A new approach to address Subset Sum Problem

Eeti Jain
CSE Department
Institute of Technology
Nirma University,Ahmedabad
Email: 11BCE027@nirmauni.ac.in

Akarsh Jain
CSE Department
Institute of Technology
Nirma University,Ahmedabad
Email: 11BCE061@nirmauni.ac.in

Sapan H. Mankad
CSE Department
Institute of Technology
Nirma University,Ahmedabad
Email: sapanmankad@nirmauni.ac.in

***Abstract*—In this paper, Subset Sum problem (Non Polynomial Complete problem) and analysis of its two solutions is discussed. These solutions are the Dynamic Solution algorithm and the Randomized algorithm. Dynamic Solution is a sound and complete algorithm that can be used to determine satisfiability and unsatisfiability with certainty. Randomized Algorithm can determine satisfiablity as well as the solution if it finds a solution, but it cannot guarantee to find a solution even if there exists one. However, Randomized Algorithm is much faster than iterative algorithm and also finds the solution, which makes it more practical. In addition, analogy between these two algorithms and two other algorithms namely PL-Resolution and Walk-Sat algorithms for 3-CNF SAT problem, which is also NPC problem is discussed and the performance of Randomized Algorithm and WalkSAT Algorithm is acceptable if the problem is not so complex.**

***Index Terms*—Subset Sum, NPC Problem, Dynamic Solution, Randomized Algorithm, PL-Resolution, Walk-SAT Algorithm**

I. Introduction

Non-polynomial problems comprise of the set of decision problems where answer to any instance of the problem is true then it can be easily proved why the solution is true[1]. Non-Deterministic Polynomial problems are the collection of problems where if the solution is true, then it provides the complexity of the problem in polynomial time. The Subset Sum problem is an NPC problem where we need to find given number summing up single or many indices of given array. The CNF-SAT is also one of the NPC problems where we check for Boolean Satisfiability . Various other solutions to such problems have been researched and improved algorithms are proposed. The work by [2][3] shows some of their proposed results.

A. NP-Complete Problem

Non deterministic Polynomial time (NP) complete complexity class is a class of decision problems in computational complexity theory. As defined by Wikipedia, ”A decision problem L is NP-complete if it is in the set of NP problems and also in the set of NP-hard problems”[4].
The first recognized problem in the field of NPC is Satisfiability(SAT) problem. This implies that no known algorithm can efficiently solve all instances of the problem and it is believed, yet not proven, that no such algorithm can exist[5]. Moreover,a cluster of problems that occured naturally are related to decision making and optimization can be converted into NP-complete to solve them more effectively[6]. This class of NPC problem is used in many practical application for computer science technologies. SAT problem is first NPC problem and it is used to prove NP completeness of other problems[7]. If we can convert an instance set of any problem to its equivalent instance set of SAT problem using polynomial order complexity, then we can prove the problem as NPC. This is how we prove 3-CNF SAT problem and Subset Sum problem as NPC problems.

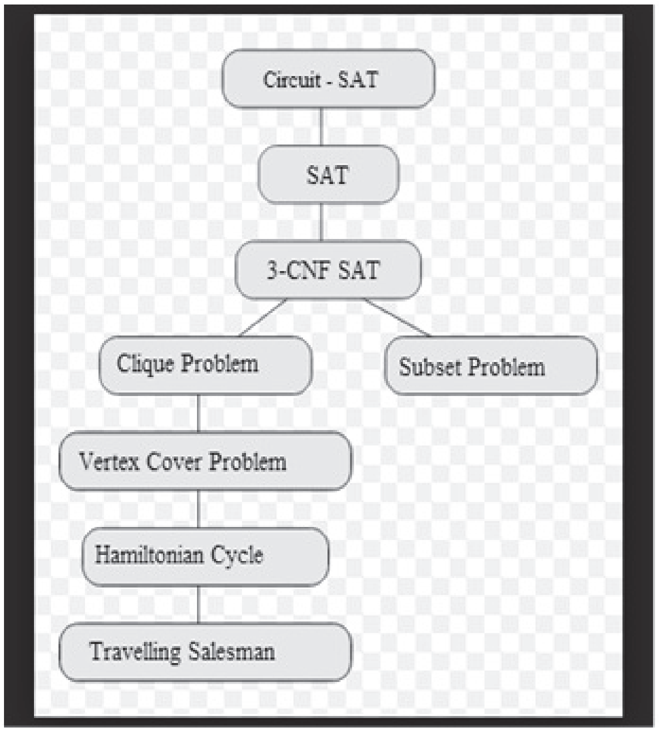

Fig. 1. NPC hierarchy[6]

B. Finding Numbers

Finding number in a subset sum problem is a problem determining whether there exist sum in array. The sum from one or many indexes should satisfy given number. If we get the required number by summing up numbers of the array then the returned value is true and the number from which sum is formed. On the other hand, if the sum can’t be made out of the numbers of the input array then returned value is false. Subset Sum problem can be applied to many fields of computer science and technology and it is also useful in studying mathematical problems. Cryptosystems[8] can also be generalized using Subset Sum problems.

C. Boolean Satisfiability

In the field of Computer Science, Satisfiability or Boolean Satisfiability(SAT), is a problem to determine any combination of boolean values which can result in the successful match according to our boolean equation. In simple terms, writing an equation using AND($\bigwedge$), OR($\bigvee$), NOT ($\neg$), variables, and parentheses, and finding any combination of variables value (TRUE or FALSE) which evaluates the complete expression to TRUE. If the result of all the possible combinations evaluates to false, then that type of expression would be considered unsatisfiable[9].

II. 3-CNF SAT Problem

It can be seen from figure 1 that 3-CNF SAT falls beneath the SAT problem. This problem is proved to be NPC problem by polynomial turing reductions or many one reductions, using SAT problem.

A. Problem

Conjunctive Normal Form (CNF)[9] is a formula made by joining number of clauses, where a clause is referred as the disjunction of literals. Any equation satisfying this condition is said to be in CNF. This is an important form to represent boolean expessions.
Similar to the satisfiability problem, determining the satisfiability of equation in CNF form where atmost three literals can be used is known as 3CNF-SAT problem. Other names for this problems are 3-SAT or 3-satisfiability problem. To reduce the unrestricted SAT problem to 3-SAT, transform each clause

$$"l_1, ..., l_n" \tag{1}$$

to a conjunction of n-2 clauses

$$"(l_1 \bigvee l_2 \bigvee x_2) \bigwedge (x_2 \bigvee l_3 \bigvee x_3) .. \bigwedge (x_{n-2} \bigvee l_{n-1} \bigvee l_n)" \tag{2}$$

where x_2,...,x_{n-2} are fresh variables not occurring anywhere. Although both formulas are not logically equivalent, they are equi-satisfiable. The formula resulting from transforming all clauses is at most 3 times as long as its original, i.e. the length growth is polynomial.

B. Analysis of Solution

There are various solutions proposed for both 3-CNF SAT and Subset Sum problem like dynamic algorithm, basic subset solution, PL-resolution, Walk SAT problem, HyperSAT, miniSAT solver, etc. The most explored solution is using molecules in DNA computers[10]. This paper has focused on PL-resolution and Walk Sat problem for 3-CNF SAT and dynamic solution for Subset sum. In addition, a new solution is proposed for Subset Sum problem which will be computing using randomized way.

1) PL-Resolution Method: The PL-Resolution[9] Algorithm provides a complete solution to find whether a solution to the problem exists or not. The main idea of this algorithm is to achieve new expression by resolving each pair of clauses , For example, clause A $\bigvee$ B and clause $\neg$A $\bigvee$ C can be resolved to a new clause B $\bigvee$ C. Then, add the new clauses . Only two results would occur in the end: (1) the resolve step gives us an empty clause, indicating there is a contradiction in knowledge base, which means expression is unsatisfiable. For instance, clause A and $\neg$A would be resolved to empty clause, besides, we know A $\bigwedge$ $\neg$A is always FALSE intuitively; (2) otherwise the new clauses we gain are already in the knowledge base, which mean there is no contradiction, and expression is satisfiable. The pseudo code of this algorithm [9] is given below:

```
  function PL-RESOLUTION
outputs:true or false
inputs:a sentence in propositional logic
  expr ← the set of clauses in the CNF
  start ← {}
  loop
  for two clauses Cm, Cn in expr do
  resolve ← PL-RESOLVE(Cm, Cn)
  if resolve contains the empty expr then return false
  start ← start ⋃ resolve
  if start ∈ expr then return true
  expr ← expr ⋃ new
```

2) Walk-SAT Algorithm: Unlike PL-Resolution, the WalkSAT Algorithm[9] can determine satisfiability (if it finds a model), but it cannot absolutely determine unsatisfiability. WalkSAT is one of the easiest and most efficient algorithms. At first, WalkSAT randomly generates a model and assigns all the variables with that model. If the sentence is unsatisfiable under this model, it will modify the model. It chooses an unsatisfied clause randomly and selects a variable in that clause to flip in every iteration. There are two ways to decide which variable should be flipped:
(1)Minimize the number of conflicts that is number of unsatisfied clauses occurring in the new state should be minimized.
(2)This step comprises of picking up any variable randomly.

Methodology followed for Walk-SAT Algorithm :

```
Name WALKSAT
inputs chances and clauses
outputs satisfying set or failure
1. Randomly give true or false to clauses.
2. If answer results to true then return satisfied set.
3. Select a randomly false clause within the clauses and try
to make it true
4. If step (3) didn't provide good result then randomly select
a variable and flip the variable.
5. Do step (2),(3) and (4) no. of times.
```

3) Graphical Analysis of Solutions: PL-resolution tells whether the result is TRUE or FALSE but does not give values on TRUE. But it never gives wrong results and we can trust its solution. Pl-Resolution complexity is very high. On the other hand Walk Sat algorithm with low complexity gives the desired value as well for the problem set. But its problem is that it fails to specify whether any solution exists or not in case of unsatisfiable instance. Figure 2 demonstrates how PL-Resolution and Walk-SAT algorithm acts when complexity of the problem is increased and Probability of these to determine satisfiability in that situation[9].

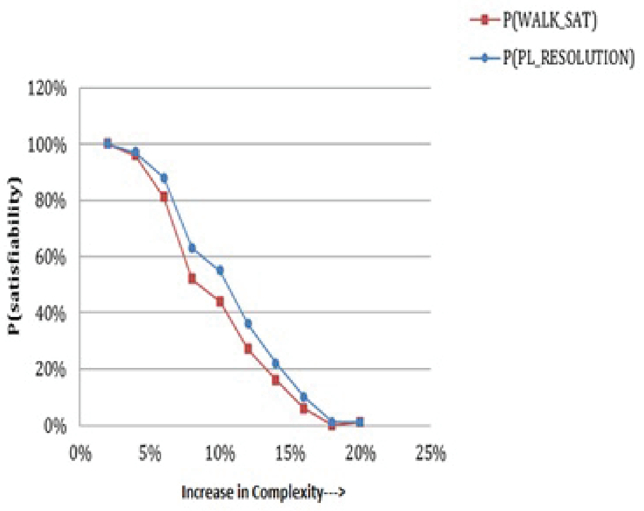

Fig. 2. Analysis of PL-Resolution and Walk-SAT Algorithm[9]

III. Subset Sum Problem

Subset Sum problem is a part of 3-CNF SAT problem. Considering the algorithms of 3-CNF SAT problems as discussed earlier, a new algorithm is devised to solve Subset Sum problem on the same methodology path as of Walk-SAT problem.

A. Problem Statement

Subset Sum Problem states that "Let there be a set A of n positive integers and a sum s. Given the set and the sum, finding the subset A' of given set A (if it exists) such that the addition of the elements of set A' is equal to the given sum s". This problem is a nice problem which introduces us to the NP-complete problems. Reason for this is that it is a decision making problem rather than optimization problem having a simple and clear definition . Approximation algorithms can be used for judgement when approximation solutions are sufficient[11].

B. Dynamic Programming Solution for Subset Sum

We can solve the Subset Sum problem in Pseudopolynomial time using method of Dynamic programming[11]. In this method, We create a boolean 2D table say sub[][] and complete it in bottom up manner. The value of sub[i][j] will be true if there is a subset existing of set[0..j-1] with sum equal to i, otherwise false. Finally, we get the result from the value of sub[sum][n]. It has complexity of O(sum*n). It can only be used to find the satisfiability condition and not the answer(that is resulting subset). Another such example of problem solved with dynamic programming method is Traveling Sales Person Problem[12].

C. Proposed Randomized Algorithm

Algorithm works as follows:

1.Input sum and array of N elements

2.Apply pigeon-hole sorting[13]

3.Check the sum if present in the array

4.Randomly pick any element and apply binary search on rest of the elements for remaining sum. Do thus for m times. (if sum composed of 2 elements)

5.If result not found then select one element randomly, select another element randomly and then go for binary search for the remaining sum. Repeat n times by keeping the first element same, and if not success then repeat the whole procedure m times. (if sum composed of 3 elements).

Constraints: Sum and Elements of the array should not be zero or less than zero.

Assumptions: m,n are constants

Complexity:O(mnN)

1) Discussion and Graphical representation: Readings in the table are approximated because we are using randomized algorithm and picking random number. Based on the readings given in table I we get the graph as shown in Figure 3 for varying size of input.

TABLE I
Results of Proposed Solution

N	m	n	Efficiency(%)	Complexity
5	1	1	34	O(N)
5	3	2	79	O(6N)
5	5	5	99	O(N^3)
10	1	1	20	O(N)
10	3	1	43	O(3N)
10	5	1	48	O(5N)
10	5	5	92	O(20N^2)
10	7	5	90	O(35N)
10	10	5	92	O(50n)
10	10	10	100	O(N^3)
100	1	1	56	O(N)
100	10	1	89	O(10N)
100	10	10	86	O(N^2)
100	50	10	87	O(5N^2)
100	100	100	100	O(N^3)
1000	1	1	62	O(N)
1000	10	10	72	O(100N)
1000	100	100	90	O(10N^2)
1000	1000	1000	100	O(N^3)

From the graph, we can deduce that though Randomized algorithm is not perfectly 100% efficient, it gets almost nearly 100% when complexity gets proportional to square of the no. of elements thus reducing the complexity. Also it gives the solution of the problem. Thus, it is far better than the dynamic algorithm which only tells about the satisfiability of

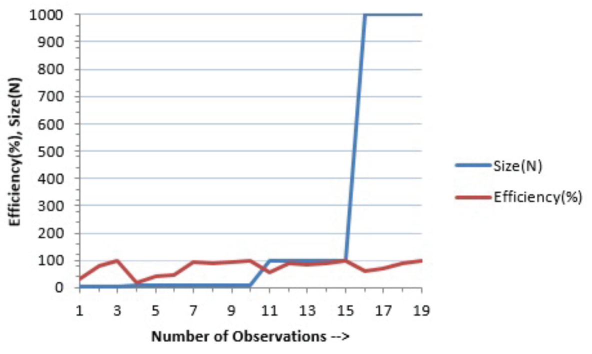

Fig. 3. Result of Proposed Solution

the problem, and not the solution. Also there is an analogy between PL-resolution of CNF-SAT problem and dynamic algorithm of Subset Sum problem and Walk-SAT Algorithm of CNF-SAT problem with Randomized Algorithm of Subset Sum Problem.

IV. Conclusion

Through the analysis we carried out in this research, we were able to solve subset-sum problem within complexity of order (N^2) that is polynomial complexity, much less than the basic subset algorithm of super polynomial complexity. Solution is similar to 3CNF- WALKSAT solution but more efficient than it.

V. Future Scope

The same algorithm with a little modification can be tried to solve other NPC problem, bottom to 3CNF set in the above graph. These algorithm can be extended to find the sum composed of 4 or many indices of the given array. This will improve the efficiency with little rise in complexity.

References

[1] NP(complexity). Wikipedia[Online]:The Free Encyclopedia. ⟨http://en.wikipedia.org/wiki/NP (complexity)⟩.

[2] GJ Woeginger. Exact algorithms for NP-hard problems: A survey. In: Combinatorial Optimization - Eureka! You shrink!. M. Juenger, G. Reinelt and G. Rinaldi (eds.), LNCS 2570, Springer, 2003

[3] Gerhard J. Woeginger. Space and time complexity of exact algorithms: some open problems. In Proc. Int. Workshop on Parameterized and Exact Computation (IWPEC),LNCS 3162, Springer, 2004.

[4] "NP Hard Problem" Wikipedia[Online]: The Free Encyclopedia. ⟨http://en.wikipedia.org/wiki/NP_hard⟩.

[5] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein, "Introduction to Algorithms".

[6] Marek Cygan et al. On Problems as Hard as CNFSAT Cornell University Library (2012).

[7] Mihai Patrascu, Ryan Williamsy, On the possibility of faster SAT algorithms,2010.

[8] R Impagliazzo, Mnaor, "Efficient Cryptographic Schemes provably as secure as Subset Sum, Journal of Cryptology,1996,springer.

[9] Wang, Xili. "A novel approach of solving the CNF-SAT problem."arXiv preprint arXiv:1307.6291(2013).

[10] Johnson, C. R.. The implementation of a DNA computer to solve n-variable 3 CNF SAT problems. University of Southern California). ProQuest Dissertations and Theses, (2004), 129-129.

[11] "Subset Sum Problem." Wikipedia[Online]: The Free Encyclopedia. ⟨http://en.wikipedia.org/wiki/Subset_Sum_Problem⟩.

[12] R. Bellman, Dynamic programming treatment of the travelling salesman problem, JACM, vol. 9, no. 1, pp. 6163, 1962.

[13] "Pigeon-Hole Sorting" Wikibooks: Open Books. ⟨http://en.wikibooks.org/wiki/Algorithm_Implementation/Sorting/Pigeonhole_sort⟩.