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 of 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 see Appendix)

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 m-items, the Grams of each item i are g_i, the Prices are p_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: $Max \sum_{i=1}^{m}(p_i \cdot x_i)$
Constraint 1: $\sum_{i=1}^{m}(g_i \cdot x_i) \le G$
Constraint 2: $x_i \in \{0, 1\}$

In convex optimization, maximizing a function f(x) is equivalent to minimizing the negative value of the function -f(x). Thus, re-writing the objective function:

$$Min \sum_{i=1}^{m}(b_i \cdot x_i)$$

Which $b_i = -p_i$

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^m 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. Additional constraints: Transforming NP-complete problems to convex optimization problems

In this paper, an additional constraint 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 variable x has two value A and B, we describe it using the function$(x - A) \cdot (x - B) = 0$ in general. It's a quadratic equation in one unknown. However, the convex equality equation can't form a convex optimization problem. We must transfer the equality constraint into inequality constraints to make it a standard convex optimization problem that can be solved. It is equivalent to:
(Assumption the value of A is less than the value of B)

$$x^2 - (A + B) \cdot x + A \cdot B \geq 0$$
$$x \geq A$$
$$x \leq B$$

(Fig.2 see Appendix)
There is another set of equivalent constraints:

$$x^2 - (A + B) \cdot x + A \cdot B \geq 0$$
$$x^2 - (A + B) \cdot x + A \cdot B \leq 0$$

(Fig.3 see Appendix)

The second set of equivalent constraints will be chosen for later argumentation. x_i￼ which have two value 0 and 1, add two constraints:

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

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: $Min \sum_{i=1}^{m}(b_i \cdot x_i)$
Constraint 1: $\sum_{i=1}^{m}(g_i \cdot x_i) \ \leq \ G$
Constraint 2: $xi \cdot (xi - 1) \geq 0$
Constraint 3: $x_i \cdot (x_i - 1) \leq 0$

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. Solving the convex problem by modified KKT conditions and Interior point method

To solve a constrained convex optimization problem, the KKT condition [3] is usually used to obtain its strong dual problem, to solve its problem equivalent.

Lemma 1 *According to the above convex optimization problem, to write the Lagrange equation:*

$$L(x_i, \{T\}, \{U_i\}, \{W_i\}) = f(x_i) + T \cdot t(x_i) + \sum_{i=1}^{m} U_i \cdot u_i(x_i) + \sum_{i=1}^{m} W_i \cdot w_i(x_i)$$

And K-K-T Conditions:

Lemma 2 *The K-K-T Conditions*

1. $\nabla f(x_i) + T \cdot \nabla t(x_i) + \sum_{i=1}^{m} U_i \cdot \nabla u_i(x_i) + \sum_{i=1}^{m} W_i \cdot \nabla w_i(x_i) = 0$

   $and\ f(x_i) = \sum_{i=1}^{m} b_i \cdot x_i;$

2.

   $t(x_i) = \sum_{i=1}^{m} g_i \cdot x_i - G \le 0\ and\ T \ge 0\ and\ T \cdot t(x_i) = 0$

3.

   $u_i(x_i) = -x_i^2 + x_i \le 0\ and\ U_i \ge 0\ and\ U_i \cdot u_i(x_i) = 0$

4.

   $w_i(x) = x_i^2 - x_i \le 0\ and\ W_i \ge 0\ and\ W_i \cdot w_i(x) = 0$

The K-K-T conditions are difficult to solve directly. There is a common method to solve convex optimization problems called the barrier interior-point method [4].

Lemma 3 *Unconstrained optimization object function based on Barrier function interior point method:*

1. $Min\, f(x_i) - \frac{1}{t}\log(-t(x_i)) - \frac{1}{t}\log(-u(x_i)) - \frac{1}{t}\log(-w(x_i))$

Above function is equivalent to:

2. $Min\, t \cdot f(x_i) - \log(-t(x_i)) - \log(-u(x_i)) - \log(-w(x_i))$

Which:

3. $f(x_i) = \sum_{i=1}^{m} b_i \cdot x_i$
4. $t(x_i) = \sum_{i=1}^{m} g_i \cdot x_i - G$
5. $u_i(x_i) = -x_i^2 + x_i$
6. $w_i(x) = x_i^2 - x_i$

In this situation, the value of each variables x will never be the value of 0 or 1, but they're going to be infinitely close to one of the two values. Once the value of x is less than some error ε, it will be confirmed its exact value is 0 or 1.

6. Solution algorithm and analysis of time complexity

Boyd, in 2004, have given a complexity analysis via self-concordance [5]. In the case that the objective function is a self-concordant function:

If the initial value of t that t (0) and the step size μ selected appropriate values, the convex programming problem will be solved in a $O(\sqrt{n})$ time complexity in the worst case, which the n is the total number of inequation constraints.

The functions f(x), t(x), u(x), w(x) can be verified as self-concordant functions, according to Nesterov and Nemirovskii's definition in 1994 [6]:

Definition 1 *self-concordant*
Let E be a finite-dimensional real vector space, Q be an open nonempty convex subset of E, F: Q —> R *be a function, a* > 0. *F is called* self-concordant *on Q with the parameter value a (a-self-concordant; notation: F ∈ Sa(Q, E)), if F* e *C^3 is a convex function on Q,* and, *for all x ∈ Q and all h £ E, the following inequality holds:*

$$|D^3F(x_i)[h,h,h]| \leq 2a^{\left(-\frac{1}{2}\right)}(D^2F(x_i)[h,h])^{\frac{3}{2}}$$

(D^k F(x} [h_1, ..., h_k] henceforth denotes the value of the kth differential of F taken at x along the collection of directions hi, ..., h_k). Note that a self-concordant on Q function F is strongly self-concordant on Q if and only if Q is the "natural domain" of F.

First, the initial value of t that t (0) will be choose by:

Algorithm 1 *choosing the value of t (0)*
1. *Give each x a feasible value, and in this case, x can be 0 or 1.*
Set x (0) = 1
2. *Compute t (0) by minimizing function g (t).*

$$g(t) = \left(t\,\nabla f(x^{(0)}) + \nabla\Phi(x^{(0)})\right)^T H_0^{-1}\left(t\,\nabla f(x^{(0)}) + \nabla\Phi(x^{(0)})\right)$$

$$and\ H_0 = t\,\nabla^2 f(x^{(0)}) + \nabla^2\,\Phi(x^{(0)})$$

$$which\ f(x^{(0)}) = \sum_{i=1}^{m} b_i \cdot x^{(0)}$$

$$which\ \Phi(x^{(0)}) = -\log\left(-t(x^{(0)})\right) - \log\left(-u(x^{(0)})\right) - \log\left(-w(x^{(0)})\right)$$

Second, choosing the value of step size μ:

Algorithm 2 *choosing the value of μ*

1. **Getting** *the total number of inequation constraints n.*

n = 2m + 1

2. **Getting** *the step size μ.*

$$\mu = 1 + \frac{1}{\sqrt{n}}$$

At last, run the optimization solving algorithm based on barrier method:

Algorithm 3 *Barrier method*

given strictly feasible x*, in this case, x can be 0 or 1.*

get the value of t (0) a*ccording to Algorithm 1.*

get the value of μ a*ccording to Algorithm 2.*

given strictly feasible x, t: = t (0) > 0, μ > 1, tolerance ε > 0.

repeat

1. *Centering step.*

Compute x(t) by minimizing t*f (0) + φ, starting at x.

2. *Update.* x: = x in (t).
3. *Stopping criterion.* quit if m/t < ε.
4. *Increase* t. t: = μt.

Computing x in (t) for a sequence of increasing values of t, until t ≥ m/ ε.

There is an upper bound on the total number of Newton steps is given by Boyd in the barrier method:

Conclusion of complexity 1

If giving a μ: $\mu = 1 + \frac{1}{\sqrt{n}}$:

There is an upper bound on the total number of Newton steps：

$$N \leq c_1 + c_2 \cdot \sqrt{n}$$

$$c_1 \leq \frac{1}{2} + c$$

$$c_2 \leq \log_2\left(\frac{n}{t(0) \cdot \epsilon}\right)\left(\frac{1}{2} + c\right)$$

$$c = \log_2 \log_2\left(\frac{1}{\epsilon}\right)$$

* The n is the number of inequation constraints.

* In practical, the ε can be set to 0.01 times the accuracy of variable x. For example, the variable as 0 or 1 have precision of single digits, ε will be 0.01.

The above process is explained there is a polynomial time algorithm about $O(\sqrt{n})$ which n is the number of inequality constraints.
Therefore, for a 0-1 knapsack problem with m items, for each variable x_i, there will be two inequation constraints for it. Adding the one linear inequality constraint there will be $2m + 1$ inequality constraints. The time complexity of solving it using the Interior-point method is $\sqrt{2m+1}$ .This is a method for solving in polynomial time.

7. The extensibility of the above method and Applications

By changing the values of A and B, the above method can be adapted to different models. For example, setting the value of A as -1 and the value of B as 1. We can solve the problem where the value of variables can be ± 1. Further, it may be helpful to understand the Ising model.

Boyd demonstrated the complexity of the algorithm based on the classical Newton method. The classical Newton method will store more hessian matrices as the number of variables increases, which will lead to more memory usage. In practical, a series of quasi-Newtonian methods such as L-BFGS [7] can be considered for the above optimization problems to reduce the memory usage as much as possible.

Like the Barrier method, the Primal-Dual method [8] is also often considered for solving convex optimization problems in industry. Although the complexity of this approach is not discussed in this article, in practice Drimal-dual method often performs as well as barrier methods.

8. Conclusion

In this paper, we make the 0-1 knapsack problem solvable in polynomial time by adding additional constraint functions. By using the Barrier interior point method, we can solve NP-Complete problems and parts of NP-hard problems in a time complexity of about $O(\sqrt{2m+1})$. 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. *Complexity of Computer Computations: Proc. of a Symp. on the Complexity of Computer Computations.* (1972)

[3] Karush, W. Minima of Functions of Several Variables with Inequalities as Side Conditions. *M.Sc. Dissertation. Dept. of Mathematics, Univ. of Chicago, Chicago, Illinois.* (1939)

[4] Fiacco, A. & McCormick, G. Computational Algorithm for the Sequential Unconstrained Minimization Technique for Nonlinear Programming. *Management Science, 10(4), 601–617.* (1964)

[5] Boyd, S. & Vandenberghe, L. Convex Optimization. *Cambridge University Press.* (2004)

[6] Nesterov, Y. & Nemirovskii, A. Interior-Point Polynomial Algorithms in Convex Programming. *Society for Industrial and Applied Mathematics.* (1993)

[7] Saputro, D. & Widyaningsih, P. Limited Memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS) Method for The Parameter Estimation on Geographically Weighted Ordinal Logistic Regression Model (GWOLR). *AIP Conf. Proc. 4 August 2017; 1868 (1): 040009.* (2017)

[8] Wright, S. Primal-dual interior-point methods. *Society for Industrial and Applied Mathematics.* (1997)

Appendix

Fig.1 Venn diagram for P $\neq$ NP (left) and P = NP (right), the right situation will be proofed.

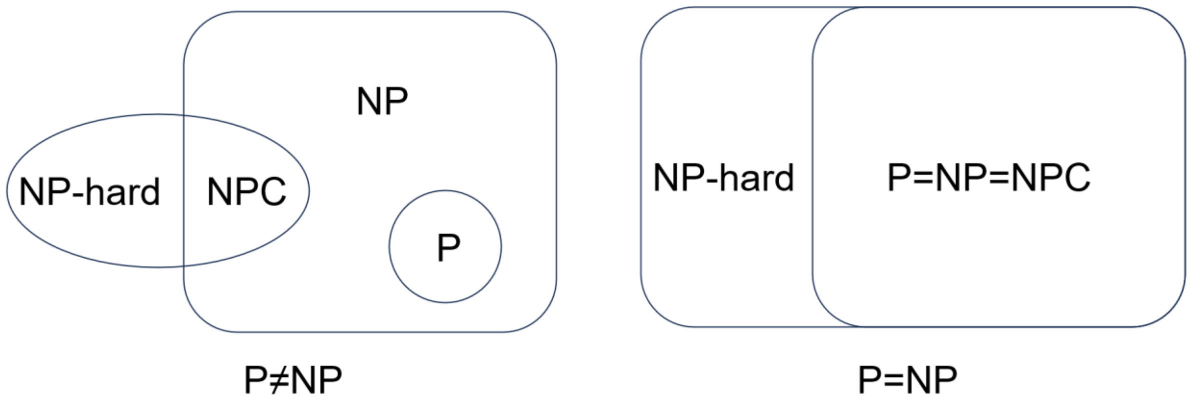


Fig.2 The blue curve is the constraint of $x^2 - (A + B) \cdot x + A \cdot B \geq 0$, and the orange part means that the constraint of $A \leq x \leq B$. Their intersection has and only has points A and B, so that the variable x can be constrained to either A-value or B-value.

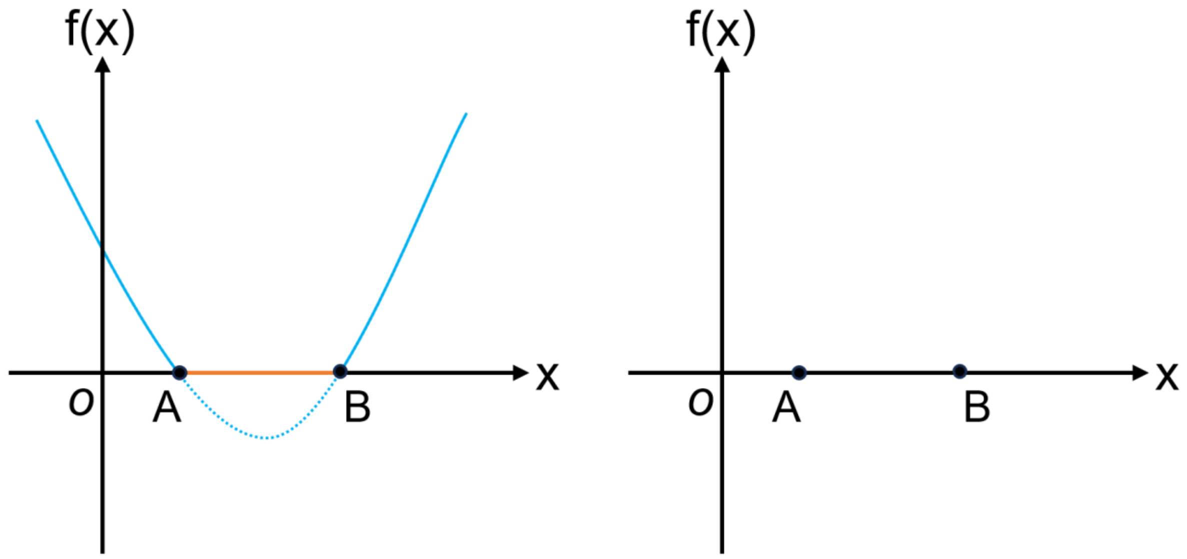
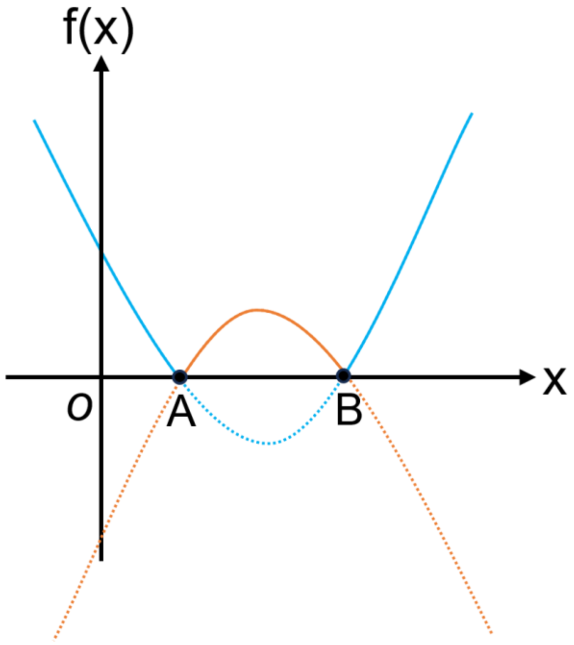
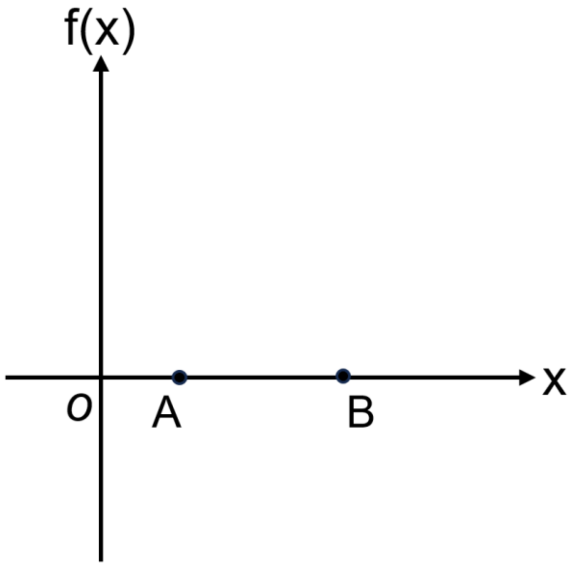


Fig.3 The blue curve is the constraint of $x^2 - (A + B) \cdot x + A \cdot B \geq 0$, and the orange curve is the constraint of $x^2 - (A + B) \cdot x + A \cdot B \leq 0$. Their intersection has and only has points A and B, so that the variable x can be constrained to either A-value or B-value.

(next page)


f(x)
x
O
A
B


f(x)
x
O
A
B