Intractability: A Geometric Representation

Sami Khuri
Department of Mathematics & Computer Science
San José State University
One Washington Square
San José, CA 95192-0103
khuri@sjsumcs.sjsu.edu

Abstract: This paper introduces a geometric representation that can be applied to illustrate the complexity of some combinatorial optimization problems. In this work, it is applied to the 0/1 knapsack problem and to a special case of a scheduling problem. This representation gives insight into the difference between tractable and intractable problems. It can therefore be used as a stepping stone to compare polynomial (P) and nondeterministic polynomial (NP) problems, before venturing into the world of NP-completeness.

1 INTRODUCTION

Whether teaching a course on algorithms as suggested in [ACM68], or similar to CS2 in [ACM79], or CO2 in [GT86], or more recently in [Tur91], we are always faced with a big challenge when the task of explaining the qualitative increase in computation in going from a P running time algorithm to an NP. The paper uses a geometric representation of the knapsack problem to explicate this increase. Understanding this geometric representation should make the formal definition of the classes of tractable and intractable problems much easier to comprehend.

The rational and integer knapsack problems are defined in the first section. Then an example and a feasible solution are given. The geometric representation of the solution is then introduced and from that representation we derive the polynomial time algorithm that yields the optimum solution to the rational knapsack problem. From the same example and its geometric representation we show how much more complicated is the integer problem than its rational counterpart.

As mentioned earlier, the geometric representation and its analysis will prepare the reader to plunge into the formal definition and study of NP problems.

SIGSCE 94- 3/94, Phoenix, Arizona,USA

The geometric representation can be used in other combinatorial optimization problems. For example, we end the article by using it for the maximum profit scheduling problem with equal deadlines.

2 THE KNAPSACK PROBLEM

In this problem, we have a knapsack of capacity M and n different objects. Each object has a weight w_i and a profit p_i. M, w_i, and p_i for $i = 1, 2, \ldots, n$ are positive integers. We would like to fill the knapsack with objects that will yield the maximum profit.

If we allow a fraction of an object to be placed in the knapsack then the problem is known as the Rational Knapsack Problem (RatKnap). On the other hand, fractions of objects are not allowed to be put into the knapsack in the 0/1, or Binary Knapsack Problem (0/1Knap).

The following is a formal definition of the knapsack problem in which we make use of Stinson's terminology for combinatorial optimization problems [Sti87].

Problem instance: Positive integers: $w_1, w_2, \ldots, w_n$; $p_1, p_2, \ldots, p_n$; and M.

Feasible solution: A vector $\vec{x} = (x_1, x_2, \ldots, x_n)$ such that $\sum_{1 \le i \le n} w_i x_i \le M$, where:

$x_i \in [0, 1]$, for the RatKnap, (thus allowing fractions of objects to be chosen); or

$x_i \in \{0, 1\}$, for the 0/1Knap, (i.e., each object is either chosen $x_i = 1$, or not $x_i = 0$)

Objective function: A profit $P(\vec{x}) = \sum_{1 \le i \le n} x_i p_i$, where $\vec{x} = (x_1, x_2, \ldots, x_n)$ is a feasible vector.

Optimal solution: Maximum-profit feasible solution.

Martello, et al., cite three reasons for which the 0/1Knap is one of the most studied discrete programming prob-

has coordinates $p_1 + p_3 = 11$ and $w_1 + w_3 = 6$, while the coordinates of C are $p_1 + p_3 + p_4 = 23$ and $w_1 + w_3 + w_4 = 14$,

to one correspondence between the chosen objects and the slopes of their corresponding line segments, ordering line

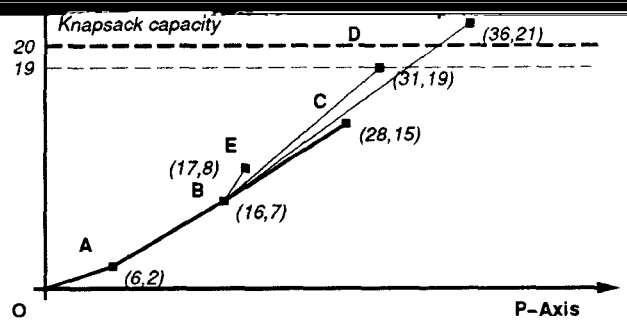

Figure 4: Different solutions of Example 1 for the 0/1Knap case.

segments from flattest to steepest (i.e., in increasing order of their slopes: $\frac{w_i}{p_i}$) is equivalent to considering the objects in decreasing order of $\frac{p_i}{w_i}$. The latter is the well known greedy algorithm which always yields an optimal knapsack for the RatKnap problem [Sti87].

STEPS OF THE GREEDY ALGORITHM

Put the objects in the knapsack according to the decreasing order of profit/weight, updating at each stage the cumulative weight. Stop if either the knapsack is full (usually by allowing a fraction of the last considered object) or if all objects were chosen without exceeding the capacity of the knapsack. □

The set of slopes is $\{1, \frac{1}{3}, \frac{1}{2}, \frac{2}{3}, \frac{4}{5}, \frac{7}{10}\}$ and applying the algorithm, we consider the objects in the order: 2, 3, 4, 6, 5, 1. The solution is given by the vector (0, 1, 1, 1, 0, $\frac{5}{14}$) with

to solve the 0/1Knap problem. *But what method will solve it?* Afterall, since the greedy technique does not work, the first two choices in Example of Section 2 are suspect. A solid class discussion yields the result that every one of the 6! paths must be examined, the optimum solution being the path whose endpoint is at or below 20 on the W-axis and farthest to the right on the P-axis. We thus conclude that solving the 0/1Knap problem requires substantially more enumeration than the RatKnap problem. Indeed, as we all know, the 0/1Knap problem is an NP-complete problem (it is in NP and the 3-Exact cover problem, which is NP-complete, can be polynomially transformed to the 0/1Knap problem [PS82]).

The geometric representation can be used to model other combinatorial problems such as the maximum profit scheduling problem with equal deadlines.

4 THE MAXIMUM PROFIT SCHEDULING PROBLEM

The maximum profit scheduling problem is a task scheduling problem. A problem instance of the general scheduling problem consists of a set T of n tasks, each of which has a length l_i, the time it takes for its execution, a deadline d_i before which the task must be scheduled and its execution completed, and a profit p_i. Scheduling the tasks of a subset S of T consists in finding the starting time of each task in S, such that at most one task at a time is performed and such that each task finishes before its deadline. Here we study the case where all jobs have identical deadlines $d_1 = d_2 = \ldots = d_n = D$. The following is a formal definition of the maximum profit

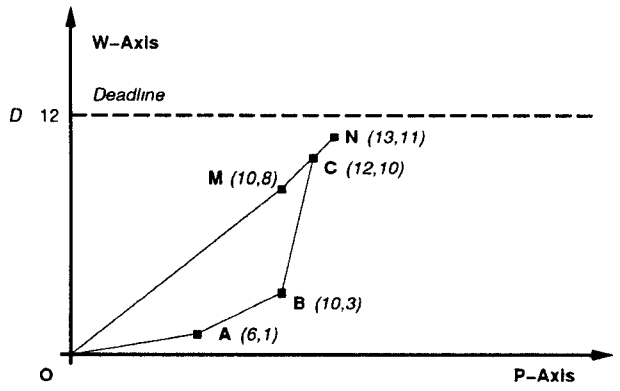

Figure 5: Geometric representation of schedules for the tasks in S and S' of Example 2.

The problem is NP-complete since the partitioning problem can be polynomially transformed into it [Kar72].

The problem can be modeled by our geometric representation. As with the knapsack problem, each alternative forms a nondecreasing path where the W-axis in Figure 2 represents cumulative task lengths, and the knapsack capacity M is replaced by the deadline D.

EXAMPLE 2

The following is an instance of the maximum profit scheduling problem with six tasks that have identical deadline values of twelve:

"knapsack capacity M" as fast as possible. Consequently, the greedy technique considers the "objects" in *increasing* order of $\frac{p_i}{w_i}$ line segments, in other words, in *decreasing* order of their slopes: $\frac{w_i}{p_i}$.

5 CONCLUSION

This paper introduces a geometric representation of the knapsack problem. It shows how the repesentation can be used to introduce the substantial enumeration difference in solving easy tractable problems (rational knapsack) and NP-complete ones (integer knapsack). The representation leads us to the greedy technique which for the rational case, runs in polynomial time and yields the optimum solution. Equivalently, one can easily plot the points corresponding to flat line segments, and compute the best profit. From the geometric representation one also concludes that the integer case is much harder and probably requires trying almost all combinations of line segments, leading to an exhaustive search and therefore an exponential (in the number of objects) running time algorithm.

In the last section we show how the geometric representation can be used for the maximum profit scheduling problem with equal deadlines. We end the section by mentioning other problems that can be modeled by the geometric representation and indicate that with minor changes, it can be used to model minimization problems.

It is our belief that more combinatorial problems can be modeled by the geometric representation, and that this pictorial representation of hard problems can be used as an educational tool to give a lot of insight into the difference between

tractable and intractable problems.

ACKNOWLEDGMENTS

The author would like to thank Dr. Frederick Stern, Thomas Bäck, and Jörg Heitkötter for their useful comments.

REFERENCES

[ACM68] ACM Curriculum Committee on Computer Science. Curriculum 68: Recommendations for academic programs in computer science. *Communications of the ACM*, 11(3):151–197, 1968.

[ACM79] ACM Curriculum Committee on Computer Science. Curriculum 78: Recommendations for the undergraduate program in computer science. *Communications of the ACM*, 22(3):147–166, 1979.

[GT86] N. Gibbs and A. Tucker. A model corriculum for a liberal arts degree in computer science. *Communications of the ACM*, 29(3):202–210, 1986.

[Kar72] R. M. Karp. Reducibility among combinatorial problems. In R. E. Miller and J. W. Thatcher, editors, *Complexity of Computer Computation*, pages 85–103. Plenum, New York, 1972.

[MT90] S. Martello and P. Toth. *Knapsack Problems: Algorithms and Computer Implementations*. John Wiley & Sons, Chichester, West Sussex, England, 1990.

[PS82] C. H. Papadimitriou and K. Steiglitz. *Combinatorial Optimization: Algorithms and Complexity*. Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1982.

[Sti87] D. R. Stinson. *An Introduction to the Design and Analysis of Algorithms*. The Charles Babbage Research Center, Winnipeg, Manitoba, Canada, 2nd edition, 1987.

[Tur91] A. J. Turner. Introduction to the joint curriculum task force report. *Communications of the ACM*, 34(6):69–79, 1991.