IJSRM
INTERNATIONAL JOURNAL OF SCIENCE AND RESEARCH METHODOLOGY
An Official Publication of Human Journals

Human Journals
Research Article
October 2016 Vol.:4, Issue:4
© All rights are reserved by Mahammad Maharram Aliyev et al.

Solving "P versus NP Problem" on Example of Subset Sum Problem

IJSRM
INTERNATIONAL JOURNAL OF SCIENCE AND RESEARCH METHODOLOGY
An Official Publication of Human Journals

Mahammad Maharram Aliyev

National Academy of Sciences of Azerbaijan, Institute of Control Systems, Baku, Azerbaijan.

Submission: 3 October 2016
Accepted: 10 October 2016
Published: 25 October 2016

HUMAN JOURNALS
www.ijsrm.humanjournals.com

Keywords: Knapsack Problem (KP), Subset Sum Problem (SSP), P-Class, NP-Class, Integer Programming, HyperPlane, HyperCircle, Hyper Arch, NPC class, P versus NP problem

ABSTRACT

In this paper exact, polynomial, determination algorithm of the Subset Sum Problem is given. Estimate order of number of simple mathematical operations in the algorithm. The order of number simple operations (addition, subtraction, multiplication, division, comparison, square root) for solving the (SSP) is O (n^4). The (SSP) is including the NPC class and we obtained all problems including this class may solve by a polynomial algorithm. Solution "P versus NP problem" on example the Subset Sum Problem is given.

INTRODUCTION

In modern Wikipedia about the **P versus NP problem** is given follow information.

"The **P versus NP problem** is a. informally speaking, it asks whether every problem whose solution can be quickly verified by a computer can also be quickly solved by a computer. It was essentially first mentioned in a 1956 letter written by Kurt Gödel to John von Neumann. Gödel asked whether a certain NP-complete problem could be solved in quadratic or linear time.[2] The precise statement of the P versus NP problem was introduced in 1971 by Stephen Cook in his seminal paper "The complexity of theorem prove major unsolved problem in computer science procedures"[3] and is considered by many to be the most important open problem in the field.[4] It is one of the seven Millennium Prize Problems selected by the Clay Mathematics Institute to carry a US$1,000,000 prize for the first correct solution.

The informal term *quickly*, used above, means the existence of an algorithm for the task that runs in polynomial time. The general class of questions for which some algorithm can provide an answer in polynomial time is called "class **P**" or just "**P**". For some questions, there is no known way to find an answer quickly, but if one is provided with information showing what the answer is, it is possible to verify the answer quickly. The class of questions for which an answer can be verified in polynomial time is called **NP**, which stands for "nondeterministic polynomial time."

The question is whether or not, for all problems for which an algorithm can *verify* a given solution quickly (that is, in polynomial time), an algorithm can also *find* that solution quickly. Since the former describes the class of problems termed NP, while the latter describes P, the question is equivalent to asking whether all problems in NP are also in P. This is generally considered one of the most important open questions in mathematics and theoretical computer science as it has far-reaching consequences to other problems in mathematics, and to biology, philosophy[4] and cryptography (see P versus NP problem proof consequences).

> " If P = NP, then the world would be a profoundly different place than we usually assume it to be. There would be no special value in 'creative leaps', no fundamental gap between solving a problem and recognizing the solution once it's found. Everyone who could "

appreciate a symphony would be Mozart; everyone who could follow a step-by-step argument would be Gauss...

Most mathematicians and computer scientists expect that P ≠ NP.[6]

The official statement of the problem was given by Stephen Cook."

Another hand the problems in computing mathematics are divided into two classes according to their difficulty: 1) The first class contains the problems in which the number of simple operations for its solution (addition, subtraction, division, comparison, square root) is polynomial expressed by the parameters (by the number of n-variables, the number of m-restriction) of the problem.

This class in said to be a polynomial class and in short, is called a P class. 2) The problems in which the number of simple operations for its solution is proportional to 2^n enter to NP class. This class is called NP hard class. The knapsack problem is one of the classic problems in the linear integer programming problems. Up to present time, it was not possible to solve this problem exactly by a polynomial algorithm.

The knapsack problem can be formulated as a solution of the following linear integer programming formulation:

$$(KP)\ \text{maximize} \sum_{j=1}^{n} p_j x_j \qquad (1.1)$$

$$\text{subject to} \sum_{j=1}^{n} w_j x_j \le c \qquad (1.2)$$

$$x_j \in \{0;1\} \quad j = 1,2,..., n \qquad (1.3)$$

We will denote the optimal solution vector by $X^* = (x_1^*, x_2^*,..., x_n^*)$ and the optimal solution value by Z^*. The set X^* denotes the optimal solution set, corresponding to the optimal solution vector.

Problem (KP) is the simplest non-linear integer programming model with binary variables, only one single constraint and only positive coefficients. Nevertheless, adding the integrality

condition (1.3) to the simple linear program (1.1)-(1.2) already puts (KP) into the class of "difficult" problems [1].

The knapsack problem has been studied for centuries as it is the simplest prototype of a maximization problem. Already in 1897, Mathews showed how several constraints may be aggregated into one single knapsack constrain. This is somehow a prototype of a reduction of a general integer program to (KP), thus proving that (KP) is at least as hard to solve as an integer program. It is, however, unclear how the name "Knapsack Problem" was invented. Danzig is using the expression in his early work and thus the name could be a kind of folklore [1].

When $p_j = w_j$ in (KP), the resulting optimization problem is known as the subset sum problem (SSP) because we are looking for a subset of the values w_i with the sum being as close as possible to, but not exceeding the given target value c.

$$(SSP) \text{maximize} \sum_{j=1}^{n} w_j x_j ;$$

$$\text{Subject to} \sum_{j=1}^{n} w_j x_j \le c$$

$$x_j \in \{0,1\}, \quad j = 1,...,n$$

Although (SSP) is a special case of (KP) it is still NP-hard and is NP- complete [1].

The class of NP-complete problems NPC is the set of decision problems Q satisfying the following two properties:

1. $Q \in NP$

2. $\forall R \in NP : R \le_p Q$

To see the latter, one may simply assume that an NP-complete problem Q could be solved in polynomial time, then by transforming R to Q we would also get a polynomial algorithm for R contradicting the assumption.

The theory of *NP*- completeness gives us a framework for showing that it is very doubtful that a polynomial algorithm for solving e.g. the subset sum problem, then we would also be able to solve numerous famous optimization problems like the travelling salesman problem, general integer programming, and we would even be able to efficiently find mathematical proofs of theorems, as stated in Cook [1].

In this paper, a new solution algorithm of knapsack problem is given by an original approach; some problems are solved exactly by a polynomial algorithm. One is subset sum problem. At first, the engineers and economists have been engaged integer problems. These problems appeared as a result of economic and engineering demands. Therefore, these problems have got engineering and economic interpretation. But these interpretations are not enough for the solution of the problem.

In order to solve any mathematical problem, it is necessary to give its geometrical interpretation. It is the work of professional mathematicians. To this end, this matter requires special attentive in this paper.

2. Problem Statement.

In this paper, using the results of [2, 3, 4, 5] we will solve the "P versus NP problem" on the example of the (SSP). This work will be done by the following scheme.

- Using theoretical knowledge given in [2, 3, 4, 5], we will construct exact, polynomial, determinate algorithm for solving (SSP).

- The scheme of the algorithm will be given.

- The order of the number simple operations in the algorithm will be estimated.

-We will show the algorithm is exact, polynomial and determinate.

The (SSP) is included in the NP-Complete class, its solution by a polynomial algorithm stipulates is solve a number of problems concerned in this class by a polynomial algorithm [1]. According to contemporary theory in this field, this proves that the "P versus NP problem" has a solution i.e. NP=P.

The structure of the algorithm is as follows:

1) Verified that the point E(1,1,…,1) is solution of the problem or not. If the point is the solution of the problem, then process ends. If the point, not the solution point, then the solution of the problem is research on every hyper-circle K_{nk}. The number of hyper circles is n-1.

2) The P_{max} point giving the maximum value to the functional F= (P, X) on K_{nk} hyper-circle is found. It is verified this point satisfies the constraint, then it is the solution on the considered K_{nk} hyper-circle. The subsequent values of *K* are considered. If this point doesn't satisfy the constraint, then the integer point W_{min} giving the minimum value to the constraint functional F(X) = (W, X) on K_{nk} hyper-circle is found. If the W_{min} point doesn't satisfy the constraint, then the problem has no solution on the considered K_{nk}. If the point W_{min} satisfies the constraint, its approximation to solution is verified. If this point satisfies the condition

$$c- (W, W_{min}) \leq N \qquad (2.1)$$

Initial point for the exact algorithm is accepted and the exact solution on K_{nk} is found. If condition (2.1) is not satisfied, the point satisfying this condition is found by the rhomb method [5]. Using the initial point satisfying condition (2.1), an initial point on another semicircle is found as well. The exact algorithm is applied on the second semicircle as well and the optimal solution is found. After performing this process on n-1 number hyper circles, the general optimal solution is chosen among the found local optimal solutions.

3. Exact, polynomial, determination algorithm of the Subset Sum Problem

1) Initial date: W, P, c, N

2) Verify the point E(1,1,…,1) is solution or not

2.1. Calculate sum $S_1=\sum_{j=1}^{N} w_j$

2.2. Verify condition $S_1>c$

2.3. If $S_1>c$ then go to 3

2.4. ELSE calculate sum $S=\sum_{j=1}^{N} p_j$

2.5. Print AMAX=S; HK (j) =1; j=1, 2,…, N

3. Order the elements of the vector W by increasing series

$$W \rightarrow WN$$

4. Order the elements of the vector P by increasing series

$$P \rightarrow PN$$

4.1.Finding projection of vector W on the M_{nk} hyperplane:

$$En\ (\frac{1}{\sqrt{N}},\frac{1}{\sqrt{N}},\ldots,\frac{1}{\sqrt{N}});\ W \rightarrow Wn;\ WP=W_n-(W_n, En)\ En$$

5. Beginning circle K=1,2,…,N-1; $r_{nk}=\sqrt{(NK-K^2)/N}$

5.1. Finding integer point XB giving maximum value to functional F= (P, X) on the K_{nk}

5.1.1. XB (j) =0; j=1, 2,…, N

5.1.2. K1=0

5.1.3. If (P (j)-P (N+1-K)>0 then XB (j) =1; K1=K1+1; j=j+1; j=1, 2,…, N

5.1.4. If (P (j)-P (N+1-K) =0 then XB (j) =1; K1=K1+1; j=j+1; j=1, 2,…, N

5.1.5. If K1=K GO TO 5.2

5.2. Verify that XB is solution point or not

5.2.1. Calculate inner product S= (W, XB)

5.2.2. If c-S≥ 0 then H (j) = HB (j); j= 1, 2,…, N; AMAX= (P, HB); K=K+1

5.3. ELSE finding integer point W_{min} giving minimum value to functional F= (W, X) on the K_{nk}. XT= W_{min}.

5.3.1. XT (j) = 0; j=1, 2,…, N

5.3.2. K1=0; If W (j) - WN (K) ≥ 0; then j= j+1; j=1, 2,…, N

5.3.3. If W (j) - WN (K) < 0 then XT (j) = 1; K1= K1+1; j=j+1

5.3.4. If W (j) - WN (K)≠ 0 then j=j+1; j=1, 2,…, N

5.3.5. If W(j)- WN(K)= 0 then XT(j)=1; K1=K1+1; j=1,2,…,N

5.3.6. If K1=K then S= (W, XT)

5.3.7. If c-S<0 then AMAX=AMAX; K=K+1

5.3.8. If c-S≥ 0 then W_{min}=XT

5.4. Finding the initial integer point near the solution on a hyper circle by the rhombus method.

5.4.1. N1=init (N/2); KO=init (K/2); q=N1-KO

5.4.2. J=q+1, q+2,…, q+K

5.4.3. I=1,2,…,N; XO(I)=0; If(W(I)-WN(j))=0 then XO(I)=1; I=I+1

5.4.4. If (W (I)-WN (j)) ≠ 0 then XO (I) =0; I=I+1; j=j+1

5.4.5. S= (W, XO)

5.4.6. If ((S≤ c) and (c-S≤ N)) then XT=XO is initial integer point of the exact algorithm. Go to 5.5.

5.4.7. W_{max}=XB

5.4.8. If S<c then W_{min}=XO;

5.4.9. V1= O_{NK}- W_{min}; V2=O_{NK}-W_{max}; V=V1+V2

5.4.10. If S>c then W_{max}= XO; V1= O_{NK}- W_{min}; V2=O_{NK}-W_{max}; V=V1+V2

5.4.11. V→ V_n; XO=O_{nk}-$r_{nk}V_n$; Go to 5.4.5.

5.5. Finding initial integer point on the other semicircle

5.5.1. b=(W,XO); WP→ WP_n

5.5.2. $T=O_{NK}+ [(b-(W,O_{NK}))/ (W,WP_n)]WP_n$

5.5.3. X2O=2T-XO; $V=O_{NK}$-X2O; $V \rightarrow V_n$

5.6. Finding initial integer point of the exact algorithm

5.6.1. Order the elements of the vector V by increasing series: V→VN

5.6.2. K1=0; XT (j) =0; j=1, 2,…, N

5.6.3. If (V (j)-VN (K)) ≥0 then XT (j) =0; j=j+1

5.6.4. If (V (j)-VN (K)) <0 then XT (j) =1; K1=K1+1; j=j+1

5.6.5. If (V (j)-VN (K)) ≠0 then j=j+1

5.6.6. If (V (j)-VN (K)) =0 then XT (j) =1; K1=K1+1

5.6.7. If K1=K Go TO 5.7.; j=j+1

5.7. Adopt point XT initial point of the exact algorithm

5.7.1. KK=1, 2,…, N; I=1, 2,…, N; T=XT; TT=T

5.7.2. S= (W, TT)

5.7.3. D=c-S

5.7.4. If [(T (I) =0) and (TT (j) =1) and (P (I)-P (j)>0)] then D1=D+W (j)-W (I)

5.7.5. If D1<0 then j=j+1

5.7.6. If D1≥0; then j1=j; I1=I; TT (I1) =1; TT (j1) =0; I=I+1

5.7.7. j= j+1; I=I+1

5.7.8. T (j) = TT (j); j= 1, 2,…, N

5.7.9. KK=KK+1

5.7.10. H (j) =T (j); j=1, 2,…, N

5.7.11. S= (P, H)

5.7.12. If S≥AMAX then AMAX=S; If S<AMAX then AMAX=AMAX

5.7.13. HK (j) =H (j); j=1, 2,…, N

5.7.14. Finding optimal solution on other semicircle

5.7.15. K=K+1

5.7.16. Print HK; AMAX

End

4. Estimating order of number of simple mathematical operations in the algorithm.

Considering the algorithm, we see that in points 1-4, order of the number simple mathematical operations is 2, i.e. the principal term of the polynomial expressing by *n* the number of simple mathematical operations is n^2.

In points (5.2 - 5.4.) the calculations consist of inner products and comparisons, their order is a unit. If we take onto amount the period *k*, order equals 2.

In points, 5.4.1- 5.4.11 of the algorithm, is inner product, comparison and implicit circle are used. Since rhomb method is a bisection algorithm, it determines the point sought among the 2^n number points located on hyper arches in regular density at *n* iteration. In order to determine the point satisfying condition (2.1), we need iteration less than *n.* The number of iterations in implicit circle is less than *n.* The order of number of simple operations in this part is equals 3.

In part 5.5.1- 5.5.3 the order of number operations is 1.

In part 5.6.1- 5.6.7 the order of the number operations is 2.

In part 5.7.1- 5.7.15 the order of the number operations is 3. In this part, we have the circles contained in each other kk=1,2,…,n; i=1,2,…,n; j=1,2,…,n and k=1,2,…,n-1. The maximum number of circles contained in each other is 4.

The exact solution method is given in [4].

The property determination is given in [2, 3, 4] throw M_{nk}, K_{nk} ,O_{nk}, r_{nk}.

We get the following results.

5. CONCLUSION

1) The order of number simple operations (addition, subtraction, multiplication, division, comparison, square root) for solving the (SSP) is O (n^4).

2) The (SSP) is including the NPC class and we obtained all problems including this class may solve by a polynomial algorithm [1].

3) The P and NP classes are equivalent: P↔NP.

4) So P=NP and the "P versus NP problem" is solved.

REFERENCES

1. Kellerer H, Pferschy U, Pisinger D. Knapsack Problems. Springer-Verlag Berlin. Heidelberg, 2004, 525 p.
2. Aliyev MM., One approach to the solution of the knapsack problem//Reports NAS of Azerb., Num. 3, 2005, p. 32-39.
3. Aliyev MM. On Solution Method of the Knapsack Problem. Proceedings of the Sixth ICMSEM.Volume I, Springer-Verlag, London, 20012, p.257-267.
4. Aliyev MM. Exact, Polynomial, Determination Solution Method of the Subset Sum Problem. Science PG. Applied and Computational Mathematics. Vol. 3, Num. 5, 2014, pp. 262-267.
5. Aliyev MM. Finding the initial integer point near the solution on a hyper circle by the rhombs method. International Journal of Modern Trends in Engineering and Research (IJMTER). Volume 02, Issue 10, [October – 2015] pp. 138-142.