SAT Instances Construction Based on Hypergraphs

ISAAC VÁZQUEZ-MORÁN, JOSÉ TORRES-JIMÉNEZ
Department of Computational Sciences
Instituto Tecnológico y de Estudios Superiores de Monterrey, Campus Cuernavaca
Apdo. Postal 99-C, Cuernavaca Morelos, 62050
México

Abstract: - The SAT problem is one of the most important combinatorial optimization problems, it consists of determining given a formula of propositional calculus in CNF (Conjunctive Normal Form) if exists an assignment of truth values for the variables, that makes the whole formula true. The 3-SAT problem is the first NP-complete problem. This paper describes a representation of SAT instances based on hypergraphs and introducing the Signed K-Dimensional Labeled Multi-Hypergraph (SKDLMH) concept.

Key-Words: - Propositional Satisfiability, SAT Instances, Hypergraph, Conjunctive Normal Form.

1 Introduction

The SAT problem constitutes de core of the computational complexity analysis, because is the first problem that have been proven to be NP-complete [1], and permits the indirect solution of many interesting problems [2]. Due to its importance within the computational theory are many SAT solving algorithms. The testing of these algorithms is dependent on the hardness of the instances used to validate the algorithm. We assumed that the connectivity of the clauses and complexity of the structure in a SAT Instance takes part in their hardness.

The first person that use the hypergraphs to represent SAT instances was Gallo [3], we use the representation of SAT Instances with hypergraphs and introduce a new concept SKDLMH that permits a design of well defined structures that are connected, this structures will be converted in SAT Instances.

2 Propositional Satisfiability Problem

The Propositional Satisfiability problem (SAT) is one of the most important optimization combinatorial problem because is the first problem that have been proven to be NP-complete [1].

SAT stands for Satisfiability, and it refers to the propositional logic problem that consists in determining the truth values for a set of literals X that defines a set of clauses C expressed in the Conjunctive Normal Form (CNF). The CNF have the form, $F = C_1 \wedge C_2 \wedge ...C_n$ where every C_i denotes a clause and it is a disjunction of literals X_j. This truth values make the whole formula F true.

The 3-SAT problem is the most simple and more used problem for testing another results of NP-Completeness [2]. The SAT problem is central to the computational complexity analysis. Many practical problems are NP-Complete, this means that they can be efficiently transformed into the SAT domain, solved in this domain and return the result to the original domain [2].

In this way it is desirable to find a good SAT solving algorithm with good performance, and the performance of SAT solving algorithms is evaluated using SAT instances, therefore is important to create programs that generate Hard SAT Instances. We assumed that to generate Hard SAT instances these must be connected and without duplicated clauses, this is because a SAT instance that is not connected in fact is a multiple small SAT instances, but in this moment the generators of SAT instances more frequently used do not ensure that the SAT instances generated are connected and without duplicated clauses [4] [5], for that reason is necessary define a formal mechanism to design connected SAT instances and without duplicated clauses.

3 Graph Theory

A graph G is defined as $G = (V, E)$, where V is a finite set different of an empty set which elements are called nodes $V = \{i \mid i = 1,..., N\}$, E has as element a pair of nodes called edges $E = \{(i, j) \mid i \in V \wedge j \in V\}$. When a graph has a two or more edges that connect the same pair of nodes, the graph is called multigraph.

A hypergraph H is defined as $H=(V,HE)$, where V is a finite set different of an empty set which elements are called nodes $V=\{i \mid i=1,...,N\}$, E has as elements thirds or more nodes called hyperedges

$$E=\{e_1,e_2,...,e_m\}\mid\left(e_i=\begin{Bmatrix}(x_a,y_a,z_a,...)\\ ,...,(x_w,y_w,z_w,...)\end{Bmatrix}\right)$$
$$\wedge(i=1,2,...,m)\wedge((x_a,y_a,z_a,...)\in V)$$
$$\wedge...\wedge((x_w,y_w,z_w,...)\in V)$$

A hypergraph is k-uniform if every hyperedge connects the same k nodes, an example of one hypergraph that connects four nodes with one hyperedge is showed in the Fig. 1, this correspond to an hypergraph with $V=\{1,2,3,4\}$ and $E=\{(1,2,3,4)\}$.

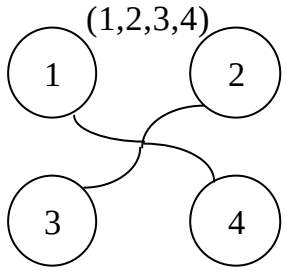

Fig. 1

3.1 Representing SAT instances with Hypergraphs

Given that a SAT instance consists of a set of clauses where each clause represents the disjunction of literals, a particular clause can be represented by a hyperedge that connects the nodes that represents the variables on the clause. In this way a k-SAT Instance is represented by a k-hypergraph in which the clauses are k-hyperedges. The Fig. 2 represents the clauses that involves the A, B, C, D, and E variables.

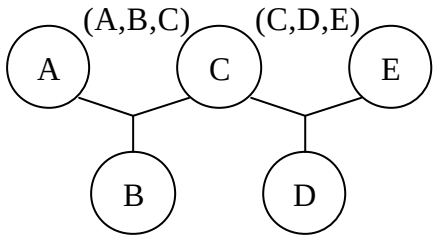

Fig. 2

In this moment the clauses defined do not have the representation of signs for every variable in the clause. For this we use a special notation for the negated variables, a circle that joins the hyperedge that connect to the node that represents the negated variable. The Fig. 3 represents the clauses (A,B,C) and $(C,\sim D,E)$.

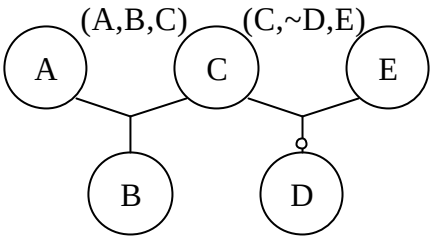

Fig. 3

A SAT instance can contain two or more clauses that involve the same variables but with a different sign combination, in this way we introduce the concept of multihypergraph to denote a graph that contains two or more hyperedges that connect the same set of nodes but with different signs. The Fig. 4 shows a multi-hypergraph that represents the clauses $(\sim A,B,\sim C)$ and $(A,\sim B,C)$.

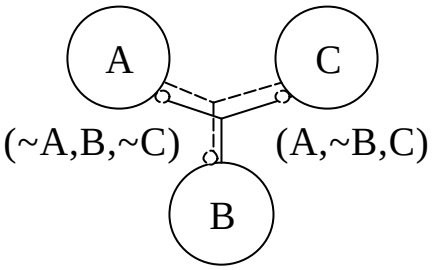

Fig. 4

4 Signed K Dimensional Labeled Multi-Hypergraph (SKDLMH)

A Signed K Dimensional Labeled Hypergraph (*SKDLH*) H is defined as $H=(V,E)$ where:

V is the K dimensional labeled vertex set (*VS*) with cardinality $|V|=n$.

E is the hyperedge set (*HES*) with cardinality $|E|=m$.

The *VS* is defined as

$$V=\{(x_1,y_1,z_1,...),(x_2,y_2,z_2,...),...,(x_n,y_n,z_n,...)\}$$
$$\mid(1\le x_i<\infty)\wedge(1\le y_i<\infty)\wedge(1\le z_i<\infty)\wedge...\wedge(n\ge 0)$$

The *HES* is defined as

$$E=\{e_1,e_2,...,e_m\}\mid\left(e_i=\begin{Bmatrix}\Diamond(x_a,y_a,z_a,...),\\ ...,\Diamond(x_w,y_w,z_w,...)\end{Bmatrix}\right)$$
$$\wedge(i=1,2,...,m)\wedge(|e_i|\ge 2)\wedge((x_a,y_a,z_a,...)\in V)$$
$$\wedge...\wedge((x_w,y_w,z_w,...)\in V)\wedge(m\ge 0)\wedge((\Diamond=-)\vee(\Diamond=\varnothing))$$

Graphically a vertex that appears as negated in a hyperedge is represented with a dot in the vertex-hyperdge joint.

The basic operations are defined in order to manipulate SKDLH objects:

Create. Specified with an equal sign, creates a SKDLH object, giving the *VS* and the *HES*. v.gr. for K=2, S={{},{}} creates an empty *2DLH* object, and R={ {(2,2),(3,4),(5,6),(7,8)}, {{-(2,2),(3,4),(7,8)}, {-(3,4),(5,6),-(7,8)} } creates the *2DLH* object whose graphic representation is given in the Fig. 5.

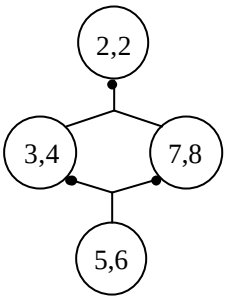

Fig. 5

Merge. Specified with $\oplus$, makes the merging of two *SKDLH* objects. Assuming that K=2 and R=(V_R, E_R) and S=(V_S, E_S) are *2DLH* objects their merging is defined as:

$$R \oplus S = \begin{cases} (V_R \cup V_S, E_R \cup E_S) & iff\ V_R \cap V_s \neq \varnothing \\ \varnothing & iff\ V_R \cap V_s = \varnothing \end{cases}$$

As can be inferred the Merge operation is commutative, i.e. $R \oplus S = S \oplus R$.

Move. Specified with K subindex, creates an isomorphic object taking as basis one previously created *SKDLH* object adjusting the labels according the K given subindexes (and also adjusts the *HES* in a consistent way). The new labels are the original labels plus the respective given K subindexes. v.gr., for K=2, and given that R={{(2,2),(3,4),(5,6),(7,8)}, {{(2,2),(3,4),(7,8)},{(3,4),(5,6),(7,8)}} and $T = R_{-1,2}$, the T object is illustrated in the Fig. 6.

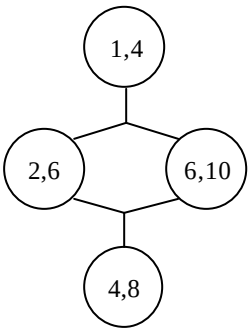

Fig. 6

Dimension Modification Operator. Specified as superindex enables the modification of the label's dimensions. The conversion of a *SKDLH* object *R* to a *SKDLH* object *S* is specified using a superindex that is added to *K* to produce the new object. v.gr. if *R* is a *4DLH* object and *S* is a *3DLH* object the creation of *S* from *R* is specified as $S = R^{-1}$, the vertex set labels for *S* are computed in a random way guaranteeing that there is no label repetition (as can be inferred this modifies also the vertex set in a consistent way), the signs of the *R HES* are conserved in the *S* object. When the result is an object of one dimension the vertex set labels for *S* are computed in random way using *n* labels and guaranteeing that there is no label repetition.

Hyperedge Sign Change. This is represented with the minus sign $-$. The operation of this unitary operator is to modify the *HES* of an *SKDLH* object, in such a way that a sign random assignment for each hyperedge is produced.

Hyperedge Sign Elimination. This operator is specified in a functional notation with the word *abs* (stands for absolute), an example is: $S = abs(T)$ creates an *S* object identical to *T* but without signs in the *HES*.

Given an expression involving many operators is necessary to define the next rules:

The order of evaluation is from left to right.

The order of operator's evaluation is: first move operator (underindex), then dimension updating (upperindex), next hyperedge sign change (unitary minus sign), next absolute operation, and at last merge.

Using parenthesis it is possible to force a desired order of evaluation of an expression.

A Signed K Dimensional Labeled Multi-Hypergraph (*SKDLMH*) is a *SKDLH* in which at least one hyperedge from *HES* appears twice with different sign combination. For example if *R* is a *SKDLH* and $S = R \oplus -R \oplus -R$, *S* is *SKDLMH*.

5 Construction of Connected SAT Instances using SKDLMH

With $K = 2$ we have a *S2DLH* structure, this permits construct the structures in the bidimensional space, after the construction of the structure we use the Hyperedge Sign Change operator, the result of this operation is a *S2DLMH* structure and we applied the Dimension Modification Operator to reduce the dimension and obtain a *S1DLMH*, the equivalence of this object and a SAT instance is explained next.

The set of variables g is denoted by $g = \{V \in VS\}$, the number of variables is $n = |g|$.

The set of clauses d is denoted by $d = \{HE \in HES\}$, the number of clauses is $m = |d|$.

Given that, the explanation of the parallel between a SAT clause with g variables and a *SKDLH* was given previously, then it can be concluded that the programs that manipulates *SKDLMH* can be used to produce random SAT instances that has the property of connectedness, i.e. beginning with an arbitrary clause we can reach all the clauses using as bridges the variables.

To construct a 3-SAT instance with a structure that will call Simple Triangles we define the next program:

$$n = \{n \mid n \in +N * \wedge n \; is \; pair \}$$
$$R = \{\{(1,1),(2,1),(2,2)\},\{(1,1),(2,1),(2,2)\}\}$$
$$S = R$$
$$For \;\; i = 1 \; To \;\; (n/2) - 2$$
$$S = S \oplus R_{i,0}$$
$$EndFor$$
$$U = \begin{Bmatrix} \{(n/2,1),(1,1),((n/2)+1,2)\}, \\ \{(n/2,1),(1,1),((n/2)+1,2)\} \end{Bmatrix}$$
$$S = -(S \oplus U)$$
$$S = S \oplus -S$$
$$T = S^{-1}$$

Where *n* is the number of variables in the instance, the Fig. 7 is the graphical representation of the *S2DLMH*. The Fig. 8 is a possible *S1DLMH* generated, with this *S1DLMH* the SAT instance generated is:

$$\begin{Bmatrix} (8 \vee 4 \vee \sim 1) \wedge (4 \vee \sim 6 \vee 7) \wedge (6 \vee 2 \vee \sim 3) \wedge (2 \vee 8 \vee \sim 5) \wedge \\ (\sim 8 \vee 4 \vee 1) \wedge (\sim 4 \vee 6 \vee \sim 7) \wedge (\sim 6 \vee \sim 2 \vee 3) \wedge (2 \vee \sim 8 \vee \sim 5) \end{Bmatrix}$$

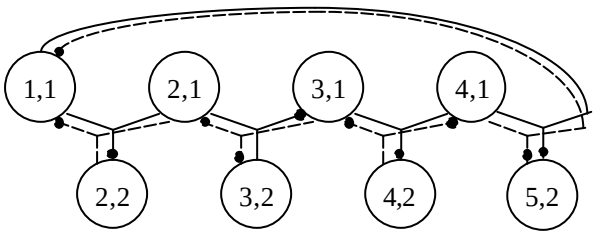

Fig. 7

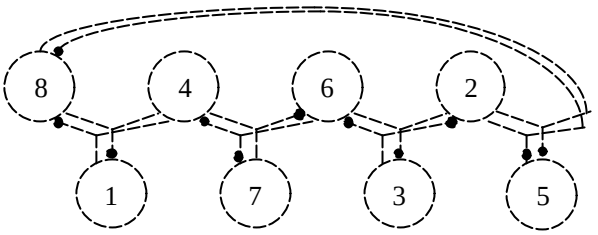

Fig. 8

6 Conclusion

This paper presents the next contributions:

A new approach to construct multihypergraphs; a novel approach to generate SAT instances based on multihypergraphs; with the SKDLMH and the operations defined is possible generate SAT instances and grouped by their structure and hardness and with every structure designed is possible generate multiple connected SAT instances without duplicated clauses.

References:

[1] S. A. Cook. The complexity of theorem proving procedures. Symposium on the Theory of Computing, New York, 1971.

[2] M. R. Garey and D. S. Johnson. Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman and Company, New York, 1979.

[3] G. Gallo and D. Pretolani. A new algorithm for the propositional satisfiability problem. Departament di informatic, University of Pisa, Corso Italy. 1992.

[4] Bart Selman, David Mitchell and Hector J. Lavesque. Generating Hard Satisfiability Problems. Department of Computer Science, University of Toronto, Toronto, Ontario, Canada, 1991.

[5] S. Kirkpatrick and B. Selman. Critical Behaviour in the Satisfiability of Random Boolean Formulae. Racah Institute of Physics and Center of Neural Computation, Hebrew University. Jesuralem, Israel, 1993.