Annals of Discrete Mathematics 4 (1979) 175–176
© North-Holland Publishing Company

Report of the Session on COMPLEXITY OF COMBINATORIAL PROBLEMS

R.L. GRAHAM (Chairman)
D. Hausmann (Editorial Associate)

Computational complexity theory as developed by theoretical computer scientists started to have a major impact on the field of discrete optimization in the early 1970's. In two remarkable papers, S.A. Cook and R.M. Karp demonstrated that a large number of combinatorial problems, notorious for their computational intractability, were all equivalent up to a polynomial transformation. This implies that a good (i.e., polynomial-time) algorithm for any of them could be used to solve all others in polynomial time as well, which would establish the equality of the problem classes $\mathscr{P}$ and $\mathscr{NP}$ [3]. In general, this equality is considered to be very unlikely, and NP-completeness of a combinatorial problem has become commonly accepted as being indicative of its inherent difficulty.

Computer scientists again must take a lot of credit for establishing membership of $\mathscr{P}$ for many traditional problems. However, the notion of a good algorithm is originally due to Edmonds [1], whose work in matroid optimization theory has been essential in getting this concept accepted and appreciated. We refer to another session report [2] for more extensive comments on the development of good algorithms. Within this area, the challenge to have upper and lower bounds on the time complexity of a problem meet each other remains as large as ever, even for seemingly simple problems. For example, finding the median of n numbers is now known to be possible in $O(n)$ time, but a lot of ingenuity seems to be required to decrease the upper bound or increase the lower bound on the multiplicative constant. It may even be possible to apply mathematical programming on a metalevel to determine optimal algorithms!

The class of problems that have been proved NP-complete is still expanding rapidly and extremely refined results are available (see, e.g. [3]). Typically, what is lacking for those problems is a good characterization for the optimality of feasible solutions, such as the one that duality theory yields for linear programming. The duality theorem provides strong evidence against the NP-completeness of linear programming, since (unlike $\mathscr{P}$) $\mathscr{NP}$ is suspected not to be closed under complementation. Linear programming, graph isomorphism and other problems that still defy classification may well turn out to be neither in $\mathscr{P}$ nor NP-complete, which has been shown to be possible, provided, of course, that $\mathscr{P} \neq \mathscr{NP}$.

An often-heard and justified complaint against NP-completeness theory is that it only yields a very coarse measure of problem complexity. Despite the inadequacy of the usual testing procedures for enumerative algorithms, it is obvious to every practitioner that some NP-complete problems are harder than others. Refinements of the complexity measure would be very welcome. Within the class of NP-complete problems, one might obtain such refinement by investigating various ways of encoding the problem data or by examining the trade-off between running time and worst-case performance of approximation algorithms. A lot of work has been done in the latter area. Questions that need much further study is how to analyze properly the average-case behavior of heuristics (which presupposes a probability distribution over the set of problem instances) and how to relate it to the worst-case behavior.

Computer science has more to offer in terms of theoretical models for the analysis of problem complexity. $\mathcal{NP}$ is included in the class of problems solvable in polynomial space. This class is the same for deterministic and nondeterministic Turing machines. Polynomial-space completeness has been established, for example, for a modification of the well-known game of hex. Beyond that, some problems have been shown to require at least exponential space, such as the "vector reachability" problem: given a finite set of integer vectors, an initial vector u and a final vector v, is it possible to add vectors from the set to u (with repetition allowed) so as to reach v, while remaining within the positive orthant? Another question of obvious interest is to extend all these concepts to parallel machine models.

Among practitioners one often encounters a lack of enthusiasm for these new tools, which probably only an equality proof of $\mathcal{P}$ and $\mathcal{NP}$ could remove completely. It is unfortunately true that many of the results obtained in this area are asymptotic and yield little concrete information for problems of "real world" size; they still have to stand the test of practical applicability. Yet we would argue that the evident success of complexity theory in determining factors that influence the inherent difficulty of a problem and in bordering the best algorithmic performance that we may expect to get, will be of ultimate benefit to theoreticans and practitioners alike.

Acknowledgements

J. Edmonds, R.L. Graham, B. Korte, E.L. Lawler and G.L. Nemhauser were among the discussants in Banff. A summary of the points they raised was prepared by J.K. Lenstra and A.H.G. Rinnooy Kan and edited by D. Hausmann.

References

[1] J. Edmonds, Paths, trees, and flowers. Canad. J. Math. 17 (1965) 449–467.
[2] J.K. Lenstra, Report of the session on "Algorithms for special classes of combinatorial optimization problems", Ann. Discrete Math. 4 (1979) this volume.
[3] J.K. Lenstra and A.H.G. Rinnooy Kan, Computational complexity of discrete optimization problems, Ann. Discrete Math. 4 (1979) this volume.