Dhaka Univ. J. Sci. 61(2): 135-140, 2013 (July)

# An Improved Subgradiend Optimization Technique for Solving IPs with Lagrangean Relaxation

**M. Babul Hasan and Md. Toha**

*Deepartment of Mathematics, Dhaka University, Dhaka-1000, Bangladesh*

(Received: 10 August 2011; Accepted: 29 July 2012)

## Abstract

The objective of this paper is to improve the subgradient optimization method which is used to solve non-differentiable optimization problems in the Lagrangian dual problem. One of the main drawbacks of the subgradient method is the tuning process to determine the sequence of step-lengths to update successive iterates. In this paper, we propose a modified subgradient optimization method with various step size rules to compute a tuning- free subgradient step-length that is geometrically motivated and algebraically deduced. It is well known that the dual function is a concave function over its domain (regardless of the structure of the cost and constraints of the primal problem), but not necessarily differentiable. We solve the dual problem whenever it is easier to solve than the primal problem with no duality gap. However, even if there is a duality gap the solution of the dual problem provides a lower bound to the primal optimum that can be useful in combinatorial optimization. Numerical examples are illustrated to demonstrate the method.

**Key Words:** Relaxation, subgradient, optimization, IP

## I. Introduction

Linear and Integer programming[1] deal with the optimization (maximization or minimization) of a function of variables known as objective function, subject to a set of linear equalities and/or inequalities known as constraints. The objective function may be profit, cost, production capacity or any other measure of effectiveness, which is to be obtained in the best possible or optimal manner.

The constraints may be imposed by different sources such as market demand, production processes and equipments, storage capacity, raw material availability, etc.

Integer programming[2,3] (IP) is a valuable tool in operations research, having tremendous potential for applications. Such problems occur quite frequently in business and industry. All assignment and transportation problems are IP problems. In these problems, the decision variables are either zero or one. i.e.

$$x_j = 0 \text{ or } 1$$

Other examples are capital budgeting and production scheduling problems. In fact, any situation involving decisions of the type "either to do a job or not to do" ("either-or") can be viewed as an IP problem. In all such situations,

$$x_j = \begin{bmatrix} 1 \quad (\text{if jth activity is performed }), \\ 0 \ (\text{if jth activity is not performed}) \end{bmatrix}$$

In addition, all allocation problems involving the allocation of men and machines give rises to IP problems, since such commodities can be assigned only in integers and not infractions.

The focus of this paper is on the subgradient optimization methods[4] which are used for solving IP problems using the methodology of Lagrangian relaxation and dualization. The goal of this paper is to employ the subgradient optimization techniques to solve a practical business related planning problem which involves certain difficult constraints but imbedded in some tractable nice mathematical structures. Therefore, in order to solve practical IP problems we may need to resort to approximation schemes and problem specific algorithms which can exploit some special structures of the problem at hand.

In Section 2, we will review the relevant literature about the approach of the Lagrangian relaxation of IP problems and formulate the related Lagrangian dual problem. We also discuss properties of the dual problem, optimality conditions as well as the general structure of the dual objective function. In Section 3, we will propose an improved subgradient optimization method and an improved algorithm to solve practical business related problems using Lagrangean relaxation technique. Section 4 concludes the paper.

## II. Lagrangean Relaxation and Duality

In the last decade, Lagrangian Relaxation[5] has been grown from a successful but largely theoretical concept to a tool that is the backbone of a number of large scale applications. There have been several surveys of Lagrangian relaxation in Geoffrin[6] and Shapiro[7]. More extensive use of Lagrangian relaxation in practice has been hampered by the lack of a "how to do it" exposition similar to the treatment usually accorded for LP, dynamic programming (DP) and IP in operation research texts. This paper is intended to at least partially fill that void and should be of interest to both developers and users of Lagrangian relaxation algorithms. Lagrangian Relaxation is based upon the observation that many difficult integer programming problems can be modeled as a relatively easy problem complicated by a set of side constraints. To exploit this observation we create a Lagrangian problem in which the complicating constraints are replaced with a penalty term in the objective function involving the amount of violation of the constraints and their dual variables. The Lagrangian problem is easy to solve and provides an upper bound (for maximization problem) on the optimal value of the original problem. It can thus be used in place of a linear programming (LP) to

Author for correspondence, *e-mai :* mbabulhasan@yahoo.com

provide bounds in a branch and bound algorithm. The Lagrangian approach offers a number of important advantages over linear and integer programming relaxation. Lagrangian dual arises from a Lagrangian relaxation. Lagrangian relaxation is a useful technique in nonlinear programming, large-scale or structural linear, convex and IP. In this paper, we restrict ourselves to case of IP. We first formulate the Lagrangian relaxation[8] concept in general terms and then demonstrate its use extensively on a numerical example. We begin with an IP problem of the following form:

(IP) Z = max $cx$

subject to: $Ax \geq b$

$$x \in X = \{x \in \mathbb{Z}_+^n : Dx \geq d\}$$

where $c \in \mathbb{R}^n$, (A,b) and (D, d) are $m \times (n+1)$ $r \times (n+1)$ matrices respectively and all other matrices have conformable dimension. Here $x \in \mathbb{Z}_+^n$ means that $x$ is an n-dimensional vector of non-negative integers. X is a set of discrete (integral) points in polyhedral and assumed to be non-empty and bounded for convenience. We call the problem (IP) the primal problem and its solution a primal solution. Suppose that the constraints $Dx \geq d$ are "nice" in a sense that an integer program with just these constraints, i.e

$$\min\{\bar{c}x : Dx \geq d, x \in \mathbb{Z}_+^n\}$$

can be "easily" solved for any choice of $\bar{c}$ while the whole problem which includes also the other constraints $Ax \geq b$ may be significantly harder to solve. We call those constraints Ax ≥ b, which make a solution procedure diffcult, the complicating constraints. A common approach to solve this problem, perhaps approximately, is to solve its Lagrangian dual problem obtained via Lagrangian relaxation[4,9,10,11]. In the Lagrangian relaxation approach, the complicating constraints Ax ≥ b are relaxed by introducing a multiplier vector u $\in \mathbb{R}^m$, called Lagrangian multiplier, and the Lagrangian function

$$L(x,u) = cx + u(b - Ax).$$

Given $u \in \mathbb{R}^m$, the Lagrangian relaxation problem is then to solve the subproblem:

SP (u): $\varphi(u) = \min L(x,u)$

$$s.t. x \in X$$

yields the function φ determined pointwise by the optimal objective value of the subproblem. Note that, for any $x \in \mathbb{R}^m$, $x \in X \cup \{x : Ax \geq b\}$, and any optimal solution $x^*$ of the (IP) it holds that

$$\varphi(u) \leq L(x,u) \leq cx \text{ and}$$

$$\varphi(u) \leq L(x^*,u) \leq cx^* = z^*.$$

The relative simplicity of solving the subproblem and the fact that $\varphi(u) \leq z^*$ allows SP(u) to be used to provide lower bounds for (IP). In general, corresponding to different values of $u$, one obtains different lower bound $\varphi(u)$ to the primal optimal value z∗. Thus, to obtain the best (greatest) lower bound of $z^*$, the best choice of $u$ would be any one which is an optimal solution to the Lagrangian dual (LD) problem:

(LD) $\varphi* = \max\{\varphi(u) : u \geq 0\}$

where $\varphi(u)$ is given pointwise by the subproblem SP$(u)$:

$$\varphi(u) = \min cx + u(b - Ax)$$

subject to: $x \in X$.

The function $\varphi$ is called the dual function. It may be noted that when m constraints that have been dualized are equality constraints of the form $Ax = b$, the corresponding Lagrangian multipliers are unrestricted in sign and the Lagrangian dual becomes

$$\varphi *= \max_{u \in \mathbb{R}^m}\{\varphi(u)\}$$

Other possible relaxation problem of the IP is a LP relaxation. For the IP problem the LP relaxation is given by

(LP) $z_{LP}^*$= min $cx$

subject to: $Ax \geq b$

$$x \in \bar{X} = \{x \in R_+^n : Dx \geq d\}.$$

There are three major questions in determining Lagrangian relaxation-based system[12]: (a) which constraints should be relaxed, (b) how to compute a good multiplier$u$, (c) how to deduce a good feasible solution to the original problem, given a solution to the relaxed problem. Roughly speaking, the answer to (a) is that the relaxation should make the problem easier, but not so easy. For (b) there is choice between a general purpose procedure called subgradient method and "smarter" methods which may be better but which are however, highly problem specific. Similarly the answer to (c) tends to be problem specific.

## III. An Improved Subgradient Optimization Method

In this section, we design a unique subgradient optimization[13] procedure that can be used to solve the Lagrangian dual of 0-1 Integer Programming. We have already seen that the Lagrangian dual problem[14] can be formulated as a linear programming problem whose numbers of constraints are equal to the number of elements of the set X that make the direct use of linear programming system impractical. The numerical approach which we used to solve the Lagrangian dual problem without using a linear programming system is a subgradient optimization method. The subgradient optimization method that we would like to consider is an iterative procedure that can be used to solve the problem of maximizing a non-differentable concave function $\varphi$ $(u)$ on a closed convex set Ω, i.e.,

max { $\varphi$ $(u)$:$u \in \Omega$}

Using the following generic procedure:

- Choose an initial point $u^0 \in \Omega$.
- Construct a sequence of points $\{u^n\} \subseteq \Omega$ which eventually converges to an optimal solution using the rule

$$u^{n+1} = P_{\Omega.}(u^n + \lambda_n v^n)$$

where $P_\Omega(.)$ is a projection on the set $\Omega, (\lambda_n > 0)$ is a positive scalar called step length and $v^n$ is a vector, called step direction, which has to be determined at each iterate point.

- Until (some stopping condition).

We want to find out the direction of motion (step direction) at each iterate point in the procedure to obtain a desired outcome depending on a particular tolerance.

*Improved Subgradient Algorithm for the Lagrangian Dual*

We have developed a modified subgradient optimization algorithm for the Lagrangean dual which is stated below:

*Step 0:* (Initialization) choose a starting point $u^0 \geq 0$ and let $k = 1$.

*Step 1:* Determine a subgradient vector $s^k$ at $u^k$ by solving the subproblem SP $(u^k)$:

$$\varphi(u^k) = \min cx + u^k(b - Ax)$$

subject to: $x \in X$

Let $x^k$ be a solution of this subproblem. Then, $s^k = b - Ax^k$.

*Step 2:* (Feasibility and Optimality Test)

If $s^k \leq 0$, then $s^k$ is an $\varepsilon$ -optimal solution to the primal problem with

$$\varepsilon = |u^k s^k|.$$

If $s^k \leq 0$ and $u^k s^k = 0$, then $x^k$ is an optimal solution of the primal problem and $u^k$ is an optimal solution of the Lagrangian dual problem. STOP. Otherwise go to Step 3.

*Step 3:* Let $u^{k+1} = P_{R_+^m}(u^k + \lambda_k s^k)$, where

$P_{R_+^m}(u) = \bar{u}$ for which its i-th component $\bar{u_\iota} = \begin{cases} u_i & if\ u_i \geq 0 \\ 0, & \text{otherwise} \end{cases}$

and ($\lambda_k \geq 0$) is a step length given by (4.4).

Let $k = k + 1$, and return to step 1.

Ideally the subgradient algorithm can be stopped when, on some iterate k, we find a subgradient which is a zero vector. However, in practice this can rarely happen since the algorithm just chooses one subgradient $s^k$ and has no way of showing $0 \in \partial\varphi(u^k)$ as a convex combination of subgradients. The stopping criteria stated in the Step 2, i.e., $s^k \leq 0$ and $u^k s^k = 0$, can happen only if the strong duality holds. However, this is not generally possible for integer programming problems. Hence the typical stopping rule is either to stop after a sufficiently large but fixed number of iterations or to stop if the value of the function has not increased (by at least a certain amount) within a given number of iterations.

*Implementation of Our Algorithm*

In this section, we use the subgradient optimization method in order to solve a real life business related problem by "relaxing" the side constraints as follows:

Stockco is considering four investments. Investment 1 will yield a net present value(NPV) of $16,000; Investment 2, an NPV of $22,000; investment 3, an NPV of $12,000; and investment 4, an NPV of $8,000; Each investment requires a certain cash outflow at the present time: Investment 1: $5000; investment 2: $7000; investment 3: $4000 and investment 4: $3000. At present, $14000 is available for investment. Formulate an IP whose solution will tell Stockco how to maximize the NPV obtained from investments 1-4.

*Solution:* As in LP formulations, we begin by defining a variable for each decision that Stockco must make. This leads us to define a 0-1 variable:

$$x_j = \begin{cases} 1, \text{if investment j is made} \\ 0 \qquad\qquad \text{otherwise} \end{cases}$$

where $j = 1,2,3,4$

For example, $x_2 = 1$ if investment 2 is made and $x_2 = 0$ if investment 2 is not made.

The NPV obtained by Stockco (in thousands of dollars) is

The total NPV of the Stockco = $16\,x_1 + 22\,x_2 + 12\,x_3 + 8\,x_4$ (1)

If $x_j = 1$ , (1) includes the NPV of investment j, and if $x_j = 0$, (1) does not include the NPV of investment j. this means that whatever combination of investment is undertaken, (1) gives the NPV of that combination of projects. This reasoning implies that Stockco's objective function is Maximize,

$$z = 16\,x_1 + 22\,x_2 + 12\,x_3 + 8\,x_4 \qquad (2)$$

Stockco faces the constraint that at most $14,000 can be invested. By the same reasoning used to develop (1), we can show that the total amount invested (in thousands of dollars) = $5\,x_1 + 7\,x_2 + 4\,x_3 + 3\,x_4$

Since at most $14,000 can be invested $x_1, x_2, x_3$ and $x_4$ must satisfy

$$5\,x_1 + 7\,x_2 + 4\,x_3 + 3\,x_4 \leq 14 \qquad (3)$$

Combining (2) and (3) with the constraints

$x_j = 0$ or $1$ $(j = 1,2,3,4)$ yields the following 0-1 IP:

Max: $Z_D = 16\,x_1 + 22\,x_2 + 12\,x_3 + 8\,x_4$
s. t. $5\,x_1 + 7\,x_2 + 4\,x_3 + 3\,x_4 \leq 14$
$x_j = 0$ or $1$ $(j = 1,2,3,4)$

*Subgradient Optimization Procedure*

Here, we will use our numerical example to develop and demonstrate the subgradient optimization method for obtaining dual variable values that produce a tight bound. Ideally, we should solve the following dual problem: $Z_D = \min_{u \geq 0} Z(u)$

Minimize: $-16\,x_1 - 22\,x_2 - 12\,x_3 - 8\,x_4$
s.t. $5\,x_1 + 7\,x_2 + 4\,x_3 + 3\,x_4 \leq 14$
$x_j = 0$ or $1$ $(j = 1,2,3,4)$

If we relax the inequality, then integrality is easy:

Min $L(x,u) = -16\,x_1 - 22\,x_2 - 12\,x_3 - 8\,x_4 + (5\,x_1 + 7\,x_2 + 4\,x_3 + 3\,x_4 - 14)$

s.t. $x_j = 0$ or $1$ $(j = 1,2,3,4)$

*Initialize:* Set iteration $k = 1$. Set the jump size $t^k = 0.5$. Set a small scalar ending tolerance $\varepsilon > 0$, say $\varepsilon = 0.01$ and pick $u^k = 0$

*Subgradient optimization procedure for $k = 1$*

Original problem:
Minimize:$-16\,x_1 - 22\,x_2 - 12\,x_3 - 8\,x_4$
s.t. $5\,x_1 + 7\,x_2 + 4\,x_3 + 3\,x_4 \leq 14$
$x_j = 0$ or 1 ($j = 1,2,3,4$)
$k = 1, u^k = 0, t^k = 0.5\,, \varepsilon = 0.01$
*Step1:* Solve $L(x, u^1)$

Minimize: $-16\,x_1 - 22\,x_2 - 12\,x_3 - 8\,x_4 + 0(5\,x_1 + 7\,x_2 + 4\,x_3 + 3\,x_4 - 14)$
s.t. $x_j = 0$ or 1 ($j = 1,2,3,4$)
Set all $x_j = 1$
*Step 2:* slack is $19 - 14 = 5$
Set $u^{k+1} = \max\{0, 0 + 0.5 \times 5\} = 2.5$
$t^{k+1} = 0.5 \times 0.9$
*Step 3:* If $u^{k+1} - u^k < \varepsilon$ then stop else set $k = k + 1$ and go to step 1.
But $|u^{k+1} - u^k| = |2.5 - 0| > \varepsilon$
So go to step 1 with $k = 2$.

The detailed result is shown as follows:

Subgradient optimization

| | | Obj-coeffs | -16 | -22 | -12 | -8 | | | | | row rhs | Epsilon |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | row coeffs | -5 | -7 | -4 | -3 | | | | | -14 | 0.001 |
| Step | t(k) | Lamda | cost1 | cost2 | cost3 | cost4 | x1 | x2 | x3 | x4 | slack | stop? |
| 1 | 0.5 | 0 | -16 | -22 | -12 | -8 | 1 | 1 | 1 | 1 | 5 | |
| 2 | 0.45 | 2.5 | -3.5 | -4.5 | -2 | -0.5 | 1 | 1 | 1 | 1 | 5 | FALSE |
| 3 | 0.405 | 4.75 | 7.75 | 11.25 | 7 | 6.25 | 0 | 0 | 0 | 0 | -14 | FALSE |
| 4 | 0.364 | 0 | -16 | -22 | -12 | -8 | 1 | 1 | 1 | 1 | 5 | FALSE |
| 5 | 0.328 | 1.8225 | 6.887 | 9.242 | -4.71 | 2.5325 | 1 | 1 | 1 | 1 | 5 | FALSE |
| 6 | 0.295 | 3.46275 | 1.313 | 2.239 | 1.851 | 2.3885 | 0 | 0 | 0 | 0 | -14 | FALSE |
| 7 | 0.265 | 0 | -16 | -22 | -12 | -8 | 1 | 1 | 1 | 1 | 5 | FALSE |
| 8 | 0.239 | 1.3286 | -9.357 | -12.7 | -6.685 | -4.014 | 1 | 1 | 1 | 1 | 5 | FALSE |
| 9 | 0.215 | 2.52434 | -3.378 | -0.329 | -1.902 | -0.427 | 1 | 1 | 1 | 1 | 5 | FALSE |
| 10 | 0.193 | 3.60051 | 2.002 | 3.203 | 2.402 | 2.8015 | 0 | 0 | 0 | 0 | -14 | FALSE |
| 11 | 0.174 | 0.88857 | -11.55 | 15.78 | -8.445 | -5.334 | 1 | 1 | 1 | 1 | 5 | FALSE |
| 12 | 0.156 | 1.76027 | -7.198 | -9.678 | -4.958 | -2.719 | 1 | 1 | 1 | 1 | 5 | FALSE |
| 13 | 0.141 | 2.54479 | -3.276 | -4.186 | -1.82 | -0.365 | 1 | 1 | 1 | 1 | 5 | FALSE |
| 14 | 0.127 | 3.25087 | 0.254 | 0.756 | 1.0034 | 1.7526 | 0 | 0 | 0 | 0 | -14 | FALSE |
| 15 | 0.114 | 1.47156 | -8.642 | -11.69 | -6.113 | -3.585 | 1 | 1 | 1 | 1 | 5 | FALSE |
| 16 | 0.102 | 2.04348 | -5.782 | -7.695 | -3.826 | -1.869 | 1 | 1 | 1 | 1 | 5 | FALSE |
| 17 | 0.092 | 2.5582 | -3.209 | -4.092 | -1.767 | -0.325 | 1 | 1 | 1 | 1 | 5 | FALSE |
| 18 | 0.083 | 3.0214 | -0.892 | -0.849 | 0.0858 | 1.0643 | 1 | 1 | 0 | 0 | -2 | FALSE |
| 19 | 0.075 | 2.85469 | -1.726 | -2.017 | -0.581 | 0.564 | 1 | 1 | 1 | 0 | 2 | FALSE |
| 20 | 0.067 | 3.0047 | -0.976 | -0.966 | 0.0191 | 1.0143 | 1 | 1 | 0 | 0 | -2 | FALSE |
| 21 | 0.06 | 2.8697 | -1.651 | -1.912 | -0.521 | 0.6091 | 1 | 1 | 1 | 0 | 2 | FALSE |
| 22 | 0.054 | 2.9912 | -1.043 | -1.061 | -0.034 | 0.9738 | 1 | 1 | 1 | 0 | 2 | FALSE |
| 23 | 0.049 | 3.1007 | -0.496 | -0.295 | 0.4027 | 1.302 | 1 | 1 | 0 | 0 | -2 | FALSE |
| 24 | 0.044 | 3.0022 | -0.988 | -0.984 | 0.0088 | 1.0066 | 1 | 1 | 0 | 0 | -2 | FALSE |
| 25 | 0.039 | 2.9135 | -1.432 | -1.604 | -0.345 | 0.7407 | 1 | 1 | 1 | 0 | 2 | FALSE |
| 26 | 0.035 | 2.9933 | -1.033 | -1.046 | -0.026 | 0.98 | 1 | 1 | 1 | 0 | 2 | FALSE |
| 27 | 0.032 | 3.0651 | -0.674 | -0.544 | 0.2605 | 1.1954 | 1 | 1 | 0 | 0 | -2 | FALSE |
| 28 | 0.029 | 3.0005 | -0.997 | -0.996 | 0.0021 | 1.0016 | 1 | 1 | 0 | 0 | -2 | FALSE |
| 29 | 0.026 | 2.9423 | -1.288 | -1.403 | -0.23 | 0.8271 | 1 | 1 | 1 | 0 | 2 | FALSE |
| 30 | 0.023 | 2.9947 | -1.026 | -1.037 | -0.021 | 0.9841 | 1 | 1 | 1 | 0 | 2 | FALSE |
| 65 | 0 | 3.001 | -0.993 | -0.99 | 0.0052 | 1.0039 | 1 | 1 | 0 | 0 | -2 | FALSE |
| 66 | 0 | 3.0001 | -0.999 | -0.999 | 0.0005 | 1.0003 | 1 | 1 | 0 | 0 | -2 | FALSE |
| 67 | 0 | 2.999 | -1.004 | -1.006 | -0.003 | 0.9971 | 1 | 1 | 1 | 0 | 2 | FALSE |
| 68 | 0 | 3 | -0.999 | -0.999 | 7.00E-50 | 1 | 1 | 1 | 0 | 0 | -2 | TRUE |

We change the step size ($t^k$) and Lagrange multiplier ($u^k$) for finding optimal value, using least iterations which are shown in Table 1.

**Table 1. Solution for different step sizes and Lagrange multipliers for finding optimal iteration.**

| Initial value of Lamda (u) | Step size (t) | x1 | x2 | x3 | x4 | No. of Iterations |
|---|---|---|---|---|---|---|
| 0 | 0.5 | 1 | 1 | 0 | 0 | 68 |
| 0 | 0.75 | 1 | 1 | 0 | 0 | 72 |
| 100 | 0.65 | 0 | 1 | 0 | 0 | 87 |
| 50 | 0.65 | 1 | 1 | 0 |  | 77 |
| 40 | 1.25 | 1 | 1 | 0 | 0 | 79 |
| 10 | 1 | 1 | 1 | 1 | 0 | 75 |
| 60 | 0.2 | 0 | 0 | 0 | 0 | 78 |
| 60 | 0.3 | 0 | 0 | 0 | 0 | 82 |
| 60 | 0.4 | 1 | 0 | 0 | 1 | 84 |
| 30 | 0.4 | 1 | 1 | 1 | 0 | 66 |
| 35 | 0.4 | 1 | 1 | 0 | 0 | 66 |

From Table 1, we conclude that the computational results show that our algorithm for Lagragian dual finishes in better solution than the general primal solution. Therefore, we are accomplished to reduce *duality gap*[15] which is ultimately the amount of discrepancy from the complementary slackness condition for the Lagrangian relaxation equation. Numerical implementations show that our solution method reduces computational time. Furthermore, comparisons of numerical results of our new algorithm with that of other alternative algorithms show its various improved features.

*Graphical representation of our problem*

We may obtain the optimal value from the graph. If we substitute any $x$ into the objective function for the Lagrangian dual problem, we obtain a linear function in $u$ (Lagrangian multiplier). Figure 1 exhibits this family of linear function for all Lagrangian relaxation solutions that are optimal for at least one value of $u$. The fact that we must maximize the Lagrangian objective means that for any particular value of u, $L(x,u)$ is equal to the largest of these linear functions. Thus, the $L(x,u)$ function is given by the upper envelope of this family of linear equations that is shown as a darkened piecewise linear function in Figure1. From this figure it is easy to see that $u = 1$ minimize $L(x,u)$.

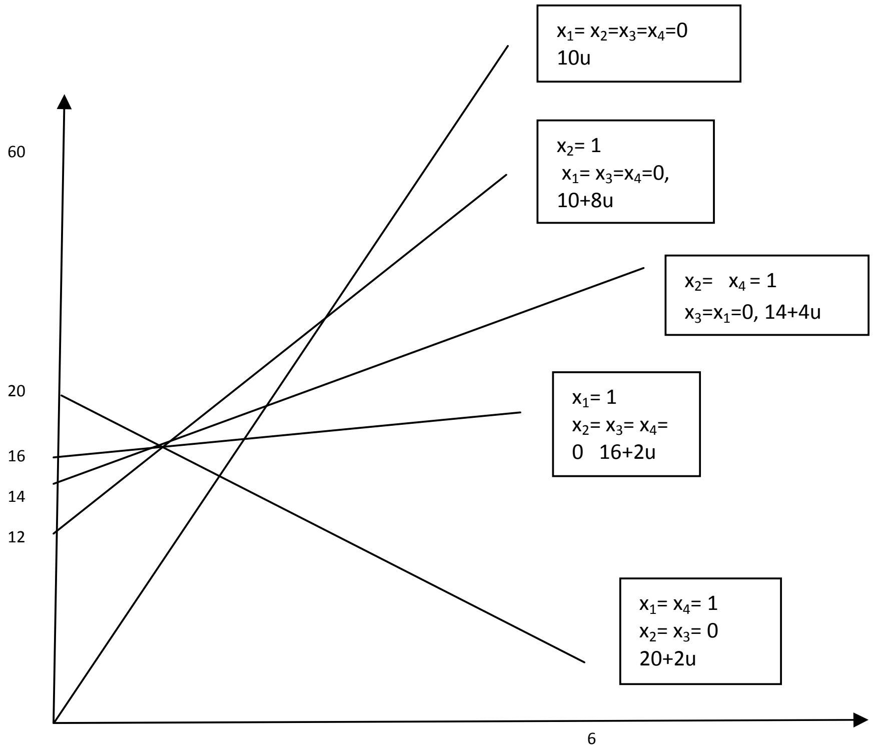

**Fig. 1.** The piecewise linear $L(x,u)$ function.

*Computational Results*

In this study, some stopping rules were applied to enhance the computational efficiency of the algorithm. Practically, however, there is no way of proving the optimality by this Lagrangian relaxation using subgradient optimization method as long as the positive duality gap exists. It is hopeful to find a feasible solution close to an optimal value more rapidly by making stopping rules appropriate to each problem. Therefore the subgradient method oscillates around dual optimal solution resulting in slow convergence. Nonetheless, the proposed solution algorithm found all the optimal solutions. Besides, it is simple to understand and easy to execute.

## IV. Conclusion

In this paper, a unified subgradient optimization method has been developed which can completely eliminate the main drawback of different subgradient optimization methods. The subgradient methods have also been employed to solve a business (Investment) related practical problem. The foundation of this work is the methodology of Lagrangian relaxation and dualization of a 0-1 integer programming problem. The method helps to remove a set of complicating constraints of an integer programming problem and give a frame work to exploit any available "nice" mathematical structure embedded in the Knapsack problem in order to solve the problem approximately. One of the central points of this research is to develop a subgradient procedure which deals with the construction of primal solutions directly from the information generated in the process of deflected or conditional subgradient optimization methods. We deduced a method that can produce a near-optimal primal solution for an integer programming problem. The goal of this paper has been to employ the subgradient optimization echniques to solve real life problems.

## References

**1.** Phillips, D. T., A. Ravindran and J. J. Solberg, 1987. Operation Research: Principles and Practices, Jan Wiley and Sons, New York, 550-650.

2. Held, M., P. Wolfe, and H. P. Crowder, 1974. Validation of subgradient optimization. *Mathematical Programming*, **6**, 62-88.

3. GEOFFRIN, A. M, Lagrangian Relaxation for Integer Programming, *Mathematical Programming Study*, North Holland, Amsterdam, Holland, **2**, 82-114.

4. Hamacher H. W. and F. Maffoli, 2003. Subgradient Optimization methods in integer programming, *Mathematical Society,* New York 1-171.

5. Camerini, P. M., L. Fratta and F. Maffioli, 1975. On improving relaxation methods by modified gradient techniques. *Mathematical Programming Study*, **3**, 26-34.

6. Geoffrin, A. M: 1974. Lagrangian relaxation and it uses in integer programming, *Mathematical Programming Study*, **2**, 82-114.

7. Shapiro, J. F, 1979. Mathematical Programming: Structures and Algorithms, John Wiley,New York.

8. Sherali, H. D. and O. Ulular, 1989. A primal-dual conjugate subgradient algorithm for specially structured linear and convex programming problems. *Applied Mathematics and Optimization*, **20**, 193-221.

9. Held, M. and R. M Karp, 1970. The travelling salesman problem and minimum spanning trees. *Operations Research*, **18**,1138-1162.

10. Held, M. and R. M Karp, 1971. The travelling salesman problem and minimum spanning trees: Part II. *Mathematical Programming*, **1**, 6-25.

11. Fisher, M. L., 1981. The Lagrangian relaxation method for solving integer programming problems. *Management Science*, **27(1)**, 1-18.

12. DANTZIG. G. B and P. WOLFE, 1960, Decomposition Principles for Linear Programs. *Operations Research*, **8**, 101-111.

13. Larsson, T., M. Patriksson and A. B. Strömberg, 1996. Conditional subgradient optimization: theory and applications, *European Journal of Operations Research*, **88**, 382-403.

14. Rockafellar, R.T., 1970, Convex Analysis. Princeton University Press.

15. Parker, R.G. and R. L. Rardin, 1988, "Discrete Optimization". Academic Press, Inc.