About fully polynomial approximability of the generalized knapsack problem

Sung-Pil Hong
School of Business, Chung-Ang University, sphong@cau.ac.kr
Bum Hwan Park
Industrial Engineering, Seoul National University, pbh@optima.snu.ac.kr

September 24, 2003

Abstract - The *generalized knapsack problem*, or *gknap* is the combinatorial optimization problem of optimizing a nonnegative linear functional over the integral hull of the intersection of a polynomially separable $0-1$ polytope and a knapsack constraint. Among many potential applications, the knapsack, the restricted shortest path, and the restricted spanning tree problem are such examples.

We establish some necessary and sufficient conditions for a gknap to admit a fully polynomial approximation scheme, or FPTAS, To do so, we recapture the scaling and approximate binary search techniques in the framework of gknap. This also enables us to find a condition that a gknap does not have an FPTAS. This condition is more general than the strong NP-hardness.

1 Introduction

Assume that Q is polynomially separable $0-1$ integral polytope, $c, d \in \mathbb{Z}_+^n$, and $B \in \mathbb{Z}_+$. Then a *generalized knapsack problem* can be written as follows:

Problem 1.1 : gknap

$$\begin{aligned} &\min, \text{ or } \max \ \ c^T x \\ &\text{s.t } \ x \in Q \\ &d^T x \leq, \text{ or } \geq B \\ &x \in \{0,1\}^n. \end{aligned}$$

Thus, given Q, there are four possible combinations of the objective-type and the inequality sign in the knapsack constraint. If, for instance, the objective is minimization and the knapsack constraint has $\leq$ inequality sign, then we denote the problem by gknap$(\min, \leq)$.

Example 1.2 Knapsack problem as a gknap$(\max, \leq)$ **or** gknap$(\min, \geq)$. Trivially, the knapsack problem is a gknap with $Q = \text{conv}\{0,1\}^n$. Let p_j and w_j, respectively, be the profit and volume of the j-th item, and W the volume of the knapsack. Then the knapsack problem is to choose a most profitable set of items that fits the knapsack capacity,

$$\begin{aligned} &\max p^T x \\ &\text{s.t } \ x \in \{0,1\}^n \\ &w^T x \leq W, \end{aligned}$$

or, to find a least profitable set of items which, if removed, makes the remaining items fit the knapsack,

$$\begin{aligned} &\min p^T x \\ &\text{s.t } \ x \in \{0,1\}^n \\ &w^T x \geq \textstyle\sum w_j - W. \end{aligned}$$

Although the two formulations are equivalent at the optimality, the approximability of one is not necessarily preserved in the other as the ratio of two optimal values, can be arbitrary. A well-known such example is the node cover and stable set problem. We will see, however, that two versions of the knapsack problem are also equivalent in approximation sense: In general, for the same Q, gknap$(\max, \leq)$ is fully polynomially approximable if and only if gknap$(\min, \geq)$ is so (See Definition 2.1 and Corollary 3.6).

Example 1.3 Restricted shortest path problem, RSP: Let $G = (V, A)$ be a directed graph. Let s, $t \in V$. Then, with a "cost", c_{ij} and a "delay", d_{ij} assigned to each arc (i, j),

the restricted shortest path problem is to find a minimum cost (s,t)-directed path among the paths whose delay is no greater than some bound B.

$$\begin{aligned} & \min c(P) \\ \text{s.t} \quad & P \in \mathcal{P}, \text{ the } (s,t)\text{-directed paths} \\ & d(P) \leq B. \end{aligned}$$

Then, it is well-known that $Q \subseteq \mathbb{R}^A$, the convex hull of the characteristic vectors of (s,t)-directed paths of G, is polynomially separable $0-1$ polytope. Hence RSP is a gknap(min, $\leq$).

The knapsack problem and RSP are known to have FPTAS. Both problems admit a pseudo-polynomial dynamic programming algorithm which finds an optimum in a pseudo-polynomial time in a single parameter, c or d. It is worth noticing that essentially every fully polynomial time approximation scheme in the literature is based on such a dynamic programming algorithm. For instance, we can devise a dynamic programming algorithm for RSP as follows: Denote by $d_j(\gamma)$, the minimum delay of a (s,j)-directed path whose cost is no greater than γ. Then, with initialization, $d_s(\gamma) = 0$ for all $\gamma \geq 0$ and $d_j(0) = \infty$, for every $j \in V - s$, we have, for $j \in V - s$, and $\gamma = 1, 2, \cdots$,

$$d_j(\gamma) = \text{Min}\{d_j(\gamma - 1), \text{Min}_{k|c_{kj} \leq \gamma}\{d_k(\gamma - c_{kj}) + d_{kj}\}\}$$

Since $d_t(\gamma)$ is monotone nonincreasing, when the recursion ends up with the first value $\gamma = \gamma^*$ such that $d_t(\gamma^*) \leq B$, we have $\gamma^* = OPT$. The running time is easily shown to be $O(|E|OPT)$.

But, not every gknap problem necessarily has such a nice dynamic programming algorithm.

Example 1.4 Restricted spanning tree problem, RST: Consider a connected undirected graph $G = (V, E)$ with $V = \{1, 2, \cdots n\}$. Each edge e is assigned two nonnegative integers, the "cost", c_e and "length", d_e. Then, RST is the problem of finding a spanning tree of the minimum cost whose length is no greater than a predetermined bound, B.

$$\begin{aligned} & \min c(T) \\ \text{s.t} \quad & T \in \mathcal{T}, \text{ the spanning trees of } G \\ & d(T) \leq B \end{aligned}$$

In [1], probably the first literature on RST, Aggarwal et al showed that RST is NP-hard. Ravi and Goemans [7] devised a polynomial time approximation scheme, or PTAS by carefully exploring the adjacency of the optima of the Lagrangian subproblem. Their scheme, however, relying on enumeration of the subsets of edges, whose number is exponential in the reciprocal of the approximation error, is not fully polynomial. The fully polynomial approximability of RST is currently open.

As mentioned earlier, every FPTAS of combinatorial optimization problem relies on a pseudo-polynomial algorithm. Therefore, one might be interested in whether RST is solvable in pseudo-polynomial time. Hong et al [3] showed that a two-variable extension of the matrix-tree theorem can be used to yield a pseudo-polynomial time algorithm for RST. It is not known, however, whether this pseudo-polynomial algorithm can lead to an FPTAS of RST as it is pseudo-polynomial in both c and d unlike the previous dynamic algorithms for the fully polynomially approximable problems.

On the other hand, to the author's best knowledge, there is no known problem for which an FPTAS is proved impossible while PTAS and a pseudo-polynomial algorithm are known. Thus, RST is in an interesting position in the map of non-MAX-SNP-hard problems.

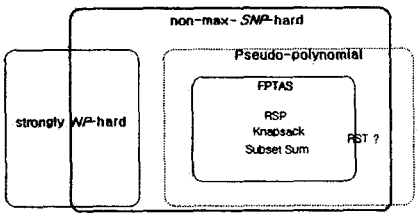

Figure 1: Non Max SNP-hard problems

2 Definitions and notation

It is very convenient to interchange the roles of c and d using the symmetry of the two parameters [2, 6].

Definition 2.1 Let Q be given. We will call gknap(min, $\geq$) and gknap(max, $\leq$) are *conjugates*. On the other hand, gknap(min, $\leq$) or gknap(max, $\geq$) is the conjugate of itself, or *self-conjugate*.

Definition 2.2 Given $\epsilon > 0$, by ϵ- approximation we mean to find a feasible solution $\hat{x}$ of a gknap satisfying

$$|c^T\hat{x} - OPT| < \epsilon OPT.$$

We will use $\langle\cdot\rangle$ to denote the binary encoding length of numbers, vectors, matrices, or even an instance of a problem. Similarly, $\|\cdot\|_\infty$ will denote the maximum absolute value of a number of a vector, matrix, or an input number of a problem instance. Note that as Q is $0-1$ integral, its vertex and facet complexity are polynomials of n. So the encoding length of gknap is determined by n, $\langle c_{\max}\rangle$, $\langle d_{\max}\rangle$, and $\langle B\rangle$.

Definition 2.3 We say a gknap is fully polynomially approximable if ϵ-approximation can be done in $\text{poly}(n, \langle c_{\max}\rangle, \langle d_{\max}\rangle, \langle B\rangle, 1/\epsilon)$.

Note that a fully polynomial approximation algorithm is also referred to as a *fully polynomial time approximation scheme* or FPTAS in the literature.

Remark 2.4 We will use the notation $\text{poly}(\cdots)$ rather abusively. They may not be the same polynomial with the same set of variables. It may be best read as "some polynomial of $\cdots$".

3 Scaling and approximate binary search

The *scaling* and *approximate binary search* have been core techniques in developing FPTAS [5, 4, 8]. We recapture them in the general framework of gknap. To do so, it is convenient to consider both *flooring* and *ceiling* operations: For fixed $\epsilon > 0$ and $C > 0$, define

$$\hat{c}_j = \lfloor \tfrac{c_j}{\epsilon C/n} \rfloor, \quad \check{c}_j = \lceil \tfrac{c_j}{\epsilon C/n} \rceil \text{ for } j = 1, \cdots, n. \quad (1)$$

Lemma 3.1 *For every $x \in \{0,1\}^n$, we have*

$$\begin{aligned} \hat{c}^T x > \lfloor \tfrac{n}{\epsilon} \rfloor &\Rightarrow c^T x > C \text{ and} \\ \hat{c}^T x \le \lfloor \tfrac{n}{\epsilon} \rfloor &\Rightarrow c^T x < (1+\epsilon)C, ; \quad (2) \\ \check{c}^T x < \lceil \tfrac{n}{\epsilon} \rceil &\Rightarrow c^T x < C \text{ and} \\ \check{c}^T x \le \lceil \tfrac{n}{\epsilon} \rceil &\Rightarrow c^T x > (1-\epsilon)C. \quad (3) \end{aligned}$$

Problem 3.2 $\widehat{\text{gknap}}$

$$\begin{aligned} \widehat{OPT} = \quad & \min \ \hat{c}^T x, \text{ or } \max \ \check{c}^T x \\ & \text{s.t } x \in Q \\ & d^T x \le \text{ or } \ge B \\ & x \in \{0,1\}^n. \end{aligned}$$

Lemma 3.3 *[8] Given a positive integer C, let $\hat{x}$ be an optimal solution of $\widehat{\text{gknap}}(\min, \cdot)$ (with c replaced by $\hat{c}$). Then,*

$$\begin{aligned} c^T\hat{x} < OPT + \epsilon C, \quad (4) \\ \hat{c}^T\hat{x} > \lfloor \tfrac{n}{\epsilon} \rfloor \Rightarrow OPT > C \text{ and} \\ \hat{c}^T\hat{x} \le \lfloor \tfrac{n}{\epsilon} \rfloor \Rightarrow OPT < (1+\epsilon)C. \quad (5) \end{aligned}$$

Similarly, let $\check{x}$ be an optimal solution of $\widehat{\text{gknap}}(\max, \cdot)$ (with c replaced by $\check{c}$). Then,

$$\begin{aligned} c^T\check{x} > OPT - \epsilon C, \quad (6) \\ \check{c}^T\check{x} < \lceil \tfrac{n}{\epsilon} \rceil \Rightarrow OPT < C, \text{ and} \\ \check{c}^T\check{x} \ge \lceil \tfrac{n}{\epsilon} \rceil \Rightarrow OPT > (1-\epsilon)C. \quad (7) \end{aligned}$$

Theorem 3.4 *If* gknap *of Problem 1.1 is solvable in a pseudo-polynomial time only in c, namely in* $\text{poly}(n, c_{\max}, \langle d_{\max}\rangle, \langle B\rangle)$, *then it is also fully polynomially approximable. The converse is also true.*

Corollary 3.5 *Given Q, a* gknap *is fully polynomially approximable if and only if its conjugate is solvable in a time that is pseudo-polynomial in the size of the knapsack constraint, namely in* $\text{poly}(n, \langle c_{\max}\rangle, d_{\max}, B)$ *time.*

Corollary 3.6 *Given Q, a* gknap *is fully polynomially approximable if and only if its conjugate is.*

Example 3.7 This corollary tells us that, in terms of fully polynomial approximability, the two formulations in Example 1.2 are also equivalent as they are conjugates.

Corollary 3.8 *Given Q, a* gknap *is fully polynomially approximable if and only if it is solvable in* $\text{poly}(n, \langle c_{\max}\rangle, d_{\max}, B)$ *time.*

The following theorem is immediate from Theorem 3.4 and Corollary 3.8.

Theorem 3.9 *For a given Q, a* gknap *is not fully polynomially approximable (unless $P = NP$) if there is an NP-hard subclass in which $d_{\max}, B = O(\text{poly}(n, \langle c\rangle))$, or $c_{\max} = O(\text{poly}(n, \langle d\rangle, \langle B\rangle))$.*

Remark 3.10 WHAT CAN WE SAY ABOUT THE NECESSITY?

Recall that the strong NP-hardness of a problem Π requires Π to have an NP-hard non-number subclass: $\Pi_p = \{\iota \in \Pi : \|\iota\|_\infty \leq p(\langle\iota\rangle)\}$ for some polynomial p. Therefore, the condition of Theorem 3.8 is more general in the sense that c or both d and B can have arbitrary values as far as the maximum absolute value of a number in the remaining parameter(s) is polynomially bounded. We will refer to these instances as *semi-number* problems.

Note that the strong NP-hardness is not useful, for instance, for probing impossibility of an FPTAS for RST. RST is known to have a pseudo-polynomial algorithm [3] and whose any non-number instance is, therefore, necessarily polynomial.

References

[1] V. Aggarwal, Y. P. Aneja, and K. P. K. Nair. Minimal spanning tree subject to a side constraints. *Comput. Ops. Res.*, 9(4):287–296, 1982.

[2] R. Hassin. Approximation schemes for the restricted shortest path problem. *Mathematics of Operations Research*, 17(1):36–42, Feb 1992.

[3] S.-P. Hong, B.-H. Park, and S.-J. Chung. A fully polynomial bicriteria approximation scheme for constrained spanning tree problem. *Manuscript, To appear in Operations Research Letters*, December 2002.

[4] E. Horowitz and S. Sahni. *Fundamentals of Computer Algorithms.* Computer Science Press, Maryland, 1989.

[5] O. Ibarra and C. Kim. Fast approximation algorithms for the knapsack and sum of subsets problems. *Journal of ACM*, 22:463–468, 1975.

[6] M. V. Marathe, R. Ravi, R. Sundaram, S. S. Ravi, D. J. Rosenkrantz, and H. B. Hunt III. Bicriteria network design problems. *Journal of Algorithms*, 28:142–171, 1998.

[7] R. Ravi and M. X. Goemans. The constrained minimum spanning tree problem. In *Proceedings of the Scandinavian Workshop on Algorithmic Theory, LNCS*, volume 1097, pages 66–75, 1996.

[8] A. Warburton. Approximation of pareto optima in multiple-objective, shortest-path problems. *Operations Research*, 35(1):70–79, Jan-Feb 1987.