ACTA MATHEMATICA VIETNAMICA
Volume 35, Number 3, 2010, pp. 55–60

RELATION BETWEEN THE HARDNESS OF A PROBLEM AND THE NUMBER OF ITS SOLUTIONS

THAN QUANG KHOAT

Abstract. In this note, we survey the effect of the number of solutions on solving problems. Theoretically, the number of solutions to a problem cannot help the problem to be easier in the sense of computation. That is, a problem having few solutions may be as easy as the one having many solutions. For the aim of giving some evidences, we show that the Subset sum problem, Knapsack problem and Bounded Integer Programming problem under the assumption that the problem has either *no solution* or *exponentially many solutions* are NP-hard. On the other hand, the Knapsack Optimization problem having at most one solution is also NP-hard.

1. Introduction

For a certain problem, studying its structure is usually the first step in solving it. The more thoroughly we study, the more efficient algorithm we can hope to obtain. Virtually, some special structures of the problem could lead to an efficient algorithm. However, some others may reveal the hardness of the problem, such as discrete structure, non-linear structure.

In this note, we consider the effects of a special structure on solving the given problem, the number of solutions. By intuition, one may believe that a problem having many solutions may be easier to solve than the one having fewer solutions; a problem having unique solution may be harder than the others.[1] However, whether or not the situation remains true in other cases is unclear.

Valiant and Vazirani [9] are the first authors considering this question. They gave a strong evidence for the conjecture that solving NP problems is as hard as solving problems with unique solution. Specifically, they showed that if Unique SAT is in P then RP=NP. Another similar result holds for the Unique Shortest Vector problem [5]. These results leave an interesting question for researchers: whether solving a problem having unique solution is equivalent to solving an NP problem. In this note, we give another evidence for the affirmative answer. More concretely, we show that Knapsack optimization problem (KOP) is still NP-hard even under the assumption that it has at most one solution.

Received July 7, 2008; in revised form December 3, 2009.
2000 *Mathematics Subject Classification.* 68Q17, 68Q25.

[1]By saying "easy" (or "hard") we mean that, in the same model of computation, the upper time bound of a problem is less (or greater) than the one of other problem.

We should remark that the hardness of problems having at most one solution closely relates to cryptography. In the area of cryptography, many well-known public key cryptosystems were built based on the hope that some problems having at most one solution are hard. For example, the security of both the Ajtai-Dwork [1] and Regev cryptosystem [7] is based on the harness of some certain approximating version of the unique shortest vector problem. Therefore, if one can establish the proper hardness (e.g., NP-hard) of this problem, it would imply that one-way functions exist - the desire of cryptographers. In view of that, our result here and the result in [4] can be considered as being on the way towards supporting the existence of some one-way function.

Another interesting question is whether the situation remains true for problems having many solutions. A negative evidence comes from a result of Calabro et al. [2]. They proposed a randomized algorithm in expected time $poly(n)(2^n/s)^{1-1/k}$ for k-SAT, where n and s are the number of variables and solutions, respectively. Their algorithm clearly runs faster as the number of solutions is large. This result leads to the hope that the more solutions a problem has, the more efficient algorithm we may have to solve it. It also raises another question of *whether a problem having many solutions is strictly easier than NP problems.* More strongly, *can we assert that a problem having exponentially many solutions are strictly easier than NP problems?*

In this note, we give a negative answer for these questions by showing that the number of solutions to a problem cannot theoretically help the problem to be easier. Specifically, we show that Subset sum problem (SSP) under the assumption that it has either no solution or exponentially many solutions is still NP-hard. Moreover, with the same assumption, the *Knapsack problem* (KP), the *Bounded Knapsack problem* (BKP), the *Bounded Integer programming problem* (BIP), and the *Integer programming problem* (IP) are still NP-hard. To obtain these results, we propose a reduction from SSP to a new SSP, whose number of solutions is as twice as that of the old one. And if we use consecutively k reductions, then the number of solutions of the final SSP is as 2^k times as that of the original SSP.

The property of exponentially many solutions of a problem has been recently exploited in the cryptographic community. For instance, the cryptosystem proposed in [10] relates to a Knapsack-type problem having exponentially many solutions. In a certain sense, the security of the cryptosystem is partially based on the hardness of finding one exactly expected from exponentially many solutions. Thus, problems with (exponentially) many solutions may be good candidates for cryptographers in building secure cryptosystems. Our result here does provide some such problems which are guaranteed to be NP-hard.

2. Main results

2.1. **Many solutions.** We consider some problems having many solutions, and show that they are not easier than NP problems. Starting point of our argument is the Subset sum problem (SSP). First of all, we should recall it (see [3] for its decision version).

Find a vector $x \in \mathbb{Z}^n$ satisfying

$$\begin{cases} a_1x_1 + \cdots + a_nx_n = b \\ 0 \le x_i \le 1, \ \forall i \end{cases} \tag{2.1}$$

where the coefficients are positive integers.

It is well-known that SSP (2.1) is NP-hard (see [3], [8]). The input of the problem consists of only the coefficients and the number of variables. If there is some more information, the problem may be easier to be solved. Here, we consider the hardness of the problem when we are given some information on the number of solutions.

Theorem 2.1. *Any SSP is polynomial-computationally equivalent to a SSP with either no solution or $2^{\Theta(n)}$ solutions, where n is the number of variables of the problem.*

This theorem immediately yields the following corollary.

Corollary 2.2. *SSP (2.1) under the assumption that the number of its solutions is either zero or exponentially many is NP-hard.*

We can prove Theorem 2.1 by reducing SSP to a new SSP satisfying the given assumption. First, we reduce SSP (2.1) to a new one such that:

(2.2) *If SSP (2.1) has no solution then neither is the new one.*

(2.3) *If SSP (2.1) has solutions then the number of solutions of the new SSP is as twice as that of the old one.*

Applying this reduction, m times to SSP will yield a new SSP having many solutions, $\Omega(2^m)$.

Now we are going to present a reduction from SSP (2.1) to a new one satisfying (2.2) and (2.3). Without loss of generality, we assume that $a_i < b$, $\forall i$. The new SSP is as follows:

Find a vector $x \in \mathbb{Z}^{n+2}$ satisfying

$$\begin{cases} \sum_{i=1}^{n} a_ix_i + tx_{n+1} + tx_{n+2} = b + t \\ 0 \le x_i \le 1, \ \forall i \end{cases} \tag{2.4}$$

where t is an optional integer such that if $(x_1, ..., x_{n+2})$ is a solution to (2.4) then $(x_1, ..., x_n)$ is also a solution to (2.1).

It is clear that, with the above assumption on t, if x^* is a solution to (2.1), then $(x^*, 1, 0)$ and $(x^*, 0, 1)$ are solutions to (2.4). That is, the number of solutions to (2.4) is at least as twice as that to (2.1).

Lemma 2.3. *Assume that t is an integer greater than $(n+1)b$. Then SSP (2.4) has a solution $(x_1^*, ..., x_{n+2}^*)$ if and only if $(x_1^*, ..., x_n^*)$ is a solution to SSP (2.1). Moreover, the number of solutions of (2.4) is as twice as that of (2.1).*

Proof. It is easy to see that if $(x_1^*, ..., x_n^*)$ is a solution to (2.1), then $(x_1^*, ..., x_n^*, 1, 0)$ and $(x_1^*, ..., x_n^*, 0, 1)$ are solutions to (2.4).

Now we assume that $(x_1^*, ..., x_{n+2}^*)$ is a solution to (2.4), then we have the equation

$$\sum_{i=1}^{n} a_i x_i^* + t.(x_{n+1}^* + x_{n+2}^*) = b + t$$

or, equivalently,

$$\sum_{i=1}^{n} a_i x_i^* = b + t.(1 - x_{n+1}^* - x_{n+2}^*) \tag{2.5}$$

Assuming that $1 - x_{n+1}^* - x_{n+2}^* \neq 0$, then $|b + t.(1 - x_{n+1}^* - x_{n+2}^*)| > nb$. By the hypothesis $a_i < b$ for all i, we immediately have $\sum_{i=1}^{n} a_i x_i^* < nb$ (due to $0 \leq x_i^* \leq 1$). Combining (2.5) with these observations yields a contrary. Thus, if $(x_1^*, ..., x_{n+2}^*)$ is a solution to (2.4), then $1 - x_{n+1}^* - x_{n+2}^* = 0$. As a consequence, we have $\sum_{i=1}^{n} a_i x_i^* = b$; that is, $(x_1^*, ..., x_n^*)$ is a solution to SSP (2.1).

The second statement is straightforward. □

This lemma reveals a reduction from a SSP with n variables to a new one with $n + 2$ variables satisfying (2.2) and (2.3). Applying consecutively the reduction m times to SSP would yield a new SSP which has $n + 2m$ variables. The new SSP has $\Omega(2^m)$ solutions, provided that the original SSP has solutions. It has no solution if the original SSP has no solution. Consequently, a suitable value of m would lead to the result of Theorem 2.1. □

In short, Theorem 2.1 implies that, theoretically, a problem having many solutions can not be easier than the one having few solutions. Despite having exponentially many solutions, the problem can remain hard. The above technique can be easily applied to the BKP, KP and BIP.

BKP can be stated similarly to (2.1), provided that the variable x_i has an upper bound c_i $(i = 1, ..., n)$. In KP, there is no upper bound on x_i's. BIP is a generalized version of the above problems. Specifically, in BIP, we are asked to find a vector $x \in \mathbb{Z}^n$ satisfying some constraints $Ax = b$ and $0 \leq x \leq c$, where some coefficients may be negative integers. These problems are known to be NP-hard [3]. However, they remain NP-hard even if we know some more information, e.g. a promise of the lower bound on the number of solutions. Indeed,

Theorem 2.4. *BKP, KP and BIP under the assumption that the number of solutions is either zero or exponentially large are NP-hard.*

We can prove the NP-hardness of BKP with the given assumption by the same technique as the one used for SSP. KP can be reduced to BKP by adding upper bound $\lceil b/a_i \rceil$ on x_i, for all i. Similarly, BIP can be reduced to BKP by using the technique of Kannan [3]. Thus, Theorem 2.4 follows. □

2.2. **Unique solution.** We have just considered the hardness of some problems with many solutions. In this subsection, we continue to examine the hardness of the problems with at most one solution. To treat this issue, we use Knapsack optimization problem (KOP) [3] as a tool to assert the claim that a problem with at most one solution may not be easier than NP problems.

Now we consider the Knapsack problem [10], [6] (see its decision version in [3]):

Find a vector $x \in \mathbb{Z}^n$ satisfying

$$\begin{cases} a \cdot x = b \\ x \geq 0, \end{cases} \tag{2.6}$$

where the coefficients are positive integers.

It is not hard to see that (2.6) is NP-hard. We are going to reduce (2.6) to KOP which has at most one solution. The new KOP is as follows:

Find a solution x to

$$\begin{array}{l} f(x) = d_1x_1 + \cdots + d_nx_n \to \min \\ \begin{cases} a \cdot x = b \\ x \in \mathbb{Z}^n, x \geq 0, \end{cases} \end{array} \tag{2.7}$$

where d_i's are optional integers such that:

$+\ d_1 > 0,$

$+\ d_i > w\sum_{j=1}^{i-1} d_j,\ w = \max\limits_k\{\lceil b/a_k \rceil\}, i > 1$

It is clear that if x^0 is a solution to (2.7), then x^0 is also a solution to (2.6); if (2.7) has no solution, then neither has (2.6). Moreover, (2.7) has at most one solution. Indeed, assume that x^1 and x^2 are different solutions to (2.7). There exists $j,\ x_j^1 \neq x_j^2$. Let r be the greatest index such that $x_r^1 \neq x_r^2$. Without loss of generality, we assume that $x_r^1 < x_r^2$. Then we have

$$\begin{aligned} f(x^1) &= d \cdot x^1 = \sum_{j<r} d_jx_j^1 + d_rx_r^1 + \sum_{k>r} d_kx_k^1, \\ f(x^2) &= d \cdot x^2 = \sum_{j<r} d_jx_j^2 + d_rx_r^2 + \sum_{k>r} d_kx_k^2 \\ &= f(x^1) + \sum_{j<r} d_jx_j^2 + d_r(x_r^2 - x_r^1) - \sum_{j<r} d_jx_j^1. \end{aligned}$$

Note that $0 \leq x_j^1 \leq w,\ \forall j$. Thus, $\sum_{j<r} d_jx_j^1 < w\sum_{j<r} d_j < d_r$ due to the hypothesis on d_i's. Combining this with the fact that $x^1, x^2 \in \mathbb{Z}_+^n$ would lead to $\sum_{j<r} d_jx_j^1 < d_r(x_r^2 - x_r^1)$. Therefore, from the above representation of $f(x^2)$, we conclude that $f(x^1) < f(x^2)$. That is, (2.7) has at most one solution. From these arguments, we obtain the following.

Theorem 2.5. *Any knapsack problem can be polynomial-computationally reduced to a knapsack optimization problem having at most one solution.*

Note that KP is NP-hard. Thus, we obtain the following.

Corollary 2.6. *The Knapsack Optimization problem under the assumption that it has at most one solution is NP-hard.*

References

[1] M. Ajtai and C. Dwork, A public-key cryptosystem with worst-case/average-case equivalence, *In Proc. of the 29th annual ACM symposium on Theory of Computing*, pp. 284-293, 1997.

[2] C. Calabro, R. Impagliazzo, V. Kabanets and R. Paturi, The complexity of Unique k-SAT: An Isolation lemma for k-CNFS, *In Proc. of the 18th IEEE annual Conf. on Computational Complexity*, pp. 135-141, 2003.

[3] R. Kannan, Polynomial-time aggregation of Integer programming problems, *Journal of the ACM* **30**(1) (1983), 133-145.

[4] T. Q. Khoat and N. H. Tan, Unique shortest vector problem for max norm is NP-hard, *Vietnam Journal of Science and Technology* **46** (5A) (2008), 86-100. Special issue on the Second International Conference on Theories and Applications of Computer Science - ICTACS'09.

[5] R. Kumar and D. Silvakumar, A note on the shortest lattice vector problem, *In Proc. of the 14th annual IEEE Conf. on Computational Complexity*, pp. 200-204, 1999.

[6] P. Q. Nguyen and J. Stern, Adapting Density Attacks to Low-Weight Knapsacks, *Advances in Cryptology - ASIACRYPT05*, Springer-Verlag, pp. 41-58, 2005.

[7] O. Regev, New lattice-based cryptographic constructions, *Journal of the ACM* **51** (6) (2004), 899-942.

[8] A. Schrijver, *Theory of Linear and Integer Programming*, John Wiley, Chichester 1999.

[9] L. Valiant and V. Vazirani, NP is as easy as detecting unique solutions, *Theoretical Computer Science* **47** (1986), pp. 85-93.

[10] B. Wang, Q. Wu and Y. Hu, A knapsack-based probabilistic encryption scheme, *Information Sciences* **177** (2007), 3981-3994.

Faculty of Information Technology, Thái Nguyên University
Thái Nguyên city, Vietnam

Current address: School of Knowledge Science
Japan Advanced Institute of Science and Technology
E-mail address: tqkhoat@jaist.ac.jp