U.C. Berkeley
CS294: PCP and Hardness of Approximation
Professor Luca Trevisan

Handout N1
January 18, 2006
Scribe: Luca Trevisan

Notes for Lecture 1

1 A Brief Historical Overview

This lecture is a "historical" overview of the field of hardness of approximation.

Hundreds of interesting and important combinatorial optimization problems are **NP**-hard, and so it is unlikely that any of them can be solved by a wost-case efficient exact algorithm. Short of proving $\mathbf{P} = \mathbf{NP}$, when one deals with an **NP**-hard problem one must accept a relaxation of one or more of the requirements of having a an *optimal* algorithm that *runs in polynomial time* on *all inputs*. Possible ways of "coping" with **NP**-hardness are to design algorithms that

1. Run in "gently exponential" time, such as, say, $O(1.1^n)$. Such algorithms can be reasonably efficient on small instances;
2. Run in polynomial time on "typical" instances. Such algorithms are extremely useful in practice. A satisfactory theoretical treatment of such algorithms has been, so far, elusive, because of the difficulty of giving a satisfactory formal treatment of the notion of "typical" instances. (Most average-case analysis of algorithms is done on distributions of inputs that are qualitatively different from the distributions arising in practice.)
3. Run in polynomial time on all inputs but return sub-optimal solutions of cost "close" to the optimum.

In this course we focus on *approximation algorithms*, that are algorithms of the third kind with a provably good worst-case ratio between the value of the solution found by the algorithm and the true optimum.

Formally, we say that an algorithm is *r-approximate* for a minimization problem (respectively, a maximization problem) if, on every input, the algorithm finds a solution whose cost is at most r times the optimum (respectively, at least $1/r$ times the optimum). The ratio r, which is always ≥ 1, is also called the *performance ratio* of the algorithm.

For some problems, it is possible to prove that even the design of an r-approximate algorithm with small r is impossible, unless $\mathbf{P} = \mathbf{NP}$. Results of this kind, called *inapproximability* results, are the subject of this survey.

The seeming intractability of many combinatorial optimization problems was observed already in the 1960s, motivating the development of suboptimal heuristic algorithms and, in particular, the notion of approximation algorithm as defined above. An early example of analysis of an approximation algorithm is a paper of Graham on scheduling problems [23].

The theory of NP-completeness [14, 30, 29], and its success in classifying the complexity of optimization problems, provided added motivation to the study of efficient suboptimal algorithms and

to the rigourus analysis of approximation algorithms. A 1973 seminal paper by Johnson [28] gave the field its current foundations. Johnson considers the problems Max SAT, Set Cover, Independent Set, and Coloring. He notes that there is a 2-approximate algorithm for Max SAT, and that this can be improved to 8/7 for Max E3SAT, the restriction of the problem to instances in which every clause contains exactly three literals. He also presents a $(1 + \ln k)$-approximate algorithm for Set Cover, where k is the size of the largest set in the collection, and he notes that natural heuristics for Coloring and Independent Set fail to give even a $n^{1-\epsilon}$ approximation for $\epsilon > 0$.

Thirty years after Johnson's paper, the design and analysis of approximation algorithms has become a large and successful research area. A book edited by Hochbaum [26] and two more recent textbooks [4, 35] give an extensive overview of the field. The book by Ausiello et al. [4] also includes a list of known results for hundreds of problems.

As the field progressed and matured, it became apparent that different combinatorial optimization problems behaved different from the point of view of approximability. For some problems, researchers were able to devise polynomial time approximation schemes (abbreviated PTAS), that is, to devise an r-approximate algorithm for every $r > 1$. For other problems, such as Max SAT and Vertex Cover, r-approximate algorithms for constant r were known, but no approximation scheme. Other problems like Set Cover had no known constant factor approximation algorithm, but were known to admit an approximation algorithm with a performance ratio that was slowly growing (logarithmic) in the input length. Finally, there were problems like Independent Set and Coloring for which not even a slowly growing performance ratio was known to be achievable in polynomial time. On the other hand, until 1990, there was no negative result showing that the existence of approximation schemes for any of the above problems would imply $\mathbf{P} = \mathbf{NP}$ or other unlikely consequences. Indeed, few inapproximability results were known at all. Furthermore, there was a general intuition that, in order to prove inapproximability for the above mentioned problems (and others), it was necessary to depart from the standard techniques used to prove the **NP**-hardness of problems.

To see the limitations of standard **NP**-hardness proofs, consider for example the proof that solving optimally the Independent Set problem is **NP**-hard. The proof starts from a 3SAT instance φ and constructs a graph G and an integer k such that if φ is satisfiable then G has an independent set of size k, and if φ is not satisfiable then all independent sets in G have size at most $k - 1$. It then follows that a polynomial time exact algorithm for solving Independent Set, together with the reduction, would imply a polynomial time algorithm for 3SAT, and, from Cook's theorem, we would have a polynomial time algorithm for every problem in **NP**. Consider now the question of the approximability of the Independent Set problem. Looking at the standard reduction from 3SAT we can see that if φ has an assignment that satisfies all the clauses except c ones, then G has an independent set of size $k - c$ and, given such an assignment, the independent set is easy to construct. Furthermore, the instances of 3SAT produced by Cook's theorem are such that it is easy to find assignments that satisfy all the clauses but one. In conclusion, the instances that we get by reducing a generic **NP** problem to Independent Set are very easy to approximate.

To derive an inapproximability result, we need a much stronger reduction. Suppose we want to prove that no 2-approximate algorithm exists for the Independent Set problem assuming $\mathbf{P} \neq \mathbf{NP}$.

Then a natural proof strategy would be to start from an **NP**-complete problem, say, 3SAT, and then reduce an instance φ of 3SAT to an instance G_φ of Independent Set, with the property that, for some k, if φ is satisfiable then

$$OPT_{\text{Independent Set}}(G_\varphi) \geq k \tag{1}$$

and if φ is not satisfiable then

$$OPT_{\text{Independent Set}}(G_\varphi) < k/2 \ . \tag{2}$$

Suppose we have such a reduction, and suppose we also have a 2-approximate algorithm for Independent Set; then the algorithm, given G_φ, will find a solution of cost at least $k/2$ if and only if φ is satisfiable. The approximation algorithm then gives a polynomial time algorithm for 3SAT, and so no such algorithm can exist if $\mathbf{P} \neq \mathbf{NP}$.

All the **NP**-completeness proofs for graph problems before 1990, however, can be essentially described as follows: we start from the computation of a generic non-deterministic Turing machine, then we encode its computation as a 3SAT formula, using the construction of Cook's theorem, and then we reduce 3SAT to the problem of interest (the reduction may be presented as a sequence of reductions involving several intermediate problems, but it can always be thought of as a direct reduction from 3SAT) by encoding variables and clauses of the formula as sub-graphs connected in a proper way. The computation of a Turing machine is very sensitive to small changes, and it seems impossible to generate an inapproximability gap starting from a fragile model and applying "local" reductions. The only inapproximability results that can be proved with such reductions are for problems that remain **NP**-hard even restricted to instances where the optimum is a small constant. For example, in the Metric Min k-Center problem it is **NP**-hard to decide whether the optimum has cost 1 or 2, and so no algorithm can have a performance ratio smaller than 2 unless $\mathbf{P} = \mathbf{NP}$ [27]. Similarly, in the Coloring problem it is **NP**-hard to decide wether the optimum has cost 3 or 4, and so no algorithm has performance ratio smaller than 4/3 unless $\mathbf{P} = \mathbf{NP}$, and Garey and Johnson [20] show that the gap can be "amplified" to k versus $2k - 4$ for constant k, ruling out also algorithms with performance ratio smaller than 2. Most interesting problems, however, become trivial when restricted to inputs where the optimum is a constant.

To prove more general inapproximability results it seemed necessary to first find a machine model for **NP** in which accepting computations would be "very far" from rejecting computations. Before such a model was discovered, an important piece of work on inapproximability was due to Papadimitriou and Yannakakis, who showed that, assuming that Max 3SAT does not have a PTAS, then several other problems do not have a PTAS [32]. Berman and Schnitger [11] proved that if Max 2SAT does not have a PTAS then, for some $c > 0$, the Independent Set problem cannot be approximated within a factor n^c.

The modern study of inapproximability was made possible by the discovery, due to Feige et al. [16] in 1990, that probabilistic proof systems could give a robust model for **NP** that could be used to prove an inapproximability result for the Independent Set problem.[1] A year later, Arora et al. [2, 1]

[1] Another, less influential, connection between probabilistic proof checking and inapproximability was discovered around the same time by Condon [13].

proved the PCP Theorem, a very strong characterization of **NP** in terms of proof systems, and showed how to use the PCP Theorem to prove that Max 3SAT does not have a PTAS. Using the reductions of [32] (and others [33, 12]), the PCP Theorem gave inapproximability results for several other problems.

The PCP Theorem was the culmination of a long line of collaborative work, that is difficult to summarize.

In telling this story, one typically starts from the introduction of the model of "interactive proof systems" due independently to Goldwasser, Micali and Rackoff [22] and to Babai [5]. In this model, a probabilistic verifier *interacts* with a prover, as opposed to receiving a fixed proof and checking its validity. The work of Goldwasser, Micali and Rackoff [22] also introduces the notion of "*zero-knowledge* proof system," which later became a fundamental primitive in the construction of cryptographic primitives. A fundamental result by Goldreich, Micali and Wigderson [21] shows that every problem in NP has a zero-knowledge proof system, assuming that a certain cryptographic assumption is true. Ben-Or et al. [10] considered a model of zero-knowledge where the verifier can interact with two (or, more generally, several) provers, who are all computationally unbounded *but* unable to communicate with each other once the protocol starts. The contribution of [10] was to show that every problem in NP has a zero-knowledge proof system in this model, without cryptographic assumptions. The model of multi-prover interactive proof (without the zero-knowledge requirement) was further studied by Fortnow, Rompel and Sipser [19]. They show that the class **MIP** of languages admitting such proof systems has the following equivalent characterization: it can be seen as the class of languages that admit exponentially long proofs of membership that can be checked in polynomial time by a randomized verifier (with bounded error probability). In modern terminology, $\mathbf{MIP} = \mathbf{PCP}[\mathrm{poly}(n), \mathrm{poly}(n)]$. This class is clearly contained in **NEXP**, where **NEXP** is the class of decision problems that admit exponentially long proofs that can be checked in exponential time in the length of the input (but, without loss of generality, in polynomial time in the length of the proof itself).

Initially, it was conjectured that **MIP** was only a small extension of **NP**, and that $\mathbf{coNP} \not\subseteq \mathbf{MIP}$. Shortly after Shamir's proof that $\mathbf{IP} = \mathbf{PSPACE}$ [34], Babai, Fortnow and Lund [7] showed that $\mathbf{MIP} = \mathbf{NEXP}$. This is a surprising result, because it says that for every language that admits exponentially long proofs, such proofs can be encoded in such a way that a polynomial-time randomized verifier can check them. The verifier will accept correct proofs with probability 1, and "proofs" of incorrect statements with probability $\leq 1/2$ (or, equivalently, with probability exponentially small in the length of the input). So the verifier becomes convinced of the validity of the proof even if it only looks at a negligible part of the proof itself.

It is natural to ask whether polynomially long proofs can be checked in polylogarithmic time. This question has to be phrased carefully, since a polylogarithmic time verifier cannot even read the instance, which makes it impossible to verify a proof for it. However if both the instance and the proof are encoded in a proper (efficiently computable) way, then Babai et al. show that polylogarithmic time verification is possible [6]. A variant of this result was also proved by Feige et al. [15]: they show that **NP**-proofs have a quasi-polynomial length encoding (i.e. an encoding of length $n^{O(\log \log n)}$) such that a polynomial-time verifier can verify the correctness of the proof in

polynomial time by using $O(\log n \log\log n)$ random bits and reading $O(\log n \log\log n)$ bits of the proof. The main result of Feige et al. [15] was to show a connection between the computational power of such a model and the hardness of approximating the Max Clique problem.[2] The result of Feige et al. [15] can be written as $\mathbf{NP} \subseteq \mathbf{PCP}[O(\log n \log\log n), O(\log n \log\log n)]$.

Arora and Safra [2] introduced several new ideas to improve on [15], and proved that $\mathbf{NP} = \mathbf{PCP}[O(\log n), O(\sqrt{\log n})]$. The main contribution of Arora and Safra is the idea of "composing" proof systems together. The next step was to realize that the reduction from PCP to Max Clique was not an isolated connection between proof checking and approximability. Sudan and Szegedy (as credited in [1]) discovered that the computations of a $(O(\log n), O(1))$-restricted verifier can be encoded as instances of the Max 3SAT problem. Then, using the web of reductions between optimization problems initiated by Papadimitriou and Yannakakis [32], this also implies that the strength of $(O(\log n), O(1))$-restricted verifiers implies the hardness of approximating several important problems including the Traveling Salesman Problem and the Steiner Minimal Tree problems in metric spaces. This was a strong motivation to prove the PCP Theorem [1], a proof that came only a few months after the initial circulation of the paper of Arora and Safra [2]. The work of Arora and others [1] introduced several new ideas and results, including the construction of a constant-query verifier requiring exponential proof length, the proof of a stronger "low degree test" (see below) and the construction of a constant-query verifier with polynomial proof length (with the proof being a string over an alphabet of super-polynomial size).

An explosion of applications to other problems soon followed, such as the work of Lund and Yannakakis [31] on Set Cover and Coloring and the work of Arora et al. [3] on lattice problems, that appeared in 1993. Later work focused on stronger inapproximability results, obtained both by proving stronger versions of the PCP Theorem and by devising better reductions. Despite great technical difficulties, very rapid progress was made in the mid 1990s. It is interesting to read a survey paper by Bellare [9], written in the middle of the most exciting period of that time. Some of the open questions raised in the paper were solved a few months later. We now know inapproximability results for Max 3SAT [25], Set Cover [17], Independent Set [24] and Coloring [18] showing that the performance ratios of the algorithms presented in Johnson's paper [28] are best possible (assuming $\mathbf{P} \neq \mathbf{NP}$ or similar complexity assumptions).

References

[1] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and hardness of approximation problems. *Journal of the ACM*, 45(3):501–555, 1998. Preliminary version in *Proc. of FOCS'92*. 3, 5

[2] S. Arora and S. Safra. Probabilistic checking of proofs: A new characterization of NP. *Journal of the ACM*, 45(1):70–122, 1998. Preliminary version in *Proc. of FOCS'92*. 3, 5

[2] A clique in a graph is a subset of vertices that are all pairwise adjacent. The Max Clique problem is, given a graph, to find the largest clique.

[3] Sanjeev Arora, László Babai, Jacques Stern, and Z. Sweedyk. The hardness of approximate optima in lattices, codes, and systems of linear equations. *Journal of Computer and System Sciences*, 54(2):317–331, 1997. 5

[4] Giorgio Ausiello, Pierluigi Crescenzi, Giorgio Gambosi, Viggo Kann, Alberto Marchetti-Spaccamela, and Marco Protasi. *Complexity and Approximation: Combinatorial Optimization Problems and Their Approximability Properties*. Springer-Verlag, 1999. 2

[5] L. Babai. Trading group theory for randomness. In *Proceedings of the 17th ACM Symposium on Theory of Computing*, pages 421–429, 1985. See also [8]. 4

[6] L. Babai, L. Fortnow, L. Levin, and M. Szegedy. Checking computations in polylogarithmic time. In *Proceedings of the 23rd ACM Symposium on Theory of Computing*, pages 21–31, 1991. 4

[7] L. Babai, L. Fortnow, and C. Lund. Non-deterministic exponential time has two-prover interactive protocols. *Computational Complexity*, 1:3–40, 1991. Preliminary version in *Proc. of FOCS'90*. 4

[8] L. Babai and S. Moran. Arthur-Merlin games: a randomized proof system, and a hierarchy of complexity classes. *Journal of Computer and System Sciences*, 36:254–276, 1988. 6

[9] M. Bellare. Proof checking and approximation: Towards tight results. *Sigact News*, 27(1), 1996. 5

[10] M. Ben-Or, S. Goldwasser, J. Kilian, and A. Wigderson. Multi-prover interactive proofs: How to remove intractability assumptions. In *Proceedings of 20th Symposium on Theory of Computing*, pages 113–131, 1988. 4

[11] P. Berman and G. Schnitger. On the complexity of approximating the independent set problem. *Information and Computation*, 96:77–94, 1992. Preliminary version in *Proc. of STACS'89*. 3

[12] M. Bern and P. Plassmann. The Steiner tree problem with edge lengths 1 and 2. *Information Processing Letters*, 32:171–176, 1989. 4

[13] A. Condon. The complexity of the max-word problem and the power of one-way interactive proof systems. *Computational Complexity*, 3:292–305, 1993. Preliminary version in *Proc. of STACS91*. 3

[14] S.A. Cook. The complexity of theorem proving procedures. In *Proceedings of the 3rd ACM Symposium on Theory of Computing*, pages 151–158, 1971. 1

[15] U. Feige, S. Goldwasser, L. Lovász, S. Safra, and M. Szegedy. Approximating clique is almost NP-complete. In *Proceedings of the 32nd IEEE Symposium on Foundations of Computer Science*, pages 2–12, 1991. 4, 5

[16] U. Feige, S. Goldwasser, L. Lovász, S. Safra, and M. Szegedy. Interactive proofs and the hardness of approximating cliques. *Journal of the ACM*, 43(2):268–292, 1996. Preliminary version in *Proc. of FOCS91*. 3

[17] Uriel Feige. A threshold of $\ln n$ for approximating set cover. *Journal of the ACM*, 45(4):634–652, 1998. 5

[18] Uriel Feige and Joe Kilian. Zero knowledge and the chromatic number. *Journal of Computer and System Sciences*, 57(2):187–199, 1998. 5

[19] L. Fortnow, J. Rompel, and M. Sipser. On the power of multi-prover interactive protocols. In *Proceedings of the 3th IEEE Conference on Structure in Complexity Theory*, pages 156–161, 1988. 4

[20] M.R. Garey and D.S. Johnson. The complexity of near-optimal graph coloring. *Journal of the ACM*, 23:43–49, 1976. 3

[21] O. Goldreich, S. Micali, and A. Wigderson. Proofs that yield nothing but their validity. *Journal of the ACM*, 38(3), 1991. 4

[22] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on Computing*, 18(1):186–208, 1989. Preliminary version in *Proc of STOC'85*. 4

[23] R.L. Graham. Bounds for certain multiprocessing anomalies. *Bell System Technology Journal*, 45:1563–1581, 1966. 1

[24] J. Håstad. Clique is hard to approximate within $n^{1-\epsilon}$. *Acta Mathematica*, 182:105–142, 1999. 5

[25] Johan Håstad. Some optimal inapproximability results. *Journal of the ACM*, 48(4):798–859, 2001. 5

[26] Dorit Hochbaum, editor. *Approximation Algorithms for NP-Hard Problems*. PWS Publishing, 1996. 2

[27] D.S. Hochbaum and D.B. Shmoys. A best possible heuristic for the k-center problem. *Mathematics of Operations Research*, 10(2):180–184, 1985. 3

[28] D.S. Johnson. Approximation algorithms for combinatorial problems. *Journal of Computer and System Sciences*, 9:256–278, 1974. 2, 5

[29] R.M. Karp. Reducibility among combinatorial problems. In R.E. Miller and J.W. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, 1972. 1

[30] L. A. Levin. Universal search problems. *Problemi Peredachi Informatsii*, 9:265–266, 1973. 1

[31] C. Lund and M. Yannakakis. On the hardness of approximating minimization problems. *Journal of the ACM*, 41:960–981, 1994. Preliminary version in *Proc. of STOC'93*. 5

[32] C. H. Papadimitriou and M. Yannakakis. Optimization, approximation, and complexity classes. *Journal of Computer and System Sciences*, 43:425–440, 1991. Preliminary version in *Proc. of STOC'88*. 3, 4, 5

[33] C.H. Papadimitriou and M. Yannakakis. The travelling salesman problem with distances one and two. *Mathematics of Operations Research*, 18:1–11, 1993. 4

[34] A. Shamir. IP = PSPACE. *Journal of the ACM*, 39(4):869–877, 1992. Preliminary version in *Proc of FOCS'90*. 4

[35] Vijay Vazirani. *Approximation Algorithms*. Springer, 2001. 2