The Second International Conference "Problems of Cybernetics and Informatics"
September 10-12, 2008, Baku, Azerbaijan. The plenary paper
www.pci2008.science.az/5/02.pdf

DISCRETE OPTIMIZATION AND DIOPHANTINE ANALYSIS

Djangir Babayev

Cox Associates, Denver, CO, USA
4834 Macintosh Place, Boulder CO, USA, 80301
djangirbabayev@aol.com

Introduction

Discrete optimization is an important area of Mathematical Optimization Theory both in theoretical and applied sense. Discrete optimization, in particular, integer programming[8] is an area appeared in 50-ties of XX century and until recently had no relation to solving equations in integer numbers in a Diophantine sense. Currently the integer programming is very developed area and found wide applications for mathematical modelling and finding optimal solutions in diverse areas of industry and management.

General **Integer Programming problem** is formulated as

(1.1) $$\text{Maximize} \sum_{i=1}^{n} c_i x_i$$

subject to

(1.2) $$\sum_{i=1}^{n} a_{ij} x_i \leq B_j \text{ , } j=1,\dots,m$$

(1.3) $$x_i \geq 0, i=1,\dots, n \text{ integer.}$$

Coefficients $a_{ij} \geq 0$, c_i, and values $B_j \geq 0$, are nonnegative integers.

An important particular case of the Integer Programming problem is the case with one constraint, case with $m=1$, **Knapsack Problem,** (1.4)-(1.6).

(1.4) $$\text{Maximize} \sum_{i=1}^{n} c_i x_i$$

subject to

(1.5) $$\sum_{i=1}^{n} a_i x_i \leq B,$$

(1.6) $$x_i \geq 0, i=1,\dots, n \text{ integer.}$$

Coefficients a_i, c_i, and value *B* are nonnegative integers.

Having a single constarint (1.5) gives to the Knapsack Problem an undoubted elegance of clearity of formulation and helped this problem to obtain wide theoretical and applied usage. But this elegance did not make the Knapsack Problem computationally easier to solve – it still belongs to the most difficult to solve class of NP-hard problems.

Wide theoretical and practical applications of Knapsak Problem created a huge volume of research on creating efficient computational methods for its solving [5], [6].

Historically integer programming problems (1.1)-(1.3) including Knapsack Problems were considered as special cases of Linear Programming problems, when variables allowed to have x_i with real values, and not restricted to be integers only. This laid an imprint on the solution methods of integer programming. And as a result currently the most efficient existing methods of integer programming are methods based on using branch and bound technique.

The Framework of new approach to developing methods for solving the Knapsack Problem

First by adding an integer nonnegative slack variable x_{n+1} with coefficients a_{n+1}=1 and c_{n+1}=0 the original Knapsack Problem (1.4)-(1.6) is reduced to the following version

(1.7) $$\text{Maximize} \sum_{i \in N} c_i x_i$$

subject to

(1.8) $$\sum_{i \in N} a_i x_i = b\,,$$

(1.9) $$x_i \geq 0,\ i \in N = 1,\ldots,n+1, \text{ integer.}$$

The structure of the original Knapsack Problem (1.4)-(1.6) allows to determine lower and upper bounds $\underline{c}$ and for $\bar{c}$ the optimal value of the Knapsack Problem

$$c^{\bullet} = \text{Maximum} \sum_{i \in N} c_i x_i\ .$$

Now the problem is reduced to a search the for value c= $c^{\bullet}$ in the interval [$\underline{c}$, $\bar{c}$]. Note that, c=c(b) is a nondecreasing function of b. The search can be caried out by testing values of c from [$\underline{c}$, $\bar{c}$] by some strategy. So, the original Knapsack Problem is reduced to the following Max Consistency problem:

Find integer nonnegative variables x_i for which, system of two Diophantine equations (2.0) and (2.1) are consistent for maximum value of c in a given interval.

Maximize $\underline{c} \leq c \leq \bar{c}$ subject to

(2.0) $$\sum_{i \in N} c_i x_i = c,$$

(2.1) $$\sum_{i \in N} a_i x_i = b\,,$$

(2.2) $$x_i \geq 0, i=1,\ldots, n+1 \text{ integer.}$$

By using Integer Equivalent Aggregation Procedure this Max Consistency2 problem can be reduced to the equivalent Max Consistency 1 problem, where two equations (2.0)-(2.1) are replaced by a single equivalent equation.

Foundations of Integer Equivalent Aggregation was presented by English mathematician G. B. Mathews to London Mathematical Society in 1897.

The principal formulation is:

Aggregation Proposition. *For an arbitrary system of m linear algebraic equaitions with integer coefficients and right hand sides and a bounded set of integer nonnegative solutions an infinite number of m-tuples of integer weights exists, so that any equation created as linear combination of the original equations with these weights has the same set of nonnegative integer solutions as the original system.*

In our case of m=2 according to ***Aggregation Proposition*** two integer multipliers are required to aggregate equations (2.0) and (2.1) into a single equivalent equation. They are denoted by v (*value*) *for equation* (2.0) and w(*weight*) for (2.1). In the search procedure for solving Max Consistency 1 problem, consistency of the aggregated equation is required to be tested for various values of c. Computationally it is highly desirable to determine multipliers only once, so that they can be valid for all values to be tested. Additionally, for the considered Knapsack Problem they should have positive values. Both of these goals are made possible by determining the multipliers by Theorem 2 of [3]. This Theorem is formulated by reference to a set of Q- a set of multi-dimensional vectors (as possibly constrained by additional inequalities and equalities of interest) and collection of eight inequality conditions:

$$\text{(3)}\quad \begin{array}{llll} C_{11}^1: & -v > U_2 - b; & C_{11}^2: & v > U_2 - b; \\ C_{12}^1: & v > (L_2 - b); & C_{12}^2: & -v > (L_2 - b); \\ C_{21}^1: & w > U_1 - c; & C_{21}^2: & -w > U_1 - c; \\ C_{22}^1: & -w > (L_1 - c); & C_{22}^2: & w > (L_1 - c); \end{array}$$

where

$$\text{(4)}\quad U_1 = \max\left(\max_{x\in Q}\sum_{J\in N} c_j x_j, c\right), \qquad L_1 = \min\left(\min_{x\in Q}\sum_{j\in N} c_j x_j, c\right);$$

$$U_2 = \max\left(\max_{x\in Q}\sum_{J\in N} a_j x_j, b\right), \qquad L_2 = \min\left(\min_{x\in Q}\sum_{j\in N} a_j x_j, b\right).$$

Each of the inequalities (3) is strict, with nonnegative right hand side, thus providing the lower bounds for the absolute values of the multipliers *v* and *w*.

Theorem 1.(Theorem 2, in [3]). *Equation*

$$\text{(5)}\qquad \sum_{j\in N} \alpha_j x_j = \beta,$$

where

$$\text{(6)}\qquad \alpha_j = vc_j + wa_j, \; j\in N, \quad \beta = vc + wb;$$

posseses the same set of nonnegative solutions as system (2.0)-(2.1), *if v and w are relatively prime integers and satisfy any pair of conditions* (C_{ip}^1, C_{kr}^2), where

$i \neq k$ or $p \neq r$. If i=k=1 in the selected pair (C_{ip}^1, C_{kr}^2), *then the multiplier w* (*if i*=*k*=2, *then the multiplier v*) *can be assigned an arbitrary integer value of any sign, relatively prime with v*(*w*).

Now the original Knapsack Problem (2.0) – (2.2) is reduced to the **Consistency 1 problem**
Find max *c* from interval $\underline{c} \le c \le \overline{c}$
subject to (5) – (6).
For solving this problem [1] presents the following
Consistency Algorithm

Step 1. Initialization. Determine $\overline{c}$ -an upper bound for $c^\bullet$.
Step 2. $c=\overline{c}$.
Step 3. Compute β and test the consistency of Diophantine equation (5).
Step 4. If (5) is inconsistent, then set *c:=c*-1 and return to Step 2.
Step 5. If Eq (5) is consistent, then $c^\bullet$=c and the corresponding solution of equation (5) is the optimal solution of the original Knapsack Problem, END.

For testing consistency of Diophantine equation (5) two new computationally efficient algorithms have been developed [1], [2], [4], [9]. In these algorithms a graph is generated based on equation (5) and a the shortest path between specific pairs of vertices is defined. Computational complexity of these algorithms are O($n\alpha_1$) and O($n+\alpha_1^2$) where $\alpha_1 = \min_{j\in N} \alpha_j$.

In a comprehensive computational experiments where Knapsack Problems with variables up to 250,000 were solved [1] the developed knapsack solution procedure, **Consistency Algorithm,** appeared to be significantly superior to advanced branch and bound methods (previously established to be the most efficient knapsack solution procedures), obtaining solutions several orders of magnitude faster for hard problems.

In another developed search method Consistency Algorithm for solving the Knapsack Problem is replaced by a binary search algorithm for $c = c^{\bullet}$ in interval [$\underline{c}$, $\overline{c}$]. In this method for Knapsack Problem the Diophantine equation testing problem is solved at most $\log_2(\overline{c} - \underline{c})$ times. This means that with the use of a polynomial Diophantine equation testing algorithm, the developed binary search Knapsack solution algorithm would be polynomial.

References

1. DJANGIR BABAYEV, FRED GLOVER, JENNIFER RYAN, 1997. New Knapsack Solution Approach by Integer Equivalent Aggregation and Consistency Determination. *INFORMS Journal of Computing, Vol. 9, No.1, Winter 1997,43-50.*
2. F. GLOVER, 1969. Integer programming over a Finite Additive Group, *SIAM Journal on Control, 7, 213-231.*
3. F. GLOVER , D. BABAYEV. 1995. New Results for Aggregating of Integer Valued Equations. *Annals of Operations Research, 58-227-242.*
4. P. HANSEN and J. RYAN, 1996. Testing Integer Knapsack for feasibility, *European Journal of Operational Research, 88, 578-582.*
5. HANS KELLERER, ULRICH PFERSCHY, DAVID PISINGER, 2004, *Knapsack Problems.*Springer, Berlin, New York.
6. S. MARTELLO P. TOTH, 1990. *Knapsack Problems: Algorithms and Computer Implementation*, JOHN WILEY & SONS, Chichester.
7. G. B. MATHEWS, 1897. On the Partition of Numbers. *Proceedings of The London Mathematical Society, 28, 486-490.*
8. GEORGE L. NEMHAUSER, LAURENCE A. WOOLSEY, 1988. *Integer and Combinatorial Optimization.* A Wiley – Interscience Publication, JOHN WILEY & SONS, New York, Chichester, Brisbane, Toronto, Singapore.
9. J. RYAN, 1990. The Structure of Integral Monoid and Integer Programming Feasibility, *Discrete applied Mathematics, 28, 251, 263.*