American Journal of Applied Sciences 4 (12): 1040-1044, 2007
ISSN 1546-9239

A Graphical Based Method for a Class of One-Stage Bounded Variables and Single Constrained Linear Programming Problems

[1]Suwitchaporn Witchakul, [1]Prapaisri Sudasna-na-Ayudthya
[1]Peerayuth Charnsethikul and [2]Kamlesh Mathur
[1]Operations Research and Management Science Units, Department of Industrial Engineering
Faculty of Engineering, Kasetsart University, Bangkok, Thailand
[2]Weatherhead School of Management, Case Western Reserve University, Cleveland, Ohio, USA

Abstract: An efficient and effective special purpose method is proposed for solving a class of one-stage single constrained linear programming problems with a finite number of right hand side scenarios and bounded variables. We compare our proposed method with a general purpose method using CPLEX interactive optimizer. For various *m* and *n*, by using elapsed computational time as the criteria, our procedure outperformed the general purpose method as the problem size grew.

Key words: Linear programming under uncertainties, bounded variables

INTRODUCTION

Consider a class of one-stage linear programming (LP) with a single constraint under a finite number of right hand sides (RHS) and bounded variables with the following formulation.

$$\underset{x,u,v}{Min}\, Z = \sum_{j=1}^{n} C_j x_j + \sum_{i=1}^{m} (g_i u_i + h_i v_i) \quad (1.1)$$

$$S.T. \sum_{j=1}^{n} a_j x_j + u_i - v_i = b_i \quad (1.2)$$

$$0 \le x_j \le t_j \quad (1.3)$$

where x_j, t_j, a_j, C_j are decision variable, upper bound of item *j*, weight coefficient of item *j*, cost coefficient of item *j*, respectively.

$$x_j, t_j, a_j, C_j \ge 0 \text{ for } j = 1,...,n \quad (1.4)$$

u_i, v_i, b_i, g_i, h_i are slack variable, surplus variable, capacity of alternative *i*, per unit cost of having u_i, per unit cost of having v_i, respectively.

$$u_i, v_i, b_i, g_i, h_i \ge 0 \text{ for } i = 1,...,m \quad (1.5)$$

$$g, h \ge 0 \quad (1.6)$$

In order to easily explain our algorithm, we have two assumptions that are following.

$$b_i \le b_{i+1}, \text{for } i = 1,...,m-1 \quad (1.7)$$

and

$$\frac{C_j}{a_j} \le \frac{C_{j+1}}{a_{j+1}}, \text{ for } j = 1,...,n-1. \quad (1.8)$$

The basic problem statement of the previous model can be described as follows. Given C_j, a_j, t_j, for $j = 1, 2, ..., n$ and g_i, h_i, b_i, for $i = 1, 2, ..., m$, the decision problem is to search for x_j, for $j = 1, 2, ..., n$ and u_i, v_i, for $i = 1, 2, ..., m$, so that the constraints (1.2) are satisfied in order to minimize the objective function (1.1).

Literature reviews: Dantzig[1] founded the general concept of linear programming (LP) including introducing LP with uncertain parameters referred to as stochastic linear programming (SLP). SLP was described by Birge and Louveaux[2]. Two basic approaches for solving SLP are chance constraint programming and two-stage modeling as presented in Wagner[3]. In general, theoretical foundations of stochastic linear and nonlinear programming are summarized by Wets[4]. The two-stage approach for SLP is quite popular among SLP researchers. One of the key research questions in this area is to develop an efficient algorithm to approximate the whole uncertainty population by a discrete set of samples leading to a deterministic equivalent LP model with manageable size and structure. Infanger[5] proposed an important sampling approach to find such

Corresponding Author: S. Witchakul, Operations Research and Management Science Units, Department of Industrial Engineering, Faculty of Engineering, Kasetsart University, Bangkok 10900, Thailand

representation and to apply Bender decomposition technique with statistical confidence intervals to solve the resulting dual angular model. In general, deterministic equivalent formulations of two-stage SLPs lead to a very large LP model with impractical sizes to be handled by a usual personal computer. Recently, research in parallel processing on large scale optimization has been received much attention as found in Benyoub and Daoudi[6] for the general constrained problem and in Meng, Tan and Zhao[7], for the SLP cases with extensions to non-linearity.

State-of-the-art LP solvers such as CPLEX, GLPK, LINGO SOPLEX and MATLAB can handle large and sparse problems. CPLEX[8] employs are primal simplex, dual simplex, barrier and network optimizers for problems with an extractable network structure. GLPK[9] implements the two-phase revised simplex method and primal-dual interior point method. LINGO[10] uses primal simplex solver, dual simplex solver and barrier solver (i.e., interior point algorithm). SOPLEX[11] is an object-oriented implementation of the primal and dual simplex algorithms. MATLAB[12] uses simplex solver and primal-dual interior point method. Moreover, there exist other LP solvers based on simplex method (i.e., CLP, lp_solve) and based on interior point method (i.e., PCx, BPMPD, HOPOM). Vanderbei[13] compared the performance of interior point method with the simplex method and concluded that simplex method is generally faster than interior point method for a small to medium scale problem but interior point method tends to be superior for large scale problem. Anderson[14] proposed a modified Schur-complement approach for handling dense columns in interior point method.

The studied problem is a class of one-stage linear programming problem in which allocation of the decision variable x_j is made to meet random RHS with a known distribution. The objective function in this problem may be a total cost. It can be divided into two parts. First part is the cost of selecting the decision variable x_j and second part is the penalty cost (i.e. shortage, storage cost). This kind of problem is a deterministic equivalent formulation of a stochastic linear programming problem with simple recourse, trivial first stage constraints and one constraint in the recourse problem. Since SLP problem with simple recourse has dual decomposition structure, dual decomposition method is another procedure for solving it as presented in Kall and Wallace[15]. In the next section, a new approach for solving the studied problem will be proposed with an optimality proof of validity.

MATERIALS AND METHODS

As stated previously, the other LP solvers can handle large and sparse problems. However, as the constraint matrix of the studied problem is dense, those LP solvers may not appropriate for large and dense problems. However, the proposed method can solve the studied problem efficiently.

The studied problem can be written as follows:

$$\underset{y,u,v}{Min}\, Z = \sum_{j=1}^{n} C'_j y_j + \sum_{i=1}^{m} (g_i u_i + h_i v_i) \tag{2.1}$$

$$S.T. \sum_{j=1}^{n} y_j + u_i - v_i = b_i \tag{2.2}$$

$$0 \le y_j \le t'_j \tag{2.3}$$

where

$$y_j = a_j x_j \tag{2.4}$$

$$C'_j = \frac{C_j}{a_j} \tag{2.5}$$

$$t'_j = a_j t_j \tag{2.6}$$

Objective function can be separated into two parts as follows.

$$\underset{\theta}{Min}\, Z = H_1(\theta) + H_2(\theta) \tag{2.7}$$

where

$$H_1(\theta) = \sum_{j=1}^{n} C'_j y_j \tag{2.8}$$

$$H_2(\theta) = \sum_{i=1}^{m} (g_i u_i + h_i v_i) \tag{2.9}$$

Let

$$\sum_{j=1}^{n} y_j = \theta \tag{2.10}$$

Let $SH1_{(t'_{j-1}<\theta<t'_j)}$ = slope of graph $H_1(\theta)$, $t'_{j-1} < \theta < t'_j$

$$SH1_{(t'_{j-1}\le\theta\le t'_j)} = C'_j \tag{2.11}$$

Since $C'_j \le C'_{j+1}$, for $j = 1,...n-1$, $H_1(\theta)$ is piecewise-linear convex function.

Let $SH2_{(b_{i-1}<\theta<b_i)}$ = slope of graph $H_2(\theta)$, $b_{i-1} < \theta < b_i$

For $0 < \theta < b_1$,

$u_i = b_i - \theta$, $i = 1,...,m$

$v_i = 0$, $i = 1,...,m$

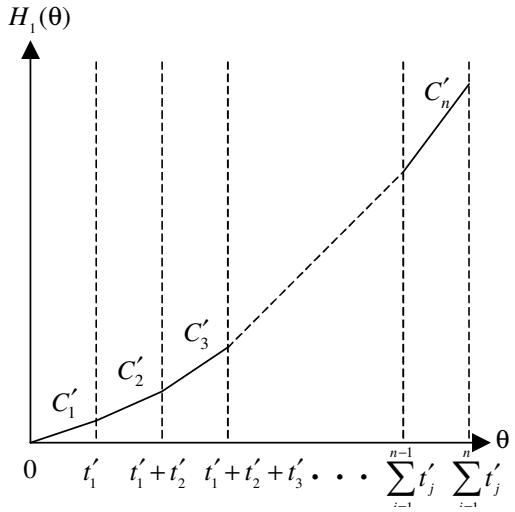

Fig. 1: Graph of $H_1(\theta)$

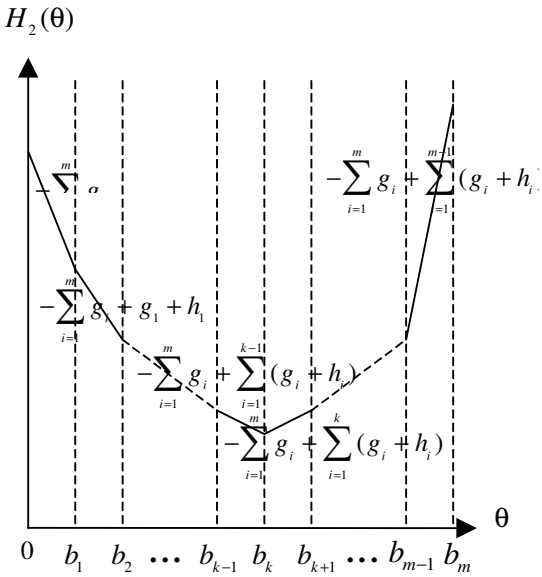

Fig. 2: Graph of $H_2(\theta)$

$$H_2(\theta) = \sum_{i=1}^{m} g_i(b_i - \theta) \tag{2.12}$$

$$SH2_{(0<\theta<b_1)} = -\sum_{i=1}^{m} g_i \tag{2.13}$$

For $b_1 < \theta < b_2$,

$v_1 = \theta - b_1$

$u_i = b_i - \theta \quad , i = 2,...,m$

$v_i = 0 \quad , i = 2,...,m$

$u_1 = 0$

$$H_2(\theta) = \sum_{i=2}^{m} g_i(b_i - \theta) + h_1(\theta - b_1)$$

$$= \sum_{i=1}^{m} g_i(b_i - \theta) + g_1(\theta - b_1) + h_1(\theta - b_1) \tag{2.14}$$

$$SH2_{(b_1<\theta<b_2)} = -\sum_{i=1}^{m} g_i + g_1 + h_1 \tag{2.15}$$

For $b_{k-1} < \theta < b_k$,

$v_i = \theta - b_i \quad , i = 1,...,k-1$

$u_i = b_i - \theta \quad , i = k,...,m$

$v_i = 0 \quad , i = k,...,m$

$u_i = 0 \quad , i = 1,...,k-1$

$$H_2(\theta) = \sum_{i=k}^{m} g_i(b_i - \theta) + \sum_{i=1}^{k-1} h_i(\theta - b_i)$$

$$= \sum_{i=1}^{m} g_i(b_i - \theta) + \sum_{i=1}^{k-1} g_i(\theta - b_i) + \sum_{i=1}^{k-1} h_i(\theta - b_i) \tag{2.16}$$

$$SH2_{(b_{k-1}<\theta<b_k)} = -\sum_{i=1}^{m} g_i + \sum_{i=1}^{k-1}(g_i + h_i) \tag{2.17}$$

For $b_k < \theta < b_{k+1}$,

$v_i = \theta - b_i \quad , i = 1,...,k$

$u_i = b_i - \theta \quad , i = k+1,...,m$

$v_i = 0 \quad , i = k+1,...,m$

$u_i = 0 \quad , i = 1,...,k$

$$H_2(\theta) = \sum_{i=k+1}^{m} g_i(b_i - \theta) + \sum_{i=1}^{k} h_i(\theta - b_i)$$

$$= \sum_{i=1}^{m} g_i(b_i - \theta) + \sum_{i=1}^{k} g_i(\theta - b_i) + \sum_{i=1}^{k} h_i(\theta - b_i) \tag{2.18}$$

$$SH2_{(b_k<\theta<b_{k+1})} = -\sum_{i=1}^{m} g_i + \sum_{i=1}^{k}(g_i + h_i) \tag{2.19}$$

Since $SH2_{(b_{i-1}\le\theta\le b_i)} \le SH2_{(b_i\le\theta\le b_{i+1})}$, for $i = 1,...m-1$, $H_2(\theta)$ is piecewise convex function.

Since $H_1(\theta)$ is a piecewise-linear convex function and $H_2(\theta)$ is a piecewise convex function, $H_1(\theta)+H_2(\theta)$ is also a piecewise convex function.

Let SH_θ = slope of graph $H_1(\theta)+H_2(\theta)$

There are $m+n+1$ extreme points. Therefore, there are $m+n$ corresponding slopes. The optimal solution can be found easily by calculating SH_θ from left to right. If SH_θ is greater than zero, the optimal solution is the starting point of that slope.

In this study, the proposed method is developed for solving LP with single constraints and bounded variables and written in MATLAB software as M-file program and compared with a general purpose method using CPLEX interactive optimizer. For the proposed method, a developed algorithm can be generalized step by step as follows.

Step 1: Input *n*, *m*

$C_j, a_j, t_j, \quad , j = 1,2,...,n$

$b_i, g_i, h_i, \quad , i = 1,2,...,m$

Step 2: Calculate $C'_j = \dfrac{C_j}{a_j} \quad j = 1,2,\ldots,n$

and $t'_j = a_j t_j$

Step 3: Find all extreme points.

$$sumtp_d = \sum_{j=1}^{d} t'_j \quad , d = 1,...,n$$

$$ep_d = \sum_{j=1}^{d} t'_j \quad , d = 1,...,n$$

$$ep_d = b_i \quad , (d,i) = (n+1,1),...,(n+m,m)$$

$$ep_d = 0 \quad , d = n+m+1$$

Sort extreme points in the ascending order:

$eps = sort(ep)$

Step 4: $r = 1$

Step 5: $r = r+1$

Check that eps_r is equal to any element in *sumtp* or b

If eps_r is equal to any element in *sumtp*, then go to step 6

Otherwise go to step 7.

Step 6: Check that eps_r is equal to which element in *sumtp*.

Let $eps_r = sumtp_{kk}$

Find the first element from eps_{r+1} to eps_{n+m+1} that is also in b.

Let that element is equal to b_{iii}

$$SH_{(eps_{r-1}<\theta<eps_r)} = C'_{kk} - \sum_{i=1}^{m} g_i + \sum_{i=1}^{iii-1} (g_i + h_i)$$

And go to step 8.

Step 7: Check that eps_r is equal to which element in b.

Let $eps_r = b_{ii}$

Find the first element from eps_{r+1} to eps_{n+m+1} that is also in *sumtp*.

Let that element is equal to t_{kkk}

$$SH_{(eps_{r-1}<\theta<eps_r)} = C'_{kkk} - \sum_{i=1}^{m} g_i + \sum_{i=1}^{ii-1} (g_i + h_i)$$

And go to step 8.

Step 8: If $SH_{(eps_{r-1}<\theta<eps_r)} > 0$, then $\theta^* = eps_{r-1}$ and go to step 9.

Otherwise, go to step 5.

Step 9: Find k^*

$k^* = k$ such that $sumtp_{k-1} \le \theta^*$ and $sumtp_k > \theta^*$

And go to step 10.

Step 10: Calculate x_j

$$x_j = t_j, \quad j = 1,...,k^*-1$$

$$x_{k^*} = \frac{\theta^* - sumtp_{k^*-1}}{a_{k^*}}$$

$$x_j = 0, \quad j = k^*+1,...,n$$

And go to step 11.

Step 11: Calculate u_i and v_i

$u_i = 0$ and $v_i = \theta^* - b_i$ for i such that $b_i < \theta^*$

$u_i = b_i - \theta^*$ and $v_i = 0$ for i such that $b_i \ge \theta^*$

And go to step 12.

Step 12: Calculate $Z = \sum_{j=1}^{n} C_j x_j + \sum_{i=1}^{m} (g_i u_i + h_i v_i)$

Experimentally for the general purpose method using CPLEX, the simplex based is selected for solving the studied problem. For the proposed algorithm, it is written as M-file format in MATLAB software. Then, an experiment is conducted by varying *m* and *n* and the elapsed time (excluding parameter generating) and solutions obtained are collected and compared. All computations are tested on PC notebook with Pentium M, 1.6 Ghz and 512 MB RAM. C_j, a_j, t_j, for

$j = 1, 2, ..., n$, are generated with uniform[0,10]. b_i for $i = 1, 2, ..., m$, are generated with uniform$[0, \sum_{j=1}^{n} a_j t_j]$. g_i, h_i for $i = 1, 2, ..., m$, are generated with uniform[0,1].

RESULTS

The results from the experiment can be summarized as shown in Table 1 and 2.

Table 1: The elapsed time (excluding parameter generating)(sec) of general purpose method using CPLEX when *m* and *n* are varied

n	*m*								
	100	250	500	750	1000	2500	5000	7500	10000
100	1.71	1.75	1.93	2.20	2.62	10.40	41.18	95.52	162.98
250	1.72	1.88	2.31	3.40	4.91	24.31	88.47	200.51	354.14
500	1.77	2.23	3.50	5.70	8.77	43.99	168.38	372.09	663.15
750	1.85	2.43	4.80	8.47	13.03	65.92	247.41	552.47	1181.97
1000	2.15	3.06	6.08	10.67	16.54	86.19	328.44	815.26	1610.47
2500	3.31	6.37	15.05	27.41	43.14	224.48	1397.91	6468.05	N/A
5000	6.34	15.02	35.52	65.17	100.35	1068.04	N/A	N/A	N/A
7500	10.27	27.61	65.22	114.27	189.86	5407.50	N/A	N/A	N/A
10000	16.46	44.24	105.45	193.24	322.96	N/A	N/A	N/A	N/A

N/A means that CPLEX cannot solve the studied problem of size (n, m).

Table 2: The elapsed time (excluding parameter generating)(sec) of the proposed method when *m* and *n* are varied

n	*m*								
	100	250	500	750	1000	2500	5000	7500	10000
100	0.00	0.01	0.01	0.02	0.02	0.11	0.37	0.80	1.42
250	0.01	0.01	0.02	0.02	0.03	0.12	0.43	0.88	1.49
500	0.01	0.02	0.03	0.04	0.05	0.15	0.50	0.94	1.58
750	0.02	0.03	0.04	0.05	0.06	0.18	0.54	1.01	1.67
1000	0.03	0.04	0.05	0.07	0.08	0.21	0.61	1.11	1.77
2500	0.14	0.15	0.17	0.21	0.23	0.45	1.19	1.82	2.50
5000	0.48	0.51	0.56	0.59	0.64	1.04	1.67	2.72	3.78
7500	1.01	1.07	1.11	1.19	1.25	1.71	2.58	3.67	4.96
10000	1.73	1.81	1.95	2.03	2.12	2.82	3.82	5.10	6.52

With both general purpose method using CPLEX and the proposed method, for all *n*, elapsed time will increase if *m* increases. Additionally, the proposed method has an average shorter elapsed time than general purpose methods using CPLEX and significantly shorter when *m* and *n* are large.

CONCLUSION

A fast special purpose method for solving LP with single constraints and bounded variables has been developed and tested. The results indicated that the proposed method is much faster than general purpose methods in the case of large-scale problems. Moreover, the proposed method can be a real time efficient algorithm for bound computing when additional integer decision variable constraints are required. Also, the proposed approach can be a foundation for extensive studies in case of multiple constraints.

REFERENCES

1. Dantzig, G.B., 1963. Linear Programming and Extensions, Princeton University Press, Princeton, New Jersey.
2. Birge, J.R. and F. Louveaux, 1997. Introduction to Stochastic Programming. Springer, New York.
3. Wagner, H.M., 1975. Principles of Operations Research. 2nd Edn., Prentice Hall, New Jersey.
4. Wets, R.J.B., 1989. Stochastic Programming, Chapter VIII, Handbooks in Operations Research and Management Science Vol. 1, Optimization, North-Holland.
5. Infanger, G., 1994. Planning under Uncertainty: Solving large scale stochastic linear programs. The Scientific Press Series, Boyd & Fraser.
6. Benyoub, A. and E.M. Daoudi, 2003. Study of the parallel continuous global optimization based on interval arithmetic. Intl. J. Comput. Eng. Sci., 4: 99-108.
7. Meng, G., R.C.E. Tan and G. Zhao, 2004. a superlinearly convergent algorithm for large scale multi-stage stochastic nonlinear programming. Intl. J. Comput. Eng. Sci., 5: 327-344.
8. ILOG. CPLEX 8.1. 2002. Reference Manual.
9. GLPK Version 4.9. 2003. Reference Manual.
10. LINGO 9.0. 2005. User's Manual.
11. Wunderling, R., A. Bley, T. Achteberg, T. Koch, B. Hiller, S. Orlowski and A. Tunchscherer, 2006. SOPLEX 1.3.0
12. The Math Works, Inc. 2000. MATLAB Manual and Instruction Sets.
13. Vanderbei, R.J., 2001. Linear Programming Foundations and Extensions. 2nd Edi, Princeton, New Jersey.
14. Anderson, K.D., 1996. A modified schur-complement method for handling dense columns in interior-point methods for linear programming. ACM Trans. Math. Software, 22: 348-356.
15. Kall, P. and S.W. Wallace, 1994. Stochastic Programming. John Wiley, New York.