Applied Mathematical Sciences, Vol. 1, 2007, no. 11, 517 - 525

A Branch and Bound-PSO Hybrid Algorithm for Solving Integer Separable Concave Programming Problems[1]

Yuelin Gao[2,3], Zihui Ren[4] and Chengxian Xu[2]

[2] School of Finance and Economics, Xi'an Jiaotong University
Xi'an,710049, China
mxxu@mail.xjtu.edu.cn

[3] Department of Information and Computation Science
Northwest Second National College
Yin Chuan 750021, China
gaoyuelin@263.net

[4] School of Mathematics and Computer
Ningxia University, Yin Chuan 750021, China
renzihui@etang.com

Abstract

A branch and bound-PSO hybrid algorithm for solving integer separable concave programming problems is proposed, in which the lower bound of the optimal value was determined by solving linear programming relax and the upper bound of the optimal value and the best feasible solution at present were found and renewed with particle swarm optimization (PSO). It is shown by the numerical results that the branch and bound-PSO hybrid algorithm is better than the branch and bound algorithm in the computational scale and the computational time and the computational precision and overcome the convergent difficulty of particle swarm optimization (PSO).

Keywords: Integer programming; separable concave programming; branch and bound method; particle swarm optimization; linear programming relax

[1]The work is supported by the Foundations of National Natural Science in Ningxia (grants NZ0676) and Post Doctor of China(20060401001), and by the Science Research Projects of Ministry of Education of China(06JA630056), and Ningxia's Colleges and Universities in 2005.

1 Introduction

Consider the integer separable concave programming problem below:

$$\text{(ISCCP)} \quad \begin{cases} \min \phi(x) = \sum_{j=1}^{n} f_j(x_j) \\ \text{s.t. } x \in D = \{x \in R^n | Ax \leq b\}, \\ \quad x \in Z^n. \end{cases}$$

where each $f_j(x_j)$ is a concave function over R, Z^n is a set which consists of all n-dimension integer points; $A = (a_{ij})_{n\times m} \in R^{m\times n}$, $b = (b_1, b_2, \cdots, b_m)^T \in R^m$, the feasible field D is a bounded convex polyhedron.

Problem (ISCCP) are encountered in a variety of areas, such as capital budgeting [2], computer-aided layout design [5], portfolio selection [3], site selection for electric message systems [6] and shared fixed costs [7], etc. The methods for solving the problem (ISCCP) has mainly method of dynamic programming, branch and bound method, the method of computational intelligence [1,4,8-10].

In the paper, we present a branch and bound-PSO hybrid algorithm for solving the problem (ISCCP), in which the lower bound of the optimal value was determined by solving linear programming relax and the upper bound of the optimal value and the best feasible solution at present were found and renewed with particle swarm optimization (PSO). It is shown by the numerical results that the branch and bound-PSO hybrid algorithm is better than the branch and bound algorithm (BBA) and overcome the convergent difficulty of particle swarm optimization (PSO).

Section 2 gives a good linear programming relax of the problem (ISCCP). Section 3 gives a particle swarm optimization algorithm based on the penalty function for solving the problem (ISCCP) to find and renew the best feasible solution of the problem (ISCCP) and the upper bound of the optimal value of it at present. Section 4 describes a branch and bound-PSO hybrid algorithm. Section 5 gives several numerical examples to show that the proposed algorithm is effective.

2 A linear programming relaxed approximation

Firstly, we give the continuous relaxed programming of the problem (ISCCP)

$$\begin{cases} \min \phi(x) = \sum_{j=1}^{n} f_j(x_j) \\ \text{s.t. } x \in D = \{x \in R^n | Ax \leq b\}, \\ \quad x \in [l, u]. \end{cases} \tag{1}$$

where $[l, u]$ is a rectangle over R^n which contains D. Because D is a bounded convex polyhedron, so we can find a rectangle $[l, u] \subseteq R^n$ such that $D \subseteq [l, u]$ by solving linear programming.

Because each $f_j(x_j)$ is a concave function over R, so its convex envelope over $[l_j, u_j]$ is a line through two points $(l_j, f_j(l_j))$ and $(u_j, f_j(u_j))$, i.e.

$$lb_j(x_j) = \frac{(f_j(u_j) - f_j(l_j))}{u_j - l_j}(x - l_j) + f_j(l_j), \tag{2}$$

thus we can obtain the best linear programming relaxed approximation of the problem (ISCCP) below:

$$\begin{cases} \min \phi(x) = \sum_{j=1}^{n} lb_j(x_j) \\ \text{s.t. } x \in D = \{x \in R^n | Ax \leq b\}, \\ \qquad x \in [l, u]. \end{cases} \tag{3}$$

The optimal value of the problem (3) is a lower bound of the optimal value of the problem (ISCCP).

3 The particle swarm optimization algorithm of integer programming

The particle swarm optimization algorithm (PSO)[8, 9, 10] is a kind of computational intelligent technology which is put forward by Kennedy and Eberhart el in 1995 and has global optimal property, but PSO's convergence is not proofed.

Below is a penalty function of the problem (ISCCP):

$$p(x) = \sum_{j=1}^{n} f_j(x_j) + M(\sum_{i=1}^{m} |\min\{0, b_k - \sum_{j=1}^{n} a_{ij}x_j\}|) \tag{4}$$

where $M > 0$ is a penalty parameter.

We now give a PSO algorithm based on the penalty function (4) for solving the problem (ISCCP). Nc is noted as the iteration times when the PSO algorithm stops, Mc is noted as the number of the particles in a particle swarm, p_{sb} is noted as the best position by which a particle has gone so far and p_{gb} is noted as the best position by which all the particles in the particle swarm has gone so far as well as x_{gb} is noted as the best feasible position by which all the particles in the particle swarm has gone so far. V_{max}^i is noted as the biggest velocity of the particle $x_i (i = 1, 2, \cdots, Mc)$.

PSO algorithm based on the penalty function

Step1. Set $t = 1, M = 1000, Nc = 100, Mc = 60$. Produce randomly a particle swarm, the initial position of each particle $x_i(i = 1, 2, \cdots, Mc)$ in which is $x_{ij}(0), j = 1, 2, \cdots, n$ and the initial velocities $v_{ij}(0), j = 1, 2, \cdots, n$. Compute the fitness of each particle. Determine p_{sb} of each particle, and p_{gb} and x_{gb} of all the particles in the particle swarm.

Step2. Set $t = t + 1, M = M \times t$. For each particle $x_i(i = 1, 2, \cdots, Mc)$, from the iteration formula below:

$$\begin{cases} v_{ij} = wv_{ij} + c_1 rand_1(p_{ij} - x_{ij}) + c_2 rand_2(p_{gj} - x_{ij}) \\ x_{ij} = x_{ij} + v_{ij}, j = 1, 2, \cdots, n. \end{cases} \tag{5}$$

where $w \in [0.2, 1.2]$ is inertia weight, $c_1 = 2$ and $c_2 = 1.7$ are acceleration constants, $rand_1$ and $rand_2$ are two random functions over $[0, 1]$. If $v_{ij} > V^i_{max}$ in (5), then $v_{ij} = V^i_{max}(j = 1, 2, \cdots, n)$.

Step3. For each particle, compute p_{sb}. For the particle swarm, compute p_{gb} and x_{gb}.

Step4. If $t = Nc$, then outcome $x_{opt} = x_{gb}$, stop; else go to step2.

All the coefficients in the PSO algorithm are determined through the numerical test in Section 5 and the PSO algorithm can find better feasible solution and better upper bound.

4 Description of a branch and bound-PSO hybrid algorithm

In this section, we describe a branch and bound-PSO hybrid algorithm for solving the problem (ISCCP), in which the branching procedure is usual integer rectangle two-partitioning one and the bounding lower procedure is by solving the problem (3) as well as in the bounding upper procedure the PSO algorithm in Section 3 is used. Denote that $R = [l, u]$.

We now describe the branch and bound-PSO hybrid algorithm (BB-PSO-HA).

Step0. (Initialization) Set $k = 0, \Omega = \{R\}$. Determine the best lower LB at present by solving the problem (3) and the best upper bound UB and the best feasible solution xx_{best} of the problem (ISCCP) at present with PSO algorithm.

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

k1.(Termination) If $\Omega = \Phi$ or $\frac{UB-LB}{UB} < eps$, then outcome xx_{best} and $OPT = UB$.

k2.(Selection Rule) Find a subrectangle R_k in Ω such that $LB(R_k) = LB$.

k3.(Branching) Partition R_k into two subrectangles $R_{k+1,1}$ $R_{k+1,2}$, and each subrectangle is reduced into integer rectangle each vertex point of which is integer point. The obtained two integer subrectangles are noted as $R_{k+1,1}$ and $R_{k+1,2}$ too. Set $\Omega = (\Omega - R_k) \bigcup \{R_{k+1,1}, R_{k+1,2}\}$.

k4.(Lower Bounding) Solve the problem (3) over $R_{k+1,1}$ and over $R_{k+1,2}$ respectively to obtain new lower LB.

k5.(Upper Bounding) Renew UB and xx_{best} with the PSO algorithm in Section 3.

k6. (Deleting Rule) Set $\Omega = \Omega - \{R \in \Omega : LB(R) \geq UB\}$ and $k = k + 1$. Go to k1.

5 Numerical Computation

We solve the three problems (6)-(8) below with BBA and BB-PSO-HA:

$$\begin{cases} \min \sum_{i=1}^{n} (c_i x_i - d_i x_i^2) \\ \text{s.t. } \sum_{i=1}^{n} a_i x_i \leq b, \\ \quad x \in [-2, 4], x \in Z, \\ \quad i = 1, 2, ..., n. \end{cases} \tag{6}$$

where $c_i \in [10, 20], d_i \in [10, 20], a_i \in [0, 50], b = 3.8sum(a) = 3.8 \sum_{i=1}^{n} a_i$.

$$\begin{cases} \min \sum_{i=1}^{n} log(c_i x_i + d_i) \\ \text{s.t.} \sum_{i=1}^{n} a_i x_i \leq b, \\ \quad x \in [1, 20], x \in Z, \\ \quad i = 1, 2, ..., n. \end{cases} \tag{7}$$

where $c_i \in [10, 20], d_i \in [10, 20], a_i \in [0, 50], b = 1.2sum(a) = 1.2 \sum_{i=1}^{n} a_i$.

$$\begin{cases} \min \sum_{i=1}^{n} (c_i x_i + x_i^{\frac{1}{d_i}}) \\ \text{s.t. } \sum_{i=1}^{n} a_i x_i \leq b, \\ \quad x \in [1, 6], x \in Z, \\ \quad i = 1, 2, ..., n. \end{cases} \tag{8}$$

where $c_i \in [-9, 9], d_i \in [1, 7], a_i \in [0, 50], b = 3.8sum(a) = 3.8 \sum_{i=1}^{n} a_i$.

The procedure of BBA and BB-PSO-HA are compiled with Matlab7.0.1 in person computer on DELL-P4-Intel865-512MB. We produce randomly twenty examples for the problems (6)-(8) in $n = 60, 100, 150, 200, 300, 500, 800, 1000, 1500, 2000, 3000, 4000$ respectively and solve these examples with BBA and BB-PSO-HA respectively. The results of the numerical computation can be seen at Table1-Table6. "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. We can say that the computation fails if BB-PSO-HA or BBA does not stop when iteration=10000.

It is shown by the numerical results from Table 1-Table 6 that BB-PSO-HA is better than BBA in the computational scale and the computational time and the computational precision.

Table 1 Numberical results for the problem 1

Ex1 $Eps = 10^{-4}$	BBA						
	Iteration			Cputime(Seconds)			failure rate
n	Avg	Max	Min	Avg	Max	MIN	*
60	7000	10000	1	472.2258	1035.8	0.0940	0.05
100	7580	10000	1	674.9794	1331.5	0.0780	0.05
150	7211	10000	1	844.7176	2574.9	0.1560	0.05
200	6776	10000	1	840.5947	3206.8	0.2970	0.05
300	8366	10000	1	1793.9932	5450.0	0.2030	0.10
500	6298	10000	3	2405.8687	8278.6	0.6400	0.10
800	5288	10000	2	4491.6621	8611.0	0.9850	0.083
1000	4357	10000	432	4143.4605	22135.0	181.0780	0.083

Table 2 Numberical results for the problem 1

Ex1 $Eps = 10^{-5}$	BBA-PSO						
	Iteration			Cputime(Seconds)			failure rate
n	Avg	Max	Min	Avg	Max	MIN	*
60	81	10000	1	584.3867	87775	9.090	0.05
100	95	648	1	795.3813	6032.4	12.1100	0
150	47	10000	1	1089.8666	9786.0	21.8590	0.2
180	122	10000	1	1462.1903	12588	21.6560	0.09523
200	178	10000	1	1510.5300	22573	29.0150	0.05
300	60	197	1	2100.8955	34889	36.0150	0
500	28	124	1	6908.9045	90178	72.7340	0
800	80	10000	1	959.0601	10874	116.3750	0.15
1000	20	10000	1	1991.250	18286	147.516	0.5

Table 3 Numberical results for the problem 2

Ex2 $Eps = 10^{-5}$	BBA						
	Iteration			Cputime(Seconds)			failure rate
n	Avg	Max	Min	Avg	Max	MIN	*
60	1076	5005	1	49.9905	325.8750	0.0460	0
100	1579	10000	1	133.3667	874.7650	0.0470	0.05
150	1655	9988	1	270.3304	2408.5	0.1870	0
200	1875	10000	1	520.5757	4111.1	0.0930	0.20
300	1081	8876	1	296.6009	2549.9	0.6100	0
500	2633	10000	1	1054.8215	4822.2	0.3130	0.05
800	558	10000	2	484.9714	8855.0	0.9850	0.05
1000	780	9888	432	490.8955	8402.8	1.5630	0

Table 4 Numberical results for the problem 2

Ex2 $Eps = 10^{-5}$	BBA-PSO						
	Iteration			Cputime(Seconds)			failure rate
n	Avg	Max	Min	Avg	Max	MIN	*
60	25	166	1	274.9855	1814.8	9.8120	0
100	8	75	1	142.8922	1488.4	17.0320	0
150	16	164	1	449.3077	4546.0	24.4690	0
180	11	175	1	171.9088	2379.8	30.0000	0
200	18	160	1	635.7583	5797.7	32.5780	0
300	14	178	1	451.2245	3947.8	49.5630	0
500	15	144	1	1017.3059	9394.1	65.3280	0
800	4	10000	1	594.2683	6732.6	137.5	0.05
1000	18	10000	1	3493.1909	50020	133.203	0.05
1500	2	5	1	297.80658	1003.2	199.593	0
2000	5	50	1	3057.3791	41250	271.218	0

Table 5 Numberical results for the problem 3

Ex3 $Eps = 10^{-5}$	BBA						
	Iteration			Cputime(Seconds)			failure rate
n	Avg	Max	Min	Avg	Max	MIN	*
60	2543	10000	1	291.1397	680.0630	0.1090	0.20
100	2942	10000	1	248.8008	1575.8	0.1250	0.15
150	3765	10000	1	348.9752	2768.2	0.1410	0.095
200	4491	10000	2	615.3112	2922.6	0.1720	0.10
300	2588	10000	1	704.9035	4100.5	0.2650	0.05
500	919	10000	1	161.2608	1721.5	0.2810	0.1428
800	2303	10000	2	664.1810	2634.0	0.4060	0.001
1000	8563	10000	1	1171.1894	4193.9	0.4310	0.047

Table 6 Numberical results for example 3

Ex3 $Eps = 10^{-5}$	BBA-PSO						
	Iteration			Cputime(Seconds)			failure rate
n	Avg	Max	Min	Avg	Max	MIN	*
60	1	3	1	10.2780	29.6410	0.1720	0
100	32	526	1	154.3984	2386.0	15.8440	0
150	2	22	1	41.8771	380.0560	23.6590	0
180	28	463	1	228.5568	3128.0	28.5470	0
200	22	374	1	525.9219	8321.0	31.7340	0
300	12	197	1	576.0852	9425.8	47.3590	0
500	10	10000	1	275.343	1324.70	78.9220	0.05
800	3	13	1	286.8236	1532.0	105.797	0
1000	2	6	1	311.4938	959.0310	159.313	0
1500	3	10000	1	595.3607	3833.5	201.25	0.05
2000	2	4	1	477.3579	1289.0	320.672	0
3000	7	38	1	1384.1188	3880.0	484.500	0
4000	24	37	1	7969.2188	23924	637.610	0

References

[1] Nemhauser G.L.and Wolsey L.A.: Integer and Combinatorial Optimization, John Wiley and sons,1988.

[2] Laughunn,D. J.: Quadratic binary programming with applications to capital-budgeting problem, Operations Research. **14** (l970)454-461.

[3] Konno, H., Watanabe, H.: Bond porfolio optimization problems and their application to index tracking: a partial optimization approach, Journal of the Operations Research Society of Japan. **39** (l996) 285-306.

[4] Barrientos,O., Correa,R., Reyes, P. and Valdebenito, A.: A brand and bound method for solving integer separable concave problems, Computational Optimization and Applications. **26** (2003) 155-171.

[5] J. Krarup,J. and Pruzan,P.M.: Computer-aided layout design, Mathematical Programming Study. **26** (1978)75-94.

[6] Witzgall, C.: Mathematical method of site selection for Electric Message Systems(EMS), NBS Internet Report, 1975.

[7] J. Rhys, "A selection problem of shared fixed costs on network flow", Management Science, **17(3)** (1970)200-207.

[8] Eberhart R.C.and Shi Y.H.: Particle swarm optimization: development, applications and resources, Proceedings of the IEEE International Conference on Evolutionary Computation,(2002)81-86.

[9] Laskari E.C., Parsopoulos K.E.and Vrahatis M.N.: Particle swarm optimization for integer programming, Proceedings of the IEEE International Conference on Evolutionary Computation, (2002)1582-15876.

[10] Eberhart R.C.and Shi Y.H.: Comparison between genetic algorithms and particle swarm optimization: development, applications and resources, Evolutionary Programming, (1998)611-615.

Received: September 30, 2006