A Heuristic Approach for Solving the Cubic Monotone 1-in-3 SAT Problem

O.Kettani
Scientific Institute, Mohammed V University, Rabat, Morocco
e-mail: kettani.o@gmail.com

***Abstract*— The present paper proposes a heuristic approach to solve a variant of the satisfiability problem (SAT) , namely the exact 3-satisfiability problem (X3SAT) which is known to remains NP-complete when restricted to expressions where every variable has exactly three occurrences, even in the absence of negated variables (Cubic Monotone 1-in-3 SAT Problem). Firstly, this problem is converted into an equivalent system of linear equations over binary variables, then it is solved using a dynamic programming heuristic based on Das algorithm requiring $O(n^4)$ time, for input formulas over n variables. Finally, a Matlab code implementation is provided.**

Keywords—X3SAT, Cubic Monotone 1-in-3 SAT Problem, NP-complete, dynamic programming, heuristic

I. Introduction

The propositional satisfiability problem (SAT) is one of the first problems that have been proven to be NP-complete [1]. The present paper is devoted to solve a variant of SAT, namely the exact 3-satisfiability problem (X3SAT) which is known to remains NP-complete for expressions where every variable has exactly three occurrences, even in the absence of negated variables (Cubic Monotone 1-in-3 SAT Problem) [2] [9].

XSAT and X3SAT have been recently investigated in [3, 4, 5, 6]. However the first breakthrough result [7] dates back to the year 1980 and provides an algorithm deciding XSAT in $O(2^{0.2441n})$ time, for input formulas over n variables. This bound has been the best known until 2003, then it has been improved to $O(2^{0.2325n})$ [8]. The best known result for unweighted X3SAT provides a solution in $O(2^{0.1379n})$ time [8].

In the present paper, the Cubic Monotone 1-in-3 SAT problem is firstly converted into an equivalent system of linear equations over binary variables, then it is solved using a dynamic programming heuristic approach in $O(n^4)$ time, based on Das algorithm [10].

II. Preliminaries

A literal is a Boolean variable a ∈ {0, 1} or its negation a. A clause C is the disjunction of different literals and is represented as a literal set. A k-clause is a clause that contains exactly k literals. A CNF formula F is a conjunction of different clauses and is represented as a clause set. A k -CNF formula is a formula that contains only clauses of a maximum length k. For a given formula F (resp. clause C), we denote the set of contained variables by V (F) (resp.V (C)). Similarly, V (l) denotes the underlying variable of a literal l. pol(l) denotes the polarity of a literal in a fixed clause of a formula. By V+ (F) (resp. V− (F)) we denote the set of all variables occurring positive (resp.negated). We call a variable (resp. a literal corresponding to it) unique if it occurs in the formula only once. We distinguish between the length of a formula F and the number of its clauses |F |. Let CNF denote the set of all formulas and let CNF+ denote the set of positive monotone formulas, i.e., each clause contains only unnegated variables. Given a formula F and a variable a, the formula F [a ← 1] (resp. F [a ← 0]) is obtained from F by setting the value of a to 1 (resp.0).

The exact 3-satisfiability problem (X3SAT) asks in its decision version, whether there exists a truth assignment t : V (F) ← {0, 1}, setting exact one literal to 1 in each clause of F . We call such an assignment t an x-model, and we denote with X3SAT the set of all exact satisfiable 3-CNF formulas. In the search version of X3SAT one has to decide whether F ∈ X3SAT, and in the positive case to find an x-model of F. X3SAT restricted to expressions where every variable has exactly three occurrences, without negated variables is called Cubic Monotone 1-in-3 SAT Problem [9].

The rest of this paper is organized as follows; in the next section, proposition 1 establishes the equivalence between the Cubic Monotone 1-in-3 SAT problem, and a system of linear equations over binary variables. Section 4 proposes a dynamic programming heuristic based on Das algorithm [10] for solving this system. Section 5 provides some examples to illustrate the effectiveness of this approach. Finally, conclusion of the paper is summarized in Section 6.

III. METHODOLOGY

Propostion 1:

Finding an x-model of F∈ Cubic Monotone 1-in-3 SAT problem is equivalent to solve the system of linear equations Ax=b, over binary variables $x \in \{0, 1\}^n$, where A is the nxn matrix defined by:

$A_{ij}=1$ if literal j appears in clause i

$A_{ij}=0$ otherwise

and b is the n vector b=(1,....,1).

Proof:

Clearly, x is a model of F∈ Cubic Monotone 1-in-3 SAT problem is equivalent $\forall i=1...n \sum_{j=1...n} A_{ij}x_j=1$, which is equivalent to Ax=b, over $\{0, 1\}^n$

Notice that F∈ Cubic Monotone 1-in-3 SAT problem implies that the number of clauses m=n=3p, where p is an integer. Indeed: Ax=b implies that $||Ax||^2=||b||^2$,

Let $x=\sum_{j\in J} \varepsilon_j$ where $J=\{j\in\{1,..,n\}\ x_j=1\}$ and $(\varepsilon_j)_{j\in\{1,..,n\}}$ is the canonical basis of R^n.

Then $Ax=A\sum_{J\in j} j$ where A_j is the j^{th} column of A matrix.

Then $||Ax||^2=||\sum_{J\in j} A_j||^2=m$

Since $\forall i,j\in JxJ\ A_iA_j=0$ because $A\sum_{J\in j} j=b$ and $||A_j||^2=3$

Then necessarily $3|J|=m=3p$ where $p=|J.|$

On the other hand, let N_1 be the number of ones of A matrix.

Since F∈ Cubic Monotone 1-in-3 SAT, if we count row by row then we obtain $N_1=3m$, and if we count column by column, then we obtain $N_1=3n$.

Thus m=n=3p.

Propostion 2:

The system of linear equations Ax=b, over binary variables $x \in \{0, 1\}^n$ can be solved using the following heuristic based on Das algorithm [10] :

Input: n,A,b=(1,....,1)
Output: $x \in \{0, 1\}^n$

```
B=b'
FOR j=n downto 1 DO
C=B-A(:,j)
A(:,j) ← NIL
e0=norm(A*(A+*B)-B)
e1=norm(A*(A+*C)-C)
IF e0 < e1 THEN x(j) ← 0
ELSE x(j) ← 1
B← C
END IF
END FOR
IF Ax=b THEN "x is a solution of the system"
END IF
```

Where A^+ is the pseudo-inverse of matrix A, x(j) denotes the j^{th} component of vector x, A(:,j) denotes the j^{th} column of matrix A and norm is the Euclidean norm.

Notice that since the main loop requires n iterations which mainly computes matrices multiplications in $O(n^3)$, then the time complexity of the proposed approach is $O(n^4)$.

IV. EXAMPLES

Example 1

n=15

F=(x1∨x15∨x3)∧(x10∨x13∨x2)∧(x3∨x13∨x12)∧(x4∨x3∨x1)∧(x11∨x5∨x8)∧(x8∨x12∨x13)∧(x8∨x15∨x11)∧(x10∨x15∨x11)∧(x7∨x4∨x9)∧(x9∨x1∨x12)∧(x4∨x6∨x9)∧(x2∨x10∨x7)∧(x6∨x14∨x5)∧(x5∨x7∨x14)∧(x14∨x2∨x6)

A=

101000000000001
010000000100100
001000000001100
101100000000000
000010010010000
000000010001100
000000010010001
000000000110001
000100101000000
100000001001000
000101001000000
010000100100000
000011000000010
000010100000010
010001000000010

solution found:

x=(100001100010100)

Example 2

n=27

F=(x22∨x14∨x11)∧(x23∨x24∨x27)∧(x6∨x24∨x9)∧(x12∨x23∨x10)∧(x12∨x3∨x19)∧(x20∨x9∨x10)∧(x20∨x27∨x19)∧(x3∨x13∨x18)∧(x15∨x4∨x8)∧(x27∨x22∨x26)∧(x10∨x6∨x19)∧(x12∨x25∨x23)∧(x11∨x20∨x8)∧(x8∨x9∨x1)∧(x25∨x7∨x14)∧(x22∨x11∨x6)∧(x21∨x17∨x14)∧(x1∨x16∨x5)∧(x5∨x1∨x13)∧(x2∨x5∨x16)∧(x3∨x7∨x4)∧(x15∨x18∨x4)∧(x26∨x16∨x21)∧(x7∨x18∨x26)∧(x13∨x15∨x25)∧(x2∨x21∨x17)∧(x24∨x2∨x17)

A=

000000000010010000000100000
000000000000000000000011001
000001001000000000000001000
000000000101000000000010000
001000000001000000100000000
000000001100000000010000000
000000000000000000110000001
001000000000100001000000000
000100010000001000000000000
000000000000000000000100011
000001000100000000100000000
000000000001000000000010100
000000010010000000010000000
100000011000000000000000000
000000100000001000000000100

000001000010000000000100000
000000000000010010001000000
100010000000000100000000000
100010000000100000000000000
010010000000000100000000000
001100100000000000000000000
000100000000001001000000000
000000000000000100001000010
000000100000000001000000010
000000000000101000000000100
010000000000000010001000000
010000000000000010000001000
solution found:
x=(111001000000011000010010010)

Example 3

n=24

F=(x21∨x16∨x7)∧(x22∨x19∨x14)∧(x22∨x9∨x15)∧(x3∨x10∨x5)∧(x20∨x18∨x17)∧(x22∨x20∨x14)∧(x1∨x3∨x13)∧(x23∨x21∨x24)∧(x9∨x8∨x19)∧(x6∨x24∨x23)∧(x9∨x17∨x18)∧(x20∨x15∨x3)∧(x14∨x2∨x19)∧(x16∨x15∨x6)∧(x18∨x1∨x21)∧(x13∨x17∨x8)∧(x7∨x4∨x23)∧(x16∨x24∨x5)∧(x12∨x2∨x13)∧(x2∨x5∨x12)∧(x12∨x4∨x11)∧(x6∨x1∨x8)∧(x10∨x4∨x11)∧(x7∨x10∨x11)

A=
000000100000000100001000
000000000000010000100100
000000001000001000000100
001010000100000000000000
000000000000000011010000
000000000000010000010100
101000000000100000000000
000000000000000000001011
000000011000000000100000
000001000000000000000011
000000001000000011000000
001000000000001000010000
010000000000010000100000
000001000000001100000000
100000000000000001001000
000000010000100010000000
000100100000000000000010
000010000000000100000001
010000000001100000000000
010010000001000000000000
000100000011000000000000
100001010000000000000000
000100000110000000000000
000000100110000000000000
solution found:
x=(011000010010000101000110)

Example 4

n=18

F=(x4∨x6∨x9)∧(x7∨x15∨x18)∧(x3∨x5∨x13)∧(x7∨x18∨x12)∧(x16∨x8∨x12)∧(x18∨x11∨x17)∧(x13∨x9∨x12)∧(x16∨x4∨x8)∧(x8∨x17∨x2)∧(x1∨x11∨x14)∧(x10∨x5∨x1)∧(x13∨x10∨x2)∧(x17∨x6∨x5)∧(x3∨x6∨x2)∧(x16∨x14∨x11)∧(x15∨x3∨x1)∧(x9∨x10∨x15)∧(x4∨x7∨x14)

A=

000101001000000000
000000100000001001
001010000000100000
000000100001000001
000000010001000100
000000000010000011
000000001001100000
000100010000000100
010000010000000010
100000000010010000
100010000100000000
010000000100100000
000011000000000010
011001000000000000
000000000010010100
101000000000001000
000000001100001000
000100100000010000

solution found:

x=(010110000011001000)

Example 5

n=39

F=(x16∨x10∨x19)∧(x20∨x16∨x38)∧(x3∨x1∨x35)∧(x30∨x38∨x14)∧(x36∨x14∨x10)∧(x38∨x8∨x34)∧(x28∨x1∨x8)∧(x10∨x39∨x4)∧(x33∨x17∨x19)∧(x3∨x29∨x24)∧(x3∨x33∨x12)∧(x8∨x7∨x6)∧(x32∨x22∨x4)∧(x9∨x31∨x27)∧(x35∨x6∨x16)∧(x20∨x36∨x22)∧(x28∨x1∨x2)∧(x19∨x28∨x29)∧(x39∨x25∨x36)∧(x6∨x11∨x7)∧(x37∨x30∨x4)∧(x7∨x24∨x17)∧(x20∨x27∨x35)∧(x21∨x5∨x24)∧(x26∨x5∨x39)∧(x21∨x30∨x13)∧(x27∨x9∨x11)∧(x14∨x29∨x18)∧(x22∨x2∨x18)∧(x18∨x26∨x13)∧(x2∨x17∨x32)∧(x5∨x15∨x31)∧(x33∨x9∨x13)∧(x26∨x37∨x31)∧(x25∨x32∨x34)∧(x25∨x11∨x34)∧(x12∨x15∨x23)∧(x21∨x23∨x12)∧(x23∨x37∨x15)

A=

000000000100000100100000000000000000000
000000000000000100010000000000000000010
101000000000000000000000000000000010000
000000000000010000000000000001000000010
000000000100010000000000000000000001000
000000010000000000000000000000000100010
100000010000000000000000000100000000000
000100000100000000000000000000000000001
000000000000000010100000000000001000000
001000000000000000000001000010000000000
001000000001000000000000000000001000000
000001110000000000000000000000000000000
000100000000000000000100000000010000000
000000001000000000000000001000100000000

000001000000000010000000000000000000010000
00000000000000000001010000000000000001000
1100000000000000000000000000100000000000
00000000000000000010000000011000000000000
000000000000000000000000100000000001001
0000011000100000000000000000000000000000
0001000000000000000000000000001000000100
0000001000000000100000010000000000000000
00000000000000000010000001000000010000
00001000000000000000100100000000000000000
000010000000000000000001000000000001
00000000000010000000100000000100000000
00000000101000000000000000001000000000000
000000000000100010000000000010000000000
0100000000000000010001000000000000000000
000000000000100001000000010000000000000
0100000000000000100000000000000010000000
000010000000000100000000000000000100000000
0000000010001000000000000000000001000000
000000000000000000000000100001000000100
0000000000000000000000010000000101000000
0000000000100000000000000010000000001000000
000000000001001000000001000000000000000000
00000000000100000000101000000000000000000
000000000000001000000010000000000000000100

solution found:

x=(10001100010110001000010010101000000110)

V. Conclusion and Future Scope

In this work, a dynamic programming heuristic approach was proposed for solving a system of linear equations Ax=b, equivalent to the Cubic Monotone 1-in-3 SAT Problem which is an NP-complete problem.

Future work will focus on the improvement of this method, and try to find when this problem has no solution.

References

[1] Cook, S.A.: “The Complexity of Theorem Proving Procedures”. In: Proceedings of the 3rd ACM Symposium on Theory of Computing, pp. 151-158, 1971

[2] Schaefer, T.J.: “The complexity of satisfiability problems”. In: Proceedings of the 10th ACM Symposium on Theory of Computing, pp. 216-226, 1978

[3] Byskov, J.M., Ammitzboll Madsen, B., Skjernaa, B.:”New Algorithms for Exact Satisfiability”. Theoretical Comp. Science 332 (2005) 515-541

[4] Dahll ̈f, V., Jonsson, P., Beigel, R.: “Algorithms for four variants of the exact satisfiability problem”. Theoretical Comp. Sci. 320 (2004) 373-394

[5] Kulikov, A., “An upper bound O(20.16254^n) for Exact 3-Satisfiability”. Journal of Mathematical Sciences,126 (2005) 1995-1999

[6] Porschen, S., Rorerath, B., Speckenmeyer, E.: “Exact 3-Satisfiability is Decidable in Time O(20.16254^n)”.Annals of Mathematics and Artificial Intelligence 43 (2005) 173-193

[7] Monien, B., Speckenmeyer, E., Vornberger, O.: “Upper Bounds for Covering Problems”. Methods of Operations Research 43 (1981) 419-431

[8] Byskov, J.M., Ammitzboll Madsen, B., Skjernaa, B.:”New Algorithms for Exact Satisfiability”. Theoretical Comp. Science 332 (2005) 515-541

[9] Cristopher Moore, John Michael Robson: “Hard Tiling Problems with Simple Tiles”. Journal of Discrete & Computational Geometry; Volume 26 Issue 4, January 2001 Pages 573-590 Springer-Verlag

[10] Subhendu Das “Binary Solutions for Overdetermined Systems of Linear Equations” Advanced Modeling and Optimization, Volume 14, Number 1, 2012

Appendix

A Matlab code implementation:

```
clear all;
p=input('input p=');
n=3*p;
aa=[];
b=[];
col=[];
for i=1:n
for j=1:n
aa(i,j)=0
end;
end;
for i=1:n
col(i)=0
end;
for i=1:n
r = randint(1,1,[1,n]);
r1=r(1);
while (col(r1)==3)
r = randint(1,1,[1,n]);
r1=r(1);
end;
r = randint(1,1,[1,n]);
r2=r(1);
while (r1==r2)| (col(r2)==3)
r = randint(1,1,[1,n]);
r2=r(1);
end;
col(r1)=col(r1)+1;
col(r2)=col(r2)+1;
r = randint(1,1,[1,n]);
r3=r(1);
if i<n
while (r3==r2)|(r3==r1)| (col(r3)==3)
r = randint(1,1,[1,n]);
r3=r(1);
end;
col(r3)=col(r3)+1;
else
r3=find(col(:)==2)
col(r3)=3
end;
a(i,1)=r1;
a(i,2)=r2;
a(i,3)=r3;
aa(i,r1)=1;
aa(i,r2)=1;
```

```
aa(i,r3)=1;
fprintf('r1( %d', i);fprintf(')= %d\t', r1);
fprintf('r2( %d', i);fprintf(')= %d\t', r2);
fprintf('r3( %d', i); fprintf(')= %d\n', r3);
end;
file_1 = fopen('a.txt','wt')
file_2 = fopen('c.txt','wt')
file_3 = fopen('s.txt','wt')
for i=1:n
for j=1:n
fprintf(' %d\t',aa(i,j));
fprintf(file_1,'%d', aa(i,j))
end;
fprintf('\n');
fprintf(file_1,'\n');
end;
for i=1:n
fprintf(file_2,'(')
fprintf(file_2,'x%d', a(i,1))
fprintf(file_2,'or')
fprintf(file_2,'x%d', a(i,2))
fprintf(file_2,'or')
fprintf(file_2,'x%d', a(i,3))
fprintf(file_2,')and')
end;
tic
b=[]
for i=1:n
b(i)=1;
end;
tic
intcon=1:n
lb=zeros(n,1)
ub=ones(n,1)
x = intlinprog(b,intcon,aa,b,aa,b,lb,ub);
toc
disp(x);
disp('TIME intlinprog');
disp(toc);
rep=input('VU')
if length(x)>0
for i=1:n
fprintf('x( %d', i); fprintf(')= %.d\n', x(i));
end;
disp('checking');
y=aa*x;
for i=1:n
fprintf('y( %d', i); fprintf(')= %.d\n', y(i));
```

```
end;
end;
%===================================================================
tic


A=aa
B=b'
for i=n:-1:1
B1=B-A(:,i)
A(:,i)=[]
e0 = norm(A*(pinv(A)*B)-B)
e1 = norm(A*(pinv(A)*B1)-B1)
if e0<e1
x(i)=0
else
x(i)=1
B=B1
end;
end
toc
disp(x');
y=aa*x;
disp('checking ...');
disp(y');
disp(toc);
if y==b'
disp('x is solution');
disp(x');
end;
end;
```