ADAPTIVE MONTE CARLO METHOD FOR SOLVING CONSTRAINED
MINIMIZATION PROBLEM IN INTEGER NON-LINEAR PROGRAMMING

Minoru Ichimura

Department of Mathematics
Okayama College of Science
Okayama 700, Japan
and

Kazumasa Wakimoto

Department of Mathematics
Okayama University
Okayama 700, Japan

We consider the non-linear integer programming where the objective function is non-linear, but the constraint is linear.

Let $\mathbf{L} = (\ell_1, \ell_2, \dots, \ell_N)$ denote an N-dimensional vector whose components ℓ_i , $i=1, 2, \dots, N$ are non-negative integers.

Let us suppose that the objective function $f(\mathbf{L})$ is non-linear. The problem is to find $\mathbf{L}^*$ which minimizes $f(\mathbf{L})$ subject to

$$(1) \quad \sum_{i=1}^N \ell_i \leq M,$$

where N and M are given positive integers.

In most practical cases, the objective function $f(\mathbf{L})$ is complicated, so it is difficult to minimize $f(\mathbf{L})$ by using the ordinary methods. For such objective function, to find $\mathbf{L}^*$, we propose the following algorithm, namely, adaptive Monte Carlo method:

- Step 1: Generate N-dimensional random vectors of size n satisfying the constraint (1), where an N-dimensional random vector means a selected vector by the equal probability from each of all possible N-dimensional vectors satisfying the constraint (1).
- Step 2: Evaluate the objective function $f(\mathbf{L})$ for each random vector and select the vector $\mathbf{L}_0$ which minimizes $f(\mathbf{L})$ in already generated vectors. Set vectors $\mathbf{L}_0$ to initial vector in next step 3.
- Step 3: Change the range of components of random vectors by the following procedures:

- i) Select the positive integer M' which is less than M .
- ii) Generate random vectors, $X^{(k)} = (x_1^{(k)}, x_2^{(k)}, \dots, x_N^{(k)})$, and $Y^{(k)} = (y_1^{(k)}, y_2^{(k)}, \dots, y_N^{(k)})$, for $k=1, 2, \dots, m$, whose components satisfy the following constraints:

$$\sum_{i=1}^N x_i^{(k)} \leq M', \quad \sum_{i=1}^N y_i^{(k)} \leq M', \quad \text{for } k=1, 2, \dots, m,$$

where all components are non-negative integers.

Put $\ell_i^{(k)} = \ell_i^0 + x_i^{(k)} - y_i^{(k)}$, $k=1, 2, \dots, m$, for $i=1, 2, \dots, N$, where $L_0 = (\ell_1^0, \ell_2^0, \dots, \ell_N^0)$.

If the vector $(\ell_1^{(k)}, \ell_2^{(k)}, \dots, \ell_N^{(k)})$ has at least a negative component, reject its vector. Return to step 2.

Example 1.

Define the objective function $f(L)$ as follows:

$$(2) \quad f(L) = \sum_{i=1}^{10} \ell_i^2$$

where the components of vector L , are non-negative integers and satisfy the following equation:

$$(3) \quad \sum_{i=1}^{10} \ell_i = 100.$$

Step 1: Generate 10-dimensional random vectors of size n satisfying Eq(3) by combinatorial method that presented in Appendix I. In this step, we use the crude Monte Carlo method.

Step 2: Evaluate the objective function $f(L)$ for each random vector and select the vector L_0 which minimizes $f(L)$ in already generated vectors. Set vector L_0 to initial vector in next step 3.

Step 3: Change the range of components of random vectors by the following procedures:

- i) Select the positive integer M' which is less than 100. In this example, $M' = 5$ or 1 are used.
- ii) Generate random vectors, $X^{(k)} = (x_1^{(k)}, x_2^{(k)}, \dots, x_{10}^{(k)})$, and $Y^{(k)} = (y_1^{(k)}, y_2^{(k)}, \dots, y_{10}^{(k)})$, for $k=1, 2, \dots, m$, whose components satisfy the following constraints:

Example 2.

Define the objective function $f(L)$ as follows:

$$(4) \quad f(L) = \sum_{i=1}^{10} l_i (1 - e^{-\frac{l_i}{\alpha_i}})$$

where α_i are constant parameters as shown in Table 2.

Table 2. Constant parameters of objective function

i	1	2	3	4	5	6	7	8	9	10
α_i	6.	7.	10.	18.	19.	23.	25.	26.	30.	47.

Let us find the vector L^* which minimizes the value of objective function $f(L)$ subject to

$$(5) \quad \sum_{i=1}^{10} l_i = 100.$$

We take the approach similar to example 1. The numerical results are shown in Table 3. Furthermore, to compare with the crude Monte Carlo method using only the step 1 and step 2, the numerical results of two methods are shown in Fig. 1.

Table 3. Numerical results in case of $n=10$ and $m=10$

cumulative numbers of generating vector	components of vector L_0	$f(L_0)$	M'
10	(1, 2, 5, 8,18, 0, 2,19,11,34)	47.396469	
20	(2, 2, 5, 7,17, 0, 2,20,11,34)	47.108032	5
30	(3, 3, 4, 9,15, 2, 3,19, 9,33)	44.611450	5
40	(3, 4, 3, 9,15, 4, 4,19, 9,30)	42.996628	5
50	(3, 5, 5, 9,13, 4, 5,18, 9,29)	41.905899	5
60	(3, 4, 3, 9,12, 7, 7,18, 8,29)	40.623596	5
70	(3, 4, 4, 9,12, 8, 8,18, 8,26)	39.853546	5
80	(3, 3, 5, 7,12, 8, 8,17, 9,28)	39.668411	5
90	(3, 3, 4, 8,13, 7, 8,15,10,29)	39.647949	5
100	(3, 3, 4, 9,11, 6,10,14,11,29)	39.153702	5
120	(3, 3, 5, 8, 9, 6,10,14,13,29)	38.887680	5
130	(3, 2, 4, 9, 9, 7,11,13,14,28)	38.588974	5

140	(3, 3, 4, 8, 7, 8,11,13,14,29)	38.527283	5
150	(2, 2, 5, 9, 9, 9, 9,14,15,26)	38.381607	5
160	(2, 2, 4, 9,10, 9,11,13,15,25)	38.176300	5
180	(2, 4, 5, 8, 8,10,13,12,15,23)	37.931122	5
190	(3, 3, 6, 9, 9, 9,12,12,15,22)	37.920959	5
210	(3, 3, 4,10, 8,10,13,12,15,22)	37.915436	5
220	(3, 4, 4,10, 8,10,12,12,15,22)	37.914230	2
240	(3, 4, 4, 9, 8,11,12,12,15,22)	37.848587	2
250	(3, 4, 5, 8, 8,11,13,12,14,22)	37.841904	2
270	(3, 4, 5, 9, 8,11,12,12,14,22)	37.815933	2
290	(3, 3, 5, 9, 8,11,12,12,14,23)	37.797485	2
310	(3, 3, 4, 9, 9,11,12,12,14,23)	37.795364	2
400	(3, 3, 5, 9, 9,11,11,12,14,23)	37.785004	2

Fig.1 Comparison between adaptive and crude Monte Carlo methods

Acknowledgments.

The authors are grateful to Dr. Koiti Takahasi, the Institute of Statistical Mathematics, Japan. Thanks are due to some members for useful discussions at the Symposium "Information and Statistics" held in Sizuoka University on February, 1973.

Appendix I.

In this appendix, two algorithms are presented for generating an N-dimensional random vector satisfying the constraint (6) as follows:

$$(6) \quad \sum_{i=1}^N \ell_i = M.$$

If an (N+1)-dimensional random vector satisfying the constraint $\sum_{i=1}^{N+1} \ell_i = M$ is generated, it is easy to generate an N-dimensional random vector $L' = (\ell'_1, \ell'_2, \dots, \ell'_N)$ satisfying the constraint:

$$(7) \quad \sum_{i=1}^N \ell'_i \leq M$$

as shown below:

Step 1: Generate an (N+1)-dimensional random vector satisfying

$$\text{the constraint} \quad \sum_{i=1}^{N+1} \ell_i = M.$$

Step 2: Put $\ell'_i = \ell_i$, for $i=1, 2, \dots, N$, and $L' = (\ell'_1, \ell'_2, \dots, \ell'_N)$.

Algorithm 1.

Step 1: Draw the random numbers of size N-1: $n_1, n_2, \dots, n_{N-1}$ without replacement from the M+N-1 consecutive positive integers (from 1 to M+N-1) and arrange them in ascending order according to their magnitude:

$$n_{(1)} < n_{(2)} < \dots < n_{(N-1)}.$$

Step 2: We get the components of a random vector by the following expression:

$$\ell_i = n_{(i)} - n_{(i-1)} - 1, \quad i=1, 2, \dots, N$$

$$\text{where } n_{(0)} = 0 \text{ and } n_{(N)} = M+N.$$

Algorithm 2.

Step 1: Draw the random numbers of size M: $n_1, n_2, \dots, n_M$, without replacement from the $M+N-1$ consecutive positive integers (from 1 to $M+N-1$) and arrange them in ascending order according to their magnitude: $n_{(1)} < n_{(2)} < \dots < n_{(M)}$.

Step 2: Put

$$k_j = n_{(j)} - (j-1), \quad j=1, 2, \dots, M.$$

Put

$$\delta_{ij} = \begin{cases} 1, & i = k_j, \\ 0, & \text{otherwise,} \end{cases}$$

for $j=1, 2, \dots, M$, and

$$l_i = \sum_{j=1}^M \delta_{ij}, \quad i=1, 2, \dots, N.$$

Then $(l_1, l_2, \dots, l_N)$ is an N-dimensional vector satisfying the constraint (6).

We can obviously prove the above algorithm 1 and 2 by repeated combination. The algorithm 2 is efficient when N is large and M is small.

Appendix II.

```

INTEGER KEKA{10},KST{10},KPA{10},KPB{10},NOKOSU{10},RES{10}
IXXC=0
JXXC=0
C.....RESET SUBPROGRAM RAN WHICH GENERATES AN UNIFORMAL RANDOM NUMBER.
      KB=RAN{IXXC,JXXC}
C.....IT = COUNT OF RANDOM VECTORS.
C.....IIT= COUNT OF EVALUATION OF THE OBJECTIVE FUNCTION.
      IT=0
      IIT=0
      KSUM=100
      KOSU=10
      SUMIN=1000000.0

```

C.....CRUDE MONTE CARLO METHOD.

DO 1 J=1,10

$$IT = IT + 1$$
$$IIT = IIT + 1$$

```
CALL RANSUM{KSUM,KOSU,KST,IXXC,JXXC}
```

SUM=0.0

DO 6 I=1,KDSU

$$\text{SUM} = \text{SUM} + \text{KST}\{\text{I}\} ** 2$$

6 CONTINUE

```
IF{SUM.GE.SUMIN} GO TO 1
```

SUMIN=SUM

DO B K=L-KOSU

8 KEKA-[K]=KST-[K]

```
WRITE{b,101} IT,IIT,{KEKA{I},I=1,KOSU},SUMIN
```

```
101 FORMAT(1H0,'KAISU=',I3,2X,'CT=',I3,5X,'{' ,10(I3,' ' ,}')',2X,F6.0)
```

1 CONTINUE

DO 7 I=1,KOSU

KST-{I}=KEKA-{I}

7 RES{I}=KEKA{I}

C.....ADAPTIVE MONTE CARLO METHOD.

KSUM=5

DO 100 KL=1,2

```
DO 2 LL=1,20
```

```
DO 3 I=1,10
```

CALL RANSUM{KSUM,KOSU,KPA,IXXC,JXXC}

CALL RANSUM{KSUM,KOSU,KPB,IXXC,JXXC}

$$IT = IT + 1$$

DO 4 J=L-KOSU

$$NOKOSU\{J\} = KST\{J\} + KPA\{J\} - KPB\{J\}$$

IF{NOKOSU{J}.LT.0} GO TO 3

4 CONTINUE

$$IIT = IIT + 1$$

C.....EVALUATE THE OBJECTIVE FUNCTION.

SUM=0.0

DO 20 J=1,KOSU

```
SUM=SUM+NOKOSU{J}**2
```

20 CONTINUE

```
IF{SUM.GE.SUMIN} GO TO 3
```

SUMIN=SUM

```
WRITE(6,101) IT,IIT,{NOKOSU{K},K=1,KOSU},SUMIN
```

DO 21 K=1,KOSU


```

21 RES{K}=NOKOSU{K}
3 CONTINUE
  DO 22 K=1,KOSU
22 KST{K}=RES{K}
2 CONTINUE
  KSUM=1
100 CONTINUE
  STOP
  END
  SUBROUTINE RANSUM(WA,KOSU,KEKA,IXR,JXR)
  INTEGER WA,KOSU,KR{100},KEKA{10}
  KUMI=WA+KOSU
  AK=KUMI
  KR{1}=0
  KP1=KOSU+1
  KR{KP1}=KUMI
  DO 2 J=2,KOSU
3 KI=RAN{IXR,JXR}*AK
  KI=MOD{KI,KUMI}
  JML=J-1
  DO 4 L=1,JML
  IF{KI.EQ.KR{L}} GO TO 3
  IF{KI.LT.KR{L}} GO TO 12
4 CONTINUE
  KR{J}=KI
  GO TO 2
12 ICN=L+1
  KWT=KR{L}
  KR{L}=KI
  DO 13 L=ICN,J
  KW=KR{L}
  KR{L}=KWT
  KWT=KW
13 CONTINUE
2 CONTINUE
  DO 5 J=1,KOSU
  KEKA{J}=KR{J+1}-KR{J}-1
5 CONTINUE
  RETURN
  END

```