ICM 1998, Berlin, Aug. 18–27

Abstracts of Plenary and Invited Lectures

Section:

14. Mathematical Aspects of Computer Science

1991 MS Classification: 68Q25,68Q99
Håstad, Johan, Royal Institute of Technology, Sweden
On approximating NP-hard optimization problems

In complexity theory algorithms are considered efficient if they run in time polynomial in the size of the input and the set of efficiently solvable problems is denoted by P. A possibly slightly larger class of problems, called BPP, is obtained if we also allow that algorithm to use random coins and only ask the algorithm to give the correct answer with high probability. Another central class of problems is NP, the set of decision problems for which a positive answer has a proof which can be verified efficiently.

Many problems encountered in practice do not seem to allow efficient algorithms and one famous class of such problems is given by the NP-complete problems. These problems belong to NP and have the property that if one problem can be solved efficiently then so can all problems in NP. It is widely believed, but still unproven, that NP is not contained in P, and thus NP-completeness is the best available proof of intractability for problems in NP.

A problem (itself not necessarily contained in NP) is said to be NP-hard if it has the property that if it can solved efficiently so can any problem in NP. In this talk we will be concerned with NP-hard optimization problems. Examples of such problems are given by graph-coloring[1], and maximum satisfiability[2].

Finding the exact optimum for these problems is computationally infeasible we resort to finding non-optimal solutions by approximation algorithms. A C-approximation algorithm is one which, for every input, the result is off by a factor at most C from optimal. Here C could either be a constant or a function depending on the size of input. The key question now is to determine, for each problem of interest, for what values of C efficient C-approximation algorithms exist.

The problem we concentrate on in this talk is Max-Lin; given an over-determined system of m linear equations in n variables over the field of two elements, find the assignment that satisfies the largest number of equations. Since a random assignment satisfies, on the average, $m/2$ equations and no assignment can satisfy more than all m equations we get an efficient

[1] Given a graph G find the minimum number of colors needed to color the nodes such that each pair of adjacent nodes have different colors.

[2] Given a collection of disjunctive clauses, find the assignment that satisfies the maximal number of clauses.

2-approximation algorithm. The main results we want to discuss is taken from [6] and says that for any $\epsilon > 0$ it is NP-hard to $(2-\epsilon)$-approximate Max-Lin.

The basic technique for proving such inapproximability results was introduced in [3]. While the technique can be described in completely combinatorial terms, the intuitively most compelling way to describe it is through *probabilistically checkable proofs* which we, as customary abbreviate PCP.

The notion of a verifier is central to a number of concepts in computer science and a standard situation would be to verify an NP-statement. A proof of the fact that a Boolean formula φ is satisfiable is a satisfying assignment and it is checked by evaluating φ on the given assignment. This verifier is deterministic, always comes to the correct conclusion but reads a large number of bits. In a PCP the verifier is probabilistic, comes to the correct conclusion with high probability, but reads very few bits. As a culmination of a series of papers it was proven in [1] that any satisfiable formula φ has a proof of polynomial size that can be verified by reading a constant, independent of the size of φ, number of bits. This verifier always accepts a correct proof of a satisfiable formula while any proof for a non-satisfiable formula is rejected with probability $1/2$. By reduction, any NP-statement has a similar proof.

The connection to approximation is, in very general terms, given by considering $\max_\pi Acc_\varphi(\pi)$ where $Acc_\varphi(\pi)$ is the probability that the verifier accepts proof π. This is a combinatorial optimization problem and, by the above reasoning, $(2-\epsilon)$-approximating this problem enables us to solve an NP-complete problem and thus this is NP-hard. Different PCPs give results for different optimization problems.

The first results [3, 1] gave in most circumstances, in the quantitative sense rather weak results. The key to improving the quantitative bounds to tight or almost tight results has been the use of very informative ways of coding binary strings. The code used for the results we describe is the long code introduced by [2] which codes a string of length n by a string of length 2^{2^n}. Strong quantitative results have also been obtained for graph-coloring [4] and max-clique[3] [5]. For both problems, for any $\epsilon > 0$ and graphs with n nodes, one cannot find an $n^{1-\epsilon}$-approximation algorithm unless NP $\subseteq$ BPP. The goal of this talk is to describe the set of ideas that lead to these efficient PCPs and discuss the connection to strong inapproximability results. For illustration we focus on the above mentioned problem Max-Lin.

References

[1] S. Arora, C. Lund, R. Motwani, M. Sudan and M. Szegedy. Proof verification and intractability of approximation problems. Proceedings of the 33rd Annual IEEE Symposium on Foundations of Computer Science, Pittsburgh, 1992, pp 14-23.

[3] Given a graph G find the maximal number of nodes that are pairwise connected.

[2] M. Bellare, O. Goldreich and M. Sudan Free Bits, PCPs and Non-Approximability- Towards tight Results. Proceedings of the 36th Annual IEEE Symposium on Foundations of Computer Science, 1995, Milwaukee, pp 422-431. See also a more complete version available from ECCC, Electronic Colloquium on Computational Complexity (http://www.eccc.uni-trier.de/eccc).

[3] U. Feige, S. Goldwasser, L. Lovász, S. Safra, and M. Szegedy. Interactive proofs and the hardness of approximating cliques. Journal of the ACM, Vol, 43:2, pp 268-292.

[4] U. Feige and J. Kilian. Zero-Knowledge and the chromatic number. Proceedings of the 11th Annual IEEE conference on Computational Complexity, Philadelphia 1996, pp 278-287.

[5] J. Håstad. Clique is hard to approximate within $n^{1-\epsilon}$. Proceedings of the 37th Annual IEEE Symposium on Foundations of Computer Science, Burlington 1996, pp 627-636.

[6] J. Håstad Some optimal inapproximability results. Proceedings of 29th Annual ACM Symposium on Theory of Computation, El Paso, 1997, pp 1-10.