CS2 : sub-transaction base
Ci : ith itemsets candidates
Input :
TB: transaction base
MinSUp: minimum support
Output:
FPB: frequent patterns base
procedure DIVIDING(TB,CS1,CS2)
    for i :0 to TB length do
        CS1 ←TB[i] or CS2 ←TB[i] according to the frequency of the items on both of CS1, CS2
        Return CS1, CS2
    end for
end procedure
procedure APRIORI(TB,FBP)
    extract 1-itemset and validate
    while (itemset to be extracted ) do
        Ci= self join the itemsets (new candidates)
        Validation(Ci)
        i=i+1
    end while
    Return FBP
end procedure
```

VI. META-APRIORI CLUSTERING FOR SAT SOLVING

In this section, we propose a novel algorithm for solving the SAT problem, where data mining collaborates with a complete resolution algorithm and incomplete one.

With the aim of reducing the problems complexity, the problems instance is divided into two groups (clusters) using, as presented in the previous section, a frequency clustering, to then execute the improved Apriori algorithm, giving as result a set of k most frequent itemsets.

Two methods are then possible; the first method merges (fusion) the itemsets of the two instances, on one unique set of itemsets which is used for creating clusters by using these itemsets as cluster's centre (If two itemsets share more than half of the elements, the two centres are merged). The

ISSN 2395-8618

problem's instance is then classified -into these clusters- using the Hamming distance[22] , so that an itemset is classified in the cluster with which it shares the maximal number of items. These clusters can be seen as new problem's instances which can be solved either by using the DPLL algorithm or the BSO algorithm according to the number of variables to be solved. The resulted solutions are then merged. The second method follows a top-down schema. Contrarily to the first, it does not merge the itemsets of both instances groups but continues splitting the instances using the same process as that described in the first method (creating the clusters using the frequent itemsets, and classify the two instances separately). Once all clusters created, the resolution of each of the clusters is made using the DPLL and the BSO algorithms. The solutions obtained by all the clusters are then combined to yield the general problem's solution.
The following figure illustrates the two presented methods.

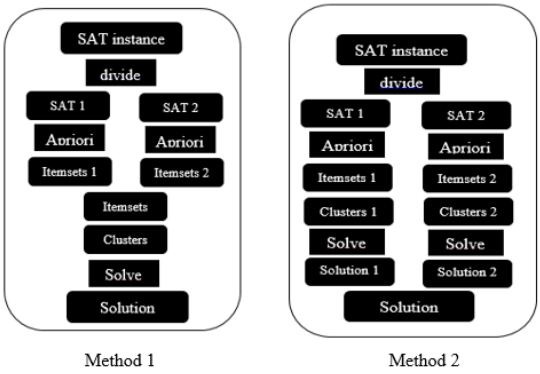

Fig. 1. META-APRIORI CLUSTERING FOR SAT SOLVING.

VII. EXPERIMENTS

To show the efficiency of the proposed approach, some experimentations were conducted on an i7 2.40 Ghz 4Go and the implementation on Microsoft visual studio CSharp 2013, and were conducted on some benchmarks which are presented in the Table 1.

TABLE I
BENCHMARKS DESCRIPTION

Benchmark	Solvability	Number of variables	Number of clauses
Benchmark1 [23]	Unsolved	99	8691
Benchmark 2 [23]	Solved	230	9975
Benchmark 3 [23]	Solved	440	9291
Benchmark 4 [23]	Solved	240	10409
Benchmark 5 [23]	Solved	260	11276
IBM 7 [24]	Solved	8710	39374
GALILEO 8 [24]	Solved	58074	276980
GALILEO 9 [24]	Solved	63624	307589

Table.1 describes the characteristics of each benchmark [23] [24], either they are solvable or not, the number of variables and clauses of each one. The sources from which these benchmarks were obtained are detailed in the references.

Table2 represents the solving rates and time solving of the Meta-Apriori Clustering DPLL-BSO vs the best time solver

TABLE II
SATISFACTION RATES AND SOLVING TIME FOR META-APRIORI CLUSTERING VS THE BEST BENCHMARK'S SOLVER.

Benchmark Name	Method	Rate (%)	Time (s)	Best Time Solver (s)	Best Solver
Benchmark1	1St Method	99, 61	37, 01		
	2Nd Method	99, 64	41, 35	-	-
Benchmark 2	1St Method	99, 10	1, 63	372, 14	Solver 1[23]
	2Nd Method	98,84	6,88		
Benchmark 3	1St Method	97,71	2,2	2088,76	Solver 2[23]
	2Nd Method	98,30	0,66		
Benchmark 4	1St Method	99,05	20,1	1257,86	Solver 3[23]
	2Nd Method	99,22	23,76		
Benchmark 5	1St Method	99,20	23,29	233,091	Solver 4[23]
	2Nd Method	99,27	27,66		

[23] in the corresponding SAT competition. It shows a significant difference between Meta-Apriori Clustering DPLL-BSO solving's time and the best time solving of each benchmark, which is due to Meta-Apriori Clustering that reduce significantly the complexity of the problem's instance, allowing an important time saving. However, we can see that the problem's instance is not 100% solved.
Comparing, for example, the 3rd benchmark for which the time solving of the best solver is about 2000s and the Meta-Apriori's time solving is about 2s which is 1000time less than the best solver even if the rate is about 97%. This rate can be handled by the use of a more efficient solver than pure DPLL and BSO.
The aim of this paper is the time saving by reducing the problem's complexity using Data Mining techniques.

TABLE III
SATISFACTION RATES AND TIME CONSUMING BETWEEN TWO CLUSTERING METHODS

	Meta-Apriori Clustering-DPLL-BSO Method 2		BSO-DM+DPLL	
Benchmark Name	Rate (%)	Time (s)	Rate (%)	Time (s)
IBM 7	199,11	13,98	93.77	24,25
GALILEO 8	99,06	617,97	97.65	1502,81
GALILEO 9	98,85	815,55	97.64	1620,47

Table 3, presents the rates and time consuming between the Meta-Apriori Clustering DPLL-BSO and the BSO-DM-DPLL[9]. These results, show that Meta-Apriori Clustering DPLL-BSO gives much better results (satisfiability rate) than the BSO-DM-BSO with time saving.

VIII. CONCLUSION

Throughout this paper, we proposed an approach based on mining frequent patterns associated with a complete algorithm and an incomplete one.
The proposed improvement of Apriori, Meta Apriori, is used as a preprocessing by extracting all the variables that appear together. The problem's instance being divided into clusters using the later itemsets, the problem's complexity is lesser, allowing the resolution of each of the clusters using either a complete algorithm or an incomplete one according to the number of variables to be solved. The later approach was applied to the Satisfiability problem because of its importance in the Artificial Intelligence community and the impact of

ISSN 2395-8618

solving such an important problem.

Many algorithms and solvers are proposed each year for solving SAT. The later approach was tested and the results of the experimentations show the impact of using frequent patterns mining as a preprocessing for solving problem.

We believe that this approach would be more efficient when used with a more efficient solver. For our future work, we will integrate this method in a solver that have proven his efficiency.

REFERENCES

[1] K. M. Han. J and P. J, "data mining, concepts and techniques," *Third Edition (The Morgan Kaufmann Series in Data Management Systems).*, 2011.

[2] J. Garey. M.R, "computers and intractability: A guide to the theory of np-completeness," *A Series of Books in the Mathematical Sciences. W. H. Freeman and Co.. pp. x+338. ISBN 0-7167-1045-5. MR 519066.*

[3] V. M. Biere.A, Heule. M and Wals.T, "Handbook of satisability ios press." *IOS press, 2009, ISBN 978-1-58603-929-5 (print)*, 2009.

[4] L. Davis. M and L. D, "a machine program for theorem proving," *Communications of the ACM 5(7)*, p. 394397, 1962.

[5] J. Hamadi. Y and S. L, "manysat: a parallel sat solver," *Journal of Satisfiability, Boolean Modeling and Computation 6(4)*, p. 245262, 2009.

[6] Gomes.C and S. B, "algorithm portfolios," *Artificial Intelligence,126(1-2)*, p. 4362, 2001.

[7] S. K. A. Marques-Silva. J.P, "grasp-a search algorithm for propositional satisfiability," *IEEE Transactions on Computers,48(5).*, vol. 232, p. 506521, May 1999.

[8] S. Hoos.H. H and Kaufmann.M, "stochastic local search : Foundations and applications," *Third Edition (The Morgan Kaufmann Series in Data Management Systems) Elsevier, ISBN: 1-55860-872-9..*, 2004.

[9] L. Selman. B and Mitchell.D.G, "a new method for solving hard satisfiability problems," *In: 10th AAAI, San Jose, CA*, p. 440446, 1992.

[10] K. Selman. B and C. B, ""local search strategies for satisfiability testing," *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, vol. 26, p. 521532, 1996.

[11] [Online]. Available: http://www.satlib.org/ubcsat/algorithms/

[12] L. C. Cai.S and S. k, "Ccanr: A configuration checking based local search solver for non-random satisfiability." *M. Heule and S. Weaver (Eds.)*, vol. 9340, p. 18, SAT 2015.

[13] L. Cai. S and Su.K, "Ccanr+glucose in sat competition 2014," *Proc. Of SAT Competition 2014*, vol. Solver and Benchmark Descriptions, p. 17, 2014.

[14] H. H. KhudaBukhsh.A.R, Xu. L and L.-B. K, "satenstein: Automatically building local search sat solvers from components," *Artificial Intelligence Journal (AIJ)*, vol. 232, pp. 20–42, March, 2016.

[15] D. H, H. C, and D. A, "datamining techniques and swarm intelligence for problem solving: Application to sat," *World Congress on Nature and Biologically Inspired Computing (NaBIC) 2013: IEEE ISBN: 978-1-4799-1414-2*, pp. 200–206, 2013a.

[16] D. A. Drias. H and H. C, "swarm intelligence with clustering for solving sat," *H. Yin et al. (Eds.): IDEAL 2013, LNCS 8206, Springer-Verlag Berlin Heidelberg*, p. 586594, 2013b.

[17] S. Agrawal.R, ""fast algorithms for mining association rules"," *1994 Int.Conf. Very Large Data Bases (VLDB94)*, vol. 99, p. 487499, Sept 1994,Santiago, Chile.

[18] G. A. Bhandari. A and D. D, "Improvised apriori algorithm using frequent pattern tree for real time applications in data mining." *In: Procedia Computer Science Vol. 46, International Conference on Information and Communication Technologies (ICICT 2015)*, vol. 46, p. 644651, 2015.

[19] S. S. Drias. H and Y. S, "a computing procedure for quantification theory," *In: Cabestany, J., Prieto, A.G., Sandoval, F. (eds.)IWANN 2005. LNCS,*, vol. vol. 3512, p. 318325, 2005.

[20] C. S. Seeley. T.D and S. J, "collective decision-making in honey bees: how colonies choose among nectar sources," *Behavioral Ecology and Sociobiology 28*, vol. 232, pp. 277–290, 1991.

[21] D. Benhamouda. N.C and Hireche.C, "Meta-apriori: a new algorithm for frequent pattern detection." *ACIIDS 2016, Part II, LNAI 9622*, vol. 99, p. 277285, 1994,Santiago, Chile.

[22] H. R., "error-detecting and error-correcting codes," *Bell System Technical Journal 29*, vol. 2, pp. 147–160, 1950.

[23] [Online]. Available: http://www.satcompetition.org/edacc/sc14/

[24] [Online]. Available: http://www.cs.ubc.ca/~hoos/SATLIB/benchm.html