Data-Mining-Based Decomposition for Solving MAXSAT Problem: Towards a New Approach

Youcef $Djenouri^1$, Zineb $Habbas^2$, Djamel $Djenouri^3$
[1] LRDSI Lab, Computer Science Department, Saad Dahleb University, Blida, Algeria
[2] Lorraine University, Metz, France
[3] CERIST Research Center, Algiers, Algeria
y.djenouri@gmail.com, zineb.habbas@lorraine.fr, ddjenouri@acm.org

***Abstract*—This paper explores advances in the data mining arena to solve the fundamental MAXSAT problem. In the proposed approach, the MAXSAT instance is first decomposed and clustered by using data mining decomposition techniques, then every cluster resulting from the decomposition is separately solved to construct a partial solution. All partial solutions are merged into a global one, while managing possible conflicting variables due to separate resolutions. Two decomposition methods are investigated. The first one is direct decomposition that uses K-means algorithm, while the second is indirect decomposition that translates the MAXSAT instance into a transactional database before decomposing the database by Apriori algorithm. For solving the sub-problems, DPLL algorithm is used. The proposed approach has been numerical evaluated on DIMACS instances and some hard Uniform-Random-3-SAT instances, and it has been compared to some state-of-the-art decomposition based algorithms. The results show that the proposed approach considerably improves the success rate, with a competitive computation time that is very close to that of the compared solutions.**

Keywords MAXSAT Problem, Knowledge Discovery, Problem Decomposition.

I. Introduction

The SATisfiability problem (SAT) is a fundamental one in computation theory. It is used for modeling many academic and real world problems, e.g., coloring problem, decision support, and automated reasoning. Formally, for a given a set of n boolean variables, $V = \{v_1, v_2, \cdots, v_n\}$, SAT calls for a Conjunctive Normal Form (CNF), which is defined as a conjunction of clauses. Every clause is a disjunction of literals, while a literal is simply a variable, v_i, from, V, or its negation, denoted by $\neg v_i$. A clause is satisfied when at least one of its literals is set to true. A CNF is satisfied if an assignment of some variables in V satisfies all the clauses. The SAT problem asks for an assignment of some variables in, V, that satisfies a CNF, F. The problem is said to be *satisfied* (SAT) if such an assignment exists, and *unsatisfied* (UNSAT) otherwise. SAT problem has been attracting different scientific communities for long time.

This paper addresses MAXSAT problem, a general form of SAT. The objective of MAXSAT is to find a complete instantiation of variables in the set, V, that satisfies the maximum number of clauses. MAXSAT is an NP-Complete optimization problem [1], for which the existing exact solvers are inefficient when dealing with large size of instances. Modern approaches are recently developed for this problem, such as decomposition techniques . On this direction, some researchers have been investigating in the improvement of DPLL (DavisPutnamLogemannLoveland) algorithm [2], which is the most largely used algorithm. It uses decomposition techniques to divide hard SAT instances into many independent but solvable sub-instances. All these methods start by considering a SAT instance as a hypergraph and use the graph theory techniques to transform the hypergraph into many independent sub-graphs.

The main contribution of this paper is to comprehensively explore data-mining techniques for solving the MAXSAT problem. While some recent solutions (that will be presented in the next section) projected some data-mining techniques into the MAXSAT problem, they are limited to the clustering step. To the best of our knowledge, no solution in the current literature provide a solver that takes advantage of the data-mining decomposition. The solution proposed herein considers all the steps and provides a DPLL solver that is based on the outcome of the clustering step. For this step, two approaches have been used, which are different from those used in the literature. The first one–called direct decomposition– decomposes a SAT instance by using Kmeans algorithm. The second one–called indirect decomposition– translates the SAT instance into a transactional database before decomposing the database by using Apriori algorithm. For the solving step, the sub-problems that represents the outcomes of the decomposition step (direct or indirect) can be solved independently by using any solving algorithm, before merging the partial solutions into a global one. This paper focuses on the use of DPLL algorithm for solving the sub-problems. The proposed approach is evaluated by an analytical study carried out on DIMACS (Center for Discrete Mathematics Theoretical Computer Science), with large and very large sat instances. The results demonstrate clear improvement over state-of-the-art solutions. The remaining of this paper is organized as follows. Section II relates on some existing MAXSAT algorithms. Section III describes the general framework of the proposed solution. In section IV, two data mining decomposition techniques are presented, i) direct decomposition with Kmeans algorithm, and ii) and indirect decomposition that applies Apriori algorithm on a constructed transactional database. Section V presents the solving algorithm. The experimental results are reported in Section VI. Finally, Section VII concludes the paper by

some remarks and perspectives.

II. Related Work

SAT and its variant MAXSAT have been attracting the research community for a long time and many algorithms have been proposed. The first canonical and the most used algorithm for the SAT problem is DPLL [2]. It uses decomposition techniques to divide hard SAT instances into many independent but solvable sub-instances. DPLL starts by constructing tree's solver, where in each level one variable is instantiated by true or false. This process is repeated until all branches are explored or all clauses are satisfied. Many DPLL-based approaches have been proposed later on, which mainly differ by the heuristics used for the branching rule [3] and [4]. In most cases, it is impossible to satisfy all clauses of a SAT instance, the problem is transformed to find a solution that satisfy the maximum number of clauses, i.e., MAXSAT. Exact solvers are inefficient when dealing with large size instances of the MAXSAT. Therefore, many heuristic algorithms have been proposed in the literature. In [5], the authors proposed a new branch and bound approach for solving MAXSAT problem by developing a new method to estimate the lower bound of the search tree. In [6], another branch and bound algorithm is proposed by exploiting all the inconsistent subsets with the max-resolution inference rule. This allows to reduce the number of generated branches that are needed to solve such a problem. In [7], a recursively restriction algorithm is proposed to reduce MAXSAT instance. The algorithm restricts one variable at a time. Another version of this algorithm is presented in [1], which uses uniform random distribution among variables to reduce MAXSAT instances. In [8], an approach based on Linear Programming (LP) model has been proposed. It solves a linear programming with CPLEX to determine the order of variables to be instantiated, then complete solver is then applied to calculate the value of each variable. Some other DPLL-based algorithms consider a SAT instance as a hypergraph and use the graph theory techniques to transform the hypergraph into many independent sub-graphs. In [3], the authors divide the whole formula of SAT instance, F, into two sub-formulas, $F1$ and $F2$, which share few variables and have the same size of clauses. The partitioning is based on the cut of hypergraph, where the aim is to minimize the cut variables. In [4], the authors propose the use of community detection techniques to identify the different communities among the SAT instances, and they propose a new strategy to manage conflict between communities. The proposed strategy gives a high priority to the variables that cause conflicts.

We are interested in this paper in another category of decomposition, which is based on data-mining-based concept. In [9], a new genetic algorithm is proposed by exploring a problem's structure in the variable linkage information. Once a linkage model between variables is done, the genetic operators is launched using the extracted knowledge. In [10], a new Stochastic Local Search (SLS) algorithm is proposed. Instead of exploring the search space by $1 - flip$ neighborhood structure (which is done in the ordinary SLS), this approach constructs $K - neighbors$ by flipping K clusters of the variables at the same time. At each step, the clusters of variables are constructed based on the unsatisfied clauses. Thereafter, the cluster of variables are flipped by true or false, i.e., all variables of the same cluster take the same value. Recently, two works for clustering SAT clauses have been proposed in [14], and [15]. In [14], the proposed clustering algorithm puts the first clause in the first cluster, then for every remaining clause, the number of common variables that are shared with the first clause is determined. Consequently, every clause is put in the nearest cluster iff this one shares variables with existing clusters. Otherwise, a new cluster labeled by this clause is created. The main drawback of the decomposition-based approaches is the quality of the derived clusters, as the size of the cut between the obtained clusters is very large.In [15], association rules mining is used to discover clusters of SAT. However, the main drawback of this algorithm is the quality of rules returned by the mining process, which degrades in the results of the clustering step. All these data-mining based solutions are limited to the clustering step, and, to our knowledge, no existing solver that takes advantage of the data-mining decomposition is available in the current literature. The present paper comprehensively explores the use of data-mining techniques to solve the MAXSAT problem and considers all its steps. First, different approach have been proposed for clustering, then an intelligent DPLL approach is proposed for the solving step. This approach takes advantage of the resulted clustering and the useful knowledge that is extracted by the data mining process. In the next section, we present the improved DPLL algorithm for solving MAXSAT problem.

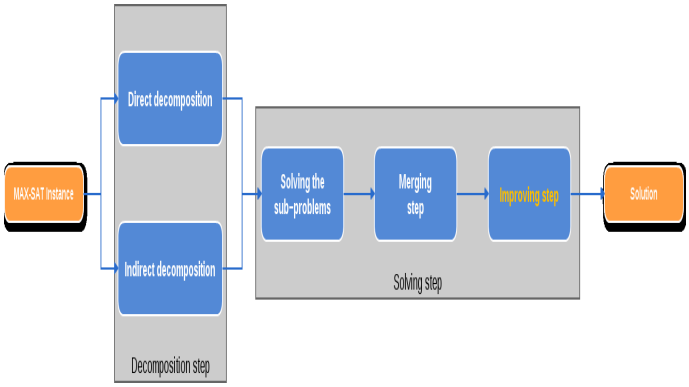

Fig. 1. General framework

III. Solution Overview

The proposed approach can be summarized by the framework presented in Figure 1, which includes three steps:

1) **Decomposition**: It step splits a MAXSAT instance into a collection of sub-problems (clusters) using data mining techniques, where a cluster can be viewed as a subset of clauses of the MAXSAT instance. The aim is to exploit the power of data mining techniques for extracting relevant knowledge, which will be used by the MAXSAT solving algorithm.
   The set of variables shared by two clusters is called a separator set. An interesting decomposition approach

is to minimize the size of separator sets, while putting into the same cluster clauses that are connected, i.e, clauses that share the maximum number of variables. In this paper, we focus on data mining decomposition techniques,and investigate two methods: i) the direct decomposition and ii) indirect decomposition. The two methods will be presented in section IV.

2) **Solving step**: The sub-problems corresponding to clusters that result from step 1 are solved independently to obtain partial solutions, which are merged into a global solution, s. Any MAXSAT solving algorithm can be used for solving the sub-problems. The obtained global solution may be improved, and several techniques may be investigated for this purpose. However, this is out of the scope of this work. A simple local research method, which is DPLL.

IV. Data Mining Decomposition Methods

A. Motivations

Data mining techniques are sometimes used as pre-treatment phase to improve the performance of solvers dealing with difficult optimization problems. Clustering and Association Rules Mining (ARM) are the most studied fundamental techniques in data mining. Clustering is the process that organizes objects into groups with members sharing similar features. A cluster is thus a collection of objects that are similar to one another and dissimilar to the objects belonging to other clusters. The notion of similarity and the quality of clustering are concepts that depend on the nature of the problem, its objective, the definition of distance, etc. ARM aims to extract association rules that satisfy the predefined minimum support and confidence from a given database [17]. It allows to extract more relevant knowledge to guide solvers to accelerate the search of the right solution.

In this paper, the goal is to use a clustering algorithm and an ARM algorithm to decompose a given MAXSAT instance, to experimentally analyze the impact of the quality of such decomposition methods on the cost of the resulting solution. The latter corresponds to the maximum rate of satisfied clauses. Without loss of generality, we chose Kmeans as the clustering algorithm and Apriori algorithm for association rules mining. The decomposition associated with Kmeans is called direct as it consists in directly splitting the SAT instance. Conversely, the decomposition associated with Apriori is called indirect because it first transforms the given SAT instance into a transactional database, before decomposing the resulted database using Apriori.

B. Direct Decomposition of MAXSAT

K-means is one of the simplest unsupervised learning algorithms for the clustering problems. It defines a simple and easy procedure to divide a given data set into a certain number of clusters, say k clusters, fixed a priori. The main idea is to define k centroids, one for every cluster. The centroids should be placed in a cunning way, since the clustering result depends on their location in the clusters. In order to optimize the outcomes, it is judicious to place them as far as possible from each other. The next step is to take each point belonging to a given data set and associate it to the nearest centroid. When no point is pending, the first step is completed and an early grouping is performed. At this stage, k new centroid are needed to be re-calculated for the new clusters that result from the previous step, and the process should be iterated. The latter stops when no more changes of the clusters are observed, i.e., when it is observed that no centroid moves any more. To adapt the K-means procedure on clustering clauses, we propose to use a new similarity and center of gravity computation for clauses that are described in the fellowing.

1) Similarity Between Clauses: The similarity (reps. dissimilarity) between two clauses represents the degree of consistency (reps. inconsistency) between them. Intuitively, they are similar when they share variables and dissimilar otherwise. We define $Dist\_clauses(c_1, c_2)$ as the similarity measure between two clauses, c_1 and c_2, with Eq.IV-B1

equation Dist_clauses(c_1, c_2) $= \max(|Set(c_1)|, |Set(c_2|) - ncv(c_1, c_2)$, (0)Where $ncv(c_1, c_2)$ is the number of common variables between c_1 and c_2, and $Set(c_i)$ return the set of variables of the clause c_i. This distance meets the mathematical properties of a metric distance function, which are:

- $\forall (c_1, c_2) \in C^2, Dist\_clauses(c_1, c_2) \in R$.
- $\forall c \in C, Dist\_clauses(c, c) = 0$.
- $\forall (c_1, c_2) \in C^2\ Dist\_clauses(c_1, c_2) = Dist\_clauses(c_2, c_1)$.
- $\forall (c_1, c_2, c_3) \in C^3\ Dist\_clauses(c_1, c_2) \leq Dist\_clauses(c_1, c_3) + Dist\_clauses(c_2, c_3)$

Example 1: Consider a SAT instance defined as the set of variables, $V = \{v_1, v_2, v_3, v_4\}$, and the two clauses, c_1 and c_2, where c_1 includes variables v_1, v_2, v_3, and c_2 includes variables v_2, v_3, v_4.

First, the set of variables of each clause is calculated. For the current example, Set(c_1)=$\{v_1, v_2, v_3\}$ and Set(c_2)=$\{v_2, v_3, v_4\}$. The set of common variables between c_1 and c_2 is then determined, which is $\{v_2, v_3\}$, i.e., $ncv(c_1, c_2) = 2$. Finally, the distance between c_1 and c_2 is calculated using IV-B1, which is 1.

2) Centroid Computation: Consider the set of clauses $C = \{c_1, c_2, ..., c_r\}$. To find the centroid clause, the frequency of every variable is calculated for all the clauses of the same cluster. The length of the clause center, denoted l, corresponds to the average number of items of all the r clauses. It is determined using the following equation: $l = \frac{\sum_{i=1}^{r} |c_i|}{m}$, where m is the number all clauses. Afterwards, the variables of the r clauses are sorted according to their concurrency, and only the l frequent variables are kept in a vector, say fq, as follows:

$$center[fq[i_1]] = 1 \vee \forall\, i_2 \neq fq[i_1]\, center[i_2] = 0$$

Example 2: Consider the following MAX-SAT problem:
$F = (\neg v_1) \wedge (\neg v_2 \vee v_1) \wedge (\neg v_1 \vee \neg v_2 \vee \neg v_3) \wedge (v_1 \vee v_2) \wedge (\neg v_4 \vee v_3) \wedge (\neg v_5 \vee v_3)$
Note that this problem is UNSAT, since there is no possible value for v_1 or v_2 that satisfies the clause $v_1 \vee v_2$. The success

ratio of this instance is 83% as only 5 clauses out of 6 can be satisfied. In this example, the frequencies of every variable are as follows: $v_1 : 4$, $v_2 : 3$, $v_3 : 3$, $v_4 : 1$, $v_5 : 1$ To compute the center of the six clauses c_1,..., c_6, the centroid l is first computed. Here $l = \frac{1+2+3+2+2+2}{6} = 2$. Then, the l frequent variables are selected. The set of the l frequent variables in the example is $\{v_1, v_2\}$, which represents $center$.

C. Indirect Decomposition of MAXSAT

The indirect decomposition consists in three steps: 1) the transformation of a MAXSAT instance into a transactional database. 2) The extraction of pertinent rules that represent the different relations between data by using Apriori algorithm. 3) The interpretation of the obtained rules for generating the clusters of clauses ?

1) **From a MAXSAT Instance to a Transactional Database***:* Let us consider the set of m clauses, C=$\{c_1, c_2, ..., c_m\}$, and the set of n variables, V=$\{v_1, v_2, ...v_n\}$. Let $score(c)$ denotes the set of variables belonging to clause c. The associated transactional database, say T=$\{t_1, t_2, ..., t_m\}$, is defined on the set of the following n different items, I=$\{I_1, I_2, ...I_n\}$, as follows:

- $\forall(t_i \in T), c_i \in C \Rightarrow t_i = c_i$
  $\forall(I_i \in I), v_i \in score(c_j) \Rightarrow (I_i = 1) \in t_j$
  $\forall(I_i \in I), (v_i \in V)\ and\ v_i \notin score(c_j) \Rightarrow (I_i = 0) \in t_j$

In other words, every clause of the SAT instance is seen as a transaction, and every variable corresponds to an item. If a variable, v_i, belongs to the score of the clause C_j, then the item I_i takes the value 1 in the transaction t_j. Otherwise, it takes the value 0 in the transaction t_j.

2) **Apriori Algorithm for Rule Extraction***:* Once the SAT instance is translated into a transactional database T , the ARM algorithm is applied to T. This is to extract relevant rules that will be used in the generation of clusters. In the literature, many ARM algorithms have been designed, such as the Apriori algorithm that we use hereafter [17]. Apriori consists of the following two steps.

1) The first step is performed in k iterations. Its aim is to find the itemsets of length k, denoted C_k, that are generated by joining the frequent itemsets of length $k-1$, L_{k-1}. At each iteration, the support of every candidate itemsets is computed. If it satisfies the minimum support constraint, it is then added to the set of frequent itemsets. This process should be repeated until the candidate itemsets of length k becomes empty.
2) The second step of the algorithm is to generate the association rules from the set of frequent itemsets. For each frequent itemsets, all the corresponding rules are generated. The confidence of each rule is then computed. If the confidence of a rule fulfils the minimum confidence constraint, then the rule is accepted. Otherwise, it is rejected. Algorithm 1 describes the framework of the Apriori algorithm.

Algorithm 1 Apriori Algorithm

Begin
Input: $T = \{t_1, t_2, ...t_m\}$: Transactional Database , Min_Sup: Minimum Support, Min_Conf : Minimum Confidence.

$L_1 \leftarrow \{Frequent1 - itemsets\}$
$k \leftarrow 2$
while $L_{k-1} \neq \emptyset$ **do**
  $C_k \leftarrow L_{k-1} \times L_{k-1}$
  $L_k \leftarrow \emptyset$
  for each c_i in C_k **do**
    $|c_i| \leftarrow |\{t/c_i \in t\}|$
    $Support(c_i) \leftarrow \frac{|c_i|}{m}$
    if $|c_i| \geq$ Min_Sup **then**
      $L_k \leftarrow L_k \cup \{c_i\}$
    end if
  end for
  $k \leftarrow k + 1$
end while
$L \leftarrow \bigcup_k L_k$
$R \leftarrow \emptyset$
for $l_i \in L$ **do**
  $r_i = \{X_i \Rightarrow Y_i\ X_i \cup Y_i = l_i \wedge X_i \cap Y_i = \emptyset\}$
  for each $r_i : X_i \Rightarrow Y_i$ **do**
    $Confidence(r_i) \leftarrow \frac{Support(r_i)}{Support(X_i)}$
    if $Confidence(r_i) \geq$ Min_Conf **then**
      $R \leftarrow R \cup \{r_i\}$
    end if
  end for
end for
return R
End

3) **Clusters Generation***:* The application of Apriori algorithm results in transactional database T that keeps only the rules with a confidence higher than Min_Conf . In the following, the process of the clustering that results from the interpretation of the rules is described. Assume $R = \{r_1, r_2, ..., r_p\}$ is the set of the returned rules, where each rule r_i is defined as,

$$r_i : X_i \Rightarrow Y_i$$

, with $X_i \subset V$, and $Y_i \subset V$.

Given a temporary list, *List-temp*, that includes all the clauses of the given instance, the set, R, is sorted in the descending order according to the support of the rules. For this purpose, the following algorithm is used.

- Initially, the first rule $r_1 : X_1 \Rightarrow Y_1$ is removed from R, and a cluster is created for the set of clauses that are defined simultaneously on X_1 and Y_1. The representative variables of the first cluster, denoted **Repres_Variables**, is the set $X_1 \cup Y_1$.
- For each rule $r_i \in R$, the sets X_i and Y_i are compared with each representative variable of the clusters that are already created. If such variables do not belong to any

existing cluster, then a new cluster is built for X_i and Y_i. $X_i \cup Y_i$ is obviously considered as the set of representative variables of this new cluster.
- This process is repeated for the p rules of R, or until *List-temp* becomes empty.

4) **Illustration***:* Consider the previous example, where **List_temp** includes all clauses $\{c_1, c_2, c_3, c_4\}$. First the set of clauses is transformed into transactional form, as follows:
- $t_1 = 1, 1, 1, 0, 0$, as the first two variables belong to c_1 and the two latest variables do not belong to c_1.
- $t_2 = 0, 0, 0, 1, 1$.
- $t_3 = 0, 1, 1, 1, 0$.
- $t_4 = 0, 1, 0, 1, 1$.

Apriori algorithm is then applied to the set of transactions from t_1 to t_4, and just the rules with high confidence are considered.
Suppose that the rules obtained are sorted according to their supports:
$r_1 : v_5 \Rightarrow v_4 \ \ (Conf = 100\%, Supp = 50\%)$.
$r_2 : v_3 \Rightarrow v_2 \ \ (Conf = 100\%, Supp = 50\%)$.
$r_3 : v_1 \Rightarrow v_2 \ \ (Conf = 100\%, Supp = 25\%)$.

Starting by the first rule, the first cluster with the clauses c_2 and c_4 is created, as both v_5 and v_4 belong to c_2 and c_4. **Repres_var**(C_1) = $\{v_4 and v_2\}$ is considered as the representative variables for the first cluster, and the clauses c_2 and c_4 are removed from **List-temp**. That is, $C_1 = \{c_2, c_4\}$, **Repres_var**(C_1) =$\{v_4, v_5\}$, **List-temp**=$\{c_1, c_3\}$. For the second rule, the same process is repeatedto obtain, C_2 =$\{c_1, c_3\}$, **Repres_var**(C_2) =$\{v_3, v_2\}$, *List-temp*=$\emptyset$.

The process stops without handling the third rule as **List_ temp** is empty. The final state of the clusters is:
C_1 =$\{c_2, c_4\}$, C_2 =$\{c_1, c_3\}$, **Repres_var**(C_2 =$\{v_4, v_5\}$, represent(2)=$\{v_3, v_2\}$

D. Quality of decomposition methods

The two decomposition methods: direct decomposition (illustrated by Kmeans) and indirect decomposition (illustrated by Apriori), just like any other decomposition method that can be investigates in the future, aim to split a MAXSAT instance into a collection of k clusters $C_1, C_2, \ldots, C_k$ where two clusters C_i and C_j are connected if they share some variables. The set of such variables is called separator set and is noted $sep_{i,j}$.
All decomposition methods do not present the same properties. To compare the decomposition methods, we measure their quality by the distance between clauses belonging to a same cluster and the distance between the clusters. Informally, a "good" decomposition is a decomposition whose clusters are clauses covering the maximum of variables, and whose separators are minimum size.

More formally, the quality of a decomposition methods is defined by MADP formula (Mean Absolute Deviation Percent) described in the equation 1. The distance between two clauses is the total number of variables minus the number of shared variables between these two clauses. The aim is to minimize this distance.

$$MADP_{min} = \sum_{l=1}^{k} \sum_{i=1}^{|Clusterl|} \sum_{j=1}^{|Clusterl|} D(C_i^l, C_j^l). \quad (1)$$

Where
C_i^l: represents the clause C_i in the l^{th} cluster.
D(C_i, C_j)=n-$vc_{i,j}$.
$vc_{i,j}$= number of shared variables between C_i and C_j.
k: is the number of clusters.
n: is the number of all variables.

V. Decomposition Based MAXSAT Resolution

Once the problem is decomposed into a collection of clusters that are connected by separators, it can be solved following different approaches. Examples of existing approaches include the bottom-up approach, and the top-down approach. The latter first solves the sub-problems induced by the separators, before solving subproblems induced by the clusters. This is by considering the variables of separators that have already been instantiated. In this paper we explore the bottom-up approach, which solves first the subproblems associated with the clusters, C_1, C_2, $\ldots C_k$. This step yields a collection of k partial solutions, s_1, s_2, $\ldots$, s_k, where k is the number of clusters. Then, all the solutions are combined into a global solution, s, via to the procedure $merge$. The global solution returned by the latter is improved by using a local search procedure, called $ImproveS$. In each step of the algorithm, the rate of satisfied clauses is returned in the variable $cost$. Algorithm 2 explains the solving step procedure.

Algorithm 2 Solving Procedure

Input a set of clusters $\{C_1, C_2, ..., C_k\}$
Output s: a solution of the problem, cost: the rate of satisfied clauses.
1: **Solving sub-problems** ($\{C_1, C_2, ..., C_k\}$, $\{s_1, s_2, ..., s_k\}$)
2: **merge** $(s_1, s_2, \ldots, s_k, \mathbf{s}, \mathbf{cost})$
3: **ImproveS**$(s, cost)$

Algorithm 2 is mainly composed of the following three steps:
- **Solving sub-problems**: Let $\{C_1, C_2, ..., C_k\}$ be the collection of clusters that result from step1. Those are previously sorted following a decreasing order on the sizes of clusters, where the size of a cluster C_i is represented by its cardinality. Each sub-problem induced by a cluster is solved by using DPLL algorithm. This procedure is described in algorithm 3. Each solution, s_i, is an instantiation of a subset of variables.

- **Merging step**: This step is crucial and directly impact the quality of the returned solution. In this paper, a very simple merging technic is proposed for the sake demonstrating the feasibility of the proposed approach

Algorithm 3 Solving sub-problems) ($\{C_1, C_2, ..., C_k\}$, $\{s_1, s_2, ..., s_k\}$)

1: **for** each cluster C_i **do**
2: s_i = DPLL(C_i)
3: **end for**
4: Return $\{s_1, s_2, ..., s_k\}$

in a simplified and easy to follow way. There is though ample room for improvement of this step in perspective to the current work. The merge procedure developed in this paper is a special concatenation of the k partial solutions, which takes into account the fact that two solutions, say s_i and s_j, are not fully independent. For variables belonging to s_j, and not to, s_i, they are copied without any update in the concatenation result. However, if a variable x belongs simultaneously to s_i and s_j, the merge procedure maintains in the concatenation result the value of, x, in, s_i. These shared variables will generate some unsatisfied clauses in the global solution s.

Algorithm 4 merge $(s_1, s_2, \ldots, s_k, \mathbf{s}, \mathbf{cost})$

1: $s \leftarrow s_1$
2: **for** each cluster s_i, $i > 1$ **do**
3:
4: **for** each variable $x \in s_j$ **do**
5: **if** $x \notin s$ **then**
6: $s \leftarrow s \odot x = v$ where v is the assigned value to x in s_j
7: **end if**
8: **end for**
9: $cost$ rate of constraints satisfied by s
10: **end for**

- **Improvement step**: The procedure DPLL returns optimal solutions for all subproblems. But the global solution obtained by the merge function cannot be optimal since the problems are not fully independent. Some shared variables may be the source of conflicts. The conflict variables are variables shared at least by two different clusters that are instantiated by different values. To improve the quality of the global solution found by the **merge** function, a local search procedure is used. In the global solution, *s*, the partial solution, s_i, is exact. First, the initial solution of the local search procedure is initialized to the global solution *s*. Then, the neighbors of *s* are computed by changing the value of each conflicted variable of the first cluster. Afterwards, starting from the best neighbor, the neighbors are determined by changing the value of each conflict variable of the second cluster. This process is repeated until all clusters are handled. Algorithm 5 explains the reparation step procedure.

Algorithm 5 Improvement Step ($\{C_1, C_2, \ldots, C_k\}$, s, $cost$)

1: **for** each cluster C_i **do**
2: s=best_neighbors(s, $Conflict - Var(C_i)$)
3: **end for**

VI. Experimentation

To validate the proposed approaches, several experiments have been carried out. We implemented the proposed approaches using *C++* environment and tested it with 21 *DIMACS* instances, 10 instances of *Uniform Random-3-SAT*, and large MAXSAT instances of $SAT2013$ industrial competition. We used a single machine (*Pentium-I3*), with *4Go* RAM. All these instances are available at SATLIB site [1].

The *DIMACS* instances used in the tests can been divided into three classes, *aim-50*, *aim-100*, and *par8*. The first class contains 8 different instances of *aim-50* class. The number of its variables is 50 and the number of clauses varies between 80 and 100. The second class includes 8 instances of *aim-100* class, its variable's size is 100, and the number of clauses varies between 160 and 200. The last class contains 5 instances of *Parity8*, with 350 variables, and clauses from 1149 to 1171. For the *Uniform Random-3-SAT* used instances, the number of variables has been varied from 20 to 250, and the number of clauses from 91 to 1065.

First, the two decomposition methods (Kmeans and Apriori) have been investigated using DIMACS SAT instances. Then, the solvers (Kmeans-DPLL and Apriori-DPLL) are compared to DPLL. Finally, the approach showing the best performances in the experiments (Apriori-DPLL) is compared to the state of the art decomposition-based MAXSAT algorithms using large instances.

A. Parameters Setting of Decomposition-based Algorithms

The decomposition-based algorithms used in the proposed approach depend on user's parameters. In fact, the number of clusters of Kmeans algorithm (K), as well as the constraints of Apriori, i.e., the minimum support ($minsup$) and the minimum confidence ($minconf$), influence the quality of the final obtained clusters. Choosing a high value of K and low values of $minsup$ and $minconf$ allows to augment the quality of the final obtained clusters. However, the computational cost of the decomposition process can also be augmented. Finding an acceptable trade-off between the computational cost and the cluster's quality is a challenging task. Intensive tests have been carried out to empirically tune parameters of Kmeans and Apriori algorithms in *Parity-8-2* Dimacs instance as follows:

1) **For Kmeans algorithm:** By varying the number of clusters K from 1 to 100, the results indicates that the computational time is enhanced. However the quality of clusters augments and stabilize at 20 clusters. The number of clusters, K, is thus set to 20 in the following experiments.
2) **For Apriori algorithm:** By reducing in both minimum support $minsup$ and $minconf$ from 100% to 20%, the results indicate that the computational time is enhanced by generating high number of association rules. However, the quality of clusters augments and stabilizes at

[1] *http://www.cs.ubc.ca/hoos/SATLIB/benchm.html*

Class	Instances	(n,m)	Intuitive Clustering	Kmeans	Apriori
	1-6yes1-1	(50,80)	612	121	98
aim50	1-6yes1-2	(50,80)	685	134	102
	1-6yes1-3	(50,80)	702	189	134
	1-6yes1-4	(50,80)	714	202	175
	2-0yes1-1	(100, 160)	1025	412	392
aim100	2-0yes1-2	(100, 160)	1136	487	402
	2-0yes1-3	(100,160)	1214	580	492
	2-0yes1-4	(100,160)	1421	670	532
	1	(350,1149)	1181	700	620
Parity8	2	(350,1157)	1562	825	712
	3	(350,1171)	1751	928	826

TABLE I
THE QUALITY OF CLUSTERS OF INTUITIVE CLUSTERING AND THE PROPOSED APPROACHES WITH DIMACS INSTANCES

Class	Instances	(n,m)	DPLL_Kmeans	DPLL_Apriori	DPLL
	1-6yes1-1	(50,80)	100	90	96
aim50	1-6yes1-2	(50,80)	100	90	96
	1-6yes1-3	(50,80)	100	89	93
	1-6yes1-4	(50,80)	100	88	93
	2-0yes1-1	(100, 160)	100	88	92
aim100	2-0yes1-2	(100, 160)	100	87	92
	2-0yes1-3	(100,160)	100	85	92
	2-0yes1-4	(100,160)	100	88	91
	1	(350,1149)	81	70	74
Parity8	2	(350,1157)	81	70	74
	3	(350,1171)	78	52	61

TABLE II
RESULTS OF KMEANS-DPLL AND APRIORI-DPLL COMPARED TO DPLL WITH DIMACS INSTANCES

$minsup = 73\%$ and $minconf = 82\%$ clusters. The minimum support and confidence are then set to 73% and 82%, respectively, in the following experiments.

B. Decomposition Approaches Performance

In this step, the decomposition based approaches (Apriori and Kmeans) are analyzed using Dimacs instances.

Table I presents the quality of the clusters obtained by our two approaches (Kmeans, Apriori), and the intuitive clustering algorithm [14] using Dimacs instances. The quality of clusters is calculated using Equation 1 (we refer to Sec. IV-D for more details). The results show a large difference between the quality of the clusters of the proposed approaches and the intuitive clustering algorithm. In fact, the clusters obtained by the former are correlated and reflect a strong dependencies between the clauses of the same cluster. A little difference is observed between Kmeans and Apriori in terms of the quality of clusters. Apriori outperforms the Kmeans in all cases, and the average value for Apriori was 407.73, while the one for Kmeans was 477.09. These results confirm that association rules clearly improve the quality of clustering SAT clauses.

C. The Proposed Approaches vs. DPLL

Tables II and III present the success rate of Kmeans-DPLL and Apriori-DPLL without improvement, vs. DPLL with *Dimacs* and *Uniform Random-3-SAT* instances. The results reveal that our approaches are close to the optimal solution found by DPLL algorithm. In fact, the success rate of the proposed approaches is higher than 90% for all *Uniform Random-3-SAT* instances. the results also reveal that Apriori-DPLL outperforms Kmeans-DPLL in all cases. This is due to the relevant rules provided by applying association rules mining process.

Instances	DPLL-Kmeans	DPLL-Apriori	DPLL
uf20-91	90	97	100
uf50-218	91	88	100
uf75-325	92	88	100
uf100-430	88	93	100
uf125-538	92	96	100
uf150-645	89	92	100
uf175-763	90	90	100
uf200-860	90	90	100
uf225-960	92	93	100
uf250-1065	89	92	100

TABLE III
RESULTS OF KMEANS-DPLL AND APRIORI-DPLL COMPARED TO DPLL WITH UNIFORM RANDOM-3-SAT INSTANCES

Class	Instances	(n,m)	DPLL_Kmeans	DPLL_Apriori	DPLL
	1-6yes1-1	(50,80)	0,98	0,08	0,05
	1-6yes1-2	(50,80)	2,35	0,12	0,08
	1-6yes1-3	(50,80)	2,95	0,15	0,07
aim50	1-6yes1-4	(50,80)	1,69	0,13	0,06
	2-0yes1-1	(100, 160)	3,53	0,65	0,17
	2-0yes1-2	(100, 160)	3,45	0,87	0,19
aim100	1-6yes1-4	(100,160)	3,87	0,95	0,21
	2-0yes1-1	(100,160)	3,53	0,65	0,17
	2-0yes1-2	(100,160)	3,45	0,87	0,19
	2-0yes1-3	(100,160)	3,87	0,95	0,21
	2-0yes1-4	(100,160)	3,92	0,62	0,28
Parity8	1	(350,1149)	21,54	8,36	5,21
	2	(350,1157)	31,99	10,65	8,12
	3	(350,1171)	50,31	18,54	11,36

TABLE IV
RUNTIME (SEC) OF KMEANS-DPLL AND APRIORI-DPLL COMPARED TO DPLL WITH DIMACS INSTANCES

Instances	DPLL-Kmeans	DPLL-Apriori	DPLL
uf20-91	0,05	0,01	16,42
uf50-218	0,1	0,03	37,15
uf75-325	0,4	0,04	54,65
uf100-430	0,8	0,1	80,72
uf125-538	4,5	0,1	100,42
uf150-645	12,5	1,5	225,32
uf175-763	18,69	2,8	400,98
uf200-860	22,65	8,9	498,63
uf225-960	26,63	15,36	556,64
uf250-1065	45,65	28,35	606,87

TABLE V
RUNTIME (SEC) OF KMEANS-DPLL AND APRIORI-DPLL COMPARED TO DPLL WITH UNIFORM RANDOM-3-SAT INSTANCES

Tables IV and V present the runtime (in sec) of the Kmeans-DPLL and Apriori-DPLL without improvement, compared to DPLL with *Dimacs* and *Uniform Random-3-SAT* instances. *The computational time of decomposition step is included*. These tables reveal that the proposed approaches largely outperforms the DPLL algorithm in terms of execution time. The runtime in the proposed approaches does not exceed $50sec$ in all instances. However, with DPLL, the runtime exceeds $600sec$ in the last instance containing 250 variables

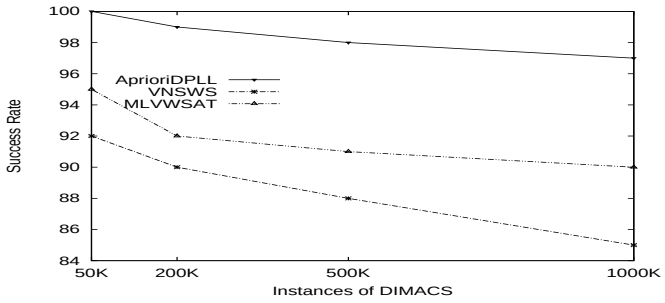

Fig. 2. Success Rate of the proposed approach and the state of the art decomposition-based MAXSAT algorithms with different number of clauses of DIMACS Instances

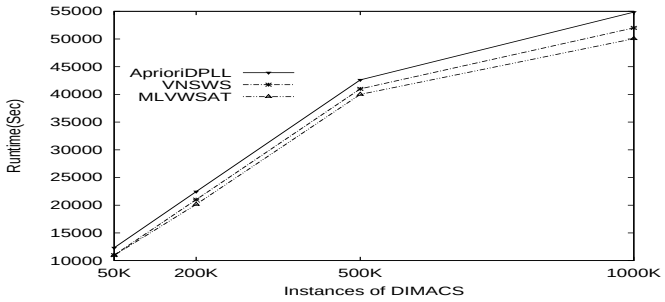

Fig. 3. Execution time (Sec) of the proposed approach and the state of the art decomposition-based MAXSAT algorithms with different number of clauses of DIMACS Instances

and 1065 clauses. These results confirm the necessity to use the decomposition methods for solving SAT instances.

From these experimentations we conclude that Apriori-DPLL outperforms Kmeans-DPLL both in terms of success rate and execution time. Therefore, Apriori-DPLL is used in the last experimentation to compare the proposed approach with the state-of-the-art MAXSAT algorithms.

D. Apriori-DPLL Vs the State-of-the-art Decomposition-based Approaches

Fig. 2 plots the success rate of the proposed approach in comparison with MLV-WSAT, VNS-WS, two approaches that are recently proposed in the literature and that uses decomposition methods in the solving process. The tests are performed using the large DIMACS instances of $SAT2013$ industrial benchmarks. The results demonstrate that the proposed approach clearly outperforms both approaches, and that it reaches the optimum (100%) in $DIMACS50K$. The success rate of all the approaches normally drops as the size of the instances goes up, but it is worthy to mention the increasing gap between the proposed approach and the other ones. The former' success rate remains beyond 97% for the largest instance ($1000K$), vs. 90% for MLVWSAT and 85% for VNSWS. This superiority is due to the clustering step that explores the knowledge extracted by Apriori algorithm and provides clusters with a higher quality than the other algorithms. This clustering influences the quality of the final results in the solving step.

Figure 3 presents the runtime in (sec). It shows the cost of the improvement in the success rate. The proposed approach inviably needs a higher runtime to reach the high performance, but the figure clearly shows that the runtime of the latter is close to that of the other approaches, and that its increase with the instance size is smooth, and in the same shape as the other approaches. This confirms the salability of the proposed approach.

VII. Conclusion

Two decomposition techniques for solving MAXSAT problem have been proposed in this paper. The first one is direct decomposition, where the clusters are obtained directly with Kmeans algorithm. Its main feature is the use of distance between clauses, and the way to determine the centroid for the set of clauses. The second approach is indirect decomposition that is based on Apriori. In this approach, the set of clauses is first transformed into transactional database, then Apriori is applied to the database to search for relevant rules that represent the SAT clauses. These rules are used to generate the clusters of clauses. These two decomposition techniques (Direct and Indirect) allow to divide a hard MAXSAT instance into several sub-instances that are easily solvable. For each cluster of clauses, the DPLL algorithm is applied to find its partial. The global solution is finally obtained in the last step by merging partial solutions. The two decomposition techniques allow to minimize the dependency between the clusters. However, the optimal solution may suffer from conflicts between the clusters, which is caused by the separators variables. To deal with this issue, we proposed two strategies. The first one is to start with clusters that have the maximum number of clauses in the solving step. Whereas, the second one is performed after finding the global solution and consists in applying a simple local search on it.

To demonstrate the performance of the two suggested approaches, several experiments have been carried out using the hard DIMACS and Uniform-Random-3-SAT instances. The results revealed that the second approach benefits from the relevant knowledge extracted in the decomposition step and improves the results obtained by the direct approach. Still, both approaches outperforms the ordinary DPLL in terms of computational time, while finding solutions that are close to the optimal one. The proposed approach has also been compared with state-of-the-art decomposition based algorithms (MLV-WSAT and VNS-WS) using very large DIMACS instances (where ordinary DPLL does not apply). The results indicate that the Apriori-DPLL outperforms the other algorithms in terms of success rate and has very competitive run time. As a perspective, we plan to investigate other heuristics to deal with the conflict problem caused by the separator variables. We are also planning to apply the two suggested approach to other optimization problems such as weighted MAXSAT, Coloring Problem, and CSP Problem. Finally, proposing a parallel version that explores HPC to solve very big MAXSAT instances is also in our agenda.

References

[1] Sakai, T., Seto, K., & Tamaki, S. (2014). Solving Sparse Instances of Max SAT via Width Reduction and Greedy Restriction. In Theory and Applications of Satisfiability TestingSAT 2014 (pp. 32-47). Springer International Publishing.

[2] Davis, M., & Putnam, H. (1960). A computing procedure for quantification theory. Journal of the ACM (JACM), 7(3), 201-215.

[3] Park, T. J., & Van Gelder, A. (1996). Partitioning methods for satisfiability testing on large formulas. In Automated Deduction Vol-13 (pp. 748-762). Springer Berlin Heidelberg.

[4] Ansategui, C., Giraldez-Cru, J., & Levy, J. (2012). The community structure of SAT formulas. In Theory and Applications of Satisfiability Testing SAT 2012 (pp. 410-423). Springer Berlin Heidelberg.

[5] Abram, A., & Habet, D. (2014, August). On the Extension of Learning for Max-SAT. In STAIRS 2014: Proceedings of the 7th European Starting AI Researcher Symposium (Vol. 264, p. 1). IOS Press.

[6] Abrame, A., & Habet, D. (2014, November). Local Max-Resolution in Branch and Bound Solvers for Max-SAT. In Tools with Artificial Intelligence (ICTAI), 2014 IEEE 26th International Conference on (pp. 336-343). IEEE.

[7] Chen, R., & Santhanam, R. (2015). Improved algorithms for sparse MAX-SAT and MAX-k-CSP. In Theory and Applications of Satisfiability Testing–SAT 2015 (pp. 33-45). Springer International Publishing.

[8] Kolokolov, A., Adelshin, A., & Yagofarova, D. (2013). Analysis and Solving SAT and MAX-SAT Problems Using an L-partition Approach. Journal of Mathematical Modelling and Algorithms in Operations Research, 12(2), 201-212.

[9] Sadowski, K. L., Bosman, P. A., & Thierens, D. (2013, July). On the usefulness of linkage processing for solving MAX-SAT. In Proceedings of the 15th annual conference on Genetic and evolutionary computation (pp. 853-860). ACM.

[10] Bouhmala, N. (2014). A Variable Neighborhood Walksat-Based Algorithm for MAX-SAT Problems. The Scientific World Journal, 2014.

[11] Jabbour, S., Sais, L., & Salhi, Y. (2013, October). Boolean satisfiability for sequence mining. In Proceedings of the 22nd ACM international conference on Conference on information knowledge management (pp. 649-658). ACM.

[12] Jabbour, S., Sais, L., & Salhi, Y. (2015). Decomposition Based SAT Encodings for Itemset Mining Problems. In Advances in Knowledge Discovery and Data Mining (pp. 662-674). Springer International Publishing.

[13] Park, T. J., & Van Gelder, A. (1996). Partitioning methods for satisfiability testing on large formulas. In Automated Deduction Vol-13 (pp. 748-762). Springer Berlin Heidelberg.

[14] Drias, H., Hireche, C., & Douib, A. (2013, August). Datamining techniques and swarm intelligence for problem solving: Application to SAT. In Nature and Biologically Inspired Computing (NaBIC), 2013 World Congress on (pp. 200-206). IEEE.

[15] Drias, H., & Djenouri, Y. (2014). Association Rules Mining: Application to Large-Scale Satisfiability.

[16] MacQueen, J. (1967, June). Some methods for classification and analysis of multivariate observations. In Proceedings of the fifth Berkeley symposium on mathematical statistics and probability (Vol. 1, No. 14, pp. 281-297).

[17] Agrawal, R., Imieliniski, T., & Swami, A. (1993, June). Mining association rules between sets of items in large databases. In ACM SIGMOD Record (Vol. 22, No. 2, pp. 207-216). ACM.