Tomasz Jankowiak 249006

3 VI 2021

Sterowanie procesami dyskretnymi

Laboratorium 5

Algorytm Carliera dla $1|r_j,q_j|C_{max}$

Prowadzący:	Dr inż. Mariusz Makuchowski
Termin zajęć:	Piątek, 11.15
Numer grupy laboratoryjnej:	E12-30a

1. Opis problemu

Mamy do wykonania *n* zadań na pojedynczej maszynie. Każde zadanie *j* opisane jest trzema parametrami:

- r_j - czas dostarczenia (termin dostępności),
- p_j - czas trwania,
- q_j - czas stygnięcia.

Zadania są dostarczane w tym samym czasie. Na maszynie w danej chwili wykonuje się co najwyżej jedno zadanie. Stygnięcie rozpoczyna się bezpośrednio po zakończeniu wykonywania zadania. Zadania mogą stygnąć równocześnie. Celem jest wyznaczenie harmonogramu pracy maszyny, który minimalizuje czas w jakim wszystkie zadania wystygną. Zatem szukamy uszeregowania o najmniejszej długości C_{max} (najpóźniej wystygnięte zadanie).

W ogólnym przypadku zagadnienie jest NP-trudne. Najlepsze znane warianty algorytmów optymalnych są w stanie rozwiązywać w stosunkowo krótkim czasie problemy z n ≤ 1000, jednak przejawiają eksplozję obliczeń dla przykładów o większym rozmiarze [1].

2. Opis algorytmu

Algorytm Carliera dla problemu $1|r_j,q_j|C_{max}$ jest algorytmem dokładnym bazującym na metodzie podziału i ograniczeń (B&B). Wykorzystuje on następujące elementy składowe:

- **algorytm Schrage dla problemu $1|r_j,q_j|C_{max}$** – wyznaczenie górnego ograniczenia **UB** (szukamy lepszych rozwiązań od tego które już mamy),
- **algorytm Schrage dla problemu $1|r_j,q_j,pmtn|C_{max}$** – wyznaczenie dolnego ograniczenia **LB** (optymalne rozwiązanie wyjściowego problemu nie może być lepsze, ponieważ każde rozwiązanie problemu $1|r_j,q_j|C_{max}$ jest także rozwiązaniem problemu $1|r_j,q_j,pmtn|C_{max}$, dla którego algorytm Schrage dostarcza rozwiązania optymalnego),
- wyznaczanie bloku oraz zadania krytycznego.

Modyfikując w czasie pracy algorytmu wartości r_j i q_j można wstawiać dane zadanie przed lub za pewien blok innych zadań. Możliwe są realizacje różnych strategii przeglądania – w głąb, wszerz oraz zachłanna.

3. Kod programu

```
#include <iostream>
#include <fstream>

using namespace std;

int cmax(int n, int*R, int*P, int *Q, int *X)
{
    int m=0,c=0;
    for(int i=0;i<n;i++){ m = max(m,R[X[i]])+P[X[i]]; c = max(c,m+Q[X[i]]); }
    return c;
}

int Schrage(int n,int*R,int*P,int*Q,int*X)
{
    int A[100],K[100], a=n,k=0,x=0,t=0;
    for(int i=0;i<n;i++) A[i]=i;
    for(int i=0; i<(n-1); i++) for(int j=0; j<(n-1); j++) if( R[A[j]] < R[A[j+1]] ) swap(A[j], A[j+1]);
    while(x!=n){
        if(a!=0){
            if(R[A[a-1]]<=t){
                K[k]=A[a-1]; k++; a--;
                for(int e=k-1;e>0;e--){ if( Q[K[e]] < Q[K[e-1]] ) swap(K[e], K[e-1]); }
                continue;
            }
        }
        if(k!=0) { X[x]=K[k-1]; k--; x++; t=t+P[X[x-1]]; continue; }
        if(0==k && R[A[a-1]]>t){ t=R[A[a-1]]; }
```

[1] Grabowski J., Nowicki E., Smutnicki C., *„Metoda blokowa w zagadnieniach szeregowania zadań"*, Warszawa 2003, str. 64.

```
    }
    return cmax(n,R,P,Q,X);
}

int Schrage_zPodzialem(int n,int*R,int*P,int*Q)
{
    int A[100],K[100], a=n,k=0,x=0,t=0,p=0,l=-1,cmaks=0,Ppom[100];
    for(int i=0;i<n;i++) A[i]=i;
    for(int i=0;i<n;i++) Ppom[i] = P[i];
    for(int i=0;i<(n-1);i++) for(int j=0;j<(n-1);j++) if(R[A[j]]<R[A[j+1]]) swap(A[j],A[j+1]);
    while(a!=0 || k!=0){
        if(a!=0){
            if(R[A[a-1]]<=t){
                K[k]=A[a-1]; k++; a--;
                for(int e=k-1;e>0;e--){ if( Q[K[e]] < Q[K[e-1]] ) swap(K[e], K[e-1]); }
                if(l!=-1) if( Q[K[k-1]] > Q[l] ){ K[k]=l; k++; swap(K[k-1],K[k-2]); l=-1; }
                continue;
            }
        }
        if(k!=0){
            if(-1==l){ l=K[k-1]; k--; }
            if(a!=0) p = min( Ppom[l], R[A[a-1]]-t );
            else p=Ppom[l];
            t = t + p; Ppom[l] = Ppom[l] - p;
            if(0==Ppom[l]){ cmaks=max(cmaks,t+Q[l]); l=-1; }
            continue;
        }
        if(0==k && a!=0) if(R[A[a-1]]>t) t=R[A[a-1]];
    }
    return cmaks;
}

void Blok(int n, int* R, int* P, int* Q, int* X, int& cI, int& cR, int& cQ)
{
   int posB = -1, m = 0, cmax = 0;
   int tmp[100];
   for (int i = 0; i < n; i++){
          int j = X[i];
          tmp[i] = (m >= R[j]);
          m = max(m, R[j]) + P[j];
          if (cmax < m + Q[j]){
            cmax = m + Q[j];
            posB = i;
          }
   }
   int i = posB,j=-1;
   int bQ = Q[X[posB]];
   int bR = R[X[posB]];
   int bP = P[X[posB]];
   while (tmp[i]){
          if (Q[X[--i]] < bQ){
                 j = X[i];
                 break;
          }
          bR = min(bR, R[X[i]]);
          bP += P[X[i]];
   }
   cI = j;
   cR = bR+bP;
   cQ = bQ+bP;
}

void Carlier(int n, int* R, int* P, int* Q, int* X, int& UB)
{
   if (Schrage_zPodzialem(n, R, P, Q) >= UB) return;
   int sCmax = Schrage(n, R, P, Q, X);
   if (sCmax < UB) UB = sCmax;
   int j, jr, jq;
   Blok(n, R, P, Q, X, j, jr, jq);
   if (j < 0) return;
   int tmpR = R[j];
   int tmpQ = Q[j];
   R[j] = jr;
```

```
    Carlier(n, R, P, Q, X, UB);
    R[j] = tmpR;
    Q[j] = jq;
    Carlier(n, R, P, Q, X, UB);
    Q[j] = tmpQ;
}

int main()
{
    int n,R[100],P[100],Q[100],X[100];

    ifstream plik("data.txt");
    string s; while(s!="data.001:") plik>>s;
    plik >> n;
    for(int i=0;i<n;i++)
        plik >> R[i] >> P[i] >> Q[i];
    plik.close();

    /*Schrage(n,R,P,Q,X);
    cout << "schr:\n" << cmax(n,R,P,Q,X) << endl;
    for(int i=0;i<n;i++) cout << X[i]+1 << " ";

    cout << "\n\nschrpmtn = " << Schrage_zPodzialem(n,R,P,Q) << endl << endl;*/

    int UB = Schrage(n,R,P,Q,X);
    Carlier(n, R, P, Q, X, UB);
    cout << "carl:\n" << UB << endl;
    for(int i=0;i<n;i++) cout << X[i]+1 << " ";
    cout << endl;

  //  cin.get();
    return 0;
}
```

4. Działanie programu

Algorytm zaimplementowano na ocenę 3 – bez właściwej strategii przeglądania i dodatkowych testów eliminacyjnych, choć z pewnością zwiększyłoby to jego efektywność. Dla wybranego z ośmiu zestawów danych (http://mariusz.makuchowski.staff.iiar.pwr.wroc.pl/download/courses/sterowanie.procesami.dyskretnymi/lab.instrukcje/lab05.carlier/carl.data/carl.data.txt) program wykonuje algorytm Carliera, a następnie wypisuje wartość funkcji celu Cmax oraz permutację. Otrzymane wyniki są zgodne z porównawczymi zawartymi w pliku.

5. Czas działania

Dane	Czas [s]
000	0.696
001	0.321
002	4.032
003	0.063
004	0.070
005	0.414
006	29.327
007	>100
008	>100

6. Podsumowanie i wnioski

Algorytm Carliera dla problemu $1|r_j,q_j|C_{max}$ w przeciwieństwie do algorytmu Schrage (który jest wykorzystywany w dwóch wersjach do znalezienia ograniczeń) znajduje rozwiązanie optymalne, choć czasem może to zająć dużo czasu (dla ostatnich danych spodziewać się można kilku godzin działania). Zatem jest to algorytm dokładny.

7. Bibliografia

1. Czesław Smutnicki – *„Algorytmy szeregowania zadań*”, Oficyna Wydawnicza PWr, Wrocław 2012, str. 160-164.
2. Józef Grabowski, Eugeniusz Nowicki, Czesław Smutnicki – *„Metoda blokowa w zagadnieniach szeregowania zadań*”, Warszawa 2003, str. 64, 83-89.
3. Mariusz Makuchowski – wykład „Algorytm Carliera dla $1|r_j, q_j|C_{max}$”, 2019, http://mariusz.makuchowski.staff.iiar.pwr.wroc.pl/download/courses/sterowanie.procesami.dyskretnymi/wyk.slajdy/SPD_w06_Carlier.pdf.