MATEC Web of Conferences **44**, 01038 (2016)
DOI: 10.1051/matecconf/20164401038

The Key Solution Algorithm of Linear Programming Model

Jun Liu [1,a] , Chuan Cheng Zhao[1] , Zhi Guo Ren[1] , Zhong Yi Feng[1] , Zheng Ping Zhu [1]

1. Lanzhou City University, School of Information Science and engineering, Lanzhou 730070, P.R.China

Abstract. Linear programming problem is a common problem, and to solve the linear model is more plagued. The paper generating algorithm is based on mathematical theory and composition. The design of feasible solution algorithm illustrates key linear programming model, then we can find a better way to solve the linear programming model solutions.

1 Introduction

According to mathematical theory of linear programming model[1-2] (1):

$$\max \quad z = \sum_{i=1}^{n} c_i x_i$$

$$s.t \begin{cases} \sum_{j=1}^{n} a_{ij} x_j \le a_{in+1} (i = 1,2,\cdots,m) \\ x_i \ge 0 (i = 1,2,\cdots,n) \end{cases} (m \ge n)$$

S,t condition of the equation

$$\sum_{j=1}^{n} a_{ij} x_j = a_{in+1} (i = 1,2,\cdots,m)$$

And $x_i = 0 (i = 1,2,\cdots,n) m + n$ equations form an optimal solution *m*+*n* plane (side) closed set *Ω*, the linear programming model (1) on the closed set *Ω* border on the objective function s,t condition of the equation：

$$z = \sum_{i=1}^{n} c_i x_i$$

From (0,0, ⋯, 0) (z = 0) moving along the surface normal direction, so that the value of *z* is gradually increased until the value of the maximum, then getting a linear programming model (1) of the optimal solution $(x_1, x_2, \cdots, x_n)$ and the corresponding *z* maximum. This method is theoretically established, but in practice it would be inconvenient to solve over three-dimensional linear programming model[3].

This paper designs the key feasible solution algorithmsof computer of graphic linear programming model.

2 The Combination Generation Algorithm

In the key feasible solution algorithms of computer of graphic linear programming model, it is supposed to discuss the combination generation algorithm of C_n^m, C_n^m linear equations are generated from m + n equations[4].

Let$C_n^0 = \emptyset$， $C_n^n = 1,2,3,\cdots,n$. When generating the combination of C_n^m, *n* numbers of the generated combination, each and all combinations are arranged by the dictionary order.

2.1 The Generation of Combination C_n^1

The combination of C_1^1 is $1, C_2^1$ is 1,2, C_3^1 is 1,2,3..., and the combination of C_n^1 is 1,2,3, ...,n. Combination can be expressed as follows (to be expressed simply, writtenC_n^m *as* $\Delta(n,m)$),borrowing documents[2] to discuss the digital triangles and trigonometric operations and "⊕" and the triangular numbers counted by "∘"):

$\Delta(1,1) = (1)^T$,

$\Delta(2,1) = (1 \quad 2)^T$,

$\Delta(3,1) = (1 \quad 2 \quad 3)^T$,

…,

$\Delta(n,1) = (1 \quad 2 \quad 3 \quad \cdots \quad n\text{–}1 \quad n)^T$

[a]authore-mail: 527876625@qq.com
This studywas supported by Lanzhou City University Ph.D. Research Fund (21265099,41361013,GS[2013]GHB1084, LZCU-BS2013-09 and LZCU-BS2013-12).

(To write conveniently, matrix form is used above.)

2.2 The Generation of Combination C_n^2

The combination of $C_2^2$is 12,$C_3^2$is 12,13,23，…，$C_n^2$is 12,13,14，…，$1n$；23,24,…$2n$；……；$n-2n-1, n-1n$；$n-1n$.

It can be expressed by digital triangle as follows:

$\Delta(2,2) = 12 = \Delta(1,2) \oplus \Delta(1,1) \circ 2$,

$$\Delta(3,2) = \begin{matrix} 12 & 13 \\ & 23 \end{matrix} = \Delta(2,2) \oplus \Delta(2,1) \circ 3$$,

$$\Delta(4,2) = \begin{matrix} 12 & 13 & 14 \\ & 23 & 24 \\ & & 34 \end{matrix} = \Delta(3,2) \oplus \Delta(3,1) \circ 4$$,

…,

$$\Delta(n,2) = \begin{matrix} 12 & 13 & 14 & \cdots & 1n-1 & 1n \\ & 23 & 24 & \cdots & 2n-1 & 2n \\ & & 34 & \cdots & 3n-1 & 3n \\ & & & & \cdots & \cdots \\ & & & & (n-2)(n-1) & (n-2)n \\ & & & & & (n-1)n \end{matrix}$$

$= \Delta(n-1,2) \oplus \Delta(n-1,1) \circ n$

2.3 The Generation of Combination C_n^3

Similarly, itcan get the digital triangle indicating the combinations of $C_3^3, C_4^3, C_5^3, \ldots, C_n^3$.

$\Delta(3,3) = 123 = \Delta(2,3) \oplus \Delta(2,2) \circ 3$,

$$\Delta(4,3) = \begin{matrix} 123 & 124 & 134 \\ & & 234 \end{matrix} = \Delta(3,3) \oplus \Delta(3,2) \circ 4$$,

$$\Delta(5,3) = \begin{matrix} 123 & 124 & 134 & 125 & 135 & 145 \\ & & 234 & & 235 & 245 \\ & & & & & 345 \end{matrix}$$

$= \Delta(4,3) \oplus \Delta(4,2) \circ 5$,

…,

$$\Delta(n,3) = \begin{matrix} 123 & 124 & 134 & 125 & 135 & 145 & \cdots \\ & 234 & & & 235 & 245 & \cdots \\ & & & & & 345 & \cdots \\ 12n & 13n & 14n & \cdots & 1(n-2)n & 1(n-1)n & \\ & 23n & 24n & \cdots & 2(n-2)n & 2(n-1)n & \\ & & 34n & \cdots & 3(n-2)n & 3(n-1)n & \\ & & & & \cdots & \cdots & \\ & & & & (n-3)(n-2)n & (n-3)(n-1)n & \\ & & & & & (n-2)(n-1)n & \end{matrix}$$

$= \Delta(n-1,3) \oplus \Delta(n-1,2) \circ n$

2.4 The Generation of Combination $\Delta(n,m)$ (C_n^m)

Similarly, itcan obtain the combination of $\Delta(n,m)$.

Thus, itcan get the $\Delta(n,m)$ of combination C_n^m generating algorithm theorem.

Theorem: the combination of $C_n^m (m \le n/2, m \ge 2)$ is $\Delta(n,m) = \Delta(n-1,m) \oplus \Delta(n-1,m-1) \circ n$.

3 The Design of Key Feasible Solution Algorithm

From the foregoing, if the feasible solutions of linear programming model (1) is the closed setΩ, and the optimal solution is in the boundary of this closed set Ω, and the possible case is unique (a vertex of closed set Ω) or infinite (an edge or a surface of closed set Ω). So it only need to find each vertex of the closed set Ω when calculating.These vertices of the closed set $\Omega(x_1, x_2, \cdots, x_n)$are called key feasible solutions, then determining the optimal solution of the linear programming model（1）from key feasible solution[5-10].

To find the key feasible solution on the closed set Ω, changing the restrictions of linear programming model to:

$$\begin{cases} \sum_{j=1}^{n} a_{ij} x_j = a_{in+1} (i = 1,2,\cdots,m) \\ x_i = 0 (i = 1,2,\cdots,n) \end{cases}.$$

This is a linear equation containing n variables and m + n equations, the key feasible solution of the closed set Ω is the intersection from m + n equations obtained C_{m+n}^{n} equations. The intersection satisfies the points of restrictions in model (1),and it corresponds to the value of the objective function *z*. Finding the maximum value from these values, there may be three cases, one (unique), two (an edge) or three (a surface).

Computer algorithm as following:

(1) Let max=0;

(2) Generating $m+n$ equations to C_{m+n}^n equation groups by the algorithm of generated combination and solving it;

(3) Substituting the linear programming model (1) restrictions, judging whether it is the key feasible solution on the closed set Ω, if so, calculating the value of objective function z;

(4) Judging whether $z >$ max is established, if so, max = z returningthe key feasible solution of equation groups is $(x_1, x_2, \cdots, x_n)$. If $z =$ max is satisfied, keeping the key feasible solution of equation groups as $(x_1, x_2, \cdots, x_n)$;

(5) Returning to 1,until getting the solution of C_{m+n}^nequation groups;

(6) Outputting the value of z and the optimal solution $(x_1, x_2, \cdots, x_n)$;

With this algorithm, we can easily calculate the optimal solution of linear programming model (1) $(x_1, x_2, \cdots, x_n)$.

4 The Advantages of Algorithm

This algorithm is based on the principle of the

existence of the optimal solution on the closed set Ω, directly calculating the intersection and determining whether it is the key feasible solution of the closed set Ω, and immediately seeking z, greatly reducing the number of calculations, and occupying less memory as a fast and effective way. Meanwhile, it will have a good reference for the algorithm of integer programming by improving this algorithm. This section will be discussed about the pull-off algorithm of the solutions of integer programming model in another text.

References

1. Prof. Dr. T. Gal. Homogene mehrparametrische lineare Programmierung[J]. Zeitschrift für Operations Research . 1972 (3)
2. K. -H. Borgwardt. The Average number of pivot steps required by the Simplex-Method is polynomial[J]. Zeitschriftfür Operations Research . 1982 (1)
3. Wolfgang Temelt. The parameter space of the general linear programming problem[J]. Soviet Mathematical Journal . 1967 (3)
4. The ellipsoid method and its consequences in combinatorial optimization[J]. Combinatorica . 1981 (2)
5. N. Karmarkar. A new polynomial-time algorithm for linear programming[J]. Combinatorica . 1984 (4)
6. Yamamura K,TanakaS.Performance evaluation of the LPtest algorithm for finding all solutions of piecewise-linearresistive circuits. International Journal of CircuitTheory and Applications . 2000
7. H. P. Benson,D. Lee. Outcome-based algorithm for optimizing over the efficient set of a bicriteria linear programming problem[J]. Journal of Optimization Theory and Applications . 1996 (1)
8. F. P. Vasil'ev,M. Yachimovich. Discrepancy method for the lexicographic linear programming problem[J]. Computational Mathematics and Modeling . 1995 (1)
9. J. Fülöp,L. D. Muu. Branch-and-Bound Variant of an Outcome-Based Algorithm for Optimizing over the Efficient Set of a Bicriteria Linear Programming Problem[J]. Journal of Optimization Theory and Applications . 2000 (1)
10. Adi Ben-Israel. A $\mathbb{Z}$-Simplex Algorithm with partial updates[J]. BIT . 1987 (1)

6 C language source code

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#define N 100
#define OK 1
#define ERROR 0
double a[N][N],c[N][N];
typedefstruct
{
double number[N];
doublereasult;
}
SeqList;
typedefstruct
{
	SeqListrecord[N];
	intNsize,Size;
}Node;
int Print(double b[][N],intn,int m)
{
	inti,j;
	for(i=1;i<=n;i++)
	{
		for(j=1;j<=m;j++)
			printf(" %lf ",b[i][j]);
		printf("\n");
	}
	return OK;
}
void Print1(Node *L)
{
	inti,j;
	for(i=1;i<=L->Size;i++)
	for(j=1;j<=L->Nsize;j++)
	{
		printf("i=%d  %lf",i,L->record[i].number[j]);
printf("\n");
		printf("\n");
	}
	printf("\n");
}
void Print2(Node *L)
{
	inti,j;
	for(i=1;i<L->Size;i++)
	{
	for(j=1;j<=L->Nsize;j++)
		printf(" %e ",L->record[i].number[j]);
		printf(" The sult is:%lf ",L->record[i].reasult);
printf("\n");
		printf("\n");
	    }
	printf("\n");
}
intCreat(Node *L,intn,int m)
{
	inti;
	int c1=1,c2=1;
	printf("未知数的个数 Nsize:");
	scanf("%d",&L->Nsize);
	for(i=n;i>1;i--)
```

```
        c1*=i;
    for(i=m-1;i>1;i--)
        c2*=i;
    for(i=n-(m-1);i>1;i--)
        c2*=i;
    L->Size=c1/c2;
    return OK;
}
int meet(Node *L,intfirst,intnumber,int m)
{
  FILE *fp;
inti,j,ok=0;
doubley,sum;
for(i=first,j=1;j<=L->Nsize;j++)
    if(L->record[i].number[j]<0
L->record[i].number==-0)
        return ERROR;
fp=fopen("wang.txt","rt");
if(fp==NULL)
  {
    printf("Can't Open!");
    exit(0);
  }
for(i=1;i<=number;i++)
  {
    sum=0;
for(j=1;j<=L->Nsize;j++)
    {
        fscanf(fp,"%lf",&y);
        sum+=L->record[first].number[j]*y;
    }
fscanf(fp,"%lf",&y);
    if(sum>y+1)
    return ERROR;
  }
fclose(fp);
return OK;
}
int Sub(Node *L,int m)
{
    inti,j,number;
    double x;
    FILE *fp;
    printf("pleaser enter s.t number:");
scanf("%d",&number);
    for(i=1;i<=L->Size;i++)
    {
    if(meet(L,i,number,m))
      {
        fp=fopen("zhu.txt","rt");
        if(fp==NULL)
          {
            printf("Can't open!");
            exit(0);
          }
          L->record[i].reasult=0;
        for(j=1;j<=L->Nsize;j++)
          {
            fscanf(fp,"%lf",&x);

L->record[i].reasult+=L->record[i].number[j]*x;
          }
        fclose(fp);
      }
    else L->record[i].reasult=0;
    }
return OK;
}
int Memory(double b[][N],Node *L,int m)
{
int j;
staticinti=1;
for(j=1;j<=L->Nsize;j++)
  {
      L->record[i].number[j]=b[j][m];
  }
i++;
return OK;
}
intTackback(double b[][N],Node *L,intn,int m)
{
    inti,j,k;
    double s;
    for(i=1;i<=n;i++)
    {
    if(b[i][i]==0) break;
      s=1.0/b[i][i];
    for(j=1;j<=m;j++)
    b[i][j]=s*b[i][j];
    }
    for(i=n;i>=1;i--)
    {
    for(j=i-1;j>=1;j--)
      {
        s=b[j][i]/b[i][i];
      if(s==0) break;
      for(k=i;k<=m;k++)
        b[j][k]=(-1*s)*b[i][k]+b[j][k];
      }
    }
Memory(b,L,m);
    return OK;
}
int change(double b[][N],inttime,intn,int m)
{
    inti,j,max;
    doublet,x,y;
    max=time;
    for(i=time+1;i<=n;i++)
    {
        x=b[max][time];
        y=b[i][time];
      if(fabs(x)<fabs(y))
        max=i;
    }
    if(max!=time)
    {
      for(j=1;j<=m;j++)
      {
        t=b[time][j];
b[time][j]=b[max][j];
        b[max][j]=t;
      }
    }
```

```
return OK;
}
intuptriangle(double b[][N],Node *L,intn,int m)
{
      inti,j,k;
      double s;
for(i=1;i<=n;i++)
      {
            change(b,i,n,m);
            for(j=i+1;j<=n;j++)
            {
                s=b[j][i]/b[i][i];
                if(s==0) break;
                for(k=1;k<=m;k++)
                  {
                      b[j][k]=(-1*s)*b[i][k]+b[j][k];
                  }
            }
      }
Tackback(b,L,n,m);
return OK;
}
int Assemble(double b[][N],Node *L,intn,int m)
{
      inti,j,k,elem[100];
      i=1; elem[i]=1;
      while(i!=0)
        {
if(elem[i]>n)
              {
                  i--;
                  elem[i]++;
              }
            else if(i==m-1)
              {
for(j=1;j<=m-1;j++)
            for(k=1;k<=m;k++)
                  c[j][k]=b[elem[j]][k];
uptriangle(c,L,m-1,m);
                  memset(c,0,sizeof(c));
            elem[i]++;
              }
            else
              {
                  i++;
                  elem[i]=elem[i-1]+1;
              }
        }
      return OK;
}
int main()
{
      FILE *fp;
      inti,j,m,n;
      Node list;
      printf("pleaser enter n and m:");
      scanf("%d%d",&n,&m);
      fp=fopen("wang.txt","rt");
      if(fp==NULL)
      {
            printf("Can't open!");
            exit(0);
      }
for(i=1;i<=n;i++)
            for(j=1;j<=m;j++)
            {
                  fscanf(fp,"%lf",&a[i][j]);
            }
creat(&list,n,m);
            assemble(a,&list,n,m);
            sub(&list,m);
            print2(&list);
return 0;
}
```