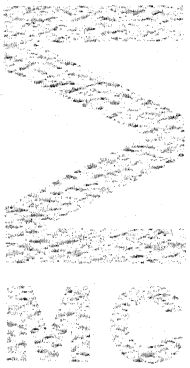
ma
the
ma
tisch

cen
trum



AFDELING MATHEMATISCHE BESLISKUNDE (DEPARTMENT OF OPERATIONS RESEARCH)

BN 32/82

JANUARI

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

AT PLAY IN THE FIELDS OF SCHEDULING THEORY

amsterdam 1982

stichting
mathematisch
centrum

MC

AFDELING MATHEMATISCHE BESLISKUNDE
(DEPARTMENT OF OPERATIONS RESEARCH)

BN 32/82

JANUARI

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

AT PLAY IN THE FIELDS OF SCHEDULING THEORY

kruislaan 413 1098 SJ amsterdam

Printed at the Mathematical Centre, 413 Kruislaan, Amsterdam.

The Mathematical Centre, founded the 11-th of February 1946, is a non-profit institution aiming at the promotion of pure mathematics and its applications. It is sponsored by the Netherlands Government through the Netherlands Organization for the Advancement of Pure Research (Z.W.O.).

1980 Mathematics Subject Classification: 90B35, 68C25

AT PLAY IN THE FIELDS OF SCHEDULING THEORY

E.L. LAWLER
University of California, Berkeley

J.K. LENSTRA
Mathematisch Centrum, Amsterdam

A.H.G. RINNOOY KAN
Erasmus University, Rotterdam

ABSTRACT

This in an informal account of our investigations in the area of deterministic sequencing and scheduling. We describe a computer aided classification system and extrapolate on the use of similar programs in other fields.

KEY WORDS & PHRASES: *machine scheduling, complexity, polynomial algorithm, NP-hardness, computer program*

AT PLAY IN THE FIELDS OF SCHEDULING THEORY

E.L. LAWLER
University of California, Berkeley

J.K. LENSTRA
Mathematisch Centrum, Amsterdam

A.H.G. RINNOOY KAN
Erasmus University, Rotterdam

Machine scheduling theory is something of a jungle, encompassing a bewilderingly large variety of problem types, as the most cursory examination of the journals reveals. It is also a marvelous playground for the algorithm designer and the complexity analyst, in that every known trick of combinatorial optimization can be applied somewhere, to one problem or another. This is an account of our explorations of this jungle-playground. Not incidentally, we shall describe a computer program we have used to help us guide our way. We conclude with some speculations about how similar, possibly more sophisticated, programs could be useful aids for researchers in other fields.

When we began our collaboration several years ago, we decided to focus our attention on *machine* scheduling problems. This meant that we excluded from consideration such worthy topics as project scheduling, timetabling, and cyclic scheduling of manpower. We also decided to concentrate on strictly *deterministic* models. Even so, this left us with an enormous number of problem types to study.

Very early on in our investigations, we decided we needed a uniform system of classification for the problems which had appeared in the literature. Starting from the classification scheme of Conway, Maxwell and Miller [1], after much debate we settled on a scheme which suited our purposes. This classification system is detailed elsewhere [4], and for present purposes can be summarized as encompassing *machine environment* (single machine, parallel

machines, open shop, flow shop, job shop), *job characteristics* (independent vs. precedence constrained, etc.), and *optimality criterion* (makespan, flow-time, maximum lateness, total tardiness, etc.).

An immediate payoff was the consummate ease with which we could communicate problem types. Visitors to our offices were sometimes baffled to hear exchanges such as: "Since $1|r_j|\sum C_j$ is NP-hard, does that imply that $1|pmtn,r_j|\sum C_j$ is NP-hard, too?" "No, that's easy, remember?" "Well, $1|d_j|\sum C_j$ is easy and that implies $1|pmtn,d_j|\sum C_j$ is easy, so what do we know about $1|pmtn,r_j,d_j|\sum C_j$?" "Nothing."

As this discussion indicates, one of our objectives was to demark as clearly as possible the boundary line between *easy* problems (solvable in polynomial time) and *NP-hard* problems. But because of the huge number of problem types and the relationships between them, it was easy to become confused. One could spend an hour trying to determine the status of a particular problem, only to realize that the issue had already been resolved - the problem was a generalization of a known NP-hard problem and therefore NP-hard as well, or a specialization of a known easy problem and therefore easy as well.

The idea of using the computer as an aid began as a joke. The afternoon of September 22, 1975, Dick Karp, Ben Lageweg, Gene Lawler and Jan Karel Lenstra met in the Mathematical Center in Amsterdam to decide on a gift to present to Alexander Rinnooy Kan on the occasion of his upcoming promotion to doctorate. Somebody made the amusing suggestion of a bound volume consisting of a computer tabulation of all the thousands of problem types with a notation for the status of each one: * for easy, ! for NP-hard, and ? for unresolved.

We were well aware that the problems in our classification system admitted of a natural partial ordering. Job shops are more general than flow shops. Precedence constrained problems are more general than problems with independent jobs. Maximum lateness is a more general optimality criterion than makespan. And so on. All that was required to produce the tabulation was to feed the computer all results in the form of known easy problems and known NP-hard problems (ignoring results that were clearly dominated by others), let the computer take account of the partial ordering, and let it churn out a properly annotated listing.

That afternoon at the Math Center, the group speculated on what the

score would be: how many *'s, !'s and ?'s would the tabulation contain? A playful attempt was made to obtain an estimate by generating a few random chains in the partial order, with everyone testing his expertise to see how far up a chain he could prove easiness and how far down NP-hardness. (Lenstra has since made the generation of random chains part of one of his stock lectures, with a member of the audience throwing a die.)

The next inevitable suggestion someone made was: "Why not have the computer list the maximal easy and minimal hard problems, and the minimal and maximal open ones as well? Wouldn't that give us a clearer picture of the situation?"

A suitable program was forthwith written by Lageweg and an initial run was made. The results were startling, for the number of easily resolvable cases it revealed in the listings of minimal and maximal open problems. During the next few weeks Lageweg, Lawler and Lenstra knocked off many targets of opportunity. The number of question marks in the tabulation was considerably smaller when, on January 28, 1976, a handsomely bound volume was presented to Alexander [5].

During the past six years there have been many developments, and Alexander's volume is now thoroughly outdated. The most impressive progress has been made in the area of preemptive scheduling of parallel machines. An elegant algorithm due to Gonzalez and Sahni (for $Q|pmtn|C_{max}$, the problem of minimizing makespan in preemptive scheduling of uniform parallel machines) [3] spawned a whole host of derivative algorithms for related problems.

At the present time, the score for 4,536 problem types stands at 81% NP-hard, 9% easy and 10% open [7]. This particular split is an artifact of our classification system, but it is certainly true that several subareas have been pretty well cleaned up. For example, the status of almost all single machine problems is known. Though open problems are still occasionally resolved, it is safe to say that nearly all the cream has been skimmed.

The problems which remain are mostly rather difficult. It is possible that they are neither NP-hard nor easy, provided that $\mathcal{P} \neq \mathcal{NP}$ (which we believe). One of the frustrations of the theory is that there is no way of proving such a result at present. For those who might care to accept a challenge, we mention two classic open problems:

(1) $P3|prec,p_j=1|C_{max}$, the problem of minimizing makespan for unit-time jobs subject to arbitrary precedence constraints on three identical parallel machines: known to be easy for two machines and NP-hard for an arbitrary number of machines.

(2) $1||\sum T_j$, the problem of minimizing total tardiness on a single machine: known to admit of a pseudopolynomial algorithm, hence not NP-hard in the strong sense (unless $\mathcal{P} = \mathcal{NP}$). Is this problem easy or is it NP-hard in the ordinary sense?

A few words about the significance of NP-completeness theory are in order. It is not always true that polynomial algorithms are *good* and that problems that admit of such algorithms are *easy* to solve in practice. It is perhaps even less true that NP-hard problems are invariably *hard* in a practical sense. Yet there is enough correspondence with reality to make the notions of *easy* and *NP-hard* more than a polite fiction. Some NP-hard problems are *really hard* to solve. For example, no one has yet solved to optimality a certain notorious 10-machine 10-job job shop problem, small as this problem instance is. (The best published solution has a makespan of 972 [8]. Lageweg has found a solution of 935. The best known lower bound is 862.)

The primary usefulness of the concept of NP-hardness is the direction it gives to the algorithm designer. With knowledge that a problem is NP-hard, he can abandon any attempt to reformulate the problem as, say, a simple network flow problem or a graphic matching problem. Instead he can concentrate his energies on developing an efficient enumerative optimization method or a well-behaving approximation algorithm. It is in this way that NP-completeness theory has probably had more impact on combinatorial optimization than any other theoretical development of the past ten years. We were pleased to observe, during the NATO Advanced Study and Research Institute on Deterministic and Stochastic Scheduling (Durham, England, July 1981) [2], that the methodology carries over to computational questions about stochastic scheduling as well.

One of our hopes when we began our project was that we might be able to determine the boundary line between easy and NP-hard problems sufficiently closely that we could gain meaningful insight into the properties that make a scheduling problem of one type or the other. This we have been able to do to some extent. When dealing with a practical scheduling problem (which is

invariably NP-hard), we found it increasingly easy to detect the particular features of the problem which were responsible for its computational intractability, since they would correspond to the crucial ingredients of an NP-hardness proof. These features then suggested certain relaxations that should be made to obtain lower bounds for a branch-and-bound procedure, or directions that could be taken in designing a heuristic.

Now to return to a discussion of the computer program and the benefits we have received from it. First, the program has provided an orderly form of record keeping for research results. Confusions and oversights have been greatly reduced. Second, the program has helped us focus our research. Listings of minimal and maximal open problems have made it easy to choose the most interesting and important ones to work on. And finally, the automatic scorekeeping has been motivational and introduced a healthy competition into our work.

A frivolous idea which occurred to us was that the computer might be programmed to produce another type of score, namely the minimum number of open problems whose resolution would resolve all remaining open problems. Alas, we found that the calculation of this score is itself an NP-hard problem [6]. We have made no attempt to devise an algorithm for its solution.

We believe that computer programs similar to ours could be applied equally well to other well-structured areas of knowledge and research. Certainly allied areas of combinatorial optimization such as location theory and, more ambitiously, algorithmic graph theory are candidates. Even the broad area of mathematical programming might be susceptible, as well as inventory theory, queueing theory, or even organic chemistry.

It would not be difficult to create a sort of automated encyclopedia. Given such a system for the field of mathematical programming, the user could make queries of the form: "What is known about a problem with such-and-such objective function and so-and-so constraints?" The system might answer: "Nothing has been reported on this specific problem, but these results have been obtained for more general and more special cases. Moreover, the following computer codes are available ..." The program would be knowledgeable of problem relationships which might be unknown to the user, even if their usefulness would be contingent on future theoretical developments. For example, it would know that maximization of a posynomial in bivalent variables is equivalent to the min-cut problem of network flow theory.

There are other types of question-answering facilities it would be useful to have. For example, for a book on scheduling theory we are writing, we should like to state a few simple rules that will enable the reader to comprehend the status of large subclasses of problems. It would be nice to be able to verify these rules by asking the system questions of the form: "Are there easy problems involving the nonpreemptive scheduling of parallel machines which do not have the objective of minimizing flowtime?" or "Are there any problems which are known to be NP-hard when preemption is permitted but easy when it is not?"

At some future date it may be possible to have computers search for problem transformations themselves. At this time, such an undertaking appears to be beyond the capabilities of artificial intelligence. Should this development come to pass, the computer would truly be an automated research assistant.

REFERENCES

1. R.W. CONWAY, W.L. MAXWELL, L.W. MILLER (1967) *Theory of Scheduling*, Addison-Wesley, Reading, MA.
2. M.A.H. DEMPSTER, J.K. LENSTRA, A.H.G. RINNOOY KAN (eds.) (1982) *Deterministic and Stochastic Scheduling*, Reidel, Dordrecht.
3. T. GONZALEZ, S. SAHNI (1978) Preemptive scheduling of uniform processor systems. *J. Assoc. Comput. Mach.* 25,92-101.
4. 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.
5. B.J. LAGEWEG, E.L. LAWLER, J.K. LENSTRA (1976) Machine scheduling problems: computations, complexity and classification. Report BN 30, Mathematisch Centrum, Amsterdam (out of print).
6. B.J. LAGEWEG, E.L. LAWLER, J.K. LENSTRA, A.H.G. RINNOOY KAN (1981) Computer aided complexity classification of combinatorial problems. Report BW 137, Mathematisch Centrum, Amsterdam.
7. 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.
8. G. McMAHON, M. FLORIAN (1975) On scheduling with ready times and due dates to minimize maximum lateness. *Oper. Res.* 23,475-482.