P = NP: Polynomial time solving method for 0-1 Knapsack problem

Zhibang Jia*

1. Abstract

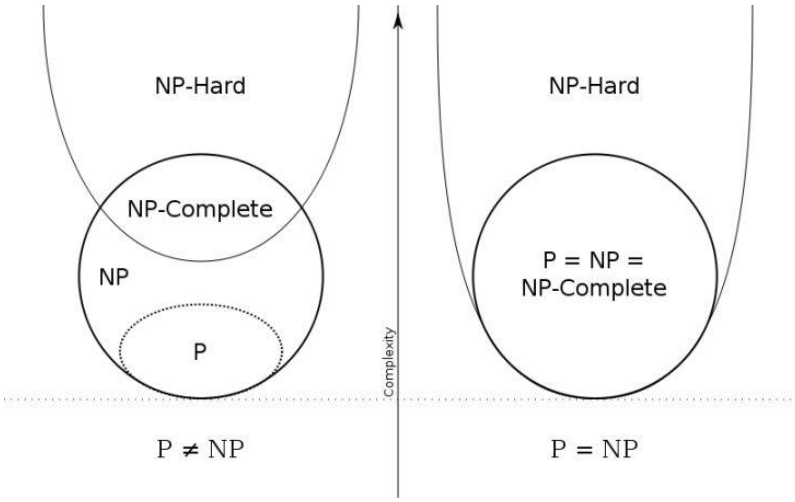
P/NP problem[1] is one of the most important problems in the field of theoretical computer science. Once P=NP is confirmed, this will mean that in polynomial time we can accurately solve complex computer problems whose complexity increases exponentially as the number of inputs increases. In Karp's 21 NP-complete problems[2], there are several forms of NPC problems. These problems exist widely in many fields. Once one form of the NP-complete problem is solved, we can easily reduce it to the other form. New breakthroughs will be made in the fields of finance, biology, chemical engineering, computer science and cryptography, industrial engineering and operations research, and other natural sciences and engineering fields.

In this paper, the NP-complete problem o f 0-1 knapsack, which is also a NP-hard problem, is taken as an example, and a method to solve this problem in polynomial time complexity is given. We will solve this problem based on the convex optimization algorithm and by constructing additional constraints.

2. Introduction

NP problems include P problems and NP-complete problems. A P problem means a deterministic problem that a deterministic Turing machine can solve in polynomial time. A NP-complete problem means that a deterministic Turing machine can verify that the solution is correct in polynomial time, but it cannot obtain an answer in polynomial time.

There is a pair of Euler diagrams that can be used to describe, respectively in P! = NP and P = NP these two different cases, the relationship between P problems, NPC problems, NP problems, and NP-hard problems: (Fig.1 We'll point out the right situation)



3. 0-1 Knapsack problem

As one of the representative problems of 01-integer programming. The 0-1 Knapsack problem is a NP-complete problem but also a NP-hard problem. Its definition is:
There are n items, the weights of each items i are w_i, the values are v_i, and the weights and prices of all items are assumed to be non-negative constants. The maximum weight that a Knapsack can bear is a constant W. Only 0 or 1 can be selected for each item.

The mathematical description is:
Objective function: Max $\sum_{i=1}^{n}(v_i \cdot x_i)$
Constraint 1: $\sum_{i=1}^{n}(w_i \cdot x_i) \leq W$
Constraint 2: $x_i \in \{0, 1\}$

As a 01-integer programming problem. In the worst case, its complexity is derived from the binary integer variable x_i. Enumerating each x_i 's different situations as 0 or 1 once, and it need to take 2^n times to get the optimal result. It's an exponential explosion. On the other hand, even if we use a dynamic programming algorithm to solve it, we can only give a pseudo-polynomial time algorithm. This is not strictly a polynomial time algorithm.

4. Jia's constraints: Transforming NP-complete problems to convex optimization problems

In this paper, an additional constraint - Jia's constraints is given, which can transform the 0-1 knapsack problem into a convex optimization problem, so that it can be solved with known polynomial-time convex optimization solving algorithms.

For each binary variable x_i, add four constraints:

$$x_i \cdot (x_i - 1) \geq 0$$

$$x_i \cdot (x_i - 1) \leq 1$$
$$x_i \geq 0$$
$$x_i \leq 1$$

So that, the result value of x_i must be 0 or 1.

So far, we have given a set of complete objective function and constraints:
Objective function: Max $\sum_{i=1}^{n}(v_i \cdot x_i)$
Constraint 1: $\sum_{i=1}^{n}(w_i \cdot x_i) \leq W$
Constraint 2: $xi \cdot (xi - 1) \geq 0$
Constraint 3: $xi \cdot (xi - 1) \leq 1$
Constraint 4: $xi \geq 0$
Constraint 5: $xi \leq 1$

Generalized to the general case, for any variable x that can take one of two different values belonging to {A, B}, we can transfer problems to convex problems by defining the constraints $A \leq a \cdot x^2 + b \cdot x + c \leq B$ and $A \leq x \leq B$.

According to the definition of convex optimization, convex programming means that if the objective function of the optimization problem is convex, the inequality constraint function is also convex, and the equality constraint function is linear (Also known as affine). For convex functions, functions on the set of real numbers can be distinguished by solving for the second derivative: if the second derivative is non-negative on the interval, it is called a convex function; if the second derivative is always greater than 0 in the interval, it is called strictly convex. Therefore, affine functions are also convex functions, but affine functions are not strictly convex.

According to the above definition, the objective function and constraint conditions we give conform to the definition of a convex optimization problem. So we proved this is a convex optimization problem.

5. Analysis of time complexity

We take the Interior-point algorithm [3] as an example, which is a classical algorithm used to solve convex optimization problems. Its time complexity is $O(n^{3.5} \cdot L)$which n is related to the number of variables and L can be a constant that depends on precision. The time complexity is independent of the number of constraints.

Therefore, for a 0-1 knapsack problem with n items, the time complexity of solving it using the

Interior-point method is $O(n^{3.5} \cdot L)$. This is a method for solving in polynomial time.

6. Conclusion

In this paper, we make the 0-1 knapsack problem solvable in polynomial time by adding additional constraint functions. By using the traditional interior point method, we can solve NP-Complete problems and parts of NP-hard problems in a time complexity of about $O(n^{3.5})$. At this point, the P/NP problem has been solved. The conclusion is that P = NP.

References:

[1] Cook, S. The P versus NP problem. *Clay Mathematics Institute.* **2**, 3 (2000).

[2] Karp, R. M. Reducibility among combinatorial problems. *Springer Berlin Heidelberg.* **8**, 219-241 (2010).

[3] Nemirovski, A. Interior point polynomial time methods in convex programming. *Lect. notes.* **42**, 3215-3224 (2004).