An Empirical Measure for Characterizing 3-SAT

Hachemi Bennaceur [1] and **Chu Min Li**[2]

Abstract. The literal encoding of the satisfiability problem (SAT) as a binary constraint satisfaction problem (CSP) allows us to make a technical comparison between the propositional logic and the constraint network fields. Equivalencies or analogies between concepts and techniques used in the two fields are then shown. For instance it is shown that arc consistency is equivalent to unit resolution [7, 2]. This result is generalized here by relating the local consistency concepts with inference rules in the propositional logic. We relate the pure literal rule with an interesting property of constraint networks called "row convexity", and we propose an empirical characterization of satisfiable and unsatisfiable 3-SAT instances.

1 Introduction

This work is based on the literal encoding of the satisfiability problem (SAT) as a binary constraint satisfaction problem (CSP) [1]. The literal encoding of the satisfiability problem (SAT) as a binary constraint satisfaction problem (CSP) allows us to make a technical comparison between the propositional logic and the constraint network fields [7].

Equivalencies or analogies between concepts and techniques used in the two fields are then shown. For instance it is shown that arc consistency is equivalent to unit resolution [7, 2]. We generalize this result by relating the path consistency concept with unit and binary resolution rules (binary resolution will be defined later), and we relate the pure literal rule with an interesting property of constraint networks called "row convexity". The constraint networks verifying this property can be solved without backtracking [6]. From the link between the pure literal rule and the row convexity property we propose an empirical characterization of satisfiable and unsatisfiable 3-SAT instances.

The organization of this paper is as follows. Section 2 reviews basic definitions on a satisfiability problem and the constraint satisfaction frameworks. Section 3 presents the literal encoding and the connection between the path consistency concept and inference rules. Sections 4 recalls the row convexity property, and presents its relation with the pure literal rule and the measure which we propose for 3-SAT instances, and section 5 deals with the computational experiments. Section 6 concludes.

2 Background

A constraint network or a Constraint Satisfaction Problem (CSP) involves the assignment of values to variables which are subject to a set of constraints. Formally, a binary CSP is defined by a quadruplet (X,D,C,R) where:

- X is a set of n variables $\{X_1, X_2, \ldots, X_n\}$;
- D is a set of n domains $\{D_1, D_2, \ldots, D_n\}$ where each D_i is a set of d_i possible values for X_i;
- C is a set of m constraints where each constraint C_{ij} between the variables X_i and X_j, $i \neq j$, is defined by its relation R_{ij};
- R is a set of m relations, where R_{ij} is a subset of the Cartesian product $D_i \times D_j$.
  The predicate $R_{ij}(v_r, v_s)$ holds iff the pair (v_r, v_s) belongs to R_{ij}.
  For the problems of our interest here, we will require that $(v_r, v_s) \in R_{ij} \Leftrightarrow (v_s, v_r) \in R_{ji}$.

A solution of the CSP is a total assignment satisfying each of the constraints.

A binary relation between variables X_i and X_j may be represented as a (0-1)-matrix with $|D_i|$ rows and $|D_j|$ columns by imposing an ordering on the domains of the variables. A zero at the intersection of row r and column c means that the pair consisting of the r^{th} element of D_i and the c^{th} element of D_j is not allowed; an one at this intersection means that the pair is allowed.

Definition 2.1 *[6] A (0-1)-matrix is row convex if in each row there is no two 1 separated by 0. A relation is row convex if its associated matrix is row convex.*

We recall now the definitions of arc consistency and path consistency.

Definition 2.2 *A domain D_i of D is arc consistent (2-consistent) iff, for each $v_r \in D_i$, and for each $X_j \in X$ such that $C_{ij} \in C$, there exists $v_s \in D_j$ such that $R_{ij}(v_r, v_s)$ holds. A CSP is arc consistent iff for each $D_i \in D$, D_i is arc consistent and D_i not empty.*

Definition 2.3 *A pair of variables X_i,X_j is path consistent (3-consistent) iff for each pair (v_r, v_s) such that $R_{ij}(v_r, v_s)$ holds, and for each $X_k \in X$ such that $C_{ik} \in C$ and $C_{kj} \in C$, there exists $v_t \in D_k$ such that $R_{ik}(v_r, v_t)$ and $R_{jk}(v_s, v_t)$ hold. A CSP is path consistent iff each pair X_i,X_j is path consistent.*

A SAT instance is a conjunction of distinct clauses. A clause is a disjunction of literals. A literal is a propositional variable x or its negation $\bar{x}$. If l denotes a positive or a negative literal, $\bar{l}$ is its complement. A SAT instance is satisfiable (has a model), if there exists an assignment of its literals to true or false which makes all its clauses true. If every clause of the SAT instance exactly has k literals, the SAT problem is called k-SAT. If exactly one literal l occurs negated in one clause and unnegated in another, the resolution rule consists in generating a resolvent on l, the clause containing all literals occurring in either parent, except the literals l and $\bar{l}$. Obviously, a SAT instance can be seen as a particular CSP with the domain of each variable is the Boolean set {True, False}.

[1] LIPN, Institut Galilée, Université Paris 13, Av. J.B. Clément 93240 Villetaneuse, France, email: hachemi.bennaceur@lipn.univ-paris13.fr
[2] LaRIA, Université de Picardie Jules Verne, 5 Rue du Moulin Neuf, 80000 Amiens, France, email: cli@laria.u-picardie.fr

3 Local consistencies and inference rules

We recall the translation of a SAT instance to a binary CSP and we express the path consistency concept in the propositional logic framework. Namely, we show that achieving path consistency is equivalent to performing unit and binary resolution rules.

3.1 From SAT to binary CSP

The SAT problem may be translated into a binary CSP as follows : let S be a SAT problem with m clauses and n literals, we associate to each clause c_k of S a variable X_k, the domain D_k of X_k is the set of literals of c_k.

For instance, if c_k is the clause $l_1 \vee \bar{l}_2 \vee l_3 \vee \bar{l}_4$ then the domain D_k of X_k is $\{l_1, \bar{l}_2, l_3, \bar{l}_4\}$.

A constraint between two variables X_i, X_j associated to two clauses c_i and c_j is created if the clause c_i contains a literal l and the clause c_j contains its complement $\bar{l}$.

For instance, if c_i and c_j are respectively the clauses $l_1 \vee \bar{l}_2 \vee l_3$ and $\bar{l}_1 \vee l_3 \vee l_4$, a constraint is created between X_i and X_j since c_i contains l_1 and c_j contains $\bar{l}_1$.

For each constraint between two variables X_i and X_j , a relation R_{ij} is defined by the Cartesian product D_i x D_j minus the tuples (l, $\bar{l}$). For instance, below is the relation R_{ij} of the above constraint.

$$\begin{aligned} R_{ij} \; = \; & \{(l_1, l_3), (l_1, l_4), (\bar{l}_2, \bar{l}_1), (\bar{l}_2, l_3), \\ & (\bar{l}_2, l_4), (l_3, \bar{l}_1), (l_3, l_3), (l_3, l_4)\} \end{aligned}$$

We sometimes also explicitly associate a relation R_{ij} defined by the Cartesian product D_i x D_j to two clauses c_i and c_j not sharing any complementary literal.

In the rest of the paper, S denotes a SAT problem and $P(S)$ its associated CSP. S is satisfiable if and only if $P(S)$ has a solution [1].

Property 3.1 *[1] If each clause of S contains at least two literals (resp. three literals), then $P(S)$ is arc consistent (resp. path-consistent).*

3.2 Path consistency and Binary Resolution

Definition 3.1 *We call Unit (resp. Binary) Resolution the resolution rule applied to two clauses of which at least one is unary (resp. binary).*

Let UR(S) denotes the Unit Resolution closure of S. It is shown in [7] that:

Theorem 1 *[7] Achieving arc consistency on $P(S)$ is equivalent to computing $UR(S)$.*

The following result, generalizing theorem 1, relates path consistency to unit and binary resolution.

Let $BUR(S)$ denote the Binary and Unit Resolution closure of S.

Theorem 2 *Achieving path consistency on $P(S)$ is equivalent to computing $BUR(S)$.*

Proof

We show by induction that a pair (l_1, l_2) is path-inconsistent iff $\bar{l}_1 \vee \bar{l}_2$ or $\bar{l}_1$ or $\bar{l}_2$ is a clause of S, or one of these clauses is produced by the Binary or Unit Resolution during the constraint propagation.

Achieving path consistency on a CSP can be done in two phases: the checking phase and the constraint propagation phase. A pair of values can be detected inconsistent in one of these phases.

(i) A pair (l_1, l_2) is detected inconsistent during the checking phase iff $\bar{l}_1$ or $\bar{l}_2$ or $\bar{l}_1 \vee \bar{l}_2$ is a clause of S, since (l_1, l_2) has no support in these latter clauses.

(ii) A pair (l_1, l_2) of R_{ij} is detected inconsistent during the constraint propagation phase iff there exists a variable X_k of P(S) such that (l_1, l_2) has no support in D_k.

Assume that initially D_k is $\{l_3, l_4, l_5\}$ (note that before the filtering process D_k may contain more than three elements, in this case we develop the same reasoning). Then, the clause c_k associated to D_k is

$$l_3 \vee l_4 \vee l_5 \tag{1}$$

Since the pair (l_1, l_2) of R_{ij} is path-inconsistent, then one pair in $\{(l_1, l_3), (l_2, l_3)\}$, one pair in $\{(l_1, l_4), (l_2, l_4)\}$, and one pair in $\{(l_1, l_5), (l_2, l_5)\}$ are already detected path-inconsistent.

By induction, one of clauses 2 and 3

$$\bar{l}_1 \vee \bar{l}_3 \tag{2}$$
$$\bar{l}_2 \vee \bar{l}_3 \tag{3}$$

and one of clauses 4 and 5

$$\bar{l}_1 \vee \bar{l}_4 \tag{4}$$
$$\bar{l}_2 \vee \bar{l}_4 \tag{5}$$

and one of clauses 6 and 7

$$\bar{l}_1 \vee \bar{l}_5 \tag{6}$$
$$\bar{l}_2 \vee \bar{l}_5 \tag{7}$$

are generated by unit or binary resolutions during the constraint propagation phase. Assume that clauses 3, 4 and 7 are the clauses generated. The following binary resolutions

$$\begin{aligned} clause\ (1) \wedge clause\ (3) \;&\rightarrow\; \bar{l}_2 \vee l_4 \vee l_5 \qquad (8) \\ clause\ (8) \wedge clause\ (4) \;&\rightarrow\; \bar{l}_1 \vee \bar{l}_2 \vee l_5 \qquad (9) \\ clause\ (9) \wedge clause\ (7) \;&\rightarrow\; \bar{l}_1 \vee \bar{l}_2 \end{aligned}$$

lead to the binary clause $\bar{l}_1 \vee \bar{l}_2$.

All other possibilities lead to one of the clauses $\bar{l}_1 \vee \bar{l}_2, \bar{l}_1, \bar{l}_2$.

■

The above theorem allows us to achieve path consistency on a SAT problem without effectively transforming the SAT problem to a CSP. The computing of the $BUR(S)$ leads to a path consistent SAT problem (see example 4.1).

4 Characterization of satisfiable and unsatisfiable 3-SAT instances

We first relate the row convexity property of CSPs with the pure literal rule and then we present a measure characterizing satisfiable and unsatisfiable 3-SAT instances.

4.1 Row convexity

Van Beek and Dechter have identified a tractable class of CSPs for which a solution can be found using a path consistency algorithm [6]. The result is based on the row convexity property.

Definition 4.1 *[6] Given an ordering of the variables $X_1, ..., X_n$, a binary CSP is directionally row-convex if each of the (0,1)-matrices of R_{ij}, where variable X_i occurs before variable X_j in the ordering, is row convex.*

Theorem 3 *[6] Let P be a path consistent binary CSP. If there exists an ordering of the variables $X_1, ..., X_n$ and of the domains $D_1, ..., D_n$ of P such that P is directionally row convex, then a solution can be found without backtracking.*

Corollary 4.1 *The satisfiability problems containing only binary clauses (2-SAT) can be solved using a path consistency algorithm.*

Proof

Let S be a 2-SAT problem. For any ordering of its clauses, $P(S)$ is row convex, and achieving a path consistency on $P(S)$ leads either to a row convex and path consistent CSP for which a solution can be found without backtracking, or to empty relation of $P(S)$ and then we conclude that $P(S)$ has no solution. ■

Example 4.1 *Let S be the 2-SAT problem defined by the following set of clauses*

$$\bar{l}_1 \vee \bar{l}_2 \quad (10)$$
$$\bar{l}_3 \vee l_2 \quad (11)$$
$$\bar{l}_3 \vee l_1 \quad (12)$$
$$l_3 \vee l_4 \quad (13)$$
$$l_3 \vee l_5 \quad (14)$$
$$\bar{l}_4 \vee \bar{l}_5 \quad (15)$$

We solve this 2-SAT instance in two corresponding ways by achieving path consistency on $P(S)$ or by computing $BUR(S)$ (See theorem 2).

The corresponding CSP $P(S)$ contains 6 variables $X_1, X_2, X_3, X_4, X_5, X_6$. The domains of these variables are $D_1 = \{\bar{l}_1, \bar{l}_2\}$, $D_2 = \{\bar{l}_3, l_2\}$, $D_3 = \{\bar{l}_3, l_1\}$, $D_4 = \{l_3, l_4\}$, $D_5 = \{l_3, l_5\}$, $D_6 = \{\bar{l}_4, \bar{l}_5\}$. The relation between variables X_2 and X_6 is

$$R_{26} = \{\bar{l}_3 \vee \bar{l}_4, \bar{l}_3 \vee \bar{l}_5, l_2 \vee \bar{l}_4, l_2 \vee \bar{l}_5\}$$

The checking phase on $P(S)$:
$(\bar{l}_3, \bar{l}_4)$ of R_{26} has no support in D_4,
$(\bar{l}_3, \bar{l}_5)$ of R_{26} has no support in D_5.
Then, the pairs $(\bar{l}_3, \bar{l}_4)$ and $(\bar{l}_3, \bar{l}_5)$ can be removed from R_{26}; Thus, the value $\bar{l}_3$ can be removed from D_2.
In the same way, we deduce that the value $\bar{l}_3$ can be removed from D_3 (using relation R_{36}).
Thus $D_2 = \{l_2\}$ and $D_3 = \{l_1\}$.

Binary Resolution Rules corresponding to the checking phase:

$$clause\ (13) \wedge clause\ (15) \quad \rightarrow \quad l_3 \vee \bar{l}_5 \quad (16)$$
$$clause\ (14) \wedge clause\ (16) \quad \rightarrow \quad l_3 \quad (17)$$
$$clause\ (11) \wedge clause\ (17) \quad \rightarrow \quad l_2 \quad (18)$$
$$clause\ (12) \wedge clause\ (17) \quad \rightarrow \quad l_1 \quad (19)$$

The constraint propagation process phase:
The pair (l_2, l_1) of R_{23} has no support in D_1, then it can be removed from R_{23} so that D_2 and D_3 become empty. Thus P(S) has no solution.
Resolution Rule corresponding to the constraint propagation phase:

$$clause\ (10) \wedge clause\ (18) \quad \rightarrow \quad \bar{l}_1 \quad (20)$$
$$clause\ (19) \wedge clause\ (20) \quad \rightarrow \quad empty\ clause$$

Thus S is unsatisfiable.

We have proved in two corresponding ways that the 2-SAT instance is unsatisfiable by achieving path consistency on its associated CSP and by computing $BUR(S)$.

4.2 Row convexity and the pure literal rule

The following result relates the row convexity property with the pure literal rule using the literal encoding of a 3-SAT instance.

Lemma 4.1 *Let S be a 3-SAT instance and $P(S)$ its associated CSP. If $P(S)$ is directionally row convex then there exists in S a pure literal.*

Proof

Assume that S does not contain any pure literal. Let c_1 be a clause, for any literal ordering of c_1, there exists a clause c_2 containing the complementary literal of the literal placed in the middle of c_1. Thus, the matrix encoding the relation between the clauses c_2 and c_1 is not row convex, hence, $P(S)$ is not directionally row convex. ■

Theorem 4 *Let m be the number of clauses of S. $P(S)$ is directionally row convex iff there exists a partition $S_1, ..., S_k : (1 \leq k \leq m)$ of the set of clauses S such that S contains a pure literal which occurs only in the clauses of S_1, and each set of clauses $S - (S_1 \cup ... \cup S_i) : (1 \leq i \leq k)$ contains a pure literal.*

Proof

$\Rightarrow$ By construction, we determine an ordering of the clauses of S such that $P(S)$ will be directionally row convex as follows : all the clauses of S_1 must be placed in the queue of the ordering and the pure literal of S must be placed at the middle of each clause of S_1. We repeat the same ordering operation for S_2, S_3,... and S_k; the clauses of S_2 will be placed before the clauses of S_1 in the ordering and the pure literal of $S - S_1$ will be placed in the middle of the clauses of S_2, and so on for others sets of clauses $S_3, ..., S_k$. Thus, the ordering clauses $S_k, S_{k-1}, ..., S_1$ leads to a directionally row convex CSP $P(S)$.

$\Leftarrow$ The partition $S_1,...,S_k : (1 \leq k \leq m)$ of S is determined as follows. Since $P(S)$ is directionally row convex. By Lemma 4.1 there exists in S a pure literal l_1, let S_1 be the set of clauses in which l_1 occurs. Since $P(S)$ is directionally row convex, $P(S - S_1)$ is directionally row convex too. Thus, by Lemma 4.1, there exits in $S - S_1$ a pure literal l_2, the set of clauses in which the literal l_2 occurs forms S_2. In the same way, $S_3, ..., S_k$ will be formed. ∎

4.3 Measure characterizing satisfiable and unsatisfiable 3-SAT instances

When the CSP is not row convex, we define the minimal row non-convexity NC of a 3-SAT instance to be the smallest number of row non-convex (0,1)-matrices representing the constraints for any variable and value ordering. We use a simple heuristic shown in figure 1 to estimate NC for a 3-SAT instance. This heuristic is based on the proof of Theorem 4.

Procedure MinimalRowNonConvexity(S)
Begin
```
NC:=0;
repeat
   let neg(x) and pos(x) denote the number of
   positive and negative occurrences of
   variable x in the current S;
   select x such that neg(x) * pos(x)
   is the smallest,
   break tie by choosing x such that
   neg(x) + pos(x) is the largest;
   remove all clauses containing x or x̄ from
   S, for every removed clause c, record its
   removing time f(c);
   Order the literals of the removed clauses
   such that x or x̄ is in the middle position;
   NC:=NC + neg(x) * pos(x);
until S becomes empty;
put all removed clauses in order of f(c);
return NC;
```
End

Figure 1.

The MinimalRowNonConvexity procedure essentially computes an ordering of clauses in the initial S and of literals in each clause. If $P(S)$ is directionally row convex, the procedure returns 0 and is similar to the FindOrder procedure in [6], otherwise it computes the number of row non-convex (0,1)-matrices under the ordering. To see this, it is sufficient to note that only the (0,1)-matrices between clauses removed at the same stage of the algorithm can have zeros separating two ones, because of the complementary literal in the middle position (if any).

Intuitively, since the MinimalRowNonConvexity procedure first treats all clauses containing x or $\bar{x}$ such that $neg(x) * pos(x)$ is the smallest, NC increases slowly. The removing of these clauses reduces the number of occurrences of remaining variables, eventually introducing some pure literals. For these reasons, we believe that the MinimalRowNonConvexity procedure computes an ordering of clauses and literals such that the number NC of row non-convex constraints is small. The counting of the number of occurrences of variables can be done in $O(n + m)$ time, where m and n are respectively the number of clauses and variables in S. So the complexity of the MinimalRowNonConvexity procedure is $O(n(n + m))$.

5 Experimentation

We use the MinimalRowNonConvexity procedure to estimate NC for a large sample of random 3-SAT instances. We vary n from 300 to 450 incrementing by 50 and m so that $m/n = 4, 4.1, 4.2, 4.25, 4.3, 4.4, 4.5$ respectively. Empirically, when the ratio m/n is near 4.25, S is unsatisfiable with the probability 0.5 and is the most difficult to solve [5, 3].

Table 1 shows the number of instances included in our study at each point $(m/n, n)$.

Table 1. The number of instances studied at each point $(m/n, n)$

	300	350	400	450
4	3000	3000	3000	3000
4.1	1000	1000	600	400
4.2	1000	1000	600	400
4.25	1000	1000	600	400
4.3	1000	1000	600	400
4.4	1000	1000	600	400
4.5	1000	1000	600	400

In order to compare the minimal row non-convexity NC of satisfiable and unsatisfiable instances, we solve these instances using Satz [4] and divide the instances at each point into satisfiable class and unsatisfiable class. Tables 2 and 3 show the average value of NC (in boldface) for each class and the number of instances in the class (between parentheses).

Table 2. Average estimated minimal row non-convexity NC of satisfiable instances at each point $(m/n, n)$

	300	350	400	450
4	**1005** (2983)	**1169** (2998)	**1335** (2999)	**1500** (2999)
4.1	**1074** (944)	**1251** (972)	**1425** (582)	**1606** (392)
4.2	**1140** (715)	**1327** (746)	**1515** (452)	**1706** (307)
4.25	**1172** (551)	**1367** (547)	**1561** (319)	**1754** (224)
4.3	**1208** (370)	**1406** (353)	**1604** (197)	**1802** (128)
4.4	**1272** (128)	**1486** (85)	**1686** (46)	**1905** (27)
4.5	**1353** (32)	**1564** (11)	**1770** (5)	**2002** (4)

Table 3. Average estimated minimal row non-convexity NC of unsatisfiable instances at each point $(m/n, n)$

	300	350	400	450
4	**1040** (17)	**1244** (2)	**1375** (1)	**1596** (1)
4.1	**1108** (56)	**1274** (28)	**1462** (18)	**1640** (8)
4.2	**1163** (285)	**1351** (254)	**1544** (148)	**1726** (93)
4.25	**1197** (449)	**1392** (453)	**1585** (281)	**1778** (176)
4.3	**1229** (630)	**1429** (647)	**1630** (403)	**1835** (272)
4.4	**1302** (872)	**1512** (915)	**1728** (554)	**1942** (373)
4.5	**1378** (968)	**1603** (989)	**1830** (595)	**2059** (396)

In the sequel, we directly denote by NC the average estimated minimal row non-convexity of a class of instances. We have empirically observed in tables 2 and 3 that the ratio NC/n tends to be a constant for a given value m/n and a class, except for the points where few instances are solved (e.g. unsatisfiable instances for $m/n = 4$, satisfiable instances for $m/n = 4.5$). This is particular true for satisfiable instances when $m/n \leq 4.25$ and for unsatisfiable instances when $m/n > 4.25$. The difference of NC/n

between satisfiable and unsatisfiables classes tends to become larger when $m/n < 4.2$ and $m/n > 4.3$, which can be used to explain the threshold phenomenon. Tables 4 and 5 show the ratio NC/n in boldface (and the number of instances averaged between parentheses) at each point $(m/n, n)$ for satisfiable and unsatisfiable instances.

Table 4. Ratio NC/n for satisfiable instances at each point $(m/n, n)$

	300	350	400	450
4	**3.35** (2983)	**3.34** (2998)	**3.34** (2999)	**3.34** (2999)
4.1	**3.58** (944)	**3.57** (972)	**3.56** (582)	**3.57** (392)
4.2	**3.80** (715)	**3.79** (746)	**3.79** (452)	**3.79** (307)
4.25	**3.90** (551)	**3.91** (547)	**3.90** (319)	**3.90** (224)
4.3	**4.02** (370)	**4.02** (353)	**4.01** (197)	**4.00** (128)
4.4	**4.24** (128)	**4.25** (85)	**4.22** (46)	**4.23** (27)
4.5	**4.51** (32)	**4.49** (11)	**4.43** (5)	**4.45** (4)

Table 5. Ratio NC/n for unsatisfiable instances at each point $(m/n, n)$

	300	350	400	450
4	**3.47** (17)	**3.55** (2)	**3.43** (1)	**3.55** (1)
4.1	**3.69** (56)	**3.64** (28)	**3.65** (18)	**3.64** (8)
4.2	**3.88** (285)	**3.86** (254)	**3.86** (148)	**3.84** (93)
4.25	**3.99** (449)	**3.98** (453)	**3.96** (281)	**3.95** (176)
4.3	**4.10** (630)	**4.08** (647)	**4.08** (403)	**4.08** (272)
4.4	**4.34** (872)	**4.32** (915)	**4.32** (554)	**4.32** (373)
4.5	**4.59** (968)	**4.58** (989)	**4.58** (595)	**4.58** (396)

To verify the phenomenon, we solve a large sample of instances at other points from $n = 220$ to $n = 380$ increasing by step 20 (excluding $n = 300$). 1000 instances are solved at each point. Tables 6 and 7 show the ratio NC/n at these points. Note that there are always few unsatisfiable instances when $m/n = 4$ and few satisfiable instances when $m/n = 4.5$.

Table 6. Ratio NC/n for satisfiable instances at each point $(m/n, n)$

	220	240	260	280	320	340	360	380
4	**3.36**	**3.36**	**3.35**	**3.35**	**3.35**	**3.34**	**3.34**	**3.34**
4.1	**3.59**	**3.58**	**3.57**	**3.58**	**3.57**	**3.56**	**3.56**	**3.56**
4.2	**3.81**	**3.81**	**3.80**	**3.80**	**3.80**	**3.79**	**3.79**	**3.79**
4.25	**3.92**	**3.91**	**3.90**	**3.91**	**3.92**	**3.89**	**3.90**	**3.90**
4.3	**4.03**	**4.02**	**4.02**	**4.02**	**4.02**	**4.00**	**4.00**	**4.01**
4.4	**4.26**	**4.23**	**4.26**	**4.25**	**4.24**	**4.22**	**4.23**	**4.24**
4.5	**4.51**	**4.47**	**4.43**	**4.48**	**4.43**	**4.44**	**4.43**	**4.51**

It appears that the ratio NC/n could be used to estimate the satisfiability of a given random 3-SAT instance. For example, we know that an instance near the threshold ($m/n = 4.25$) is satisfiable with an empirical probability 0.5. However if $NC/n < 3.90$, it should be satisfiable with a more important probability. One might also think that a reason for the unsatisfiability of an instance is the large number of its row non-convex (0-1)-matrices.

6 Conclusion

We have found that it is possible to associate to each local consistency level in the constraint network equivalent inference rules in the propositional logic. We have expressed the path consistency technique in propositional logic and relating an interesting property of CSPs called Row Convexity with the pure literal rule. We have then proposed an emperical measure based on this property for identifying satisfiable and unsatisfiable 3-SAT instances. We have empirically observed that the measure tends to be a constant for a given value m/n. This result might be used to partly explain why an instance is unsatisfiable.

Table 7. Ratio NC/n for unsatisfiable instances at each point $(m/n, n)$. At points (400, 4) and (450, 4), none of the 1000 solved instances is unsatisfiable

	220	240	260	280	320	340	360	380
4	**3.47**	**3.47**	**3.51**	**3.46**	**3.45**	**3.53**	**-**	**-**
4.1	**3.70**	**3.69**	**3.69**	**3.65**	**3.65**	**3.66**	**3.69**	**3.61**
4.2	**3.89**	**3.89**	**3.89**	**3.88**	**3.86**	**3.86**	**3.86**	**3.85**
4.25	**4.00**	**4.00**	**3.99**	**3.99**	**3.99**	**3.97**	**3.97**	**3.96**
4.3	**4.12**	**4.11**	**4.10**	**4.09**	**4.09**	**4.07**	**4.08**	**4.07**
4.4	**4.35**	**4.35**	**4.34**	**4.34**	**4.33**	**4.31**	**4.32**	**4.31**
4.5	**4.61**	**4.60**	**4.60**	**4.59**	**4.59**	**4.57**	**4.57**	**4.57**

REFERENCES

[1] Bennaceur H., *The Satisfiability problem regarded as constraint-satisfaction problem*, In the proceeding of the 12th European Conference on Artificial Intelligence, Page 125-130, Budapest, Hungary, August 1996.

[2] Bennaceur H. *A comparison between SAT and CSP techniques*, Research Report LIPN 2001-10.

[3] Crawford, J.M., Auton L.D., Experimental Results on the Crossover Point in Random 3-SAT, *Artificial Intelligence*, 81, 1996.

[4] Li C.M., Anbulagan, *Heuristics Based on Unit Propagation for Satisfiability Problems*, In Proceedings of IJCAI-97, ISBN 1-55860-480-4, Page 366-371, Nagoya, Japan, August 1997.

[5] Mitchell, D, Selman, B., Levesque H., Hard and Easy Distributions of SAT Problems, In *AAAI'92*, pages 459-465, San Jose, CA, 1992.

[6] Van Beek P., Dechter R., *On the Minimality and Global Consistency of Row-Convex Constraint Networks*, Journal of the ACM, 42:543-561, 195.

[7] Walsh T., *SAT v CSP*, Proceedings of CP-2000, pages 441-456, Springer-Verlag LNCS-1894, 2000.