ZASTOSOWANIA MATEMATYKI
APPLICATIONES MATHEMATICAE
XVI, 3 (1979)

ALGORITHM 66

A. ADRABIŃSKI and M. WODECKI (Wrocław)

AN ALGORITHM FOR SOLVING THE MACHINE SEQUENCING PROBLEM WITH PARALLEL MACHINES

1. Procedure declaration. The procedure *SEQPRO* finds an optimal sequence of operations for the machine sequencing problem with sets of identical machines, i.e., for the sequencing problem with parallel machines [5]. The optimal sequence minimizes the total time spent for processing all operations.

Data:

Q	—	number of sets of parallel machines;
M	—	number of operations;
T	—	number of pairs of operations for which the precedence relations are given;
MC	—	number of cousins of all parallel machine sets;
LIP	—	number of machines;
INF	—	maximum positive number of type **integer**;
$MAKS$	—	allowed number of iterations of the algorithm;
$NI[1:Q]$	—	array of numbers of operations such that $NI[k]$ is the number of operations which are to be carried out on the machines from the k-th set of parallel machines;
$BK[1:Q]$	—	array of numbers of machines such that $BK[k]$ is the number of machines of the k-th set of parallel machines;
$RTP[1:T]$, $RTK[1:T]$	—	these arrays contain pairs of operations expressing the technological requirements put on the order of operation; the array *RTP* contains the predecessor numbers and the array *RTK* contains the successor numbers;
$C[1:MC]$	—	array of processing times of operation; $C[(i-1)\times BK[j]+1:i\times BK[j]]$ are the processing times of the i-th operation, where $i=N[j-1]+1, N[j-1]+2, \ldots, N[j]$, on

the machines of the j-th set of parallel machines, where $j = 1, 2, \ldots, Q$ and $N[0] = 0$;

PP[1:*LIP*] — array of access times of machines; *PP*[*i*] is the time from which on the i-th machine can be used.

Results:

LK[1:*M*] — array of cousin numbers in the optimal selection;

SO[1:*M*] — array of the earliest start times of operations in the optimal sequence;

SK[1:*M*] — array of the general reserve times of operations;

LOPT — value of the total time spent for processing all operations.

Other parameters:

END — label outside of the body of the procedure *SEQPRO* to which a jump is made if the expected number of iterations *MAKS* is smaller than that required by the algorithm.

2. Method used. An improved version of the algorithm due to Grabowski (see [4] and [5]) has been used in the procedure *SEQPRO*. For definitions and notation see [5].

The computations start with the graph $D_{11}^0 = \langle A, U^{\prime}; S_1; S_1^0\rangle$ which represents the root of the solution tree H and with sets $F_1 \neq \emptyset$, $F_1^0 \neq \emptyset$, $F_1^t \neq \emptyset$, $F_1^{t_0} \neq \emptyset$ described in [5]. The first lower bound L^* in Step 1 (test step) is the critical path length of the graph

$$D(S_1^1 \cup S_1^0) = \langle A, U^0; S_1^1; S_1^0\rangle,$$

where sets A and U^0 have been defined in [5], S_1^0 is an initial selection of sets (selection of minimal length cousins) and $S_1^1 \subset S_1$, S_1 being an initial complete selection. It can easily be proved that $L(S_1^1 \cup S_1^0)$ is a better initial lower bound than that proposed by Grabowski.

Remarks on Steps 3 and 4. Let us consider any node $D_{rp}^0 \in R_{D_{di}}^{0'}$ of the solution tree H corresponding to the $(r+p)$-th iteration of the algorithm. Let

$$K = R_r' \cup R_{rp} \cup K_p^{0'} \tag{1}$$

be the set of candidates in the graph D_{rp}^0, i.e., K is the set of reverse and empty arcs which are prepared for complementing and eliminating, respectively. If $R_{rp} \neq 0$ and the implicit condition is satisfied for some arcs from the set R_{rp}, then we choose an arc from K which belongs to the set R_{rp} and has a minimal delta, i.e., such that

$$\delta_{rp}\{\langle y, x\rangle, \langle u, v\rangle\} = \min_{\langle c, d\rangle \in R_{rp}} \Delta_{rp}[\langle a, b\rangle, \langle c, d\rangle].$$

```
procedure SEQPRO(Q,M,T,MC,LIP,INF,MAKS,NI,BK,RTP,RTK,C,PP,
 LK,SO,SK,LOPT,END);
 value Q,M,T,NI,BK,RTP,RTK,C,PP,MAKS;
 integer Q,M,T,INF,LIP,LOPT,MAKS,MC;
 integer array NI,BK,RTP,RTK,C,PP,LK,SO,SK;
 label END;
 begin
  integer A,B,D,E,F,G,H,I,J,K,L,N,P,R,S,W,Z,MAX,MIN,NR1,NR2,
  P1;
  Boolean B1,B2;
  integer array LPP[1:M],WK[0:M],KC[1:M+M],SD,DET[0:MAKS],
  DEL,NDEL[1:if MC=M then 1 else MC-M];
  Boolean array FKT[1:M];
  SD[0]:=D:=R:=0;
  for I:=1 step 1 until Q do
   begin
    J:=NI[I];
    D:=D+J×(J-1)
   end I;
  D:=D/2;
  for I:=1 step 1 until M do
   begin
    for J:=1 step 1 until T do
     if RTP[J]=I
      then go to LAB;
    R:=R+1;
    SO[R]:=I;
LAB:
   end I;
  P1:=D+T+R;
```

```
P:=P1+M;
W:=P+M;
N:=M+M+2;
S:=D+M;
begin
 integer array AP,AK,APT,AKT,PU[1:W],APZ,AKZ,APP,AKP[1:D],
 REP,BD[1:S],LSD[0:S],LOX,LOXPR,LXZ,LXZPR,MDK[1:N];
 Boolean array FR,FRH,FRT[1:D];
 integer procedure CPM(PU,AP,AK,LOX);
  integer array PU,AP,AK,LOX;
  begin
   integer I,J,K,F,G,H,U,MAX;
   Boolean array SW[1:N],SU[1:W];
   U:=0;
   for I:=1 step 1 until N do
    begin
     SW[I]:=SU[I]:=true;
     PU[I]:=0
    end I;
   for I:=N+1 step 1 until W do
    begin
     SU[I]:=true;
     PU[I]:=0
    end I;
   for K:=1 step 1 until N do
    begin
     for I:=1 step 1 until N do
      if SW[I]
       then
       begin
```

```
for J:=1 step 1 until W do
 if AK[J]=I
  then
  begin
   if SW[AP[J]]
    then go to NEXTI
    else
     if SU[J]
      then
      begin
       U:=U+1;
       SU[J]:=false;
       PU[U]:=J;
       MAX:=LOX[AP[J]];
       if J>P1
        then MAX:=MAX+KC[J-P1];
       G:=AK[J];
       if B1
        then
        begin
         if MAX>LOX[G]
          then
          begin
           LOXPR[G]:=LOX[G];
           LOX[G]:=MAX;
           MDK[G]:=J
          end MAX>LOX[G]
          else
           if MAX>LOXPR[G]
            then LOXPR[G]:=MAX;
```

```
                                go to E1
                              end B1;
                             if MAX>LOX[G]
                              then LOX[G]:=MAX;
E1:                         end SU[J]
                   end AK[J]=I;
                 SW[I]:=false;
                 go to NEXTK;
NEXTI:         end SW[I];
NEXTK:end K
      end CPM;
     L:=S:=0;
     for J:=1 step 1 until Q do
      begin
       MIN:=NI[J];
       MAX:=MIN-1;
       G:=L+L;
       for I:=1 step 1 until MAX do

        begin
         H:=I+L;
         K:=H+H;
         for F:=I+1 step 1 until MIN do
          begin
           S:=S+1;
           AKP[S]:=Z:=K;
           AP[S]:=APZ[S]:=Z+1;
           AK[S]:=AKZ[S]:=Z:=F+F+G;
           APP[S]:=Z+1
          end F
        end I;
```

```
    L:=L+MIN
   end J;
 for I:=1 step 1 until T do
   begin
    S:=S+1;
    F:=RTP[I];
    APT[S]:=AP[S]:=F+F+1;
    H:=RTK[I];
    AKT[S]:=AK[S]:=H+H
   end I;
 for I:=1 step 1 until R do
   begin
    S:=S+1;
    G:=SO[I];
    APT[S]:=AP[S]:=G+G+1;
    AKT[S]:=AK[S]:=N
   end I;
 L:=0;
 for I:=1 step 1 until Q do
   begin
    MIN:=NI[I];
    K:=L+L;
    for J:=1 step 1 until MIN do
     begin
      S:=S+1;
      APT[S]:=AP[S]:=1;
      AKT[S]:=AK[S]:=J+J+K
     end J;
    L:=L+MIN
   end I;
```

```
for I:=1 step 1 until M do
 begin
  S:=S+1;
  L:=I+I;
  APT[S]:=AP[S]:=L;
  AKT[S]:=AK[S]:=L+1
 end I;
L:=LOPT:=Z:=NR2:=LSD[0]:=LOX[1]:=WK[0]:=0;
for I:=1 step 1 until Q do
 begin
  NR1:=NI[I];
  for J:=1 step 1 until NR1 do
   begin
    L:=L+1;
    LPP[L]:=LOPT;
    R:=1;
    S:=BK[I];
    MIN:=INF;
    for K:=1 step 1 until S do
     begin
      Z:=Z+1;
      B:=C[Z];
      if MIN>B
       then
       begin
        MIN:=B;
        R:=K
       end MIN>B
     end K;
    REP[D+L]:=SO[L]:=SK[L]:=R;
```

```
KC[M+L]:=MIN;
KC[L]:=PP[LOPT+R];
WK[L]:=Z;
FKT[L]:= S=1;
LK[L]:=S-1;
A:=NR2;
B:=R-1;
for K:=1 step 1 until B,R+1 step 1 until S do
 begin
  NR2:=NR2+1;
  DEL[NR2]:=C[WK[L-1]+K]-MIN;
  NDEL[NR2]:=K
 end K;
S:=S-1;
K:=-S;
for K:=K+2 while K<0 do
 begin
  R:=S+K;
  for F:=1 step 1 until R do
   begin
    for H:=F step K until 1 do
     begin
      G:=A+H;
      E:=G-K;
      MIN:=DEL[G];
      MAX:=DEL[E];
      if MIN≤MAX
       then go to ENDF
       else
       begin
```

```
                    DEL[G]:=MAX;
                    DEL[E]:=MIN;
                    MIN:=NDEL[G];
                    NDEL[G]:=NDEL[E];
                    NDEL[E]:=MIN
                  end MIN≤MAX
                end H;
ENDF:         end F
            end K
          end J;
         LOPT:=LOPT+BK[I]
        end I;
       L:=1;
       for A:=1 step 1 until D do
        begin
         FR[A]:=FRT[A]:=FRH[A]:=false;
         B:=APP[A];
         G:=AKP[A];
         J:=APZ[A];
         K:=AKZ[A];
         E:=F:=1;
         for S:=P+E while AP[S]≠GVAK[S]≠J do
          E:=E+1;
         for S:=P+F while AP[S]≠KVAK[S]≠B do
          F:=F+1;
         if SK[E]≠SK[F]
          then APT[A]:=AKT[A]:=REP[A]:=0
          else
          begin
           APT[A]:=APZ[A];
```

```
          REP[A]:=1;
          AKT[A]:=AKZ[A]
        end SK[E]=SK[F]
      end A;
     for J:=2 step 1 until N do
      LOX[J]:=-1;
     B1:=false;
     CPM(PU,APT,AKT,LOX);
     LOPT:=LOX[N];
     for I:=1 step 1 until D do
      APT[I]:=AKT[I]:=0;
     go to KROK2;
KROK1:
     LOX[1]:=0;
     for J:=2 step 1 until N do
      LOX[J]:=-1;
     CPM(PU,APT,AKT,LOX);
     if LOX[N]≥LOPT
      then go to KROK4;
KROK2:
     for J:=1 step 1 until N do
      begin
       LOX[J]:=LOXPR[J]:=LXZ[J]:=LXZPR[J]:=-1;
       MDK[J]:=0
      end J;
     H:=LOX[1]:=LXZ[N]:=0;
     B1:=true;
     CPM(PU,AP,AK,LOX);
     B1:=false;
     for I:=1 step 1 until D do
```

```
  if AP[I]=0
   then H:=H+1;
 for J:=W-H step -1 until 1 do
  begin
   K:=PU[J];
   R:=AP[K];
   MAX:=LXZ[AK[K]];
   if K>P
    then MAX:=MAX+KC[K-P1];
   if MAX>LXZ[R]
    then
    begin
     LXZPR[R]:=LXZ[R];
     LXZ[R]:=MAX
    end MAX>LXZ[R]
    else
     if MAX>LXZPR[R]
      then LXZPR[R]:=MAX
  end J;
 MAX:=LOX[N];
 if LOPT>MAX
  then
  begin
   LOPT:=MAX;
   for J:=1 step 1 until D do
    REP[J]:=if FRT[J]∧(¬FR[J]) then (if FRH[J] then -1
            else 0) else 1;
   for J:=1 step 1 until M do
    REP[D+J]:=SK[J]
  end LOPT>MAX;
```

```
B:=I:=0;
R:=LSD[L-1]+1;
A:=MDK[N];
for J:=0 while B≠1 do
 begin
  if A≤D
   then
   begin
    if ¬FRT[A]
     then
     begin
      I:=I+2;
      G:=R+1;
      F:=APZ[A];
      H:=AKZ[A];
      if G≥MAKS
       then go to KON;
      SD[R]:=SD[G]:=A;
      MIN:=LOXPR[H]-LOX[H];
      MAX:=LXZPR[F]-LXZ[F];
      Z:=if MIN>MAX then MIN else MAX;
      MAX:=MIN+MAX+KC[MDK[APP[A]]-P1]+KC[MDK[F]-P1];
      MAX:=if Z>MAX then Z else MAX;
      if FR[A]
       then
       begin
        if FRH[A]
         then
         begin
          DET[R]:=INF+1;
```

```
           DET[G]:=Z
         end FRH[A]
         else
         begin
           DET[R]:=MAX;
           DET[G]:=INF+1
         end ¬FRH[A]
       end FR[A]
       else
       begin
         DET[R]:=MAX;
         DET[G]:=Z
       end ¬FR[A];
      R:=G+1
    end ¬FRT[A]
  end A≤D
  else
   if A>P
    then
    begin
     S:=A-P;
     if ¬FKT[S]
      then
      begin
       F:=LK[S];
       if R+F-2≥MAKS
        then go to KON;
       E:=WK[S]-S;
       for K:=E+1-F step 1 until E do
        begin
```

```
              I:=I+1;
              SD[R]:=A;
              DET[R]:=DEL[K];
              R:=R+1
             end K
           end ¬FKT[S]
         end A>P;
      B:=AP[A];
      A:=MDK[B]
     end J;
    if I=0
     then go to KROK4;
    LSD[L]:=LSD[L-1]+I;
KROK3:
    MIN:=B:=INF;
    Z:=LSD[L-1]+1;
    NR1:=LSD[L];
    for I:=Z step 1 until NR1 do
     if DET[I]<INF
      then
      begin
       A:=SD[I];
       MAX:=DET[I];
       if A≤D
        then
        begin
         if SD[I-1]=A∧I>Z
          then
          begin
           G:=APP[A];
```

```
F:=APZ[A];
if ¬(FKT[MDK[G]-P]∧FKT[MDK[F]-P])
 then go to XYZ
 else
 begin
  E:=NR2:=1;
  for S:=P+E while AP[S]≠AKP[A]∨AK[S]≠F do
   E:=E+1;
  for H:=P+NR2 while AP[H]≠AKZ[A]∨AK[H]≠G do
   NR2:=NR2+1;
  if SK[E]≠SK[NR2]
   then
   begin
    if B>MAX
     then
     begin
      B:=MAX;
      R:=I
     end B>MAX
   end SK[E]≠SK[NR2]
   else
   begin
    DET[I]:=INF;
    FR[A]:=true;
    FRT[A]:=FRH[A]:=false
   end SK[E]=SK[NR2]
 end(FKT[MDK[G]-P]∧FKT[MDK[F]-P]);
go to XYZ
end SD[I-1]=A
end A≤D;
```

```
        if MIN>MAX
         then
         begin
          MIN:=MAX;
          K:=I
         end MIN>MAX;
XYZ: end DET[I]<INF;
    if B<INF
     then
     begin
      MIN:=B;
      K:=R
     end B<INF;
    if MIN≥INF
     then
ALFA:
     begin
      S:=LSD[L];
      K:=LSD[L-1]+1;
      for I:=K while I≤S do
       begin
        A:=SD[I];
        if A≤D
         then
         begin
          FRT[A]:=false;
          AP[A]:=APZ[A];
          AK[A]:=AKZ[A];
          APT[A]:=AKT[A]:=0;
          B2:=DET[I]>INF;
```

```
          FR[A]:= B2∨DET[I+1]>INF;
          FRH[A]:= B2;
          K:=K+2
        end A≤D
        else
        begin
          H:=0;
          for F:=I+H while SD[F]=A∧F≤S do
           H:=H+1;
          K:=K+H;
          F:=A-P;
          LK[F]:=H;
          H:=SO[F];
          KC[F]:=PP[LPP[F]+H];
          KC[F+M]:=C[WK[F-1]+H];
          FKT[F]:=false
        end A>D
      end I;
     go to KROK4
   end MIN≥INF;
 A:=SD[K];
 if A≤D
  then
  begin
    FRT[A]:=true;
    FR[A]:=false;
    if SD[K-1]=A∧K>Z
     then
     begin
       FRH[A]:=false;
```

```
          AP[A]:=AK[A]:=0
        end SD[K-1]=A
        else
        begin
          AP[A]:=APT[A]:=APP[A];
          AK[A]:=AKT[A]:=AKP[A];
          FRH[A]:=true
        end SD[K-1]≠A
      end A≤D
      else
      begin
        R:=A-P;
        H:=SK[R]:=NDEL[WK[R]-R+1-LK[R]];
        KC[R]:=PP[LPP[R]+H];
        KC[R+M]:=C[WK[R-1]+H];
        FKT[R]:=true
      end A>D;
    BD[L]:=A;
    L:=L+1;
    go to KROK1;
KROK4:
    L:=L-1;
    if L=0
     then go to KON;
    A:=BD[L];
    E:=1;
    S:=LSD[L-1];
    for I:=S+E while SD[I]≠A do
     E:=E+1;
    if A≤D
```

```
then
begin
 DET[I]:=INF;
 FR[A]:=true;
 if FRH[A]
  then
  begin
   if DET[I+1]≥INF
    then
    begin
     AP[A]:=APT[A]:=APZ[A];
     AK[A]:=AKT[A]:=AKZ[A]
    end DET[I+1]≥INF
    else
    begin
     FRT[A]:=false;
     AP[A]:=APZ[A];
     AK[A]:=AKZ[A];
     APT[A]:=AKT[A]:=0
    end DET[I+1]<INF
  end FRH[A]
  else go to ALFA
end A≤D
else
begin
 R:=A-P;
 Z:=LK[R]:=LK[R]-1;
 SK[R]:=F:=SO[R];
 H:=0;
 NR2:=LSD[L];
```

```
      for E:=I+H while SD[E]=A∧E≤NR2 do
       H:=H+1;
      DET[E-Z-1]:=INF;
      KC[R]:=PP[LPP[R]+F];
      KC[R+M]:=C[WK[R-1]+F];
      if Z≠0
       then FKT[R]:=false
     end A>D;
   go to KROK3;
KON:
   E:=F:=H:=0;
   for I:=1 step 1 until D do
    begin
     B1:=REP[I]=1;
     B2:=REP[I]=0;
     AP[I]:=if B1 then APZ[I] else if B2 then 0 else APP[I];
     AK[I]:=if B1 then AKZ[I] else if B2 then 0 else AKP[I];
     if AP[I]=0
      then H:=H+1
    end I;
   for I:=1 step 1 until M do
    begin
     E:=REP[D+I];
     KC[I]:=PP[LPP[I]+E];
     KC[M+I]:=C[WK[I-1]+E]
    end;
   for J:=1 step 1 until N do
    LOX[J]:=LXZ[J]:=-1;
   LOX[1]:=LXZ[N]:=0;
   B1:=false;
```

```
CPM(PU,AP,AK,LOX);
for J:=W-H step -1 until 1 do
 begin
  F:=PU[J];
  E:=AP[F];
  MAX:=LXZ[AK[F]];
  if F>P1
   then MAX:=MAX+KC[F-P1];
  if MAX>LXZ[E]
   then LXZ[E]:=MAX
 end J;
for I:=1 step 1 until M do
 begin
  F:=P+I;
  E:=AP[F];
  H:=AK[F];
  SO[I]:=G:=LOX[E];
  MAX:=G+KC[I+M];
  SK[I]:=LOPT-MAX-LXZ[H];
  LK[I]:=REP[D+I]
 end I;
if L≠0
 then go to END
end
end SEQPRO
```

The motivation is as follows. Let $\overline{\langle u, v\rangle} \in K \cap R_{rp}$ be the empty arc which has a minimal delta and for which the implicit condition is satisfied. Then the set K given by (1) can be defined as

$$K = \{\overline{\langle u, v\rangle}\} \cup \{\langle u, v\rangle\} \cup K^{(*)},$$

where $\langle u, v\rangle$ is the reverse arc of $\langle x, y\rangle \in S_r$, and $K^{(*)}$ is the set of all other candidates from K. Eliminating the normal arc from D_{rp}^0 we obtain a new graph D_{sp}^0 of the solution tree H. Let $R_D^{(1)} = \{D_{ij}^0\}_{i\geqslant s, j\geqslant p}$ denote the family of all possible successors of the graph D_{sp}^0 in H, let $R_D^{(2)}$ be the family of all possible successors of the graph D_{rp}^0 which can be obtained by complementing the normal arc $\langle y, x\rangle$, and let $R_D^{(3)}$ denote the family of all possible successors of the graph D_{rp}^0 which can be obtained by eliminating and complementing arcs from the set $K^{(*)}$. Note that for all $D' \in (R_D^{(2)} \cup R_D^{(3)})$ there exists $D'' \in R_D^{(1)}$ such that D'' is a subgraph of D'. Thus the critical path of any graph $R_D^{(2)} \cup R_D^{(3)}$ is as long as the critical path in any graph of $R_D^{(1)}$. Therefore, when backtracking (Step 4) to the predecessor D_{rp}^0 of D_{sp}^0 we abandon all graphs of the family $R_D^{(2)} \cup R_D^{(3)}$ and then we backtrack to the predecessor D_{kp}^0 or D_{rl}^0 of D_{rp}^0.

If $R_{rp} = \emptyset$ or the implicit condition is not satisfied for any arc of R_{rp}, then we choose the arc of K which satisfies the criteria of Step 3 of Grabowski's algorithm.

3. Computational results. The procedure *SEQPRO* has been verified on the ODRA 1305 computer. In all cases the computations started with a pseudo-random initial selection S_1 generated by the procedure *SEQPRO* and with the initial selection of sets S_1^0 containing cousins of minimal lengths. The computation times of all examples are significantly dependent on the problem configuration and on the lengths of cousins. Table 1 shows the results of computations.

Acknowledgements. We are grateful to Dr. J. Grabowski for helpful discussion on the implementation of the algorithm and to Dr. Maciej M. Sysło for help in preparing the paper.

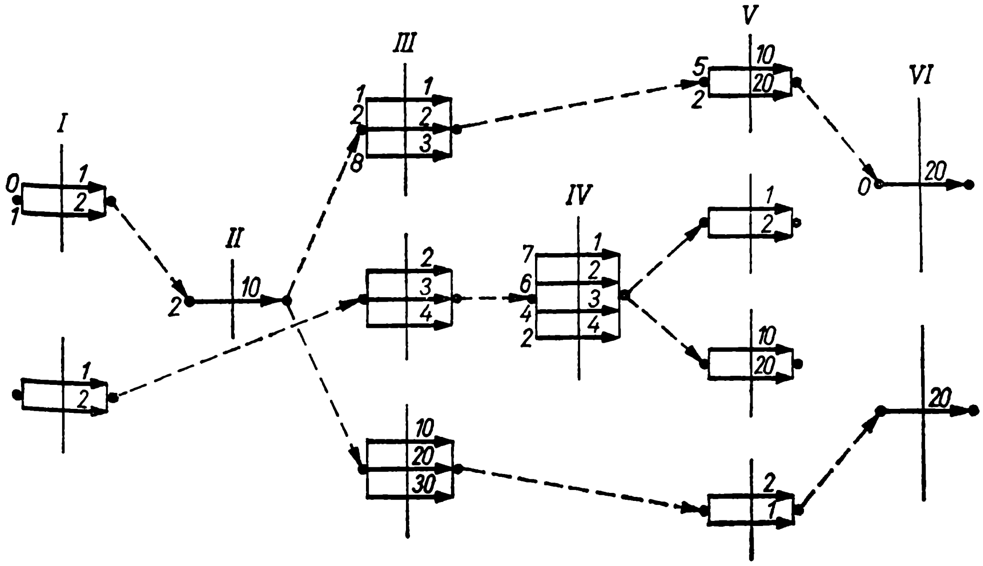

Fig. 1

TABLE 1

No.	Source of examples	Data					Length of a critical path		Number of			Number of optimal iterations	Time (sec.)
		Q	M	T	MC	LIP	starting	optimal	iterations	tests	improvements		
1	Fig. 1 (1)	6	13	11	28	13	62	62	24	12	0	1	15
2	Fig. 1 (2)	6	13	11	28	13	63	63	12	6	0	1	21
3	Fig. 2	3	9	6	27	9	7	6	822	231	1	706	127
4	Fig. 3	4	13	8	30	9	16	15	161	24	1	79	65
5	Fig. 4	2	6	3	6	2	34	31	12	7	1	3	2
6	Fig. 5	2	5	4	17	5	11	7	39	12	1	16	3
7	Fig. 6	5	20	17	43	11	38	38	13	1	0	1	64
8	Fig. 7	4	8	8	19	7	11	8	318	115	3	262	47
9	Fig. 8	4	14	10	38	11	14	9	103	10	1	20	53
10	Fig. 9	4	13	8	13	4	23	13	142	69	6	123	79
11	Fig. 9 (2)	4	13	8	13	4	23	16	166	84	4	8	89

(1) In Figs. 1-9, numbers at the beginning of the arrows indicate the access time of machines, numbers at the arrow-head indicate the processing time of operation, and the type of machine is denoted by Roman numerals.

(2) All the access times are equal to zero.

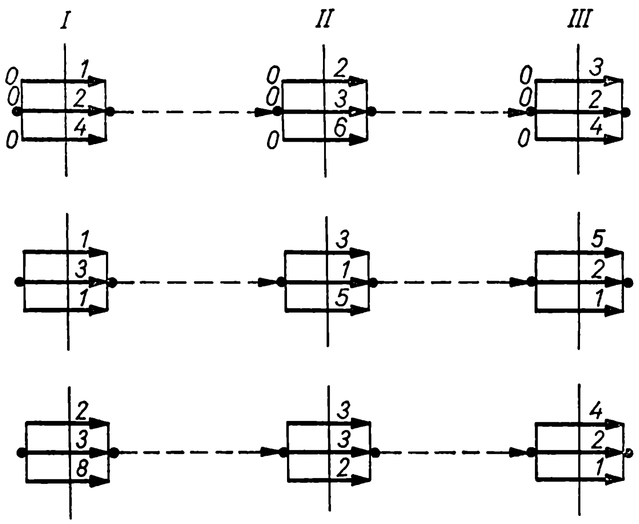

Fig. 2

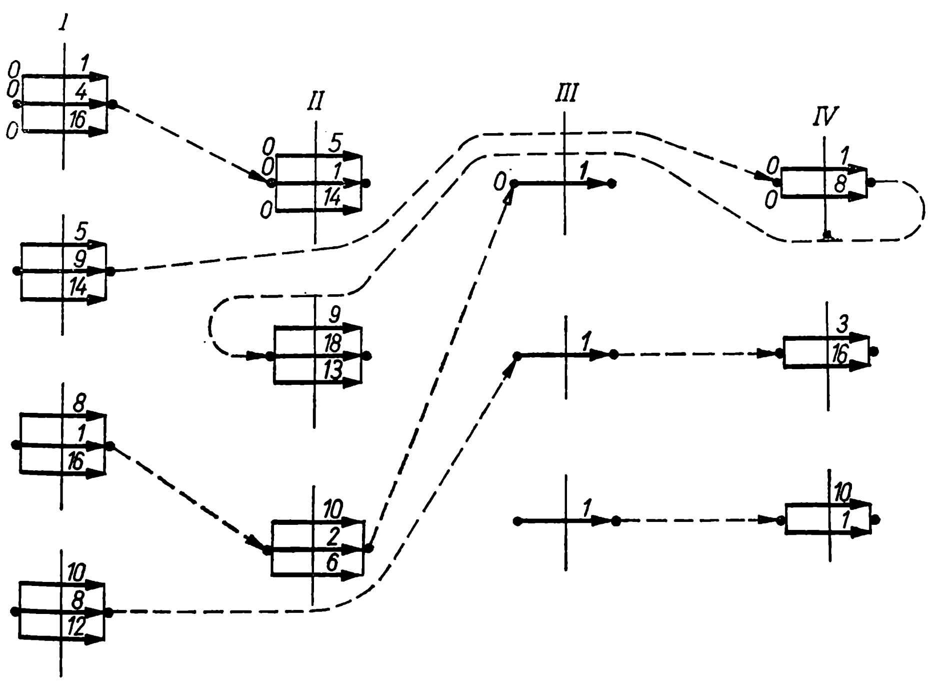

Fig. 3

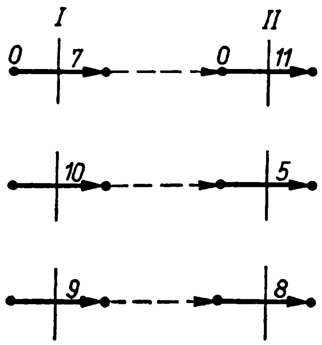

Fig. 4

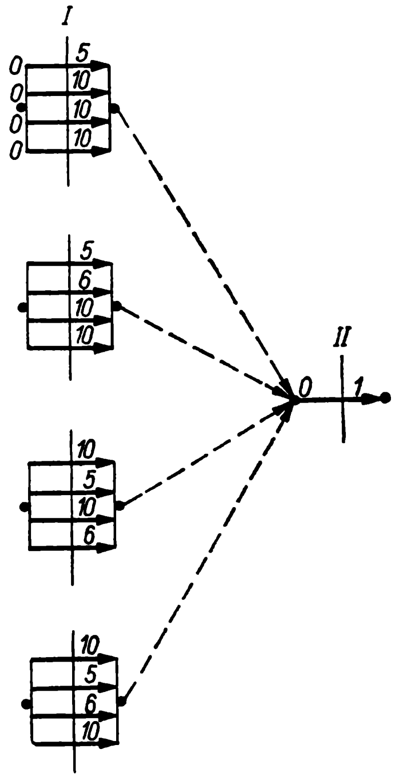

Fig. 5

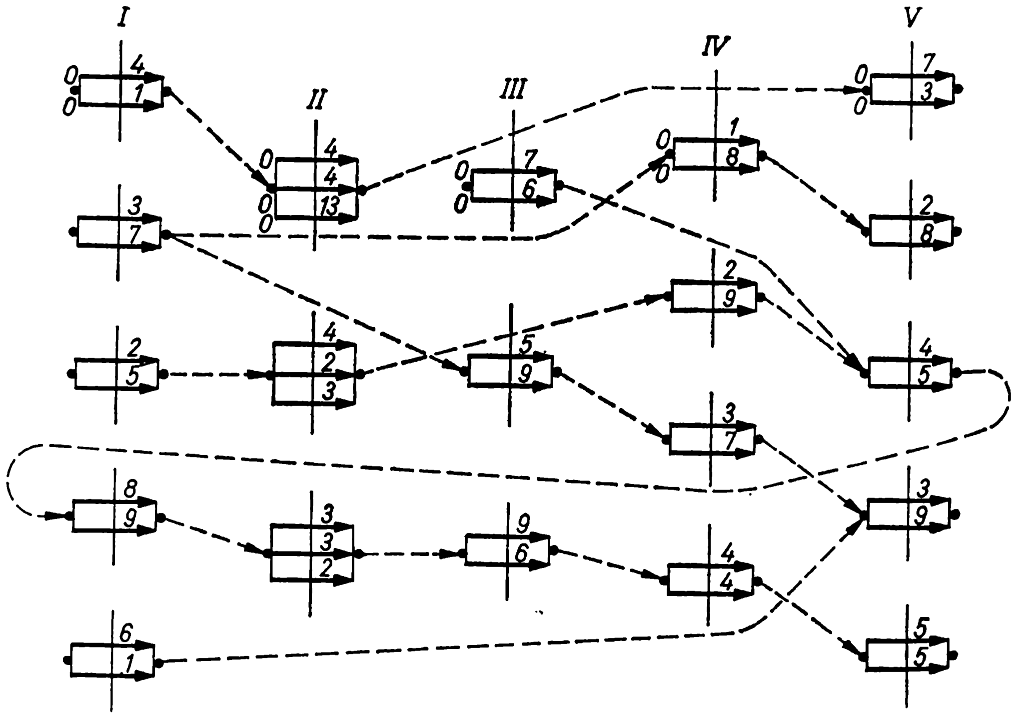

Fig. 6

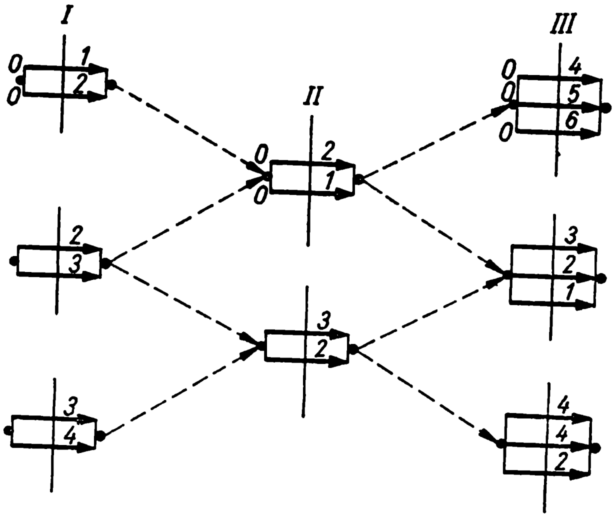
I
0 1
0 2
2
3
3
4
II
0 2
0 1
3
2
III
0 4
0 5
0 6
3
2
1
4
4
2

Fig. 7

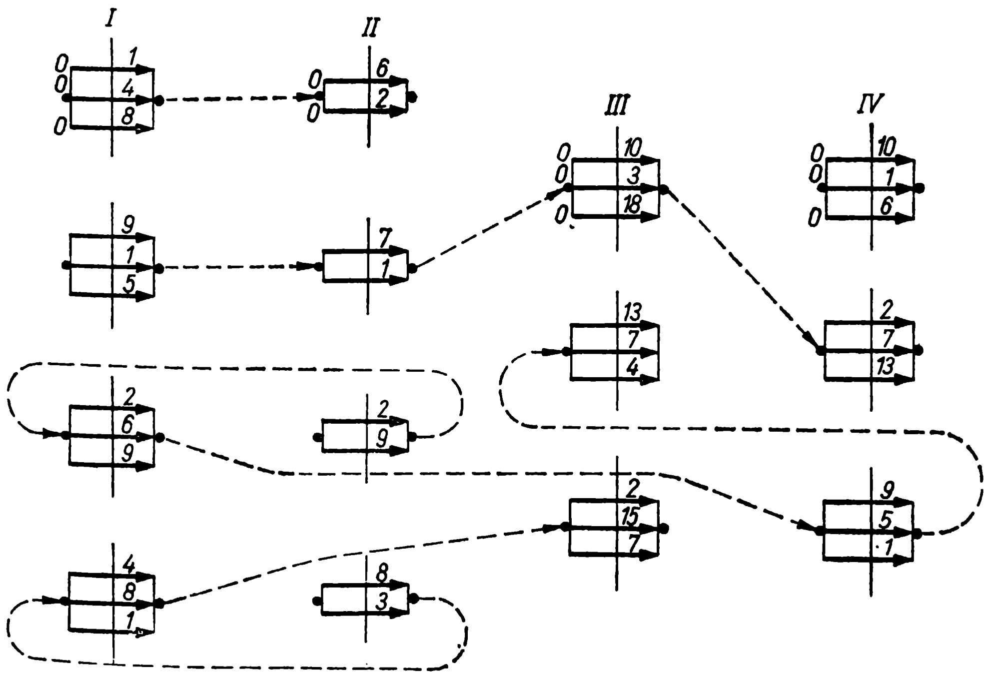
I
0 1
0 4
0 8
9
1
5
2
6
9
4
8
1
II
0 6
0 2
7
1
2
9
8
3
III
0 10
0 3
0 18
13
7
4
2
15
7
IV
0 10
0 1
0 6
2
7
13
9
5
1

Fig. 8

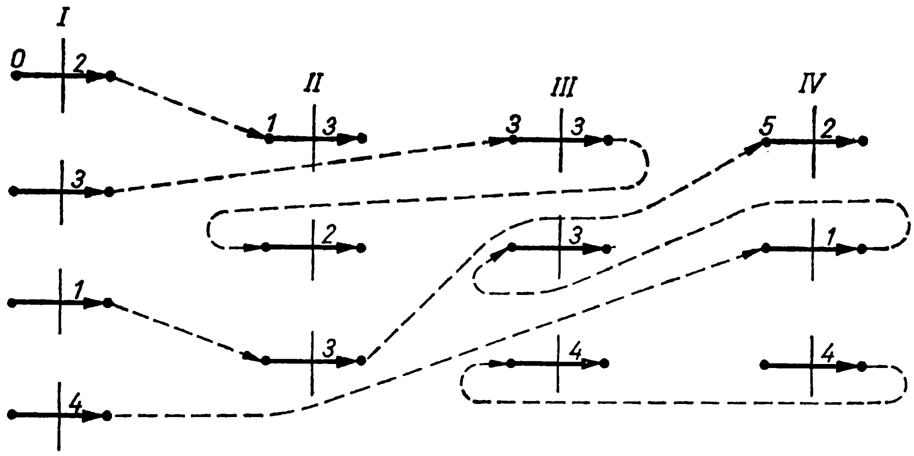

Fig. 9

References

[1] A. Adrabiński and J. Grabowski, *An algorithm of solving the machine sequencing problem*, Zastosow. Matem. 15 (1977), p. 523-542.

[2] E. Balas, *Discrete programming by the filter method*, Opns. Res. 15 (1967), p. 915-967.

[3] — *Machine sequencing via disjunctive graphs — an implicit enumeration algorithm*, ibidem 19 (1969), p. 941-957.

[4] J. Grabowski, *A new formulation and solution of the sequencing problem: algorithm*, Zastosow. Matem. 15 (1977), p. 463-474.

[5] — *Formulation and solution of the sequencing problem with parallel machines*, ibidem 16 (1978), p. 215-232.

INSTITUTE OF COMPUTER SCIENCE
UNIVERSITY OF WROCŁAW
50-384 WROCŁAW

Received on 15. 5. 1977

ALGORYTM 66

A. ADRABIŃSKI i M. WODECKI (Wrocław)

ALGORYTM ROZWIĄZANIA ZAGADNIENIA KOLEJNOŚCIOWEGO ZE ZBIORAMI MASZYN RÓWNOLEGŁYCH

STRESZCZENIE

Procedura *SEQPRO* znajduje taką optymalną kolejność operacji dla zagadnienia kolejnościowego ze zbiorami maszyn równoległych, że całkowity czas wykonania wszystkich operacji jest minimalny.

Dane:

- Q — liczba zbiorów maszyn równoległych;
- M — liczba operacji;
- T — liczba par operacji, wyrażających wymagania technologiczne porządku operacji;
- MC — liczba kuzynów wszystkich zbiorów maszyn równoległych;
- LIP — liczba maszyn;
- INF — maksymalna liczba dodatnia typu **integer**;
- $MAKS$ — maksymalna liczba iteracji algorytmu;
- $NI[1:Q]$ — tablica liczb operacji; $NI[k]$ jest liczbą operacji, które mogą być wykonane na maszynach z k-tego zbioru maszyn równoległych;
- $BK[1:Q]$ — tablica liczb maszyn; $BK[k]$ jest liczbą maszyn w k-tym zbiorze maszyn równoległych;
- $RTP[1:T]$, $RTK[1:T]$ — tablice zawierające pary operacji, które wyrażają wymagania technologiczne porządku operacji; tablice RTP i RTK zawierają odpowiednio numery poprzedników i następników każdej pary operacji;
- $C[1:MC]$ — tablica czasów wykonania operacji; $C[(i-1)\times BK[j]+1: i\times BK[j]]$ są czasami wykonania i-tej operacji ($i = N[j-1]+1, N[j-1]+2, \ldots, N[j]$) na maszynach z j-tego zbioru maszyn równoległych ($j = 1, 2, \ldots, Q$ oraz $N[0] = 0$);
- $PP[1:LIP]$ — tablica czasów dostępu maszyn; $PP[i]$ jest momentem czasu, od którego i-ta maszyna jest dostępna.

Wyniki:

- $LK[1:M]$ — tablica numerów kuzynów w optymalnej reprezentacji;
- $SO[1:M]$ — tablica najwcześniejszych momentów rozpoczęcia operacji dla optymalnej kolejności ich wykonania;
- $SK[1:M]$ — tablica ogólnych rezerw czasów wykonania operacji;
- $LOPT$ — całkowity czas trwania wszystkich operacji.

Inne parametry:

- END — etykieta, do której następuje skok z procedury *SEQPRO*, jeżeli liczba iteracji $MAKS$ jest mniejsza niż wymagana przez algorytm.

Procedura *SEQPRO* realizuje algorytm Grabowskiego [5] z małymi poprawkami. Działanie procedury sprawdzono na maszynie ODRA 1305. Obliczenia kontrolne, przeprowadzone dla wielu przykładów, wykazały poprawność przedstawionej procedury.