On the efficient approximability of constraint satisfaction problems

Johan Håstad

Abstract

We discuss some results about the approximability of constraint satisfaction problems. In particular we focus on the question of when an efficient algorithm can perform significantly better than the algorithm that picks a solution uniformly at random.

1 Introduction

The most famous problem in theoretical computer science is the question of whether the two complexity classes P and NP are equal.

Here P is the set of problems[1] that can be solved in time which is polynomial in the size of the input. This is the mathematical definition aimed to correspond to problems which can be solved efficiently in practice on fairly large instances. One might object that there are very large polynomials but this has rarely been a problem and most problems known to be in P are efficiently solvable in the everyday meaning of the concept.

The class NP is the set of decision problems such that for instances with a positive answer there is a short proof of this state of affairs that can be verified efficiently. One of the most famous problems in NP is the traveling salesman problem, TSP, in which we are given n cites and distances $d(c_i, c_j)$ between the cities. The task is, given an upper bound K, to find a tour that visits all the cities and returns to its origin and is of total length at most K. If there is such a tour then, given the tour, it is easy to verify that indeed it is of the desired quality. Formally "easy" should here be interpreted as computable in time which is polynomial in the input length. Formulated differently there is a proof, sometimes called a certificate, that the instance has a positive answer and this short proof can be verified efficiently.

The existence of an easily verifiable certificate says little about the difficulty of finding the certificate. Of course we can always try all certificates but apart from this general procedure, the existence of the certificate seems to be of little help and indeed there is no general method to find the optimal tour for TSP in time that is subexponential in the number of cities n. Note also that the problem is very non-symmetrical in that it is very hard, in general, to convince someone that indeed there is no tour shorter than K. In other words there are probably no short certificates proving tight lower bounds on tour lengths.

The question whether P equals NP is the question of whether, for any decision problem where a positive answer has a short proof that can be verified efficiently, this proof can be found efficiently. From an everyday perspective this would seem absurd in that it rules out the brilliant idea, the idea that is hard to come by but which can immediately be verified as being excellent. In the terms of mathematical

[1]To be precise P only contains decision problems but for simplicity let us ignore this formal definition in this informal discussion.

research NP=P would, informally, be the same as saying that whenever a theorem has a short proof then this proof can be found quickly. We all "disprove" this statement frequently but on the other hand one should not compare mathematicians to computers.

In any case, most people working in complexity theory have a strong conviction that NP does indeed contain problems not in P. It seems, however, that a proof of this requires new insights. To prove that a problem does not belong to P one is required to prove that any algorithm that solves the given problem must take a long time. Another way to phrase this is to say that any algorithm that runs quickly must make a mistake on some input. This quantification over algorithms is very difficult to handle. There are many ways to proceed in a computation and for some problems there are very counter-intuitive algorithms that turn out to be both efficient and correct. Before continuing with our main topic let us take a small detour.

Consider ordinary arithmetic with large integers. It is easy to add two n-bit numbers with $O(n)$ operations. This is clearly, up to constants, the best we can do as each input bit needs to participate in some operation.

Multiplication done the standard, grade school, way requires $O(n^2)$ operations to multiply two n-bit numbers. There are more efficient methods and the most efficient algorithm [24] is based on the discrete Fourier transform and runs in time $O(n \log n \log\log n)$. It is unknown if this is best possible or even more embarrassingly it is unknown whether multiplication can be done in time $O(n)$. In other words we do not know whether multiplication is a more difficult operation to perform than addition or whether it is simply that we have not found the best way to do it. It is not obvious which is the most difficult problem, to prove that NP$\neq$ P, or to prove that multiplication needs super-linear time, but with our inability to prove lower bounds for computation a major new idea seems to be needed to succeed with either task. Let us return to our main line of reasoning.

Even if we have not been able to prove that NP contains difficult problem we have identified the best candidates for such problems; the NP-complete problems. Loosely speaking a problem in NP is NP-complete if it belonging to P is equivalent to NP=P. Thus, as we strongly believe that NP$\neq$ P, being NP-complete is strong evidence that a problem is computationally difficult. A slight variant is to prove that a problem is NP-hard. Such a problem has the property that it belonging to P implies that NP=P, but as opposed to the NP-complete problem such a problem need not itself belong to NP. For instance it might not be a decision problem or it might be a decision problem of even higher complexity.

Many problems are known to be NP-complete problem and in particular TSP discussed above is NP-complete. Ever since Cook [5], in 1971, first defined the class of NP-complete problems, one problem, Satisfiability, has turned out to play a central role. In Satisfiability we are given a Boolean formula and the question is to find an assignment that satisfies the formula. One example is given by the formula

$$\varphi = (x_1 \vee \bar{x}_2 \vee x_3) \wedge (x_2 \vee \bar{x}_3 \vee \bar{x}_4) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee x_4).$$

This formula has the further property that it is a conjunction of disjunctions of

literals[2] and this type of formula is usually called 3-CNF, where 3 is a bound on the number of literals in each disjunction (or clause) and CNF is short for Conjunctive Normal Form. It was proved by Cook that satisfiability of 3-CNF formulas is NP-complete. One might note that satisfiability of 2-CNF formulas is decidable in polynomial time and we invite the reader to find an efficient algorithm in this case when each disjunction only contains at most 2 literals.

The problem of deciding satisfiability of 3-CNF formulas is known as 3-Sat. It is a prime example of what it is called a Constraint Satisfaction Problem (CSP) as it is a collection of constraints each on a small set of variables. As 3-Sat would seem like a rather simple CSP one might expect that most CSPs are NP-complete and this is indeed correct. Already in 1978 Schaefer [24] constructed the short list of classes[3] of Boolean CSPs for which the problem is in P while in all other cases it is NP-complete. Over larger domains the situation is more complicated and a complete characterization over the domain of three values was found only recently by Bulatov [4]. The general case is still not resolved.

In this paper, however, we focus on the Boolean case where each variable only takes two values. As mentioned above almost all problems in their basic form are NP-complete but we turn to a more refined question. Above we described a black or white world where we want to satisfy all constraints. We now turn to a world of shades of gray where we want to satisfy as many constraints as possible. Let us try to formulate the central question.

Consider an instance φ of 3-Sat where each clause is of length exactly 3. We know that it is NP-complete to decide whether we can satisfy all the constraints, but maybe we are happy if we can satisfy almost all the constraints. It is easy to see that a random assignment satisfies a fraction $\frac{7}{8}$ of the constraints and hence we want to find an assignment that satisfies significantly more than this fraction. There are instances where no assignment satisfies more than a fraction $\frac{7}{8}$ of the constraints and hence the question is most interesting when we, for one reason or another, are guaranteed that there is some unknown assignment that satisfies all, or almost all of the constraints.

It turns out that for 3-Sat we cannot efficiently find a good assignment even in this case while for some other NP-complete problems we can find an assignment that does asymptotically better than a random assignment. The goal of the current paper is to survey these results.

2 Efficient computation

As our arguments are quite informal almost any definition of efficient computation would be sufficient. A reader desiring a precise definition can think of the Turing machine or consult any standard text such as [21].

Many of our efficient algorithms are randomized. Again the precise details are not important but as we in some cases count the number of possible random choices let us fix a model.

[2] A literal is a variable or a negated variable.

[3] 0-valid, 1-valid, weakly positive, weakly negative, affine or 2-CNF: consult [24] for definitions.

We assume that the algorithms has the possibility to choose independent random bits to use in the computation. These random bits are recorded on the work tape of the Turing machine. As is commonly done we sometimes refer to these bits as the "random coin flips" made by the algorithm.

3 Maximal constraint satisfaction problems

On the input side it turns out that it is more convenient to use $\{-1, 1\}$ rather than $\{0, 1\}$ for our two possible values. A predicate P of arity k is a mapping $\{-1, 1\}^k \to \{0, 1\}$. An instance of Max-CSP-P is given by a collection $(C_i)_{i=1}^m$ of k-tuples of literals [4]. When we want to emphasize the arity of P we call Max-CSP-P a k-CSP.

For an assignment to the variables, a particular k-tuple is satisfied if P, when applied to values of the literals, returns 1. For an instance I and an assignment x we let $N(I, x, P)$ be the number of constraints of I satisfied by x under the predicate P.

We could allow positive weights giving different constraints different importance. As we do allow repetition of the same constraints we can, to a large extent, simulate weights and the existence of weights turn out not to be of any significant importance for our discussion.

Definition Max-CSP-P is the problem of, given an instance I, to find the assignment x that maximizes $N(I, x, P)$.

A key parameter for a Max-CSP is the number of assignments that satisfy the predicate P.

Definition The *density*, $d(P)$, of a predicate P on k Boolean variables is defined as $p2^{-k}$ where p is the number of assignments in $\{-1, 1\}^k$ that satisfy P.

Definition A k-CSP where each constraint is a disjunction of at most k literals is an instance of *Max-k-Sat*. The subproblem where each constraint is the disjunction of exactly k literals is denoted by *Max-Ek-Sat*.

Next we look at linear equations modulo 2. As we use $\{-1, 1\}$ as our two values with -1 corresponding to true, addition modulo 2 is in fact multiplication.

Definition A k-CSP where each constraint is that a product of at most k literals equals a constant is an instance of *Max-k-Lin*. If each product contains exactly k variables we denote it by *Max-Ek-Lin*.

Note that, by Gaussian elimination, if all equations can be satisfied then such an assignment can be found in polynomial time. Thus the interesting case of the problem is when we cannot satisfy all the constraints.

In this paper we mostly consider linear equations modulo 2 but many results apply to linear equations modulo m for larger values of m which need not even

[4] Note that we allow both variables and negated variables.

be prime. This problem is most natural in the case when the variables are also allowed to take values modulo m and in such a case we would speak of the problems Max-k-Lin-m and Max-Ek-Lin-m.

Normally we allow negation of variables for free but to make the next problem both a Max-CSP and a graph problem we make an exception.

Definition A 2-CSP where each constraint is an inequality $x_i \neq x_j$ is an instance of *Max-Cut*.

To see that Max-Cut is a graph problem think of each pair (i, j) that appears as a constraint as an edge between vertices i and j in a graph. The task is now to divide the vertices into two parts in such a way as the number of edges between the two classes is maximized. Note also that Max-Cut is a special case of Max-E2-Lin with each equation of the form

$$x_i x_j = -1.$$

4 Approximation algorithms

In the best of all worlds we would, given a Max-CSP, efficiently find the optimal solution. As stated in the introduction, however, for almost all Max-CSPs this is an NP-hard task and hence we have to ask for less. We focus on algorithms that are guaranteed to return a reasonably good solution.

Definition Let O be a maximization problem and let $C \leq 1$ be a real number. For an instance x of O let $OPT(x)$ be the optimal value. A *C-approximation algorithm* is an algorithm that on each input x outputs a number V such that $C \cdot OPT(x) \leq V \leq OPT(x)$.

One might require the algorithm to return a proof, in the case of a Max-CSP an assignment of the given quality, that indeed its output satisfies the given condition. In fact all algorithms we discuss will have this property and thus they do more than required. On the other hand it turns out that when we prove that some computational problem is hard we usually prove that already finding the approximate answer in the above sense is hard.

We have of course a similar definition for minimization problems but as we here only deal with maximization problems we do not state it formally.

Definition An *efficient* C-approximation algorithm is a C-approximation algorithm that runs in worst case polynomial time.

We also allow randomized approximation algorithms in which case we require the upper bound $V \leq OPT(x)$ to always hold while $V \geq C \cdot OPT(x)$ is relaxed to hold only in expectation. Note that the expectation is taken only over the random coin flips of the algorithm and is assumed to hold for each individual input. In particular, there is no randomness associated with the input.

Most of the algorithms described can, at an arbitrarily small loss, be derandomized and in any case running the algorithm repeatedly can make sure that, with very high probability, we get an output within (almost) a factor C of optimal.

The formulation "having approximation ratio C" is sometimes used as an alternative to saying "being a C-approximation algorithm".

Any Max-CSP-P has an approximation algorithm with constant approximation ratio.

Theorem 4.1 *Max-CSP-P admits a polynomial time approximation algorithm with approximation ratio $d(P)$.*

Proof Assume that we are given an instance with m constraints. A random assignment satisfies any given constraint with probability $d(P)$ and thus it satisfied $d(P)m$ constraints on the average. As the optimal assignment can satisfy at most all m constraints we get a randomized $d(P)$-approximation algorithm. □

Let us note that it is not difficult to deterministically find an assignment that satisfies a fraction $d(P)$ of the constraints by the method of conditional expectation. We leave the details to the reader.

The random assignment algorithm finds an assignment that satisfies $d(P)m$ constraints independent of $OPT(x)$. If we want to have a better approximation ratio then it is sufficient to do better in the case when $OPT(x)$ is close to the maximal value m.

The main classification we study in this paper is inspired by this fact and is slightly different from approximation ratio. We concentrate on what is needed from the optimal solution in order for an efficient algorithm to find an assignment that does significantly better than the naive randomized algorithm used above which simply picks a random assignment.

Definition A Max-CSP given by predicate P on k variables is *approximation resistant on satisfiable instances* if for any $\epsilon > 0$ it is NP-hard to distinguish instances where all constraints can be simultaneously satisfied from those where only a fraction $d(P) + \epsilon$ of the constraints can be simultaneously satisfied.

The existence of the arbitrarily small constant ϵ is somewhat annoying. It is not difficult to see that it cannot be the case that it is NP-hard to distinguish satisfiable instances from instances where exactly a fraction $d(P)$ of the constraints can be simultaneously satisfied, and thus some weakening is needed. The chosen weakening turns out to be convenient but there are other alternatives and in particular one could ask for ϵ to be a function of the size of the input and tend to 0 as this size increases.

While the previous definition does not exactly correspond to a statement about approximation ratio, the next definition is equivalent to saying that, again up to an arbitrary $\epsilon > 0$, the approximation ratio given by Theorem 4.1 is the best possible.

Definition A Max-CSP given by predicate P on k variables is *approximation resistant* if for any $\epsilon > 0$ it is NP-hard to distinguish instances where a fraction $(1 - \epsilon)$ of the constraints can be simultaneously satisfied from those where only a fraction $d(P) + \epsilon$ of the constraints can be simultaneously satisfied.

Next we have a class of problems that are of intermediate complexity. It is possible to find an assignment of non-trivial quality for almost satisfiable instances but when the quality of the optimal solution is below a certain threshold this is no longer possible.

Definition A Max-CSP given by predicate P on k variables is *somewhat approximation resistant* if it is not approximation resistant and there is some $\delta > 0$ such that for any $\epsilon > 0$ it is NP-hard to distinguish instances where a fraction $d(P) + \delta$ of the constraints can be simultaneously satisfied from those where only a fraction $d(P) + \epsilon$ of the constraints can be simultaneously satisfied.

Finally we have the class of problems where we can, as soon as the optimum is significantly better than the random assignment, find an assignment that also does significantly better than the random assignment.

Definition A Max-CSP given by predicate P on k variables is *always approximable* if for each $\delta > 0$ there is an $\epsilon_\delta > 0$ and an efficient algorithm that given an instance where a fraction $d(P) + \delta$ of the constraints can be simultaneously satisfied finds an assignment that satisfies at least a fraction $d(P) + \epsilon_\delta$ of the constraints.

We proceed to give some positive results in next section.

5 Constraints on two variables

The main technique used for deriving efficient approximation algorithms Max-CSPs is through semi-definite programming which was introduced in this context by Goemans and Williamson [9] as a tool to attack several problems. As the basic algorithm is very beautiful and quite easy to state, especially in the case of Max-Cut, we describe their approach for this problem here.

Let us formalize Max-Cut as a quadratic program

$$\max_{x \in \{-1,1\}^n} \sum_{(i,j) \in E} \frac{1 - x_i x_j}{2}. \tag{5.1}$$

This sum measures exactly the size of the maximal cut as each term is one if the edge is cut and zero otherwise. Let us relax (5.1) by introducing variables y_{ij} for the products $x_i x_j$ giving the program

$$\max_{y} \sum_{(i,j) \in E} \frac{1 - y_{ij}}{2}.$$

Allowing the variables y_{ij} to be completely independent would, of course, make the problem uninteresting and the key is to require that the numbers y_{ij} form a positive symmetric semidefinite matrix with ones on the diagonal. Let us write this as follows

$$\max_{y \succeq 0, y_{ii}=1} \sum_{(i,j) \in E} \frac{1 - y_{ij}}{2}. \tag{5.2}$$

Note that this is a relaxation of the original problems as if $y_{ij} = x_i x_j$ then indeed this matrix fulfills the conditions.

The magic now comes from the fact that the optimization problem (5.2) can, by a result by Alizadeh [1], be solved to any desired accuracy in polynomial time. For simplicity of discussion we ignore that we can only find an almost optimal solution and assume that we find the true optimum. This slight inaccuracy does not affect our results as we state them.

Let us phrase the problem (5.2) slightly differently. Remember that an $n \times n$ matrix Y is symmetric positive semidefinite iff there is another $n \times n$ matrix V such that

$$Y = V^T V.$$

This implies that there are vectors (in fact the columns of V) such that

$$y_{ij} = (v_i, v_j)$$

and the requirement that $y_{ii} = 1$ is equivalent to v_i being a unit vector. Thus (5.2) is equivalent to

$$\max_{(\|v_i\|=1)_{i=1}^n} \sum_{(i,j)\in E} \frac{1-(v_i,v_j)}{2}. \tag{5.3}$$

In this formulation it is obvious that that (5.3) is a strict generalization of (5.1) as we can interpret the x_i as vectors forced to lie in a single dimension.

While it is easy to interpret a one-dimensional solution as a high dimensional solution the challenging and interesting part is to take a high dimensional solution and produce a good one-dimensional solution. The inspired solution by Goemans and Williamson is to pick a random vector $r \in \mathbf{R}^n$ and set

$$x_i = \mathrm{sign}((r, v_i)), \tag{5.4}$$

where, in the probability 0 event that $(r, v_i) = 0$, we set $x_i = 1$.

Let us analyze this rounding from the approximation ratio perspective. Assume that the angle between v_i and v_j is θ_{ij}. The contribution to the objective function is then

$$\frac{1-\cos\theta_{ij}}{2}$$

while the probability that the edge is cut, i.e. that $\mathrm{sign}((r, v_i)) \neq \mathrm{sign}((r, v_j))$ is exactly $\frac{\theta_{ij}}{\pi}$. Define the real number α_{GW} by

$$\alpha_{GW} = \min_{0\leq\theta\leq\pi} \frac{\frac{\theta}{\pi}}{\frac{1-\cos\theta}{2}}.$$

The numeric value of α_{GW} is approximately .878. We get the following chain of inequalities

$$E\left[\sum_{(i,j)\in E} \frac{1-x_i x_j}{2}\right] = \sum_{(i,j)\in E} \frac{\theta_{ij}}{\pi} \geq \alpha_{GW} \sum_{(i,j)\in E} \frac{1-\cos\theta_{ij}}{2} \geq \alpha_{GW} \cdot OPT.$$

The last inequality follows as the maximum of the relaxed problem is at least the true maximum. We conclude that the given algorithm is an α_{GW}-approximation

algorithm. It is randomized but it can be derandomized [20] with an arbitrarily small loss in the quality of the obtained solution.

Let us turn to Max-2-Sat. Remember that we are using -1 to denote true and hence a clause $x_i \vee x_j$ is equivalent to the quadratic expression

$$\frac{3 - x_i - x_j - x_i x_j}{4}. \tag{5.5}$$

Negations are handled in the natural way and a clause $\bar{x}_i \vee x_j$ corresponds to

$$\frac{3 + x_i - x_j + x_i x_j}{4}, \tag{5.6}$$

with similar formulas for the other types of clauses.

The expressions (5.5) and (5.6) look essentially different from the corresponding terms for Max-Cut in that they contain linear terms and to remedy this we introduce an extra variable x_0 which always will be true. With this trick (5.5) turns into

$$\frac{3 + x_0 x_i + x_0 x_j - x_i x_j}{4}, \tag{5.7}$$

and we can relax this to

$$\frac{3 + (v_0, v_i) + (v_0, v_j) - (v_i, v_j)}{4}, \tag{5.8}$$

with unit length vectors v_i. This suggests the following approximation algorithm for Max-2-Sat.

Make an objective function by summing all quadratic expressions corresponding to clauses and solve the resulting semi-definite program. To retrieve Boolean values pick a random vector r and set x_i to true if $\text{sign}((v_i, r)) = \text{sign}((v_0, r))$.

It turns out that this algorithm gives an approximation ratio of α_{GW} for Max-2-Sat, i.e. the same constant as obtained for Max-Cut. To see this note that (5.8) can be written as

$$\frac{1 + (v_0, v_i)}{4} + \frac{1 + (v_0, v_j)}{4} + \frac{1 - (v_i, v_j)}{4}, \tag{5.9}$$

and the old argument can be applied to each term separately. That we have denominator 4 instead of 2 and that we might have plus signs instead of minus signs does not affect that analysis.

The fact that signs do not matter implies that the algorithm approximating Max-Cut can, essentially without change, be applied to Max-E2-Lin giving the same constant α_{GW} also for this problem.

Let us make a couple of observations about the algorithm for Max-2-Sat. It is clearly needed to have a non-symmetrical relation between "true" and "false" for Max-2-Sat while there is complete symmetry between the two sides in Max-Cut. Thus there is some need for v_0 or some other mechanism for breaking the symmetry between "true" and "false".

Secondly it turns out that the direction of v_0 is very special and this can be used to obtain better approximation ratios [8, 19]. Using a search over a large set

of rounding procedures Lewin et al [19] obtain and algorithm whose approximation ratio probably[5] is approximately .940.

Although it does not follow from the given analysis one can modify the rounding to show that any Boolean 2-CSP is always approximable according to our definition. Furthermore, it turns out that this result generalizes to any constant size domain [13] and thus in our terminology any Max-CSP where each constraint depends on two variables is always approximable.

6 Some approximation resistant predicates

Let us start by stating one result. We then discuss some consequences and only later, very briefly, discuss some of the ideas behind the proof.

Theorem 6.1 *[12] For any $\epsilon > 0$ it is NP-hard to approximate Max-E3-Lin within a factor $\frac{1}{2} + \epsilon$. Said equivalently Max-E3-Lin is approximation resistant.*

Stated in other terms, for any $\epsilon > 0$ it is NP-hard, given a system of linear equations modulo 2, to determine whether there is a solution that satisfies a fraction $1 - \epsilon$ of the equations or if no assignment satisfies more than a fraction $\frac{1}{2} + \epsilon$ of the equations.

Note that we cannot strengthen Theorem 6.1 to prove Max-Lin approximation resistant on satisfiable instances as Gaussian elimination gives an efficient procedure to determine whether all equations are simultaneously satisfiable. It is possible to prove a variants Theorem 6.1 with a sub-constant value for ϵ. Results along those lines have been obtain by Khot and Ponnuswami [18].

One can also note that, if variables are allowed to take values in $[m]$, Theorem 6.1 can be extended [12] to give inapproximability $\frac{1}{m} + \epsilon$ of Max-E3-Lin-m and the result even extends to equations over non-Abelian groups [7].

Let us postpone the discussion of the proof of Theorem 6.1 and first use it to obtain results for some other Max-CSPs of interest.

Theorem 6.2 *[12] For any $\epsilon > 0$ it is NP-hard to approximate Max-E3-Sat within a factor $\frac{7}{8} + \epsilon$. Said equivalently, Max-E3-Sat is approximation resistant.*

Proof We give a reduction from Max-E3-Lin. We are given a system of equations modulo 2 with three variables in each equation and we want to produce an instance of Max-E3-Sat. Since we are using $\{-1, 1\}$, addition modulo 2 is conveniently represented as multiplication and a linear equation containing three variables can be written as

$$x_i x_j x_k = b \tag{6.1}$$

for some indices i, j, and k and $b \in \{-1, 1\}$. Assume for the time being that $b = 1$

[5] This value has only been determined numerically and has not been proved analytically.

and consider the following clauses

$$(x_i \vee x_j \vee \bar{x}_k)$$
$$(x_i \vee \bar{x}_j \vee x_k)$$
$$(\bar{x}_i \vee x_j \vee x_k)$$
$$(\bar{x}_i \vee \bar{x}_j \vee \bar{x}_k).$$

Then if (6.1) is satisfied so are all four clauses while if (6.1) is not satisfied then exactly three clauses are true. If $b = -1$ then we use the four clauses

$$(x_i \vee x_j \vee x_k)$$
$$(x_i \vee \bar{x}_j \vee \bar{x}_k)$$
$$(\bar{x}_i \vee x_j \vee \bar{x}_k)$$
$$(\bar{x}_i \vee \bar{x}_j \vee x_k)$$

with the same property. Thus if we start with a system of m equations we get $4m$ clauses and an assignment that satisfies v of the equations satisfies exactly $3m + v$ clauses. It is now easy to check that the existence of an algorithm which gives a $\frac{7}{8} + \epsilon$ approximation for Max-E3-Sat with $\epsilon > 0$ implies a $\frac{1}{2} + \epsilon'$ approximation for Max-E3-Lin with $\epsilon' > 0$. Theorem 6.2 follows from Theorem 6.1. □

For Max-E3-Sat we could "hope" for approximation resistance on satisfiable instances and this is true.

Theorem 6.3 *[12] Max-E3-Sat is approximation resistant on satisfiable instances.*

There does not seem to be any fast way of deriving Theorem 6.3 from Theorem 6.1 and major parts of the proof have to be modified in a substantial way.

We will only briefly touch upon the ideas of these theorems but before we do even this we need to discuss proof systems in general.

7 Proof systems

The complexity class NP can be seen as a proof system. We have a prover P and a verifier V. P finds the witness of membership and sends it to V who then efficiently can check the proof. As an example, to prove that a formula φ is satisfiable P would supply the satisfying assignment which V then would verify by evaluating the formula.

In this standard situation V reads the entire proof, but we want to model a situation where V does spot checks and only reads a small fraction of the proof. We count the number of bits V reads from the proof and to make this easy to formalize we envision the proof in the shape of an oracle. Let Σ^* be the set of all finite binary strings.

Definition An *oracle* is a function $\Sigma^* \to \{0, 1\}$.

Written proofs are identical with proofs using oracles where reading the i'th bit of the written proof corresponds to evaluating the oracle function on the binary

representation of i. The entire concept of oracle Turing machines is just to be able to formally count the numbers of bits accessed by V when checking the proof. The reader unfamiliar with oracle Turing machine might be more comfortable by disregarding the notion and simply count the number of bits read by V in a less formal way.

A typical verifier $V^\pi(x, r)$ is a probabilistic Turing machine where π is the oracle, x the input and r the (internal) random coins of V. We say that the verifier *accepts* if it outputs 1 (written as $V^\pi(x, r) = 1$) and otherwise it *rejects*. As is standard in complexity theory we identify a decision problems with a language which is simply the set of inputs with the answer "yes". We can thus speak of the "language of satisfiable formulas" and use membership as the primitive notion.

Definition Let c and s be real numbers such that $1 \geq c > s \geq 0$. A probabilistic polynomial time Turing machine V is a verifier in a *Probabilistically Checkable Proof (PCP)* with soundness s and completeness c for a language L iff

- For $x \in L$ there exists an oracle π such that $Pr_r[V^\pi(x, r) = 1] \geq c$.
- For $x \notin L$, for all π $Pr_r[V^\pi(x, r) = 1] \leq s$.

An important property turns out to be whether the identity of later bits read by V depends on the values obtained for earlier bits read. If they do, V is called "adaptive", with the opposite being called "non-adaptive".

In the current paper we always have perfect completeness in that a correct proof of a correct statement is always accepted and hence c is always equal to one in the above definition. Using smaller values of c might seem counterintuitive but this is used in the proof of Theorem 6.1.

We are interested in a number of properties of the verifier and one property that is crucial to us is that V does not use too much randomness.

Definition The verifier V uses *logarithmic randomness* if there is an absolute constant c such that on each input x and proof π, the length of the random string r used by V^π is bounded by $c \log |x|$.

Using logarithmic randomness makes the total number of possible sets of coin flips for V polynomial in $|x|$ and hence all such sets can be enumerated in polynomial time.

We also care about the number of bits V reads from the proof.

Definition The verifier V reads c bits in a PCP if, for each outcome of its random coins and each proof π, V^π asks at most c questions to the oracle.

To get a feeling for why we discuss PCPs let us envision a proof of Theorem 6.3. An NP-hardness proof is essentially always a reduction. To prove that it is NP-hard to distinguish objects of class X from objects of class Y one describes a polynomial time algorithm that takes a Boolean formula φ as input and produces an output which is of class X if φ is satisfiable while it is of class Y if φ is not satisfiable.

In particular to prove Theorem 6.3 we would expect to have a polynomial time reduction which on input φ and a number $\epsilon > 0$, produces another formula ψ in 3-CNF with the following properties.

- If φ is satisfiable so is ψ.
- If φ is not satisfiable then any assignment satisfies only a fraction $\frac{7}{8} + \epsilon$ of the clauses of ψ.

Clearly this would prove Theorem 6.3 as an algorithm A distinguishing satisfiable formulas from formulas where only a fraction $(\frac{7}{8} + \epsilon)$ of the clauses can be simultaneously satisfied could be used to determine satisfiability. Given φ one could simply compute ψ and then run the algorithm A on input ψ.

Let us see that what we have at our hands is in fact a PCP. The written proof would now not be a satisfying assignment for φ but a satisfying assignment of ψ. This would be checked by a verifier V that given φ first constructs ψ, picks a random clause of ψ, reads the values of the three variables in the proof and accepts if the clause is satisfied. Let us check the properties of this verifier.

If φ is satisfiable then so is ψ and the prover can specify the oracle accordingly and hence V would accept with probability 1 giving perfect completeness $c = 1$.

If φ is not satisfiable then no assignment satisfies more than a fraction $\frac{7}{8} + \epsilon$ of the clauses of ψ and hence independently of the proof, the verifier would accept with probability at most $\frac{7}{8} + \epsilon$.

Note that the only randomness used by V is to pick a random clause of ψ and since ψ is of polynomial size this can be done with a logarithmic number of random coins. Finally also note that V only reads three bits of the proof.

Our type of reduction is more or less equivalent to the existence of a PCP which is limited to reading three bits. Let us sketch the reverse reduction in the case when the verifier is non-adaptive. The correspondence also exists in the adaptive case but is less tight.

Suppose there is a PCP to determine if φ is satisfiable, which reads 3 bits of the proof, has completeness one, soundness s, and where the verifier uses a logarithmic number of random bits and is non-adaptive.

Consider the "proof optimization problem" where we put ourselves in the shoes of a prover that wants to find the proof that maximizes the probability that the verifier accepts. Having not determined the proof yet we use a Boolean variable x_i to be determined as the value of the i'th bit of the proof. For each set of coinflips, r, of V it reads three bits i_1^r, i_2^r and i_3^r and accepts given a condition $C_r(x_{i_1^r}, x_{i_2^r}, x_{i_3^r})$ on these bits. Any condition on three bits can be written as a formula that is a 3-CNF and hence let us assume that C_r is of this form. Then the probability that V accepts is essentially given by the number of clauses satisfied in the formula

$$\psi = \wedge_r C_r(x_{i_1^r}, x_{i_2^r}, x_{i_3^r}).$$

We encourage the reader to work out the details of this reduction. Note that it is important that the resulting formula is of polynomial size and for this it is crucial that we only have a polynomial number of different r's. This is equivalent to saying that we only have a logarithmic number of random coins.

The existence of a PCP with even the gross behavior of what we need, i.e. reading a constant number of bits, completeness one and soundness s, a constant strictly smaller than 1 and using logarithmic randomness is already mind-boggling. It is remarkable that it is possible to efficiently verify an arbitrary NP-statement of arbitrary size reading only a constant number of bits of the proof. The existence of such a PCP was established in a sequence of papers and the final construction was given by Arora et el [2].

Theorem 7.1 *[2] There is a universal integer c such that any language in NP has a PCP with soundness $1/2$ and completeness 1 where V uses logarithmic randomness and reads at most c bits of the proof.*

Remark This theorem is often called the "PCP-theorem" or referred to as "ALMSS" after the initials of the authors.

Remark Although the number of bits read is independent of which language in NP we are considering, this is not true for the amount of randomness. The number of random bits is $d \log n$ for any language L, but the constant d depends on L.

To establish Theorem 6.1 and Theorem 6.3 one needs to find PCPs with very good constants. As the construction builds on a number of results it is not possible to present them here and we refer the interested reader to [12]. Let us give some results for other Max-CSPs.

8 Constraints on three Boolean variables

Zwick has determined good bounds for the approximation constants for all predicates on three variables [25]. We here focus on the classification into our four groups which gives us fewer details to consider.

To discuss the results it is convenient to write the predicate as a multilinear polynomial. We already used this representation when we discussed 2-CSPs but let us now do it more formally.

Theorem 8.1 *Any predicate P on k variables $x_1, x_2 \ldots x_k$, can in a unique way be written as real sum*

$$P(x) = \sum_{S \subseteq [k]} c_S^P \prod_{i \in S} x_i.$$

There are a number ways to prove this theorem and readers familiar with the discrete Fourier transform might realize that the coefficients c_S^P are exactly the Fourier coefficients. Let us give an example. If P accepts the strings $(-1,-1,-1)$, $(-1,-1,1)$, $(-1,1,-1)$ $(1,-1,-1)$ and $(1,-1,1)$ then

$$P(x) = \frac{1}{8}(5 - x_1 - 3x_2 - x_3 - x_1x_2 + x_1x_3 - x_2x_3 + x_1x_2x_3) \tag{8.1}$$

For predicates on three variables the question whether $c_{\{1,2,3\}}$ equals 0 is of crucial importance.

Theorem 8.2 *A predicate P on three variables is always approximable iff $c^P_{\{1,2,3\}} = 0$.*

Sketch of proof. If the coefficient is 0 then P can be written as a real weighted sum of predicates each depending on two variables. Such sums can be efficiently approximated as discussed in Section 5.

The proof of the reverse direction uses essentially the same reduction as used in our sketch of proof of Theorem 6.2 from Theorem 6.1. The fact that $c^P_{\{1,2,3\}} \neq 0$ is equivalent to P not accepting the same number of strings of even parity and odd parity. Assume that P accepts a strings of even parity and b strings of odd parity where $a > b$. In our example (8.1) above $a = 3$ and $b = 2$.

Now given an equation

$$x_i x_j x_k = 1 \tag{8.2}$$

we write down the four constraints

$$P(x_i, x_j, x_k)$$
$$P(x_i, \bar{x}_j, \bar{x}_k)$$
$$P(\bar{x}_i, x_j, \bar{x}_k)$$
$$P(\bar{x}_i, \bar{x}_j, x_k).$$

An assignment that satisfies (8.2) satisfies a of these constraints while an assignment that does not satisfy (8.2) satisfies b constraints. Equations of the form $x_i x_j x_k = -1$ are handled by adding one negation. We produce an instance with $4m$ constraints where an assignment that satisfies v of the linear equations satisfies $bm + v(a-b)m$ of the P-constraints. As $d(P) = (a+b)/8$ a small calculation is sufficient to prove that P is at least somewhat approximation resistant. We leave the details with the reader.

Next we look at approximation resistance.

Theorem 8.3 *A predicate P on three Boolean inputs is approximation resistant iff it is implied by parity or or the negation of parity.*

Sketch of proof. Looking more closely at the previous proof, and working out the numbers, if $a = 4$ then the given reduction establishes approximation resistance.

The proof that no other predicates are approximation resistant goes by giving an efficient approximation algorithm for each of the other predicates. We refer to [25] for the details.

As stated in the introductory sections we allow negation for free. Another degree of freedom is the order of the inputs to P and hence any two predicates that can be transformed into each other by negations of inputs together with a permutation of the inputs are, in our eyes, equivalent. Viewed this way, for $d = 4, 5, 6, 7$ there is only one predicate accepting d inputs proved to be approximation resistant by Theorem 8.3. For each of these predicates we can ask what happens on satisfiable instances.

As stated several times, parity itself is not approximation resistant on satisfiable instances while Theorem 6.3 states that the predicate we get that accepts 7 inputs does indeed have the property. This result can be extended to the predicate that accepts 6 inputs.

Theorem 8.4 *Let P be a predicate on three Boolean inputs implied by parity which accepts 6 or 7 inputs. Then P is approximation resistant on satisfiable instances.*

The only remaining question for predicates on three Boolean inputs with regards to our classification is whether a predicate P that is implied by parity and which accepts 5 inputs is approximation resistant on satisfiable instances. Such a predicate P have several equivalent formulations but one convenient way is to define it to accept the input unless exactly one of the three input bits is true. To determine whether this predicate is approximation resistant on satisfiable instances is still an open question. We known that it is approximation resistant and there is no obvious way to make use of the fact that an instance is satisfiable and not only almost satisfiable. In particular it is NP-complete to determine whether an instance is satisfiable or not.

9 Max-CSP on Boolean variables of higher width

The approximation resistance of predicates on more than three variables have been studied but results are far less complete.

Some of the results extend without problems and in particular we have the following.

Theorem 9.1 *[12] For any $k \geq 3$, parity on k variables as well as any predicate implied by parity on k variables is approximation resistant.*

From this theorem it possible to conclude that always approximable predicates are very few.

Theorem 9.2 *A predicate P is always approximable iff $c_S^P = 0$ for any S of size at least 3.*

The proof of this is a very slight generalization of the proof of Theorem 8.2. If indeed all the coefficients are 0 then we again can write P as a weighted sum of 2-CSPs.

If we have some S of size $k \geq 3$ with $c_S^P \neq 0$ we can use this as a basis of a reduction in a similar way as was sketched in the proof of Theorem 8.2. We leave the details to the interested reader.

Theorem 9.2 leads to a full characterization of the always approximable predicates. The only predicate that depends on four variables that has this property is

$$P(x) = \frac{2 + x_1x_3 + x_1x_4 + x_2x_3 - x_2x_4}{4}$$

while there is no predicate that depends on at least 5 variables and that is always approximable.

Hast [11] classified many predicates on four variables as to whether they were approximation resistant, ignoring whether approximation resistance held for satisfiable instances.

When we identify predicates that can be made equal by permuting and negating inputs and ignore the constant predicates there are are 400 different predicates on

four Boolean variables. Of these 79 were determined to be approximation resistant, 275 were found not to be approximation resistant while Hast was not able to determine the status of the remaining 46 predicates.

It is interesting to look at the division into groups depending on the number of accepting 4-tuples.

Accepted inputs	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Non-resistant	1	4	6	19	27	50	50	52	27	26	9	3	1	0	0
Resistant	0	0	0	0	0	0	0	16	6	22	11	15	4	4	1
Unknown status	0	0	0	0	0	0	6	6	23	2	7	1	1	0	0

From this table it seems evident that the more inputs a predicate accepts the more likely it is to be resistant. This is partly true but not fully true as the following theorem indicates.

Theorem 9.3 *[11] There are predicates on four variables P and Q such that P implies Q, P is approximation resistant while Q is not approximation resistant.*

Setting

$$P = ((x_1 \vee (x_2 = x_3)) \wedge (\bar{x}_1 \vee (x_2 = x_4)))$$

and

$$Q = ((x_2 = x_3) \vee (x_2 = x_4))$$

gives an example. Approximation resistance of P was proved in [10] while an approximation for Q was given in [25]. The fact that Q only depends on three variables might be taken as a drawback of this example but this can be remedied. In fact, if we make Q accept also the string $(1, 1, -1 - 1)$ then it remains only somewhat approximation resistant.

The intuition that predicates that accept few inputs are easy to approximate while predicates that accept most inputs are hard to approximate, can, however be given some formal support.

Theorem 9.4 *[11] Any predicate on k variables accepting as most $2\lfloor k/2 \rfloor + 1$ inputs is not approximation resistant.*

There are, however, some predicates that accept rather few inputs but are still approximation resistant. A prime example was given by Samarodnitsky and Trevisan [22].

Theorem 9.5 *[22] Assume $l_1 + l_2 + l_1 l_2 = k$ then there is a predicate P_{ST} on k variables which accepts $2^{l_1+l_2}$ inputs which is approximation resistant.*

The predicate is the conjunction of $l_1 l_2$ linear constraints. It is possible to prove [11] that any predicate implied by P_{ST} is also approximation resistant. This can be used to prove the following:

Theorem 9.6 *[11]Assume $l_1 + l_2 + l_1 l_2 = k$ then any predicate on k variables that accepts at least $2^k + 1 - 2^{l_1 l_2}$ inputs is approximation resistant.*

Håstad and Khot [15] extended, at the cost of slightly worse bounds, the work of Samorodnitsky and Trevisan to achieve approximation resistance on satisfiable instances.

Theorem 9.7 *[15] Assume $2l_1 + 2l_2 + l_1l_2 = k$ then there is a predicate P_{HK} on k variables which accepts $2^{2l_1+2l_2}$ inputs which is approximation resistant on satisfiable instances.*

10 Exact constants and the unique games conjecture

Many basic questions with regards to approximability of NP-hard questions remain. Due to lack of space let us not introduce more problems but instead mention some recent developments.

The approximation constant α_{GW} obtained for Max-Cut has not been improved since the original Goemans-Williamson paper over a decade ago. There is now evidence that it might be the correct constant for Max-Cut. To be more precise Khot [16] formalized in 2002 a conjecture about a game characterization of NP known as the "Unique Games Conjecture", (UGC) . This has turned out to be a strong conjecture with many important consequences, one [17] being that the Max-Cut constant α_{GW} is indeed correct.

Of the problems we discussed also the approximability of Max-2-Sat can more or less be resolved using UGC. Austrin [3] proved that, again assuming UGC, the numerically found approximation ratio for the algorithm by Lewin et al [19] (around .940) is an upper bound for the approximation ratio of any efficient approximation algorithm. Thus in the likely case that [19] did find the correct approximation ratio for their algorithm the rather natural question of the best approximation constant for Max-2-Sat might be determined and the true answer is the optimal value of an ugly but well defined optimization problem and happens to be around .940.

Samorodnitsky and Trevisan [23] proved that if d is the smallest number such that $k \leq 2^d - 1$ then, based on the UGC, there is an approximation resistant predicate on k inputs that accepts 2^d inputs and thus Theorem 9.4 might be quite close to the truth. Håstad [14] used this result to show that, again based on UGC, for large k, a random predicate P of width k is with high probability approximation resistant.

The truth of the UGC is uncertain and there seems to be no compelling evidence either to believe it or doubt it. As the number of interesting consequences of UGC increases, the urgency of proving or disproving it is mounting and UGC is now one of the main open problems of the area of PCPs and approximability of NP-hard optimization problems.

11 Conclusions and open problem

We have discussed the classification of Max-CSPs mainly in the Boolean case. Our knowledge of Max-CSPs over larger domains is far less complete[6] and a lot of work remains to be done. There are results also for larger size domain, in particular Theorem 6.1, Theorem 9.5 and Theorem 9.7 do extend [12, 6, 15] but as we do not have space to discuss these problems here we refer to those papers for a discussion.

[6] Indeed, even classical NP-hardness is just resolved for size three domains [4].

Approximation resistance on satisfiable instances is maybe the ultimate hardness condition for a CSP. Even though there is some assignment that satisfies all the constraints, efficient computation cannot do essentially better than picking an assignment at random. We do believe that already approximation resistance is a central property of a CSP a and much better evidence of hardness than the standard NP-completeness that is abundant and hence not very informative.

A few open questions have been mentioned in the text. In the best of all worlds one would have a complete characterization of which predicates fall into which category. Currently there is such a characterization only for the miniature class of always approximable predicates. The optimist could hope for a full characterization of all our classes. Of course once we have such a characterization there are many more detailed questions to study.

Acknowledgment. I am grateful to Per Austrin and an anonymous referee for comments on a preliminary version of this manuscript.

References

[1] F. Alizadeh. Interior point methods in semidefinite programming with applications to combinatorial optimization. *SIAM Journal on Optimization*, 5:13–51, 1995.

[2] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and the hardness of approximation problems. *JACM: Journal of the ACM*, 45:501–555, 1998.

[3] P. Austrin. Balanced Max-2-Sat might not be the hardest. ECCC Technical report 06/088, 2006.

[4] A. Bulatov. A dichotomy theorem for constraint satisfaction problems on a 3-element set. *JACM: Journal of the ACM*, 53:66–120, 2006.

[5] S. Cook. The complexity of theorem proving procedures. In *3rd Annual ACM Symposium on Theory of Computing*, pages 151–158, 1971.

[6] L. Engebretsen. The nonapproximability of non-boolean predicates. *SIJDM: SIAM Journal on Discrete Mathematics*, 18:114–129, 2004.

[7] L. Engebretsen, J. Holmerin, and A. Russell. Inapproximability results for equations over finite groups. *Theoretical Computer Science*, 312:17–45, 2004.

[8] U. Feige and M. Goemans. Approximating the value of two prover proof systems, with applications to Max-2-Sat and Max Dicut. In *3rd Israeli Symposium on the theory of Computing and Systems*, pages 182–189, 1995.

[9] M. Goemans and D. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems, using semi-definite programming. *Journal of the ACM*, 42:1115–1145, 1995.

[10] V. Guruswami, D. Lewin, M. Sudan, and L. Trevisan. A new characterization of NP with 3 query PCPs. In *Proceedings of 39th Annual IEEE Symposium of Foundations of Computer Science*, pages 8–17, 1998.

[11] G. Hast. Beating a random assignment. Ph.D. Thesis, Royal Institute of Technology, 2005.

[12] J. Håstad. Some optimal inapproximability results. *Journal of the ACM*, 48:798–859, 2001.

[13] J. Håstad. Every 2-CSP allows nontrivial approximation. In *Proceedings of the 37th Annual ACM Symposium on Theory of Computation*, pages 740–746, 2005.

[14] J. Håstad. On the approximation resistance of a random predicate. unpublished manuscript, 2006.

[15] J. Håstad and S. Khot. Query efficient PCPs with perfect completeness. *Theory of Computing*, 1:119–149, 2005.

[16] S. Khot. On the power of unique 2-prover 1 round games. In *Proceedings of the 34th Annual ACM Symposium on Theory of Computing*, pages 767–775, 2002.

[17] S. Khot, G. Kindler, E. Mossel, and R. O'Donnell. Optimal inapproximability results for Max-Cut and other 2-variable CSPs. In *Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science*, pages 146–154, 2004.

[18] S. Khot and A. K. Ponnuswami. Better inapproximability results for Max-Clique, Chromatic number and Min-3-Lin-Deletion. In *ICALP (1)*, pages 226–237, 2006.

[19] M. Lewin, D. Livnat, and U. Zwick. Improved rounding techniques for the Max 2-Sat and Max Di-Cut problems. In *Proceedings of the 9th IPCP, Lecture Notes in Computer Science 2337*, pages 67–82, 2002.

[20] S. Mahajan and H. Ramesh. Derandomizing approximation algorithms based on semidefinite programming. *SIAM Journal on Computing*, 28:1641–1663, 1999.

[21] C. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.

[22] A. Samorodnitsky and L. Trevisan. A PCP characterization of NP with optimal amortized query complexity. In *Proceedings of the 32nd Annual ACM Symposium on Theory of Computing*, pages 191–199, 2000.

[23] A. Samorodnitsky and L. Trevisan. Gowers uniformity, influence of variables and PCPs. In *Proceedings of the 38th Annual ACM Symposium on Theory of Computing*, pages 11–20, 2006.

[24] T. Schaefer. The complexity of satisfiability problems. In *Conference record of the Tenth annual ACM Symposium on Theory of Computing*, pages 216–226, 1978.

[25] U. Zwick. Approximation algorithms for constraint satisfaction problems involving at most three variables per constraint. In *SODA*, pages 551–560, 1998.

Johan Håstad
School of Computer Science and Communication
KTH-Royal Institute of Technology
S-100 44 Stockholm
SWEDEN
johanh@kth.se