stichting
mathematisch
centrum

MC

AFDELING MATHEMATISCHE BESLISKUNDE BW 137/81 MAART
(DEPARTMENT OF OPERATIONS RESEARCH)

B.J. LAGEWEG, E.L. LAWLER, J.K. LENSTRA, A.H.G. RINNOOY KAN

COMPUTER AIDED COMPLEXITY CLASSIFICATION OF COMBINATORIAL PROBLEMS

Preprint

kruislaan 413 1098 SJ amsterdam

Printed at the Mathematical Centre, 413 Kruislaan, Amsterdam.

1980 Mathematics Subject Classification: 90B35, 68C25

COMPUTER AIDED COMPLEXITY CLASSIFICATION OF COMBINATORIAL PROBLEMS

B.J. LAGEWEG (*Mathematisch Centrum, Amsterdam*)
E.L. LAWLER (*University of California, Berkeley*)
J.K. LENSTRA (*Mathematisch Centrum, Amsterdam*)
A.H.G. RINNOOY KAN (*Erasmus University, Rotterdam*)

ABSTRACT

We describe a computer program that has been used to maintain a record of the known complexity results for a class of 4,536 machine scheduling problems. The input of the program consists of a listing of known "easy" problems and a listing of known "hard" problems. The program employs the structure of the problem class to determine the implications of these results. The output provides a listing of essential results in the form of maximal easy and minimal hard problems as well as listings of minimal and maximal open problems, which are helpful in indicating the direction of future research. The application of the program to a restricted class of 120 single-machine problems is demonstrated. Possible refinements and extensions to other research areas are suggested. It is also shown that the problem of determining the minimum number of results needed to resolve the status of all remaining open problems in a complexity classification such as ours, is itself a hard problem.

KEY WORDS & PHRASES: *combinatorial optimization, single-machine scheduling, computational complexity, polynomial algorithm, NP-hardness, computer program.*
NOTE: This report will be submitted for publication in a journal.

1. INTRODUCTION

For several years the authors have been investigating the computational complexity of deterministic machine scheduling problems with the objective of delineating, as precisely as possible, the erratic boundary between "easy" and "hard" problem types. We say that a problem is *easy* when it can be solved in polynomial time; by *hard* we mean NP-hard.

There are 4,536 members in the class of combinatorial optimization problems encompassed by our investigation. Simply maintaining a manual record of known complexity results is a tedious and time-consuming task. Consequently, we developed a computer program, MSPCLASS, for this purpose. Its input consists of a listing of known easy problems and a listing of known hard problems. The program systematically employs a certain *partial ordering* defined on the problem class not only to derive the status of each problem (*easy*, *open* or *hard*), but also to determine four subclasses of *maximal easy*, *minimal open*, *maximal open* and *minimal hard* problems. The output listings produced by the program have proved to be of great benefit.

We describe the program MSPCLASS in general terms in Section 2. Next, we define a class of 120 single-machine scheduling problems in Section 3, and demonstrate the application of the program to this restricted class in Section 4. (The state of the art for the complete class is given in [11]; a preliminary report for a class of 4,158 problems was presented in [10].) We suggest possible refinements and extensions to other research areas in Section 5.

One refinement of a program like MSPCLASS would be for it to determine the minimum number of research results that would completely resolve the status of all remaining open problems in the class under consideration. In the Appendix, we show that this problem is itself NP-hard.

2. THE PROGRAM MSPCLASS

We assume that the reader is familiar with the basic concepts of computational complexity theory and that we can dispense with a discussion of certain well-known definitions. For an excellent survey of the theory of NP-completeness, see [5].

The program MSPCLASS deals with a well-defined class S of problem types. For any two problems $P,P' \in S$, the program is able to determine whether or not $P \rightarrow P'$, where $\rightarrow$ is a certain given *partial ordering*. This relation has the property that
(i) if $P \rightarrow P'$ and P' is easy, then P is easy, and
(ii) if $P \rightarrow P'$ and P is hard, then P' is hard.
For example, $P \rightarrow P'$ may hold because it is evident from the definitions of P and P' that the set of instances of P is included in the set of instances of P'. In general, $P \rightarrow P'$ implies that P is *reducible* to P'.

The input of the program consists of research results in the form of a class I^{*} of known easy problems and a class $I^{!}$ of known hard problems. The program then partitions the class S into three classes of easy, hard and open problems:

$$S^{*} = \{P \in S \mid \exists\, P' \in I^{*}: P \rightarrow P'\},$$
$$S^{!} = \{P \in S \mid \exists\, P' \in I^{!}: P' \rightarrow P\},$$
$$S^{?} = S - (S^{*} \cup S^{!}).$$

(Obviously, S^{*} and $S^{!}$ are disjoint; if not, either a mistake has been made or a very surprising result has been obtained.) The program also determines four subclasses of problems that are *minimal* or *maximal* with respect to the relation $\rightarrow$:

$$S^{*}_{\max} = \{P \in S^{*} \mid \neg \exists\, P' \in S^{*}-\{P\}: P \rightarrow P'\},$$
$$S^{?}_{\min} = \{P \in S^{?} \mid \neg \exists\, P' \in S^{?}-\{P\}: P' \rightarrow P\},$$
$$S^{?}_{\max} = \{P \in S^{?} \mid \neg \exists\, P' \in S^{?}-\{P\}: P \rightarrow P'\},$$
$$S^{!}_{\min} = \{P \in S^{!} \mid \neg \exists\, P' \in S^{!}-\{P\}: P' \rightarrow P\}.$$

The output of the program provides a count of the membership of S^{*}, $S^{?}$ and $S^{!}$ as well as complete listings of $S^{*}_{\max}$, $S^{?}_{\min}$, $S^{?}_{\max}$ and $S^{!}_{\min}$.

The sizes of S^{*}, $S^{?}$ and $S^{!}$ indicate a score by which the progress of the research can be judged. The problems in $S^{*}_{\max}$ and $S^{!}_{\min}$ might be said to represent the *essential results* since all other known results can be deduced from them by invoking the relation $\rightarrow$; accordingly, the minimum input is given by

$I^* = S^*_{\max}$ and $I^! = S^!_{\min}$. The members of $S^?_{\min}$ and $S^?_{\max}$ are suitable targets for further research.

Remark. For the purpose of defining the classes S^*, $S^?$ and $S^!$, it is not necessary for the relation $\rightarrow$ to be a partial ordering. However, if $\rightarrow$ is not a partial ordering, the above definitions of the subclasses $S^*_{\max}$, $S^?_{\min}$, $S^?_{\max}$ and $S^!_{\min}$ will not suit our purposes. There may be certain sets of problems of equivalent complexity with respect to the relation $\rightarrow$; they correspond to strongly connected components of the digraph with node set S and arcs for $\rightarrow$. All members of such a set may, for example, be maximal easy problems, and yet none belongs to $S^*_{\max}$, as defined.

3. A CLASS OF SINGLE-MACHINE PROBLEMS

Because the class of machine scheduling problems under study is so large and unwieldy to describe, we have chosen to demonstrate the application of the program MSPCLASS to a restricted class of 120 single-machine problems. Even this requires a certain amount of explanation and a number of definitions.

Generally speaking, each instance of a single-machine problem calls for an optimal feasible schedule to be found for a set of n *jobs* J_j $(j = 1,\ldots,n)$. A *schedule* consists of a set of nonoverlapping time intervals and the designation of the job which the machine is to process in each interval. *Feasibility* of a schedule is determined by various specified parameter values and conditions. *Optimality* is judged with reference to a given objective function.

More particularly, each problem instance specifies the following information, either explicitly or implicitly.

(1) For each job J_j $(j = 1,\ldots,n)$, a nonnegative integer *processing requirement* p_j is given. In order to be feasible, a schedule must provide an amount of processing for each J_j equal to p_j. For any schedule, let S_j and C_j denote the *start time* and *completion time* of J_j respectively, *i.e.*, the time at which J_j is first (last) processed.

(2) If the schedule is required to be *nonpreemptive*, then each J_j must be processed continuously from S_j until $C_j = S_j+p_j$. If the schedule is allowed to be *preemptive*, then the processing of any job may arbitrarily often be interrupted and resumed at a later time, without penalty.

(3) For each J_j, a nonnegative integer *release date* r_j is given. A feasible schedule must be such that $r_j \leq S_j$ for each J_j.

(4) *Precedence constraints* between the jobs are given in the form of an acyclic digraph $G = (\{1,\ldots,n\},A)$. A feasible schedule must be such that $C_j \leq S_k$ whenever $(j,k) \in A$.

(5) With respect to the optimality criterion, a nonnegative integer *due date* d_j and/or an integer *weight* w_j may be given for each J_j. For any schedule, let $L_j = C_j - d_j$ denote the *lateness* of J_j, $T_j = \max\{0, C_j - d_j\}$ its *tardiness*, and $U_j = 0$ if $C_j \leq d_j$, $U_j = 1$ if $C_j > d_j$ its *unit penalty*. The *objective function*, which is to be minimized, is specified to be one of the following eight types:

- the *maximum completion time* $C_{\max} = \max_j\{C_j\}$;
- the *maximum lateness* $L_{\max} = \max_j\{L_j\}$;
- the *total completion time* $\sum_j C_j$;
- the *total weighted completion time* $\sum_j w_j C_j$;
- the *total tardiness* $\sum_j T_j$;
- the *total weighted tardiness* $\sum_j w_j T_j$;
- the *number of late jobs* $\sum_j U_j$;
- the *weighted number of late jobs* $\sum_j w_j U_j$.

In this class of scheduling problems, each problem type is defined by a sextuple $(\pi_0, \pi_1, \pi_2, \pi_3, \pi_4, \pi_5)$, where each component indicates a property shared by all problem instances. The component π_0 specifies the *machine environment*, the components $\pi_1, \pi_2, \pi_3, \pi_4$ indicate *job characteristics*, and the component π_5 refers to the *objective function*. These components take on mnemonically encoded values as follows. Let $\circ$ denote the empty symbol.

(0) $\pi_0 = 1$: single-machine.

(1) $\pi_1 \in \{p_j = 1, \circ\}$, where

$\pi_1 = p_j = 1$: all $p_j = 1$ $(j = 1,\ldots,n)$;

$\pi_1 = \circ$: the p_j are arbitrary nonnegative integers.

(2) $\pi_2 \in \{\circ, \mathit{pmtn}\}$, where

$\pi_2 = \circ$: preemption is not allowed;

$\pi_2 = \mathit{pmtn}$: preemption is permitted.

(3) $\pi_3 \in \{\circ, r_j\}$, where

$\pi_3 = \circ$: all $r_j = 0$ $(j = 1,\ldots,n)$;

$\pi_3 = r_j$: the r_j are arbitrary nonnegative integers.

(4) $\pi_4 \in \{\circ, tree, prec\}$, where

$\pi_4 = \circ$: G has no arcs (the jobs are independent);

$\pi_4 = tree$: G has either indegree at most one for each vertex or outdegree at most one for each vertex;

$\pi_4 = prec$: G is an arbitrary acyclic digraph.

(5) $\pi_5 \in \{C_{max}, L_{max}, \sum C_j, \sum w_jC_j, \sum T_j, \sum w_jT_j, \sum U_j, \sum w_jU_j\}$.

There are thus $1\times2\times2\times2\times3\times8 = 192$ problem types. For human consumption, they are written in the form $1|\pi_4,\pi_3,\pi_2,\pi_1|\pi_5$ (*cf.* [6]).

Several of the properties defined by the components of the sextuple are simple generalizations of others. For example, *prec* is a generalization of *tree*, and $\sum w_jC_j$ and $\sum T_j$ are generalizations of $\sum C_j$. This suggests an approach to the construction of the relation $\rightarrow$.

The possible values of the component π_i correspond to the vertices of the digraph G_i shown in Figure 1 $(i = 1,\ldots,5)$. There is an arc from π to π' if π' is a direct generalization of π. For two problems $P = 1|\pi_4,\pi_3,\pi_2,\pi_1|\pi_5$ and $P' = 1|\pi_4',\pi_3',\pi_2',\pi_1'|\pi_5'$, we have $P \rightarrow P'$ if either $\pi_i = \pi_i'$ or G_i contains a directed path from π_i to π_i', for $i = 1,\ldots,5$.

The relation $\rightarrow$ can be augmented to take into account relationships between problem types that are subtler than simple generalization. The dashed arcs in G_5 from L_{max} to $\sum T_j$ and $\sum U_j$ can be added, by the following argument. For any instance of a problem with the L_{max} objective function, the minimum value of L_{max} is equal to the minimum value of L for which there exists a schedule with $\sum T_j' = \sum U_j' = 0$, where T_j' and U_j' are defined with respect to due dates $d_j' = d_j+L$. The minimum L can be found by a bisection search, requiring a polynomially bounded number of applications of any algorithm capable of solving the problem with L_{max} replaced by $\sum T_j$ or $\sum U_j$. This establishes the (Turing) reducibility from any L_{max} problem to the corresponding $\sum T_j$ and $\sum U_j$ problems.

A further augmentation of the relation $\rightarrow$ is provided by some observations about preemption. First, in preemptive single-machine scheduling there is no reason to interrupt the processing of a job other than at an integer point in time. Secondly, there is no advantage to preemption if all release dates are equal. It follows that, if $\pi_1 = p_j{=}1$ or $\pi_3 = \circ$, we may as well assume $\pi_2 = \circ$. This leaves only five relevant combinations of values for the components π_1, π_2 and π_3. They correspond to the vertices π_6 of the digraph G_6 shown in

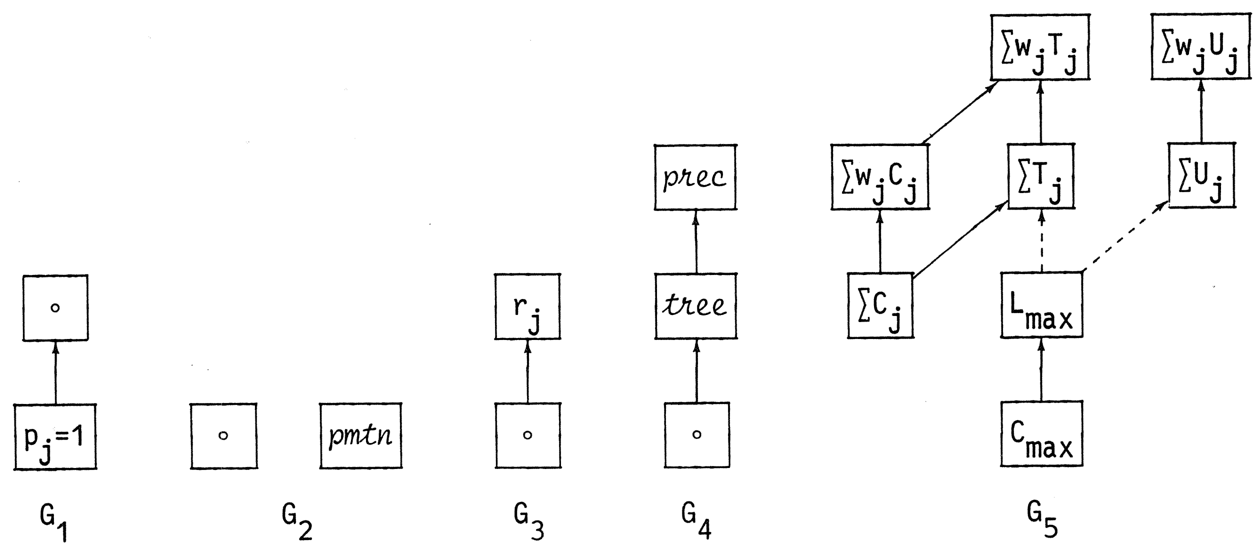


Figure 1. The graphs G_i $(i = 1,\ldots,5)$.

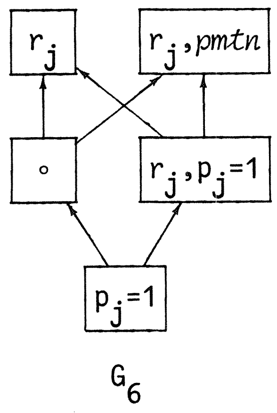


Figure 2. The graph G_6.

Figure 2, with arcs derived from G_1, G_2 and G_3. The total number of distinguishable problem types is now reduced from 192 to 120, each being written in the form $1|\pi_4,\pi_6|\pi_5$.

4. APPLICATION OF MSPCLASS TO THE CLASS OF SINGLE-MACHINE PROBLEMS

The program MSPCLASS has been implemented in PASCAL on the Control Data Cyber 170-750 of the SARA Computing Centre in Amsterdam. Its application to the class of single-machine scheduling problems defined in the previous section is demonstrated below.

We note that separate runs are made for two different inputs, corresponding to different assumptions about the encoding of problem instances containing numerical data. Under a standard *binary* encoding, the easy problems are solvable in *strictly polynomial* time and the hard problems are NP-hard in the *ordinary* sense. Under a *unary* encoding, the easy problems are solvable in *pseudopolynomial* time and the hard problems are NP-hard in the *strong* sense (*cf.* [4;18]).

The output of the latest runs is shown in Figures 3 and 4. The listings present useful additional information in the form of references to the literature where the results in question can be found, an indication of general algorithmic techniques applicable to some easy problems, the symbol # for open problems that are both minimal and maximal, and acronyms of convenient "starting problems" for transformations in the case of hard problems.

Combining the results for binary and unary encodings, we get six classes of problems:

- unary and binary easy (51 problems);
- unary and binary open (4 problems, all involving the $\sum T_j$ criterion);
- unary and binary hard (62 problems);
- unary easy, binary open (1 problem: $1||\sum T_j$);
- unary easy, binary hard (2 problems: $1||\sum w_jU_j$ and $1|r_j,pmtn|\sum w_jU_j$);
- unary open, binary hard (0 problems).

We shall be happy to award a suitable prize to the first person resolving the status of any one of the open problems: a bottle of California champagne in case of a polynomial algorithm, a Dutch cheese in case of an NP-hardness proof.

5. CONCLUDING REMARKS

We have found the program MSPCLASS of great help in recording the state of the art in machine scheduling theory, an area of very rapid development in recent years. The program has been useful in interpreting the implications of new results and also in guiding our research efforts.

The system lends itself to many refinements, such as the elaboration of the relation $\rightarrow$ suggested in Section 2 and a further analysis of the class of

```
SINGLE MACHINE SCHEDULING, BINARY ENCODING

DATE: 81:03:05

NUMBER OF PROBLEMS          MINIMAL      TOTAL   MAXIMAL

EASY                                       51         8
OPEN                              2         5         3
HARD                             12        64

TOTAL                                     120

MAXIMAL EASY PROBLEMS

1/RJ,PJ=1/SUMWJTJ            LAGEWEG (TRANSPORTATION PROBLEM)
1/RJ,PJ=1/SUMWJUJ            LAGEWEG (TRANSPORTATION PROBLEM)
1/RJ,PMTN/SUMCJ              BAKER 1974
1/RJ,PMTN/SUMUJ              LAWLER 1981 (DYNAMIC PROGRAMMING)
1/TREE/SUMWJCJ               HORN 1972; SIDNEY 1975
1/PREC,RJ,PJ=1/SUMCJ         LAWLER: COFFMAN GRAHAM 1972
1/PREC,RJ,PMTN/LMAX          BLAZEWICZ 1976
1/PREC,RJ/CMAX               LAWLER 1973

MINIMAL OPEN PROBLEMS        MAXIMAL OPEN PROBLEMS

1//SUMTJ                     1/RJ,PMTN/SUMTJ
1/TREE,PJ=1/SUMTJ            1/TREE,RJ,PJ=1/SUMTJ
                             1/TREE/SUMTJ

MINIMAL HARD PROBLEMS

1//SUMWJTJ                   3PT  LAWLER 1977; LENSTRA RINNOOY KAN BRUCKER 1977
1//SUMWJUJ                   KS   KARP 1972
1/RJ,PMTN/SUMWJCJ            3PT  LABETOULLE LAWLER LENSTRA RINNOOY KAN 1979
1/RJ/LMAX                    3PT  LENSTRA RINNOOY KAN BRUCKER 1977
1/RJ/SUMCJ                   3PT  LENSTRA RINNOOY KAN BRUCKER 1977
1/TREE,PJ=1/SUMWJTJ          3PT  LENSTRA RINNOOY KAN 1980
1/TREE,PJ=1/SUMUJ            S3P  LENSTRA RINNOOY KAN 1980
1/TREE,RJ,PJ=1/SUMWJCJ       3PT  LENSTRA RINNOOY KAN 1980
1/TREE,RJ,PMTN/SUMCJ         3PT  LENSTRA 1980
1/PREC,PJ=1/SUMWJCJ          LA   LAWLER 1978; LENSTRA RINNOOY KAN 1978
1/PREC,PJ=1/SUMTJ            CL   LENSTRA RINNOOY KAN 1978
1/PREC/SUMCJ                 LA   LAWLER 1978; LENSTRA RINNOOY KAN 1978
```

Figure 3.

open problems. For example, one might like the program to determine the minimum number of research results that would completely resolve the status of all remaining open problems. This refinement is likely to be difficult to implement, since the problem to be solved is itself NP-hard, as we show in

```
SINGLE MACHINE SCHEDULING, UNARY ENCODING

DATE: 81:03:05

NUMBER OF PROBLEMS           MINIMAL     TOTAL   MAXIMAL

EASY                                      54         8
OPEN                               2       4         3
HARD                              11      62

TOTAL                                    120

MAXIMAL EASY PROBLEMS

1/RJ,PJ=1/SUMWJTJ             LAGEWEG (TRANSPORTATION PROBLEM)
1//SUMTJ                      LAWLER 1977
1/RJ,PMTN/SUMCJ               BAKER 1974
1/RJ,PMTN/SUMWJUJ             LAWLER 1981 (DYNAMIC PROGRAMMING)
1/TREE/SUMWJCJ                HORN 1972; SIDNEY 1975
1/PREC,RJ,PJ=1/SUMCJ          LAWLER: COFFMAN GRAHAM 1972
1/PREC,RJ,PMTN/LMAX           BLAZEWICZ 1976
1/PREC,RJ/CMAX                LAWLER 1973

MINIMAL OPEN PROBLEMS         MAXIMAL OPEN PROBLEMS

1/RJ,PMTN/SUMTJ             # 1/RJ,PMTN/SUMTJ
1/TREE,PJ=1/SUMTJ             1/TREE,RJ,PJ=1/SUMTJ
                              1/TREE/SUMTJ

MINIMAL HARD PROBLEMS

1//SUMWJTJ                    3PT  LAWLER 1977; LENSTRA RINNOOY KAN BRUCKER 1977
1/RJ,PMTN/SUMWJCJ             3PT  LABETOULLE LAWLER LENSTRA RINNOOY KAN 1979
1/RJ/LMAX                     3PT  LENSTRA RINNOOY KAN BRUCKER 1977
1/RJ/SUMCJ                    3PT  LENSTRA RINNOOY KAN BRUCKER 1977
1/TREE,PJ=1/SUMWJTJ           3PT  LENSTRA RINNOOY KAN 1980
1/TREE,PJ=1/SUMUJ             S3P  LENSTRA RINNOOY KAN 1980
1/TREE,RJ,PJ=1/SUMWJCJ        3PT  LENSTRA RINNOOY KAN 1980
1/TREE,RJ,PMTN/SUMCJ          3PT  LENSTRA 1980
1/PREC,PJ=1/SUMWJCJ           LA   LAWLER 1978; LENSTRA RINNOOY KAN 1978
1/PREC,PJ=1/SUMTJ             CL   LENSTRA RINNOOY KAN 1978
1/PREC/SUMCJ                  LA   LAWLER 1978; LENSTRA RINNOOY KAN 1978
```

Figure 4.

the Appendix. This complexity result can be interpreted as a theorem concerning the difficulty of determining the minimum cost of completing a research project.

Even without such refinements, we believe than an extension of our approach to other large classes of combinatorial optimization problems could yield comparable benefits. Resource constrained scheduling, vehicle routing, location and allocation, and network design are possible areas that come to mind.

APPENDIX. DETERMINING A MINIMUM COMPLETE RESEARCH PROGRAM

Suppose we are given the state of the art in the form of an acyclic digraph $G = (S,\rightarrow)$ whose nodes are labeled *easy*, *open* and *hard* in such a way that
(i) if $P \rightarrow P'$ and P' is labeled *easy*, then P is labeled *easy*, and
(ii) if $P' \rightarrow P$ and P' is labeled *hard*, then P is labeled *hard*.
A *complete research program* (CRP) for G is defined as a subset of *open* nodes with the property that it is possible to relabel each of them either *easy* or *hard* in such a way that application of (i) and (ii) allows us to relabel each other *open* node either *easy* or *hard* as well. We are interested in determining a CRP of minimum cardinality.

At first glance, this problem seems to be closely related to a known easy problem. If we are asking for the minimum number of *open* nodes that must be relabeled *solved* such that for each other *open* node P there exists a *solved* node P' with $P \rightarrow P'$ or $P' \rightarrow P$, then Menger's Theorem could be applied to the transitive reduction of G to answer the question in polynomial time. However, we are asking for the minimum number of *open* nodes that must be relabeled either *easy* or *hard* such that for each other *open* node P there exists either an *easy* node P' with $P \rightarrow P'$ or a *hard* node P' with $P' \rightarrow P$. This is, in fact, an NP-hard problem.

We will prove this assertion by means of a transformation from the following NP-complete problem:

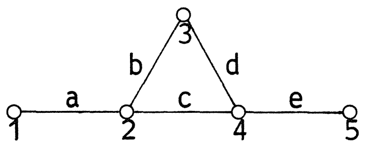
VERTEX COVER [5,[GT1]]: Given a graph $G = (V,E)$ and an integer k, does V include a subset U of size at most k such that every edge in E is incident with at least one vertex in U?

Let $\ell = |V|+k$. Given any instance of VERTEX COVER, we construct a digraph $G = (S,\rightarrow)$ as follows (*cf.* Figure 5):

- for each vertex $v \in V$, there are $\ell+2$ nodes $v_1,\ldots,v_\ell,\bar{v},\bar{\bar{v}} \in S$ and $\ell+1$ arcs $v_1 \rightarrow \bar{v}$, ..., $v_\ell \rightarrow \bar{v}$, $\bar{v} \rightarrow \bar{\bar{v}}$;

VERTEX COVER

Instance: G = (V,E):

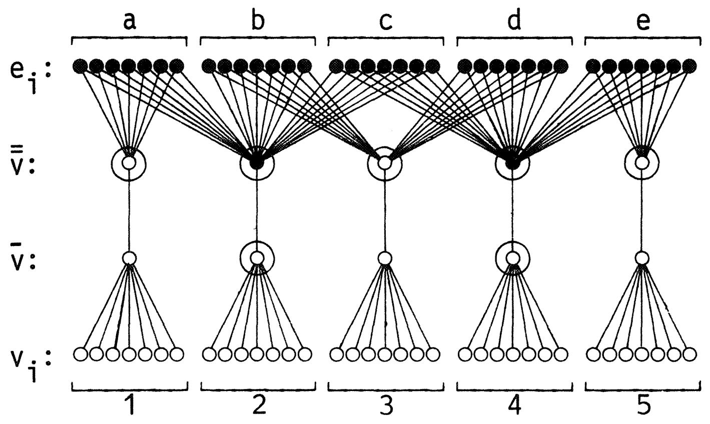


$k = 2$.

Solution: $U = \{2,4\}$.

CRP PROBLEM

Instance: $\mathcal{G} = (\mathcal{S},\rightarrow)$ (all arcs are directed upwards; arcs implied by transitivity are not shown):



$\ell = 7$.

Solution: CRP = {◎,◉}; ○: *easy*; ●: *hard*.

Figure 5. Illustration of the transformation.

- for each edge $e = \{v,w\} \in E$, there are ℓ nodes $e_1,\ldots,e_\ell \in S$ and 2ℓ arcs $\bar{\bar{v}} \rightarrow e_1$, ..., $\bar{\bar{v}} \rightarrow e_\ell$, $\bar{\bar{w}} \rightarrow e_1$, ..., $\bar{\bar{w}} \rightarrow e_\ell$;
- all other arcs are implied by transitivity;
- all nodes in S are labeled *open*.

We claim that VERTEX COVER has a solution if and only if there exists a CRP of size at most ℓ.

Suppose that G has a vertex cover $U \subseteq V$ of size at most k. A CRP of size at most ℓ is then obtained by relabeling each $\bar{\bar{v}}$ ($v \in U$) *hard* and relabeling each $\bar{\bar{v}}$ ($v \in V-U$) and each $\bar{v}$ ($v \in U$) *easy*; application of (i) and (ii) allows us to relabel each e_i ($i = 1,\ldots,\ell$; $e \in E$) *hard* and each $\bar{v}$ ($v \in V-U$) and v_i ($i = 1,\ldots,\ell$; $v \in V$) *easy*.

Conversely, suppose that G has a CRP $\subset S$ of size at most ℓ. It is easily argued that, due to the choice of ℓ, the nodes v_i and e_i $(i = 1,\ldots,\ell)$ can only be relabeled *easy* and *hard* respectively, and, moreover, that none of these nodes is contained in the CRP. For each $v \in V$, the CRP contains either one node from $\{\bar{v},\bar{\bar{v}}\}$ or both. In the former case, this has to be $\bar{\bar{v}}$, labeled *easy*. In the latter case, $\bar{v}$ is labeled *easy* and $\bar{\bar{v}}$ is labeled *hard*. The *hard* nodes $\bar{\bar{v}} \in S$, of which there can be no more than $\ell - |V| = k$, allow us to relabel each e_i $(i = 1,\ldots,\ell;\ e \in E)$ *hard* as well. It follows that the corresponding vertices $v \in V$ constitute a vertex cover of size at most k. This completes the proof.

ACKNOWLEDGMENTS

The authors are indebted to R.M. Karp for his active participation in the creation of this project. This research was partially supported by NATO Special Resoarch Grant 9.2.02 (SRG.7) and by NSF Grant MCS78-20054.

REFERENCES

1. K.R. BAKER (1974) *Introduction to Sequencing and Scheduling*, Wiley, New York.
2. J. BLAZEWICZ (1976) Scheduling dependent tasks with different arrival times to meet deadlines. In: E. GELENBE (ed.) (1976) *Modelling and Performance Evaluation of Computer Systems*, North-Holland, Amsterdam, 57-65.
3. E.G. COFFMAN, JR., R.L. GRAHAM (1972) Optimal scheduling for two-processor systems. *Acta Informat.* 1,200-213.
4. M.R. GAREY, D.S. JOHNSON (1978) "Strong" NP-completeness results: motivation, examples and implications. *J. Assoc. Comput. Mach.* 25,499-508.
5. M.R. GAREY, D.S. JOHNSON (1979) *Computers and Intractability: a Guide to the Theory of NP-Completeness*, Freeman, San Francisco.
6. R.L. GRAHAM, E.L. LAWLER, J.K. LENSTRA, A.H.G. RINNOOY KAN (1979) Optimization and approximation in deterministic sequencing and scheduling: a survey. *Ann. Discrete Math.* 5,287-326.

7. W.A. HORN (1972) Single-machine job sequencing with treelike precedence ordering and linear delay penalties. *SIAM J. Appl. Math.* 23,189-202.

8. R.M. KARP (1972) Reducibility among combinatorial problems. In: R.E. MILLER, J.W. THATCHER (eds.) (1972) *Complexity of Computer Computations*, Plenum Press, New York, 85-103.

9. J. LABETOULLE, E.L. LAWLER, J.K. LENSTRA, A.H.G. RINNOOY KAN (1979) Preemptive scheduling of uniform machines subject to release dates. Report BW 99, Mathematisch Centrum, Amsterdam.

10. B.J. LAGEWEG, E.L. LAWLER, J.K. LENSTRA (1976) Machine scheduling problems: computations, complexity and classification; in honour of A.H.G. Rinnooy Kan upon the occasion of the defense of his doctoral thesis, January 28, 1976. Report BN 30, Mathematisch Centrum, Amsterdam (out of print).

11. B.J. LAGEWEG, E.L. LAWLER, J.K. LENSTRA, A.H.G. RINNOOY KAN (1981) Computer aided complexity classification of deterministic scheduling problems. Report BW 138, Mathematisch Centrum, Amsterdam.

12. E.L. LAWLER (1973) Optimal sequencing of a single machine subject to precedence constraints. *Management Sci.* 19,544-546.

13. E.L. LAWLER (1977) A "pseudopolynomial" algorithm for sequencing jobs to minimize total tardiness. *Ann. Discrete Math.* 1,331-342.

14. E.L. LAWLER (1978) Sequencing jobs to minimize total weighted completion time subject to precedence constraints. *Ann. Discrete Math.* 2,75-90.

15. E.L. LAWLER (1981) Unpublished result.

16. J.K. LENSTRA (1980) Unpublished result.

17. J.K. LENSTRA, A.H.G. RINNOOY KAN (1978) Complexity of scheduling under precedence constraints. *Oper. Res.* 26,22-35.

18. J.K. LENSTRA, A.H.G. RINNOOY KAN (1979) Computational complexity of discrete optimization problems. *Ann. Discrete Math.* 4,121-140.

19. J.K. LENSTRA, A.H.G. RINNOOY KAN (1980) Complexity results for scheduling chains on a single machine. *European J. Oper. Res.* 4,270-275.

20. J.K. LENSTRA, A.H.G. RINNOOY KAN, P. BRUCKER (1977) Complexity of machine scheduling problems. *Ann. Discrete Math.* 1,343-362.

21. J.B. SIDNEY (1975) Decomposition algorithms for single-machine sequencing with precedence relations and deferral costs. *Oper. Res.* 23,283-298.