数理解析研究所講究録
第 864 巻 1994 年 83-90

An Application of Modular approach to Separable Nonlinear Programming Problem

岡山理科大学 岩崎 彰典 (Akinori Iwasaki)
四国大学 疋田 光伯 (Mitsunori Hikita)
岡山理科大学 仲川 勇二 (Yuji Nakagawa)
岡山理科大学 成久 洋之 (Hiroyuki Narihisa)

Abstract

A discrete optimization method, which is called modular approach, is proposed for solving a separable nonlinear programming problem. By dividing seach space of variables, the nonlinear programming problem is translated into a discrete optimization problem that is equivalent to nonlinear knapsack problem. When the nonlinear knapsack problem is solved, we do not need the convexity and differentiability of original problem. The nonlinear knapsack problem can be solved efficiently by modular approach. It is shown that modular approach can be applied to a nonlinear programming problem by computational experiments.

1. Introduction

A separable nonlinear programming problem with one constraint function is written as follows:

<N>

$$\text{maximize} \quad f(\boldsymbol{x}) = \sum_{i\in I} f_i(x_i), \tag{1}$$

$$\text{subject to} \quad g(\boldsymbol{x}) = \sum_{i\in I} g_i(x_i) \leq b, \tag{2}$$

$$x_i \in S_i \quad (i \in I), \tag{3}$$

where $I = \{1, 2, \ldots, n\}$, and $S_i \subset R$ is a seach space, and b is a maximum amount of available resource.

We divide the seach space S_i into finite set A_i for each $i-th$ variable:

<K>

$$\text{maximize} \quad f(\boldsymbol{x}) = \sum_{i\in I} f_i(x_i), \tag{4}$$

$$\text{subject to} \quad g(\boldsymbol{x}) = \sum_{i\in I} g_i(x_i) \leq b, \tag{5}$$

$$x_i \in A_i \subset S_i \quad (i \in I), \tag{6}$$

where $I = \{1, 2, \ldots, n\}$, $A_i = \{a_{i1}, a_{i2}, \ldots, a_{ij}, \ldots, a_{ik_i}\}$. The search space S_i is represented by k_i points $\{a_{i1}, \ldots a_{ik_i}\}$.

The original problem <N> is translated into discrete optimization problem <K> that is equivalent to the nonlinear knapsack problem. Solving the nonlinear knapsack problem by discrete optimization method, the convexity and differentiability of original problem are not required.

We use modular approach(MA) for solving the nonlinear knapsack problem <K>. MA can solve the large scale nonlinear knapsack problem.

The optimal solution of the problem <K> is a near optimal solution of the original problem <N>. The search space S_i of the original problem can be reduced to the neighborhood of the near optimal solution. The new problem with reduced search spaces is created and solved by MA. By repetition of the above procedures, the near optimal solutions converge into the optimal solution of original problem <N>.

2. Modular approach

Nakagawa[1] proposed a new solution method called modular approach (MA) for solving discrete optimization problem. MA is a bottom-up scheme as well as Dynamic Programming. First, MA considers an optimization system corresponding to a given discrete optimization problem. Next, MA executes the following items 1) 2) recursively until the number of variables I becomes one.

1) The set A_i is reduced by fathoming tests.

2) Integrate two variables into one variable.

As for fathoming tests, we use dominance test, bounding test and feasibility test, which are techniques commonly used by Branch-and-Bound.

To integrate means to introduce a new set A_{NEW} that is corresponding to cartesian product of the two sets as follows:

$$A_{NEW} = A_j \times A_m \tag{7}$$

and j and m are removed from the set I.

There are four ways to select the sets j and m in the set I. Let k_i be the number of elements in the set A_i.

1) j and m such that k_j and k_m are the least.
2) j such that k_j is the least, and m such that k_m is the most.
3) j and m such that k_j and k_m are the most.
4) j and m in order of $i \in I$.

We choose the item 2) that is the fastest and can solve the largest scale problem. [3]

MA written by pseudo code is shown in figure 1.
The input of $ModularApproach()$ is a data sequence of Problem $< PC >$ and Quosi-Optimal Solution $< NEAR >$. Problem $< PC >$ contains a data sequence of current problem $< P >$ and a data sequence of $< T >$ that is required to translate the current problem $< P >$ into primal problem. The Quosi-Optimal Solution $< NEAR >$ is given by Recursive Greedy method[2]. Function $Fathom()$ reduces the set A_i by fathoming tests, and renews the current problem$< P >$. Function $ChoiceIM()$ selects two sets A_j and A_m. Function $Integrate()$ integrates the two sets A_j and A_m into one set A_{NEW}. After repeating the fathoming tests and integration, function $FindOptimalSolution()$ gives the optimal solution of the one variable problem.

3. Computational experiments

We divide the serch space S_i of given problem into finite set A_i . We create the nonlinear knapsack problem from the set A_i. The next two steps are repeated until required precision is given.

1) MA is applied to the nonlinear knapsack problem, and near optimal solution of given problem is given.

2) The neighborhood of the near optimal solution is divided, and the new nonlinear knapsack problem with reduced seach spaces is created.

3.1 Example 1

We consider a convex and differentiable problem as follows:

$$\text{maximize} \quad f(\boldsymbol{x}) = \sum_{i=1}^{10} (a_i + b_i x_i)^2, \tag{8}$$

$$\text{subject to} \quad g(\boldsymbol{x}) = \sum_{i=1}^{10} (c_i + d_i x_i)^2 \leq e, \tag{9}$$

$$x_i \in R. \tag{10}$$

Coeficients a_i, b_i, c_i, d_i and e are shown in Table 1.

This problem is solved by numerical computation and MA. Each results are shown in Table 2. Generally the results of MA are agreement with the results of numerical computation.

3.2 Example 2

We consider the nonconvex and undifferentiable problem as follows:

$$\text{maximize} \quad f(\boldsymbol{x}) = \sum_{i=1}^{10} f_i(x_i), \tag{11}$$

$$\text{subject to} \quad g(\boldsymbol{x}) = \sum_{i=1}^{10} g_i(x_i), \tag{12}$$

$$f_i(x_i) = \begin{cases} a_{i1}|\sin(x_i + b_{i1})| & (0.0 \leq x_i < 1.0) \\ a_{i2}|\cos(x_i + b_{i2})| & (1.0 \leq x_i < 2.0) \\ a_{i3}\ln(x_i + b_{i3}) & (2.0 \leq x_i < 3.0) \\ a_{i4}\sqrt{x_i + b_{i4}} & (3.0 \leq x_i < 4.0) \\ a_{i5}\exp(x_i/5 + b_{i5}) & (4.0 \leq x_i < 5.0), \end{cases} \tag{13}$$

$$g_i(x_i) = (x_i + c_i)^2 \leq e. \tag{14}$$

Coefficients $a_{i1}, \ldots, a_{i5}, b_{i1}, \ldots, b_{i5}, c_i$ and e are shown in Table 3.

This problem is solved by MA and the results are shown in Table 4. First, the seach space is divided into 100 elements, and the near optimal solution is given by MA. Next, the neighborhood of the near optimal solution is divided into 100 elements, and the second near optimal solution is also given by MA. The second near optimal solution exhausts the resource of constraint.

4. Concluding remarks

Solving two examples, it is shown that MA can solve nonconvex and undifferentiable nonlinear programming problem.

References

[1] Nakagawa Y.:"A New Method for Discrete Optimization Problems", *Trans. IEICE*, Vol.J73-A No.3 pp.550-556 (1990)

[2] Nakagawa Y.,Ohtagaki.H:"A modification of greedy procedure for solving nonlinear knapsack class of reliability optimization problems",
Trans. IEICE, Vol.J74-A No.3 pp.535-541 (1991)

[3] Nakagawa Y.,Hikita T.,Iwasaki A.:"A Fast Exact Method for the Multiple Choice Knapsack Problem",
Trans. IEICE, Vol.J75-A No.11 pp.1752-1754 (1992)

```
DATADEF
    < K >= {I, {K_1, K_2, ..., K_I}};
    < f >= {{f_1(1), ..., f_1(K_1)}, ..., {f_I(1), ..., f_I(K_I)}};
    < g >= {{g_1(1), ..., g_1(K_1)}, ..., {g_I(1), ..., g_I(K_I)}};
    < P >= {< K >, < f >, < g >, b};
    < PC >= {< P >, < T >};
    < NEAR >= {f^NEAR, {x_1^NEAR, ..., x_I^NEAR}};
    < OPT >= {f^OPT, {x_1^OPT, ..., x_I^OPT}};
    < M >= {m_1, m_2};
ENDDEF
FUNCTION  ModurarApproach()
INPUT  Problem< PC >,Quosi-Optimal Solution< NEAR >;
BEGIN      yes ← 1; No ← 0; IsNearSolOptimal ← No;
    WHILE  I ≥ 2  DO
        {< PC >, < NEAR >} ⇐ Fathom(< PC >, < NEAR >);
        IF  exist i ∈ {1, ..., I}  such that K_i = 0   THEN
            IsNearSolOptimal ← Yes;
            {< OPT >} ← {< NEAR >};
            EXITWHILE
        ENDIF
        {< M >} ⇐ ChoiceIM(< PC >);
        {< PC >} ⇐ Integrate(< M >, < PC >);
    ENDWHILE
    IF  IsNearSolOptimal = No  THEN
        {< OPT >} ⇐ FindOptimalSolution(< PC >);
        IF  f^OPT < f^NEAR  THEN
            {< OPT >} ← {< NEAR >)};
        ENDIF
    ENDIF
RETURN Optimal Solution< OPT >
END
```

figure 1. Modular approach

Table 1. Coefficient of Example. 1

e=1000

i	a_i	b_i	c_i	d_i
1	-2.2073	2.8969	-0.7442	-1.0398
2	2.4402	4.3	3.7626	-4.3934
3	-0.7114	5.52	-1.8898	1.871
4	1.2999	7.832	1.9367	3.4018
5	-2.2473	5.5	3.2593	4.4436
6	-4.9337	-4.98	1.0066	2.43
7	2.9042	2.9	3.1389	-1.7736
8	1.3991	-4.2178	2.8799	2.4
9	-1.0905	3.0	-3.4813	-1.283
10	-0.6581	4.54	-3.4915	-2.3995

Table 2. Results of Example. 1

		x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	f	g
Numerical	Solution	-6. 498558	1. 0223189	17. 132994	-1. 120442	-0. 962257	0. 0719372	2. 9085138	-1. 976559	-7. 244958	-2. 47795	10301. 902	1000
N=1000	Solution	-6. 56	1	17. 12	-1. 12	-0. 96	0. 08	2. 88	-1. 96	-7. 32	-2. 48	10301. 736	999. 99833
d=0. 04	Errors	0. 061442	0. 0223189	0. 0129939	-0. 000442	-0. 002257	-0. 008063	0. 0285138	-0. 016559	0. 075042	0. 0020505		
N=4000	Solution	-6. 53	1. 02	17. 13	-1. 12	-0. 96	0. 06	2. 91	-1. 97	-7. 25	-2. 48	10301. 888	999. 9996
d=0. 01	Errors	0. 031442	0. 0023189	0. 0029939	-0. 000442	-0. 002257	0. 0119372	-0. 001486	-0. 006559	0. 005042	0. 0020505		
N=10000	Solution	-6. 508	1. 024	17. 132	-1. 12	-0. 964	0. 072	2. 904	-1. 976	-7. 244	-2. 48	10301. 899	999. 99994
d=0. 004	Errors	0. 009442	-0. 001681	0. 0009939	-0. 000442	0. 0017431	-6. 28E-05	0. 0045138	-0. 000559	-0. 000958	0. 0020505		
N=40000	Solution	-6. 495	1. 022	17. 133	-1. 12	-0. 962	0. 072	2. 909	-1. 977	-7. 248	-2. 478	10301. 901	1000
d=0. 001	Errors	-0. 003558	0. 0003189	-6. 14E-06	-0. 000442	-0. 000257	-6. 28E-05	-0. 000486	0. 0004414	0. 003042	5. 047E-05		
N=1000*1000	Solution	-6. 49936	1. 0224	17. 13296	-1. 12016	-0. 96224	0. 07184	2. 90832	-1. 97648	-7. 24496	-2. 47792	10301. 902	1000
d=0. 00016	Errors	0. 000802	-8. 11E-05	3. 386E-05	-0. 000282	-1. 69E-05	9. 721E-05	0. 0001938	-7. 86E-05	2. 034E-06	-2. 95E-05		
N=4000*1000	Solution	-6. 49824	1. 02232	17. 13304	-1. 12048	-0. 96224	0. 07184	2. 90832	-1. 97656	-7. 24488	-2. 47808	10301. 902	1000
d=0. 00008	Errors	-0. 000318	-1. 05E-06	-4. 61E-05	3. 779E-05	-1. 69E-05	9. 721E-05	0. 0001938	1. 374E-06	-7. 8E-05	0. 0001305		

N:The number of division.(4000*1000 means 1000 division after 4000 division)

d:Division width

Table 3. coefficients of Example. 2

e=150

i	a_{i1}	a_{i2}	a_{i3}	a_{i4}	a_{i5}	b_{i1}	b_{i2}	b_{i3}	b_{i4}	b_{i5}	c_i
1	3. 6	3. 5	1. 6	1. 1	0. 7	4. 5	2. 2	1. 5	0. 8	0. 4	0. 8
2	2. 4	2. 2	0. 6	0. 2	0. 2	2. 0	1. 9	0. 6	0. 3	0. 7	1. 5
3	5. 0	4. 8	3. 1	2. 5	0. 8	4. 0	4. 2	0. 8	1. 0	0. 5	2. 0
4	2. 0	1. 9	0. 7	0. 2	0. 1	3. 6	4. 1	0. 2	0. 2	0. 6	0. 5
5	4. 5	4. 0	3. 0	2. 4	0. 7	2. 9	1. 2	0. 9	0. 9	0. 5	0. 2
6	2. 8	3. 0	1. 1	0. 8	0. 5	4. 7	1. 6	1. 2	0. 5	0. 8	0. 9
7	3. 9	4. 1	2. 9	2. 5	1. 0	4. 4	3. 8	0. 4	1. 3	0. 3	1. 3
8	3. 2	3. 4	1. 2	0. 9	0. 6	3. 3	0. 7	1. 8	1. 1	0. 9	1. 8
9	4. 7	5. 0	2. 7	2. 7	0. 6	0. 8	3. 6	0. 1	0. 7	0. 4	0. 7
10	2. 7	2. 7	1. 0	0. 5	0. 1	1. 1	2. 3	1. 9	0. 6	0. 2	0. 3

Table 4. Results of Example. 2

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	f	g
N=100	0. 10	0. 00	3. 40	0. 85	3. 90	0. 00	3. 80	4. 90	3. 90	0. 35	39. 437	149. 99
N=100*100	0. 182	0. 000	3. 526	0. 998	4. 000	0. 000	3. 866	4. 999	4. 000	0. 442	39. 444	150

N:The number of division