International Journal of Computer Applications Technology and Research
Volume 2– Issue 5, 526 - 529, 2013, ISSN: 2319–8656

Algorithm for P versus NP Problem on Sets by JEEVAN – KUSHALAIAH Method

Neelam Jeevan Kumar
Electrical and Electronics Engineering, JNTU Hyderabad
Rangashaipet, Warangal, Andhra Pradesh, India.

Abstract: P versus NP [1-2] Problems are one of the most important open questions in mathematics and theoretical computer science. Jeevan – Kushalaiah Method is a method to find the possible number of combinations between n-elements. This article explains about Algorithm to solve subset sum problem quickly and easily. The problems on subset sum problems perform arithmetic operations that can be calculated in terms of exponential time - polynomial time. Major part of the article deals with Class-P type problems which in be solved on a deterministic Turing Machine. This article is mainly prepared on the basis of an article in Wikipedia.

Keywords: Jeevan-Kushalaiah Method; Jeevan – Kushalaiah Algorithm; Time Complexity; Exponential Time; Polynomial Time;

1. INTRODUCTION

In computational complexity theory an answer to the P = NP [3-4] question would determine whether problems that can be verified in polynomial time, like the subset-sum problem, can also be solved in exponential- polynomial time. If it turned out that P $\neq$ NP, it would mean that there are problems in NP (such as NP-complete problems) that are harder to compute than to verify: they could not be solved in polynomial time, but the answer could be verified in polynomial time. The algorithm named as Jeevan – Kushalaiah Algorithm because it uses Jeevan-Kushalaiah Method. The Time taken to produce the output is in Exponential -polynomial time

$$\mathbf{T(n1) = EXPTIME = 2^{poly(n1)} \ldots\ldots(I.a)}$$

$$\mathbf{T(n1) = P = (2^{O(\log n)} = poly(n)) \ldots\ldots (I.b)}$$

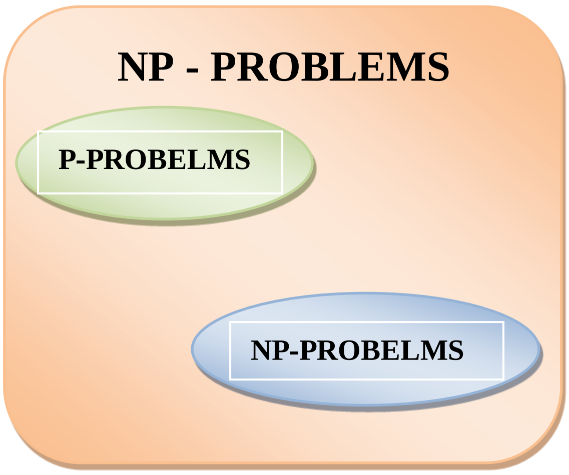


Figure.1: Diagram of complexity classes provided that P ≠ NP. The existence of problems within NP but outside both P and NP-complete, under that assumption, was established by Ladner's theorem

2. JEEVAN – KUSHALAIAH METHOD

Jeevan – Kushalaiah Method [5] is a method to know possible different combinations between n-elements (either elements are constants or variables).

Let there are n-elements $\mathbf{a_1, a_2, a_3 \ldots a_n}$

The possible number of Combinations are in sets with addition operation are, N

N={0,[(a_1),(a_2),(a_3),….(a_2)],[(a_1+a_2),(a_1+a_3),(a_1+a_4),…..(a_1+a_n),(a_2+a_3),(a_2+a_4),(a_2+a_5),… (a_2+a_n),(a_3+a_4), ….. (a_3+a_n),….(a_{n-1}+a_n)],[(a_1+a_2+a_3),(a_1+a_2+a_4),(a_1+a_2+a_5), …..(a_{n-2}+a_{n-1}+a_n)]……. [….]……. [(a_1+a_2+a_3+…+a_n)]}

N= {Θ_0 , Θ_1, Θ_2,Θ_{n-1}, Θ_n }

Where Θ_1 = [(a_1),(a_2),(a_3),….(a_2)] ……. Θ_n = [(a_1+a_2+a_3+…+a_n)]

Θ_n = [$\phi_{n,1}$, $\phi_{n,2}$, …. $\Phi_{n,p}$]

Maximum value of $\phi_{n,p}$ is

$$\mathbf{\Phi_{n,p} = {}^{n}C_p = \binom{n}{p} = (n!)/(n-p)!(p)! \ldots\ldots (II)}$$

$$\mathbf{\Theta_n = \sum_{i=1}^{\binom{n}{p}} \phi_{n,\binom{n}{p}} \ldots\ldots (III)}$$

The possible number of Combinations, N

$$\mathbf{N = \sum_{i=1}^{n} \Theta_n \ldots.. (IV)}$$

3. JEEVAN – KUSHALAIAH ALGORITHM

To find Subset of Set is Predefined value, x by adding elements of Set. To find which subset combination has given predefined value by adding has to Jeevan-Kushalaiah Algorithm.

Jeevan – Kushalaiah Algorithm must satisfy three conditions on Sets before going to algorithm process.

Condition-1: Create Subsets with positive and negative numbers and name as P_n and N_n respectively

Condition-2: The sum all values of either set should not be less than minimum value of other set

$Mod\{\sum P_n\}$ not $<$ $Mod\{min[N_n]\}$ or
$Mod\{\sum N_n\}$ not $<$ $Mod\{min[P_n]\}$

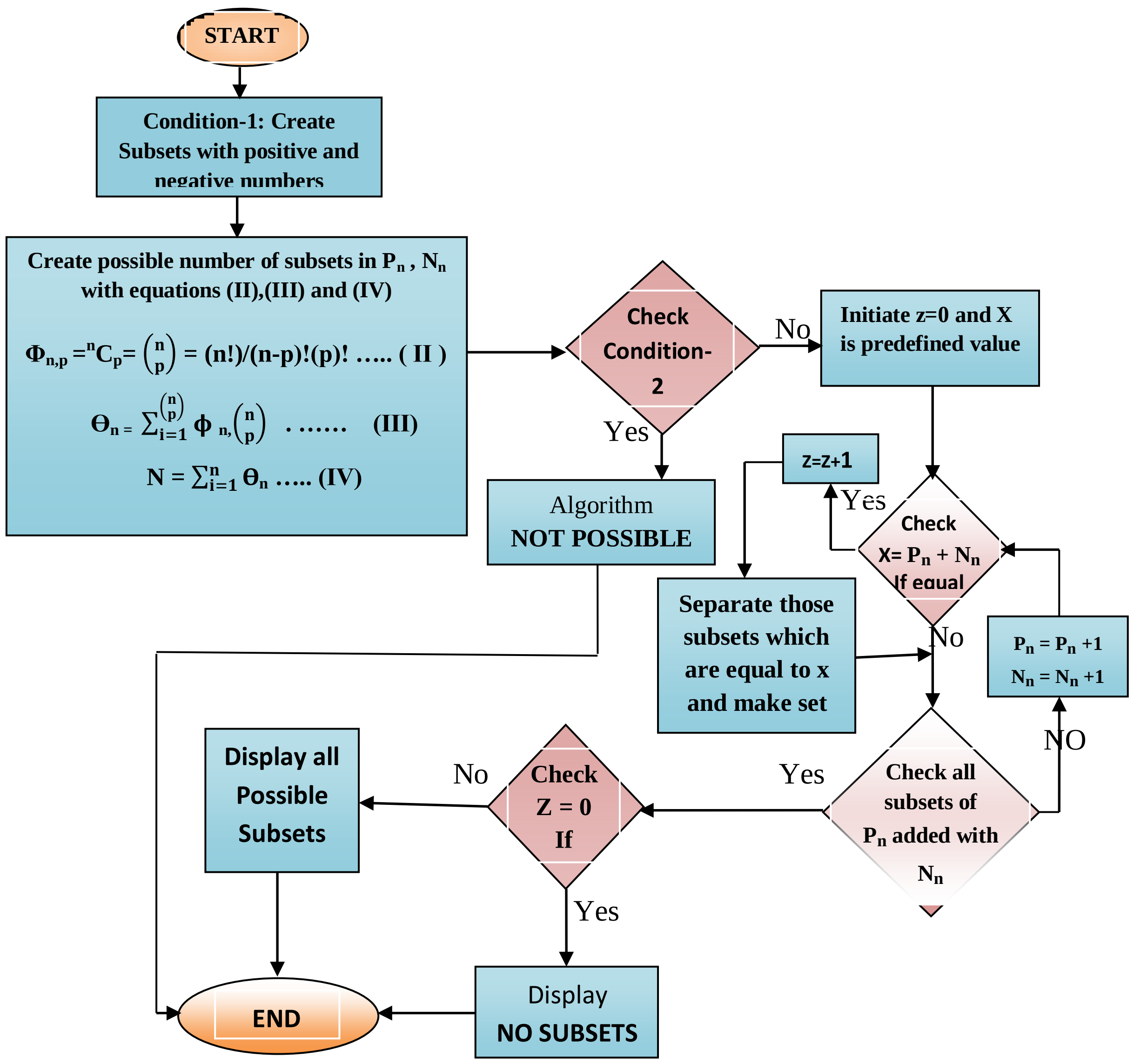
Condition-3: Eliminate or remove or delete Θ_0 value from P_n and N_n

Condition-4: The set should not contain all same sign numbers

The algorithm steps:

Step-1: Start the Program
Step-2: Perform condition-1 on given set.
Step-3: Create possible number of subsets in P_n , N_n with equations (II),(III) and (IV)
Step-4:Check condition-2 if it fails display output as "NOT POSSIBLE' then go to step-10 otherwise go to step-5
Step-5: Set Z = 0, Where Z is the number of Predefined valued subsets.
Step-6: Add each subset of P_n (or N_n) with N_n or (P_n) and equate the result with predefined value.
Step-7: Increase Z= Z+1when result is equal to X and Separate those subsets which are equal to x and make a set.
Step-8: Check all possible of combinations are done
Step-9: then check Z value if it is Zero display 'NO SUBSETS' otherwise "Display all Possible Subsets'. Then Go to Step-10
Step-10: Terminate (or END) the Program

4. FLOW CHART FOR JEEVAN – KUSHALAIAH ALGORITHM


START
Condition-1: Create Subsets with positive and negative numbers
Create possible number of subsets in P_n , N_n with equations (II),(III) and (IV)
$\Phi_{n,p} = {}^nC_p = \binom{n}{p} = (n!)/(n-p)!(p)!$ ….. (II)
$\Theta_n = \sum_{i=1}^{\binom{n}{p}} \Phi_{n,}\binom{n}{p}$ ……. (III)
$N = \sum_{i=1}^{n} \Theta_n$ ….. (IV)
Check Condition-2
No
Yes
Initiate z=0 and X is predefined value
Algorithm NOT POSSIBLE
Z=Z+1
Yes
Check X= P_n + N_n If equal
No
Separate those subsets which are equal to x and make set
P_n = P_n +1
N_n = N_n +1
NO
Display all Possible Subsets
No
Check Z = 0 If
Yes
Check all subsets of P_n added with N_n
Yes
END
Display NO SUBSETS

- Number of iterations of algorithm = [no.of subsets of **P_n**]*[no.of subsets of **N_n**] **… (V)**
- The area of Jeevan – Kushalaiah Algorithm table is NxN (Where N is from equation (IV))

5. COMPUTATIONAL TIME

The total computational time[6-7] **T(n)** = EXPTIME(P) is directly proportionate to N_1*N_2(time taken for each addition)

Where EXPTIME = exponential time[8] for calculating subset elements of P_n and N_n and P = polynomial time[9] for calculating arithmetic operation (i.e., addition) between two elements between P_n and N_n

N_1 = Number of subsets of P_n and N_2 is number of subsets of N_n

EXPTIME = $2^{poly(n1)}$ ………….. (from equation I.a)

P= $2^{O(\log n2)}$ ……………..(from equation I.b)

T(n) = EXPTIME(P) = $(2^{poly(n1)})(2^{O(\log n2)})$ ……..(VI.a)

The arithmetic time is directly proportional to $N_1 * N_2 = n_2$

T(n) α EXPTIME(n_2)…… (VI.b)

6. EXAMPLE

Does a subset of the set {−2, −3, 15, 14, 7, −10} add up to x=0 ?

Checking Jeevan-Kushalaiah Algorithm Conditions

Condition-1: Create Subsets with positive and negative numbers and name as P_n and N_n respectively

P_n = {15,14,7}and N_n = {-2,-3,-10}

Condition-2: The sum all values of either set should not be less than minimum value of other set

Mod{$\sum P_n$} not < Mod{min[N_n]} or Mod {$\sum N_n$} not < Mod{min[P_n]}

Mod(14+14+7) > Mod(-2) and Mod(-2-3-10) <Mod(7), Condition – 2 is satisfied

Condition-3: Eliminate or remove or delete Θ_0 value from P_n and N_n

$P_n = \{ \Theta_1, \Theta_2, \Theta_3\}$ and $N_n = \{ \Theta_1, \Theta_2, \Theta_3\}$: number of subsets, N = 7 from equations (I),(II) and (III)

P_n = { [15],[14],[7],[(15+14=29),(15+7=22),(14+7=21)],[15+14+7 = 36]} = {[15],[14],[7],[(29),(22),(21)][36]}

N_n = { [-2],[-3],[-10],[(-2-3=-5),(-2-10 = -12),(-3-10 = -13)],[-2-3-10 = -15]} ={[-2],[-3],[-10],[(-5),(-12),(13)],[-15]}

P_n:Θ_1 = [(15),(14),(7)] , Θ_2 = [(29),(22),(21)] , Θ_3 = [36]; **N_n** :Θ_1 = [(-2),(-3),(-10)] , Θ_2 = [(-5),(-12),(-13)] , Θ_3 = [-15]

By the algorithm flow chart equation – (VI):

Number of iteration = [no.of subsets of **P_n**]*[no.of subsets of **N_n**] = [7]*[7] =49

7. CONCLUSION

The manuscript explained and proposed about Subsets of sets which are equal to predefined value, X by adding. The elements of the Set are either Constants like Trigonometry, Logarithmic, Exponential Functions or Integers like K1, K2…. Any type of arithmetic operations like addition, subtraction, multiplication and division is performed on the sets. Many computer scientists believe that P≠NP and Author too believe. The main reasons for failing of Jeevan – Kushalaiah algorithm is conditions on sets and Computational Program for Set. Even though all of them are satisfied there may be a chance of getting solution with Z=0(zero) its mean the solution is cannot be determined by Computer program but can be calculated the paper work and major drawback is there is no Algorithm program in the world to perform operation on sets. The Example shows only for 6-element set as the elements of subset increases the complexity of the problem increases Exp (poly)..ly of computational time, T(n).

Table.1 : Jeevan – Kushalaiah Algorithm table to get Predefined value, X = Pn + Nn

X= $P_n + N_n$ = 0	**X= $P_n + N_n$ = 0**	**X= $P_n + N_n$ = 0**	**X= $P_n + N_n$ = 0**	**X= $P_n + N_n$ = 0**	**X= $P_n + N_n$ = 0**	**X= $P_n + N_n$ = 0**
15 – 2 =13	14 – 2 =12	7 – 2 =5	29 – 2 =27	22 – 2 =20	21 – 2 =19	36 – 2 =34
15 - 3 =12	14 - 3 =11	7 - 3 =4	29 - 3 =26	22 - 3 =19	21 - 3 =18	36 - 3 =33
15 – 10 =5	14 – 10 =4	7 – 10 = -3	29 – 10 =4	22 – 10 =12	21 – 10 =11	36 – 10 =26
15 -5 =10	14 -5 =19	7 -5 = 2	29 -5 =24	22 -5 =17	21 -5 =16	36 -5 =29
15 -12 =3	14 -12 =1	7 -12 = -5	29 -12 =17	22 -12 =10	21 -12 =9	36 -12 =24
15 – 13 =2	14 – 13 =1	7 – 13 = -6	29 – 13 =16	22 – 13 =9	21 – 13 =8	36 – 13 =23
15 – 15 =0*	14 – 15 =-1	7 – 15 = -8	29 – 15 =14	22 – 15 =7	21 – 15 =6	36 – 15 =21

Subsets which are equal to **X= $P_n + N_n$ = predefined value (i.e., 0)**

ANSWER is {15,-2,-3,-10}

8. REFERENCES

[1]. Cook, Stephen, "The **P** versus **NP** Problem". Clay Mathematics Institute. April 2000, Retrieved 18 October 2006

[2]. Cook, Stephen "The complexity of theorem proving procedures". Proceedings of the Third Annual ACM Symposium on Theory of Computing. 1971, pp. 151–158.

[3]. Ben-David, Shai; Halevi, Shai (1992). On the independence of P versus NP. Technical Report 714. Technion

[4]. Elvira Mayordomo. "P versus NP" Monografías de la Real Academia de Ciencias de Zaragoza 26: 57–68 (2004).

[5]. Neelam Jeevan Kumar, Neelam Kushalaiah, "JEEVAN-KUSHALAIAH METHOD TO FIND THE COEFFICIENTS OF CHARACTERISTIC EQUATION OF A MATRIX AND INTRODUCTION OF SUMMETOR", IJSER, pp 1553-1562, 4(8), ISSN 2229-5518, Aug-2013

[6]. Lance Fortnow, Steve Homer, "The Computational Complexity Column", NEC Laboratories America, Princeton, NJ 08540, USA.

[7]. Christos Papadimitriou , "Computational Complexity". Addison-Wesley. ISBN 0-201-53082-1. 1991, page 491.

[8]. J. M. Robson, "N by N checkers is Exptime complete". SIAM Journal on Computing, 1984, 13 (2): 252–267

[9]. Terr, David. "Polynomial Time." From MathWorld--A Wolfram Web Resource, created by Eric W. Weisstein.

[10]. Impagliazzo, R.; Paturi, R.; Zane, F. "Which problems have strongly exponential complexity?", Journal of Computer and System Sciences, 2001, 63 (4): 512–530

Wikipedia Links:

[1]. http://en.wikipedia.org/wiki/Millennium_Prize_Problems

[2]. http://en.wikipedia.org/wiki/P_versus_NP_problem

[3]. http://en.wikipedia.org/wiki/EXPTIME

[4]. http://en.wikipedia.org/wiki/Polynomial_time#Polynomial_time

Books:

[1]. Christos Papadimitriou , "Computational Complexity". Addison-Wesley. ISBN 0-201-53082-1. 1991, page 491

[2]. Schrijver, Alexander, "Preliminaries on algorithms and Complexity". Combinatorial Optimization: Polyhedra and Efficiency 1, 2003, Springer. ISBN 3-540-44389-4.

[3]. Carlson, James; Jaffe, Arthur; Wiles, Andrew, eds. (2006). The Millennium Prize Problems. Providence, RI: American Mathematical Society and Clay Mathematics Institute. ISBN978-0-8218-3679-8