Linear Programming Relax-PSO Hybrid Bound Algorithm for a Class of Nonlinear Integer Programming Problems*

Yuelin Gao[1,2], Chengxian Xu[2], and Jimin Li[1]

[1] Department of Information and Computation Science, Northwest Second National College, Yin Chuan 750021, China
gaoyuelin@263.net
[2] School of Finance and Economics, Xi'an Jiaotong University, Xi'an Jiaotong University, Xi'an 710049, China
mxxu@mail.xjtu.edu.cn

Abstract. The paper researches a class of nonlinear integer programming problems the objective function of which is the sum of the products of some nonnegative linear functions in the given rectangle and the constraint functions of which are all linear as well as strategy variables of which are all integer ones. We give a linear programming relax-PSO hybrid bound algorithm for solving the problem. The lower bound of the optimal value of the problem is determined by solving a linear programming relax which is obtained through equally converting the objective function into the exponential-logarithmic composite function and linearly lower approximating each exponential function and each logarithmic function over the rectangles. The upper bound of the optimal value and the feasible solution of it are found and renewed with particle swarm optimization (PSO). It is shown by the numerical results that the linear programming relax-PSO hybrid bound algorithm is better than the branch-and-bound algorithm in the computational scale and the computational time and the computational precision and overcomes the convergent difficulty of PSO.

1 Introduction

Integer programming problems are encountered in a variety of areas, such as capital budgeting [6], computer-aided layout design [7], portfolio selection [8], site selection for electric message systems [9] and shared fixed costs [10] etc. The methods for solving the Integer programming problems have mainly method of dynamic programming, branch and bound method, the method of computational intelligence [1,2,3,11,12, 13].

* The work is supported by the Foundations of Post-doctoral Science in China (grants 2006041001) and National Natural Science in Ningxia (2006), and by the Science Research Projects of National Committee in China and the Science Research Project of Ningxia's Colleges and Universities in 2005.

Y. Wang, Y. Cheung, and H. Liu (Eds.): CIS 2006, LNAI 4456, pp. 29–35, 2007.

In the paper, we consider a class of nonlinear integer programming problems below:

$$\begin{cases} \min \phi(x) = \sum_{i=1}^{t} \prod_{j=1}^{p} (c_{ij}^T x + d_{ij}) \\ \text{s.t.} \quad Ax \le b, \\ \qquad x \in Z^n \cap [l, u]. \end{cases} \tag{1}$$

where $t, p_i \in Z_+ - \{0\}, \prod_{i=1}^{t} p_i \ge 2, p = \prod_{i=1}^{t} p_i; d_{ij} \in R_+, c_{ij} = (c_{ij1}, c_{ij2}, \cdots, c_{ijn})^T \in R_+^n$, in $R = [l, u], A = (a_{ij})_{m \times n} \in R^{m \times n}, b \in R$. Z is noted as the set which consist of all the integers, $l, u \in Z^n$.

We will give a new linear programming relax-PSO hybrid bound algorithm of the problem (1) by making use of branch-and-bound method (BBA) and PSO. It will be shown by the numerical results that the algorithm to be proposed is better than BBA in the computational scale and the computational time and the computational precisionand that it overcomes the convergent difficulty of PSO. In Section 2, we give a linear relaxed approximation so as to determine a lower bound of the optimal value of the problem (1). In Section 3, we give a PSO algorithm based on the penalty function of the problem (1) so as to find and renew the feasible solutions and the upper bound of the problem (1). In Section 4, the numerical computation is done so as to test the property of the proposed algorithm. Section 5 is conclusions.

2 Linear Programming Relaxed Approximation

Firstly, we convert equally the problem (1) into the non-linear integer programming problem below:

$$\begin{cases} \min \phi = \sum_{i=1}^{t} exp(\sum_{j=1}^{p_i} log(c_{ij}^T x + d_{ij})) \\ \text{s.t.} \quad Ax \le b, \\ \qquad x \in Z^n \cap [l, u]. \end{cases} \tag{2}$$

Secondly,the problem(2) is continuously relaxed to the problem below:

$$\begin{cases} \min \phi = \sum_{i=1}^{t} exp(\sum_{j=1}^{p_i} log(\sum_{k=1}^{n} c_{ijk} x_k + d_{ij})) \\ \text{s.t.} \quad Ax \le b, \\ \qquad x \in [l, u]. \end{cases} \tag{3}$$

For $i = 1, 2, \cdots, t, j = 1, 2, \cdots, p_i$, let $\phi_{ij} = \log y_{ij}$,where $y_{ij} = c_{ij}^T x + d_{ij} = \sum_{k=1}^{n} c_{ij} x_k + d_{ij}$. From $x \in [l, u], y_{ij} \in [l_{ij}, u_{ij}]$,where

$$l_{ij} = \sum_{k=1}^{n} min\{c_{ijk} l_k, c_{ijk} u_k\} + d_{ij}, \tag{4a}$$

$$l_{ij} = \sum_{k=1}^{n} max\{c_{ijk}l_k, c_{ijk}u_k\} + d_{ij}, \tag{4b}$$

Because $log(y_{ij})$ is a strictly increase concave function in $(0, +\infty)$, it can be seen that the convex envelope of ϕ_{ij} over $[l_{ij}, u_{ij}]$ is a line which is through two points $(l_{ij}, log(l_{ij}))$,$(u_{ij}, log(u_{ij}))$, i.e. the line is the best lower approximate linear function of ϕ_{ij} in $[l_{ij}, u_{ij}]$:

$$z_{ij} = \frac{log(u_{ij}) - log(l_{ij})}{u_{ij} - l_{ij}}(y_{ij} - l_{ij}) + log(l_{ij}) = \overline{c}_{ij}y_{ij} + \overline{d}_{ij}. \tag{5}$$

where

$$\overline{c}_{ij} = \frac{log(u_{ij}) - log(l_{ij})}{u_{ij} - l_{ij}}, \tag{6}$$

$$\overline{d}_{ij} = \frac{u_{ij}log(l_{ij}) - l_{ij}log(u_{ij})}{u_{ij} - l_{ij}}. \tag{7}$$

Let $\overline{l}_i = \sum_{j=1}^{p_i} log(l_{ij}), \overline{u}_i = \sum_{j=1}^{p_i} log(u_{ij}), \overline{z}_i = \sum_{j=1}^{p_i} log(z_{ij})$,$\psi_i = exp(z_i)$. Because $exp(z_i)$ is a strictly increasing convex function in $(-\infty, +\infty)$, so the best lower approximate linear function of ψ_i on z_i in $[\overline{l}_i, \overline{u}_i]$ is a line through two points $(\overline{l}_i, exp(\overline{l}_i))$ and $(\overline{u}_i, exp(\overline{u}_i))$ and tangents with $\psi_i = exp(z_i)$, i.e. it is the linear function $ll_i(z_i) = \overline{c}_i z_i + \overline{d}_i$,where

$$\overline{c}_i = \frac{exp(\overline{u}_i) - exp(\overline{l}_i)}{\overline{u}_i - \overline{l}_i}, \tag{8}$$

$$\overline{d}_i = \frac{exp(\overline{u}_i) - exp(\overline{l}_i)}{\overline{u}_i - \overline{l}_i}(1 - log(\frac{exp(\overline{u}_i) - exp(\overline{l}_i)}{\overline{u}_i - \overline{l}_i},)). \tag{9}$$

So, we obtain a lower approximate linear function of ψ on $z = (z_1, z_1, \cdots, z_t)$ over $[\overline{l}_i, \overline{u}_i]$ where $\overline{l} = (\overline{l}_1, \overline{l}_2, \cdots, \overline{l}_t,)$ and $\overline{u} = (\overline{u}_1, \overline{u}_2, \cdots, \overline{u}_t,)$:

$$\omega = \sum_{i=1}^{t} ll_i(z_i). \tag{10}$$

Thus, the linear programming relaxed approximation of the problem(1) is

$$\begin{cases} \min \omega = \sum_{i=1}^{t} ll_i(z_i) \\ \text{s.t. } Ax \le b, \\ \quad z_i = \sum_{j=1}^{p_i} z_{ij}, i = 1, 2, \cdots, t, \\ \quad z_{ij} = \overline{c}_{ij}y_{ij} + \overline{d}_{ij}, i = 1, 2, \cdots, t, j = 1, 2, ..., p_i, \\ \quad y_{ij} = c_{ij}^T y_{ij} + d_{ij}, i = 1, 2, \cdots, t, j = 1, 2, ..., p_i, \\ \quad x \in [l, u]. \end{cases} \tag{11}$$

Obviously, the optional value of the problem(11) is sure to be a lower bound of the problem(1).

3 A PSO Algorithm Based on The Penalty Function

The particle swarm optimization algorithm (PSO) is a kind of computational intelligent which is put forward by Kenney and Eberhart etc. in 1995 and has global optimization property but is not proven in convergence[11,12,13]. We only give a PSO algorithm based on the penalty function.

Firstly,we give a penalty function of the problem(1) below:

$$p(x) = \phi(x) + M(\sum_{i=1}^{m} \mid min\{0, b_i - \sum_{j=i}^{n} a_{ij}x_j\} \mid) \tag{12}$$

where the penalty coefficient $M > 0$ can be any number large enough.

N_c represents the biggest iteration of PSO, M_c represents the particle number in particle swarm, p_{sb} represents the best position by which a particle swarm has gone so far and p_{gb} represents the best position by which all the x_{gb} represents the best feasible position in the particle swarm at present. V_{max}^i represents the biggest velocity of a particle x_i.

The PSO algorithm based on the penalty function(IP-PSO) is described below:

Step1. Set $t = 1, M = 1000, N_c = 100$.Produce randomly a particle swarm in Scale M_c.The initial position of each particle x_i is $x_{ij}(0)(j = 1, 2, \cdots, n)$ and the initial velocity is $v_{ij}(j = 1, 2, \cdots, n)$, compute each particle's fitness and determine p_{sb} and p_{gb} and x_{gb}.

Step2. Set $t = t + 1$. For each particle from the next formula:

$$\begin{cases} v_{ij} = wv_{ij} + c_i r_i(p_{ij} - x_{ij}) + c_2 r_2(p_{gj} - x_{ij}) \\ x_{ij} = x_{ij} + v_{ij} \\ i = 1, 2, \cdots, M_c, j = 1, 2, \cdots, n. \end{cases} \tag{13}$$

where $w \in [0.2, 1.2]$ is inertia weight, $c_1 = 2, c_2 = 1.7$ are acceleration constants, r_1, r_2 are two random functions over [0,1].If $v_{ij} > V_{max}^i$ in (13), then $v_{ij} = V_{max}^i$. Renew p_{sb} and p_{gb} as well as x_{gb}.

Step3. If $t = N_c$, outcome the best particle $x_{opt} = x_{gb}$; else, go to Step2.

All the coefficients in the IP-PSO are determined through the numerical test in Section 5 and the IP-PSO can find better feasible solution and better upper bound of the problem(1).

4 Description of Linear Programming Relax-PSO Hybrid Bound Algorithm

In the section,we describe a linear programming relax-PSO hybrid bound algorithm (BB-PSO-HA). In the algorithm,branching procedure is simple integer rectangle two-partitioning one and lower bounding procedure needs solving the problem(11) in each sub-rectangle as well as upper bounding procedure needs the algorithm IP-PSO.

BB-PSO-HA

Step0.(Initialization) $k := 0, \Omega = \{R\}$. Solve the problem(12), and determine the lower bound LB of the problem(1). Use Algorithm IP-PSO to determine the best feasible solution x_{best} so far.

Step$k$.($k = 1, 2, \cdots$)

k1(termination) If $\Omega = \Phi$ or $\frac{UB-LB}{UB} < Eps$, then outcome $z_{opt}, Optv = UB$.

k2(Selection Rule) In Ω, find a rectangle R_k such that $LB(R_k)$.

k3(Branching Rule) Partition R_k into two sub-rectangle with rectangle simple two equally-partition technique,and reduce each sub-rectangle to make vertex point integer, and obtain two integer sub-rectangle R_{k1} and R_{k2}. Set $\Omega = (\Omega - R_k) \cup \{R_{k+1,1}, R_{k+1,2}\}$

k4(Lower Bounding) Solve the problem(11) in $R_{k+1,1}$ and $R_{k+1,2}$ respectively so as to renew LB.

k5(Upper Bounding) Solve the problem(1) in $R_{k+1,1}$ and $R_{k+1,2}$ respectively with IP-PSO to renew x_{best} and $UB = \phi(x_{best})$.

k6(deleting Rule) $\Omega = \Omega - \{R \in \Omega : LB(R) \geq UB\}$, $k = k + 1$, go to k_1.

5 Numerical Analysis

In the problem(1), let $t = 1, p_1 = n, c_{ij}^n x = c_i x_i$, then, we obtain the next example:

$$\begin{cases} \min \omega = \prod_{i=1}^{n} (c_i x_i + d_i) \\ \text{s.t.} \ \sum_{i=1}^{n} a_i x_i \leq b, \\ \quad x_i \in [1, 20], \\ \quad x_i \in Z, \\ \quad i = 1, 2, \cdots, n. \end{cases} \tag{14}$$

where $c_i \in [-20, 20], d_i \in [21, 52], a_i \in [0, 50], b = 1.2 sum(a) = \sum_{i=1}^{n} a_i$.

The procedures of BBA and BB-PSO-HA are compiled with Matlab7.0.1 in personal computer DELL-P4-Intel1865-512MB. We produce randomly twenty examples for the problems (14) in n=60,100,150,200,300,500,800,1000,1500,2000. and solve the examples with BBA and BB-PSO-HA respectively. The results of the numerical computation are seen at Table1-Table2 where Ex1=$Eps_1 = 10^{-4}$ and Ex2=$Eps_2 = 10^{-5}$. "Iteration" and "Cputime" are noted as the iteration times and computational time respectively. "Avg, Max, Min" are noted as the iteration times and computational time of "average, maximum, minimum" respectively.

It is shown by the numerical results from Table 1 and Table 2 that BB-PSO-HA is better than BBA in computational scale, computational time and computational precision.

Table 1.

Ex1	BBA					
	Iteration			Cputime(Seconds)		
n	Avg	Max	Min	Avg	Max	MIN
60	7000	10000	1	472.2	1035.8	0.09
100	7580	10000	1	674.9	1331.5	0.07
150	7211	10000	1	844.7	2574.9	0.15
200	6776	10000	1	840.5	3206.8	0.29
300	8366	10000	1	1793.9	5450	0.2
500	6298	10000	3	2405.8	8278.6	0.64
800	5288	10000	2	4491.6	8611	0.98
1000	4357	10000	432	4143.4	22135	181
1500	*	*	*	*	*	*
*	*	*	*	*	*	*

Table 2.

Ex2	BBA-PSO					
	Iteration			Cputime(Seconds)		
n	Avg	Max	Min	Avg	Max	MIN
60	25	166	1	274.9	1814.8	9.8
100	8	75	1	142.8	1488.4	17
150	16	164	1	449.3	4546	30
180	11	175	1	171.9	2379.8	30
200	18	160	1	635.7	5797.7	32.5
300	14	178	1	451.2	3947.8	49.5
500	15	144	1	1017.3	9394.1	65.3
800	4	43	1	594.2	6732.6	137.5
1000	18	256	1	3493.1	50020	133.2
1500	2	5	1	297.8	1003.2	199.5
2000	5	50	1	3057.3	41250	271.2

6 Conclusion

We give a new linear programming relax-PSO hybrid bound algorithm for solving a class of nonlinear integer programming problems. The lower bound of the optimal value of the problem is determined by solving a linear programming relax which is obtained through equally converting the objective function into the exponential-logarithmic composite function and lower approximating each exponential function and each logarithmic function with the best linear function. The upper bound of the optimal value and the feasible solution of it are found and renewed with PSO.

It is shown by the numerical results that the linear programming relax-PSO hybrid bound algorithm is better than BBA in computational scale, computational time and computational precision and overcomes the convergent difficulty of PSO.

References

1. Nemhauser, G.L., Wolsey, L.A.: Integer and Combinatorial Optimization. John Wiley and sons, New York (1988)
2. Kuno, T.: Solving a class of multiplicative programs with 0-1 knapsack constraints. Journal of Optimization Theory and Applications 103, 121–125 (1999)
3. Barrientos, O., Correa, R., Reyes, P., Valdebenito, A.: A brand and bound method for solving integer separable concave problems. Computational Optimization and Applications 26, 155–171 (2003)
4. Horst, R., Tuy, H.: Global optimization, deterministic approaches. Springer, Heidelberg (1996)
5. Gao, Y.L., Xu, C.X, Wang, Y.J., Zhang, L.S.: A new two-level linear relaxed bound method for geometric programming problem. Applied Mathematics and Computation 164, 117–131 (2005)
6. Laughunn, D.J.: Quadratic binary programming with applications to capital-budgeting problem. Operations Research 14, 454–461 (1970)
7. Krarup, J., Pruzan, P.M.: Computer-aided layout design. Mathematical Programming Study 9, 75–94 (1978)
8. Markovitz, H.M.: Portfolio selection. Wily, New York (1978)
9. Witzgall, C.: Mathematical method of site selection for Electric Message Systems(EMS), NBS Internet Report (1975)
10. Rhys, J.: A selection problem of shared fixed costs on network flow. Management Science 17, 200–207 (1970)
11. Eberhart, R.C., Shi, Y.H.: Particle swarm optimization: development, applications and resources. In: Proceedings of the IEEE International Conference on Evolutionary Computation, pp. 81–86 (2002)
12. Laskari, E.C., Parsopoulos, K.E., Vrahatis, M.N.: Particle swarm optimization for integer programming. In: Proceedings of the IEEE International Conference on Evolutionary Computation, pp. 1582–1587 (1978)
13. Eberhart, R.C., Shi, Y.H.: Comparison between genetic algorithms and particle swarm optimization: development, applications and resources, Evolutionary Programming, pp. 611–615 (1998)