Projekt zespołowy

Temat: Algorytm Balasa

Opracował:
Damian Wolański

Opis działania programu.

Zero-jedynkowy addytywny algorytm Balasa pozwala na rozwiązanie następującego binarnego programu liniowego:

Znaleźć minimum:

$$z = c_1x_1 + c_2x_2 + ... + c_nx_n$$

Przy ograniczeniach:

$$a_{i1}x_1 + a_{i2}x_2 + ... + a_{in}x_n \leq b_i$$

gdzie:
i = 1, 2, ..., m
x_j *= 0 lub 1*
j = 1, 2, ..., n.

Zasada działania algorytmu.

Do śledzenia postępu przeglądu używa się dwu wektorów: $u^T = (u_1, u_2, ..., u_n)$ oraz $x^T = (x_1, x_2, ..., x_n)$. Jeśli w skład rozwiązania częściowego wchodzą zmienne $x_{j1}, x_{j2}, ..., x_{js}$, wybrane w takim właśnie porządku, to składowe wektora *x* odpowiadające tym zmiennym mają wartości albo 0 albo 1. Pozostałe składowe wektora *x*, odpowiadające zmiennym swobodnym, mają wartości -1. Składowe wektora *u* są postaci:

$$u = \begin{cases} jk & \text{jeżeli } x_{jk} = 1 \text{ i jego dopełnienie nie było rozważane} \\ -jk & \text{jeżeli } x_{jk} = 0 \text{ lub } 1 \text{ i jego dopełnienie było rozważane} \\ 0 & \text{jeżeli } k > s \end{cases}$$

Krok początkowy.
Krok 1.
Sprawdzić, czy $b \geq 0$. Jeśli tak, to optymalnym rozwiązaniem jest *x* = 0. W przeciwnym razie podstawić

$$u^T = (0, 0, ..., 0)$$
$$x^T = (-1, -1, ..., -1)$$
$$z' = \infty \text{ (dostatecznie duża liczba)}$$

Kroki iteracyjne
Krok 2.
Obliczyć:

$$y_i = b_i - \sum_{j \in J} a_{ij} x_j$$
$$y' = \min y_i \text{ , dla } i = 1, 2, ..., m$$
$$z = \sum_{j \in J} c_j x_j$$

gdzie *J* jest zbiorem wskaźników zmiennych o ustalonych wartościach (rozwiązanie częściowe). Jeśli $y' \geq 0$ oraz $z < z'$, to przyjąć $z = z'$ oraz $x' = x$. Wektor x' jest nowym potencjalnym rozwiązaniem. Przejść do kroku 6 (cofanie się). W przeciwnym razie iść dalej.

Krok 3.
Utworzyć następujący podzbiór *T* zmiennych swobodnych x_j:

$$T = \{ j: z + c_j < z' , a_{ij} < 0 \text{ dla } i \text{ takich, że } y_i < 0 \}$$

Jeśli $T = \{\}$ (zbiór pusty), to przejść do kroku 6. W przeciwnym razie iść dalej.

Krok 4.
TEST NIEDPOPUSZCZALNOŚCI. Sprawdzić, czy istnieje taki wskaźnik *i*, że:

$$y_i < 0 \text{ oraz}$$
$$y_i - \sum_{j \in T} \min (0, a_{ij}) < 0$$

Jeśli tak, to przejść do kroku 6. W przeciwnym razie iść dalej.

Krok 5.
TEST ROZGAŁĘZIEŃ BALASA. Dla każdej zmiennej swobodnej *x* utworzyć zbiór:

$$M_j = \{ i: y_i - a_{ij} < 0 \}$$

Jeśli wszystkie zbiory M_j są puste, przejść do kroku 6. W przeciwnym razie obliczyć:

$$v_j = \sum_{i \in M_j} (y_i - a_{ij})$$

dla każdej zmiennej swobodnej x_j, gdzie $v_j = 0$, gdy $M_{,j} = \{\}$ (zbiór pusty).
Powiększyć bieżące rozwiązanie częściowe o tę zmienną x_j, dla której v_j osiąga wartość największą. Wróć do kroku 2.

Krok 6.
COFANIE SIĘ:

1. Zmienić w wektorze *u* znak najbardziej w prawo położonej składowej dodatniej. Nadać wszystkim składowym leżącym na prawo od niej wartość 0. Jest to nowe rozwiązanie częściowe. Wrócić do kroku 2.
2. Jeśli wektor *u* nie ma dodatniej składowej, to przegląd pośredni został zakończony. Optymalnym rozwiązaniem jest x', a odpowiadającą mu wartością funkcji celu jest z'. Jeśli $z' = \infty$, to nie ma rozwiązania dopuszczalnego.

Fragmenty kodu.

Przygotowanie programu

```
void PrepareBalas(unsigned int N, unsigned int M, int** A, int* B, int* C, int* O, bool MAX)
{
// Ustawienie wszystkich ograniczeń na mniejsze równe
for(unsigned int i=0; i<M; i++)
{
if(O[i]>0) // Jeśli ograniczenie większe równe mnożymy przez -1
{
for(unsigned int j=0;j<N;j++)
A[i][j]=-A[i][j];
B[i]=-B[i];
}
}
// Zmiana problemu maksymalizacji na problem minimalizacji
if(MAX)
{
for(unsigned int j=0;j<N;j++)
{
C[j]=-C[j];
if(C[j]<0)
{
for(unsigned int i=0; i<M; i++)
{
A[i][j]=-A[i][j];
B[i]=B[i]+A[i][j];
}
}
}
}

}
```

```
// Krok 2
for(unsigned int i=0; i<M; i++)
{
Y[i]=B[i];
for(unsigned int j=0;j<N; j++)
{
if(XX[j]>0)
Y[i]=Y[i]-A[i][j]*XX[j];
}
}
YMIN=*(min_element(Y.begin(), Y.end()));

Z=0;
for(unsigned int j=0;j<N; j++)
{
if(XX[j]>0)
Z=Z+C[j]*XX[j];
}

if(YMIN>=0 && Z<FVAL)
{
X=XX;
FVAL=Z;
EXIST=true;
break;
}

// Krok 3
T.clear();
for(unsigned int j=0; j<N; j++)
{
for(unsigned int i=0;i<M; i++)
{
if(XX[j]<0 && A[i][j]<0 && Z+C[j]<FVAL && Y[i]<0)
{
T.push_back(j);
break;
}
}
}
if(T.size()==0)
break;

// Krok 4
FLAGUE=false;
for(unsigned int i = 0; i < M; i++)
{
```

```
if(Y[i]<0)
{
SUM=0;
for(unsigned int j = 0; j < T.size(); j++)
SUM=SUM+min(0, A[i][T[j]]);

if(Y[i]-SUM<0)
{
FLAGUE=true;
break;
}
}
}
if(FLAGUE==true)
break;

// Krok 5
MT.clear();
FLAGUE=false;
for(unsigned int j = 0; j < N; j++)
{
for(unsigned int i = 0; i < M; i++)
{
if(Y[i]-A[i][j]<0)
MT[j].push_back(i);
}

V[j]=0;
for(unsigned int i = 0; i < MT[j].size(); i++)
V[j]=V[j]+Y[MT[j][i]]-A[MT[j][i]][j];

if(MT[j].size()>0)
FLAGUE=true;
}
if(FLAGUE==false)
break;

FLAGUE=false;
for(unsigned int j = 0; j < N; j++)
{
if(V[j] >= *(max_element(V.begin(), V.end())))
{
XX[j]=1;
for(unsigned int k=0; k<N; k++)
if(U[k]==0)
{
U[k]=j;
```

```
break;
}

FLAGUE=true;
break;
}
}
if(FLAGUE==false)
break;

}

// Krok 6
R=-1;
for(int k=N-1; k>=0; k--)
{

if(U[k]>0)
{
XX[U[k]]=0;
U[k]=-U[k];
R=k+1;
break;
}
}
if(R<0)
break;
for(unsigned int k=R; k<N; k++)
U[k]=0;
}
if(EXIST)
{
cout << "x = (";
for(unsigned int i=0; i<N; i++)
{
if(X[i]<0)
X[i]=0;
cout << X[i];
if(i==N-1)
cout << ");";
else
cout << ",";
}
cout << endl;
cout << "Wartosc funkcji " << FVAL << endl;
}
return EXIST;}
```

Screeny z działającego programu

Czynności:
- podajemy rozmiar macierzy NxM,
- wprowadzamy wektor wartości funkcji celu,
- wprowadzamy wektor ograniczeń,
- wprowadzamy macierz współczynników,

```
N - liczba zmiennych
M - liczba ograniczen
A - Macierz wspolczynnikow
B - wektor ograniczen
C - wektor wartosci funkcji
Podaj N
4
Podaj M
3
Podaj C0
10
Podaj C1
14
Podaj C2
21
Podaj C3
42
Podaj B0
-12
Podaj B1
-14
Podaj B2
-10
Podaj A[0,0]
-8
Podaj A[0,1]
-11
Podaj A[0,2]
-9
Podaj A[0,3]
-18
Podaj A[1,0]
-2
Podaj A[1,1]
-2
Podaj A[1,2]
-7
Podaj A[1,3]
-14
Podaj A[2,0]
-9
Podaj A[2,1]
-6
Podaj A[2,2]
-3
Podaj A[2,3]
-6
x = (1,0,0,1);
Wartosc funkcji 52

--------------------------------
```