FPTAS and pseudo-polynomial separability of integral hull of generalized knapsack problem

Sung-Pil Hong
Chung-Ang University

October 2004

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 prove via the ellipsoid method the equivalence between the fully polynomial approximability and a certain pseudo-polynomial separability of the gknap polytope.

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

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

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,

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

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

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

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.

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

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.

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.

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

2 Preliminaries

It is very convenient to interchange the roles of c and d using the symmetry of the two parameters [2, 4].

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 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 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 $\cdots$".

The following result has been established by Hong and the proof is omitted.

Theorem 2.5 *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* poly$(n, \langle c_{\max}\rangle, d_{\max}, B)$ *time.*

3 Pseudo-polynomial separation and fully polynomial approximation

3.1 Ellipsoid method with separation subroutine.

Grötschel, Lovász and Schrijver, in their seminal work [1], observed an equivalence between the separation and optimization on

polyhedra. Even when the polyhedron is not given by an explicit set of linear inequalities, the ellipsoid method can be applied to solve the feasibility problem in polynomial time if the separation problem is solvable in polynomial time.

Unlike when a linear system is given explicitly, when the polyhedron P is not full-dimensional, the technique that perturbing P into an equivalent full dimensional polyhedron does not seem to be promising as, for instance, the separation of P and that of the perturbed P seem to be quite different problems: the perturbation will mar the combinatorial problem of P that is usually essential for an efficient separation algorithm. The following theorem is adopted from [5].

Theorem 3.1 *Let $P \subseteq \mathbb{R}^n$ be a polyhedron whose facet complexity is bounded by φ. And suppose a separation algorithm $\mathcal{A}$ is given. Then there is a polynomial $\Psi(\cdot,\cdot)$ such that the feasibility problem is solvable in $T \times \Psi(n,\varphi)$, where T is the maximum time required by $\mathcal{A}$ for inputs of size at most $\Psi(n,\varphi)$.*

This theorem shows certain independence between the main algorithm and the separation subroutine. It requires only that the separation algorithm $\mathcal{A}$ correctly finds a separation hyperplane. Then the main algorithm guarantees that the size of the current solution bounded by $\Psi(n,\varphi)$ determined *a priori* and independently of $\mathcal{A}$. And $\mathcal{A}$ only affects the complexity of each iteration. In particular, $\mathcal{A}$ does not have to be polynomial in Theorem 3.1. This enables us to derive an analogy between fully polynomial approximation and the pseudo-polynomial separability on gknap.

3.2 Pseudo-polynomial separation and fully polynomial approximation.

Write $P = Q \cap \{x : d^T x \leq, \text{ or } \geq B\}$. Also denote by P_I, the integral hull, namely the convex hull of the integral solutions of $P = Q \cap \{x : d^T x \leq, \text{ or } \geq B\}$. If necessary, we will also use notation $P^{\leq}$ or $P^{\geq}$, respectively, depending on whether knapsack constraint is $\leq$ of $\geq$-type.

Then we can find OPT of gknap by finding a minimum or maximum γ such that the polytope $P_I(\gamma) = P_I \cap \{x : c^T x \leq, \text{ or } \geq \gamma\}$ is feasible. Thus to find OPT it suffices to solve the feasibility problem of $P_I(\gamma)$ for a fixed $0 \leq \gamma \leq nc_{\max}$.

First, the complexity of P_I is poly(n), since, by assumption, every vertex of Q is $0-1$ integral. Its complexity, namely the maximum of an encoding length of a vertex is $O(n)$. Then it is well-known that the facet complexity is poly(n) [5]. Hence the facet complexity of $P_I(\gamma)$, is essentially bounded by the encoding length of $c^T x \leq, \text{ or } \geq \gamma$.

Lemma 3.2 *The facet complexity φ_γ of $P_I(\gamma)$ is* poly$(n, \langle c_{\max}\rangle)$.

This essentially determines the number of iterations of ellipsoid method.

Notice that given a separation algorithm for P_I, it is a trivial matter to devise a separation algorithm for $P_I(\gamma)$ which runs in essentially the same computation time. Now we denote by sep$(y \mid P_I)$ the problem to separate y from P_I.

Lemma 3.3 *Let $y \in \mathbb{Q}^n$ be any vector. If there is a separation algorithm $\mathcal{A}$ which solve* sep$(y \mid P_I)$ *in* poly$(n, \langle y\rangle, d_{\max}, B)$, *then* gknap *is solvable in* poly$(n, \langle c_{\max}\rangle, d_{\max}, B)$

Proof. According to Theorem 3.1 we may assume $\langle y\rangle$ is bounded by $\Psi(n, \varphi_\gamma)$ which is, from Lemma 3.2, poly$(n, \langle c_{\max}\rangle)$. Then the complexity of $\mathcal{A}$ becomes poly$(n, \langle c_{\max}\rangle, d_{\max}, B)$.

Hence by Theorem 3.1 the total running time for solving feasibility problem is $O(\text{poly}(n, \langle c_{\max}\rangle, d_{\max}, B) \times \Psi(n, \varphi_\gamma))$ which is poly$(n, \langle c_{\max}\rangle, d_{\max}, B)$. Hence OPT is also solvable in poly$(n, \langle c_{\max}\rangle, d_{\max}, B)$. □

Remark 3.4 Since $P_I(\gamma) \subseteq [0,1]^n$. We can the initial point of the ellipsoid method independent of c. However, in general iterations, it seems to be inevitable that the next current point y is determined by c (See Figure 1). Thus it seems to be essential that the running time of $\mathcal{A}$ is polynomial in $\langle y \rangle$.

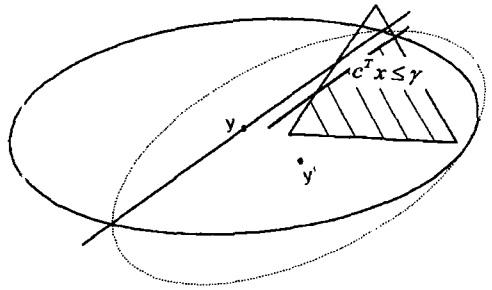

Figure 1: The next current point y' is determined by c.

Theorem 3.5 *The* gknap *is fully polynomially approximable if for any $y \in \mathbb{Q}^n$ the separation problem for P_I of its conjugate,* $\mathrm{sep}(y \mid P_I)$ *is solvable in* $\mathrm{poly}(n, \langle y_{\max} \rangle, d_{\max}, B)$.

Proof. From Theorem 2.5 and Lemma 3.3. □

Let's consider the converse of Theorem 3.5 which is true if, by Theorem 2.5, the solvability of its conjugate in $\mathrm{poly}(n, \langle c_{\max} \rangle, d_{\max}, B)$ would imply the solvability of the separation of the integral hull P_I, $\mathrm{sep}(y \mid P_I)$ in $\mathrm{poly}(n, \langle y_{\max} \rangle, d_{\max}, B)$. The "if part" of the above statement seems to be plausible considering the following theorem [5].

Theorem 3.6 *Let $y \in \mathbb{Q}^n$ and $P \subseteq \mathbb{R}^n$ be a polyhedron whose facet complexity is bounded by φ. And suppose an algorithm $\mathcal{A}$ is given for the optimization problem defined by $P \subseteq \mathbb{R}^n$ and $y \in \mathbb{Q}^n$. Then there is an algorithm for* $\mathrm{sep}(y \mid P)$ *whose running time is* $\mathrm{poly}(n, \varphi, \langle y \rangle, \tau(y, P))$, *where $\tau(y, P)$ is the running time of $\mathcal{A}$.*

If gknap is fully polynomially approximable, then by Theorem 2.5, $\mathrm{opt}(y \mid P_I)$ on P_I of its conjugate is solvable in $\mathrm{poly}(n, \langle y_{\max} \rangle, d_{\max}, B)$. Therefore, by Theorem 3.6, $\mathrm{sep}(y \mid P_I)$ is solvable in $\mathrm{poly}(n, \varphi, \langle y \rangle, \tau(y, P_I))$ which is $\mathrm{poly}(n, \langle y_{\max} \rangle, d_{\max}, B)$.

However, we need to be careful since in gknap we assumed the $c \geq 0$. Hence we can apply the above theorem only if $y \geq 0$. But, if $y_j < 0$ for some j its separation from P_I is a trivial problem. Hence,

Theorem 3.7 *The* gknap *is fully polynomially approximable if and only if for any $y \in \mathbb{Q}^n$ the separation problem for P_I of its conjugate,* $\mathrm{sep}(y \mid P_I)$ *is solvable in* $\mathrm{poly}(n, \langle y_{\max} \rangle, d_{\max}, B)$.

References

[1] M. Grotschel, L. Lovasz, and A. Schrijver. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica*, 1(2):169–197, 1981.

[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] 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.

[5] A. Schrijver. *Theory of linear and integer programming.* Wiley, Chichester, 1986.