The (Im)possibility of Simple Search-To-Decision Reductions for Approximation Problems

Alexander Golovnev ✉
Georgetown University, Washington, D. C., USA

Siyao Guo ✉
Shanghai Frontiers Science Center of Artificial Intelligence and Deep Learning, NYU Shanghai, China

Spencer Peters ✉
Cornell University, Ithaca, NY, USA

Noah Stephens-Davidowitz ✉
Cornell University, Ithaca, NY, USA

Abstract

We study the question of when an approximate search optimization problem is harder than the associated decision problem. Specifically, we study a natural and quite general model of black-box search-to-decision reductions, which we call *branch-and-bound reductions* (in analogy with branch-and-bound algorithms). In this model, an algorithm attempts to minimize (or maximize) a function $f : D \to \mathbb{R}_{\geq 0}$ by making oracle queries to $h_f : \mathcal{S} \to \mathbb{R}_{\geq 0}$ satisfying

$$\min_{x \in S} f(x) \leq h_f(S) \leq \gamma \cdot \min_{x \in S} f(x) \tag{1}$$

for some $\gamma \geq 1$ and any subset S in some allowed class of subsets $\mathcal{S}$ of the domain D. (When the goal is to maximize f, h_f instead yields an approximation to the maximal value of f over S.) We show tight upper and lower bounds on the number of queries q needed to find *even a γ'-approximate minimizer* (or maximizer) for quite large γ' in a number of interesting settings, as follows.

- For arbitrary functions $f : \{0,1\}^n \to \mathbb{R}_{\geq 0}$, where $\mathcal{S}$ contains all subsets of the domain, we show that no branch-and-bound reduction can achieve $\gamma' \lesssim \gamma^{n/\log q}$, while a simple greedy approach achieves essentially $\gamma^{n/\log q}$.
- For a large class of MAX-CSPs, where $\mathcal{S} := \{S_w\}$ contains each set of assignments to the variables induced by a partial assignment w, we show that no branch-and-bound reduction can do significantly better than essentially a random guess, even when the oracle h_f guarantees an approximation factor of $\gamma \approx 1 + \sqrt{\log(q)/n}$.
- For the Traveling Salesperson Problem (TSP), where $\mathcal{S} := \{S_p\}$ contains each set of tours extending a path p, we show that no branch-and-bound reduction can achieve $\gamma' \lesssim (\gamma - 1)n/\log q$. We also prove a nearly matching upper bound in our model.

These results show an oracle model in which approximate search and decision are strongly separated. (In particular, our result for TSP can be viewed as a negative answer to a question posed by Bellare and Goldwasser (SIAM J. Comput. 1994), though only in an oracle model.) We also note two alternative interpretations of our results. First, if we view h_f as a data structure, then our results unconditionally rule out black-box search-to-decision reductions for certain data structure problems. Second, if we view h_f as an efficiently computable heuristic, then our results show that any reasonably efficient branch-and-bound *algorithm* requires more guarantees from its heuristic than simply Eq. (1).

Behind our results is a "useless oracle lemma," which allows us to argue that under certain conditions the oracle h_f is "useless," and which might be of independent interest. See also the full version [7].

2012 ACM Subject Classification Theory of computation → Problems, reductions and completeness

Keywords and phrases search-to-decision reductions, oracles, constraint satisfaction, traveling salesman, discrete optimization

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2023).
Editors: Nicole Megow and Adam D. Smith; Article No. 10; pp. 10:1–10:20

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2023.10

Category APPROX

Related Version *Full Version*: https://eccc.weizmann.ac.il/report/2021/141/

Funding *Siyao Guo*: Supported by National Natural Science Foundation of China Grant No. 62102260, Shanghai Municipal Education Commission (SMEC) Grant No. 0920000169, NYTP Grant No. 20121201 and NYU Shanghai Boost Fund.
Spencer Peters: Supported in part by the NSF under Grant No. CCF-2122230.
Noah Stephens-Davidowitz: Supported in part by the NSF under Grant No. CCF-2122230.

1 Introduction

We study *branch-and-bound search-to-decision* reductions for approximate optimization problems. These are algorithms that attempt to find an approximate minimizer (or maximizer) of a function $f : D \to \mathbb{R}_{\geq 0}$ by making oracle queries to an oracle h_f that yields an approximation to $\min_{x \in S} f(x)$ (or to $\max_{x \in S} f(x)$) for certain subsets $S \subseteq D$.[1] In the introduction, we will typically assume that $D = \{0,1\}^n$.

1.1 Background and motivation

To explain our motivation, we first recall that any *exact* optimization problem comes in two flavors. The *decision* problem asks us to compute $\min_{x \in \{0,1\}^n} f(x)$ (or to compute the maximum) for some input objective function f.[2] The *search* problem asks us to find an $x \in \{0,1\}^n$ that optimizes $f(x)$. In practice, for *many* problems of interest, search and decision seem to be more-or-less equally hard, in the sense that the fastest known algorithm for the decision problem effectively already solves the search problem. In other words, often it seems that the best way to determine whether something exists is to look for it. (Of course, this is certainly not always the case!)

One can try to formally explain this phenomenon by showing a *search-to-decision reduction*. In other words, to show that search and decision are essentially equivalent, we can prove that we can efficiently solve the search problem using an oracle for the decision problem.

Indeed, there is a well known and very elegant greedy approach to search-to-decision reductions that works for many *exact* optimization problems. Specifically, to minimize some function $f : \{0,1\}^n \to \mathbb{R}_{\geq 0}$, one can define $f_{x_1} : \{0,1\}^{n-1} \to \mathbb{R}_{\geq 0}$ for a bit $x_1 \in \{0,1\}$ to be the function $f_{x_1}(x') := f(x_1, x')$, i.e., the function f with its first input bit set to x_1. One can then use a decision optimization oracle to find $V_{x_1} := \min_{x' \in \{0,1\}^{n-1}} f_{x_1}(x')$. (Here, we are assuming for simplicity that the decision problem is expressive enough to represent f_{x_1}. More generally, one can define f_0 and f_1 to be restrictions of f onto some sets S_0, S_1 that partition the domain.) Notice that $V_{x_1} \leq V_{1-x_1}$ if and only if there is an x that minimizes f

[1] The name "branch-and-bound reduction" comes from an analogy with *branch-and-bound algorithms* for optimization problems, which are ubiquitous in both theory and practice. (See, e.g., [14] and the references therein.) In the branch-and-bound literature, one typically uses some efficiently computable heuristic h_f to estimate $\min_{x \in S} f(x)$ in order to find an (exact or approximate) minimizer of f. See Section 1.4.

[2] Formally, to make this a true decision problem, we should include a threshold r in the input, and the problem should be to determine whether $\min_{x \in \{0,1\}^n} f(x) \leq r$. However, these two problems are essentially equivalent, as a simple binary-search-based reduction shows. We therefore ignore such subtleties.

and whose first bit is x_1. We can then repeat the process on f_{x_1} with x_1 chosen such that V_{x_1} is minimal, finding a second bit $x_2 \in \{0,1\}$ such that there exists a minimizer with first bit x_1 and second bit x_2, etc. Eventually, we will have found all of the bits of a minimizer x of f.[3]

This elegant and natural idea has many applications. Perhaps most importantly, this shows that if the decision problem of computing the minimal value of a function over subsets of the domain is solvable in time $T_D(n)$, then the search problem of *finding* a minimizer is solvable in time $T_S(n) \leq O(n) \cdot T_D(n)$, so that the search and decision variants of this problem are equivalent in quite a strong sense. Indeed, this reduction even shows a type of "instance-wise equivalence," in the sense that it shows that the search problem of minimizing any *specific* function f is essentially equivalent to the problem of computing the minimal value of simple restrictions of f. Slight variants of this simple greedy reduction work for many *exact* optimization problems of interest, so that one can conclude that search and decision are essentially equivalent (and even instance-wise equivalent) for *many* important optimization problems. (However, Bellare and Goldwasser showed that this is not always the case by proving that there exist search problems that are not solvable in polynomial time but whose decision problems *are* solvable in polynomial time [2] (assuming a certain reasonable complexity-theoretic conjecture).)

It is then natural to ask what happens when we move to *approximate* optimization problems. Indeed, Feige first raised the question of whether *approximation* can be harder than *estimation* [5], and Feige and Jozeph later showed that there exist problems for which approximation is harder than estimation if and only if $\mathsf{FP} \neq \mathsf{TFNP}$ [6]. Here, we have adopted Feige's terminology, in which the task of *finding* $x \in \{0,1\}^n$ such that $f(x) \leq \gamma \min_{x \in \{0,1\}^n} f(x)$ is called an *approximation problem*, while the task of determining $y \geq 0$ such that $\min_{x \in \{0,1\}^n} f(x) \leq y \leq \gamma \min_{x \in \{0,1\}^n} f(x)$ is called an *estimation problem*. In other words, approximation is the approximate search problem, and estimation is the approximate decision problem.

In fact, if the estimation problem is NP-complete for some approximation factor γ, then there is a certain rather weak sense in which approximation is provably equivalent to estimation (where both problems have the same approximation factor γ). For example, since such problems are polynomial-time equivalent to various exact optimization problems for which the above search-to-decision reduction works (e.g., MAX-3-SAT), one can simply combine reductions to and from such an exact optimization problem together with the search-to-decision reduction described above to reduce approximation to estimation for any NP-complete approximate optimization problem. However, this reduction is much less satisfying than the "greedy" search-to-decision reduction described above because, e.g., it can increase the size of the input instance by an arbitrary polynomial and will not in general preserve properties of the input instance (i.e., it will not prove instance-wise equivalence).[4]

[3] For a more concrete example, consider the maximization problem MAX-3-SAT. In this example, the reduction iteratively finds an assignment $(x_1, \ldots, x_n) \in \{0,1\}^n$ that maximizes the number of satisfied clauses one bit at a time, using a decision MAX-3-SAT oracle. Specifically, it uses its decision oracle to determine x_1 such that such an assignment exists with first bit x_1, then to determine x_2 such that such an assignment exists with first two bits (x_1, x_2), etc.

[4] For example, the greedy reduction described above implies that $T_S(n) \leq O(n) \cdot T_D(n)$, where $T_S(n)$ and $T_D(n)$ are "the fastest possible running times" for exact search and decision optimization problems. In contrast, the above NP-completeness-based reduction only guarantees that $T_{A,\gamma}(n) \leq T_{E,\gamma}(n^C)$ where $T_{A,\gamma}(n)$ and $T_{E,\gamma}(n)$ are "the fastest possible running times" for approximation and estimation respectively for some fixed optimization problem for which γ-estimation is NP-complete (each with approximation factor γ), and C is an arbitrarily large constant. Since $T_{E,\gamma}(n)$ is superpolynomial in n (unless $\mathsf{P} = \mathsf{NP}$), this is not a very useful conclusion. E.g., it could be the case that $T_{E,\gamma}(n) = 2^n$, while $T_{A,\gamma}(n) = 2^{n^{100}}$. The fundamental issue is that this NP-completeness-based reduction can increase the instance size n by an arbitrary polynomial.

What we would really like is a *simple* reduction from approximation to estimation, that is, a reduction roughly like the "greedy" search-to-decision reduction for exact optimization described above. Such a reduction would ideally not increase the size n of the input instance. And, it would be even better if such a reduction only called its oracle on "subinstances" of the input instance, again in analogy with the simple greedy reduction. We might even be willing to sacrifice in the approximation factor in exchange for this simplicity – reducing γ'-approximation to γ-estimation for some $\gamma' > \gamma$.

Indeed, we originally arrived at this question in the context of lattice problems, in which preserving the size and structure of the input instance is often even more important than preserving the approximation factor. (See, e.g., [13, 15, 16].)

1.2 Our model

Our definition of branch-and-bound reductions can be viewed as one way of formalizing such "simple" reductions. Specifically, our model captures reductions that work via black-box access to an estimation oracle for "subinstances," i.e., an oracle h_f that estimates the optimal value of f up to a factor of γ over a subset S of the domain D.

In more detail, for an unknown objective function $f : D \to \mathbb{R}_{\geq 0}$ over a domain D (from some family of objective functions), such (possibly randomized) reductions have access to an oracle $h_f : \mathcal{S} \to \mathbb{R}_{\geq 0}$, where $\mathcal{S} \subseteq 2^D$ is a collection of subsets of the domain D. We assume that the oracle h_f satisfies

$$\min_{x \in S} f(x) \leq h_f(S) \leq \gamma \cdot \min_{x \in S} f(x)$$

for some not-too-large $\gamma \geq 1$, i.e., that h_f is a γ-*approximate estimation oracle* that solves the estimation problem on restrictions of f to various subsets. The goal of such a reduction is to find an explicit x such that $f(x) \leq \gamma' \min_{x \in D} f(x)$ for some not-too-large γ'.

We measure the running time of the reduction entirely in terms of the number of *queries* q made to this oracle. This makes our model quite strong in some sense. In particular, the lower bounds that we describe below are lower bounds on the number of queries, and therefore even rule out algorithms that perform a bounded number q of queries but otherwise perform arbitrary unbounded computation. (To be clear, queries may even be adaptive, i.e., the result of a previous query may be used to decide what to query next.) And, we study the best achievable approximation factor γ' for the problem of optimizing f that is achievable in this model, as a function of the number of queries q and the estimation approximation factor γ.

We stress that the *only* information that these reductions have about the function f comes from the oracle h_f. In particular, these reductions do *not* take as input a description of the function f. (Formally, the input to the reduction is actually empty.) This can be viewed as a weakness of our model. But, it does capture a wide class of reductions, and even more importantly, it is precisely this choice that will allow us to prove such strong *unconditional* lower bounds on the approximation factor γ' that is achievable after a certain number of oracle queries (without, e.g., requiring us to prove that $\mathsf{FP} \neq \mathsf{TFNP}$ along the way). Indeed, if we gave our reductions direct access to the input, then our model would actually be *stronger* than the standard model(!), and we would therefore have little hope of proving unconditional lower bounds in such a model.

We call reductions in our model *branch-and-bound reductions* in analogy with the paradigm of branch-and-bound algorithms. (See Section 1.4.) E.g., the simplest branch-and-bound reductions use a greedy approach like the one described above. In other words, they find

$x^* \in \{0,1\}^n$ that approximately minimizes $f : \{0,1\}^n \to \mathbb{R}_{\geq 0}$ as follows. The reduction first *branches*, i.e., it chooses subsets $S_1, \ldots, S_k \subseteq \{0,1\}^n$ that partition the input space $\{0,1\}^n$. It then uses h_f to compute values $h_f(S_1), \ldots, h_f(S_k)$ such that $h_f(S_i) \approx \min_{x \in S_i} f(x)$. It then chooses an i such that $h_f(S_i)$ is minimal and repeats the procedure on S_i – partitioning S_i into subsets $T_1, \ldots, T_k \subseteq S_i$, computing estimates $h_f(T_j)$, selecting T_j that minimizes this estimate, and so on. Eventually, the reduction is left with a single element $x^* \in \{0,1\}^n$, and we hope that $f(x^*)$ is relatively close to the optimal value $\min_{x \in \{0,1\}^n} f(x)$. It is not hard to see that this greedy reduction makes at most $q \leq k(n/\log k + 1)$ queries and outputs x^* with

$$f(x^*) \leq \gamma^{n/\log k + 1} \min_{x \in \{0,1\}^n} f(x) \approx \gamma^{n/\log q} \min_{x \in \{0,1\}^n} f(x) \, .$$

(See Proposition 10 for a formal statement and proof.)

Even this quite large approximation factor $\gamma' \approx \gamma^{n/\log q}$ has proven to be quite useful in some rather specific contexts (e.g., [9, 16]). But, we of course would like to know if we can do better. I.e., is there a branch-and-bound reductions that makes at most q queries but achieves an approximation factor significantly better than $\gamma^{n/\log q}$? For example, a significant improvement to this approximation factor would resolve an important open problem in lattice-based cryptography [16]. (Indeed, this work originally arose from an effort to improve [16].)

1.3 Our results

Our first main result shows that the greedy reduction described above is essentially optimal in general. That is, there exist(s a distribution over) functions $f : \{0,1\}^n \to \mathbb{R}_{\geq 0}$ and a γ-approximate estimation oracle $h_f : 2^{\{0,1\}^n} \to \mathbb{R}_{\geq 0}$ such that no algorithm making significantly fewer than q queries to h_f can find $x^* \in \{0,1\}^n$ with $f(x^*) \ll \gamma^{n/\log q} \min_{x \in \{0,1\}^n} f(x)$. We use the notation $y \leftarrow \mathcal{A}^{h_f}()$ for sampling from the output distribution of the oracle algorithm $\mathcal{A}^{h_f}$, which formally takes no input.

▶ **Theorem 1** (Lower bound for arbitrary functions f. See Section 4). *For every $\gamma \geq 1$ and positive integer ℓ, there exists a distribution over functions $f : \{0,1\}^n \to \mathbb{R}_{\geq 0}$ and a γ-approximate estimation oracle $h_f : 2^{\{0,1\}^n} \to \mathbb{R}_{\geq 0}$ for f such that for any oracle algorithm $\mathcal{A}$ making at most q queries to h_f,*

$$\Pr[x^* \leftarrow \mathcal{A}^{h_f}() \ : \ f(x^*) \leq \gamma' \min_{x \in \{0,1\}^n} f(x)] \leq \varepsilon$$

where $\gamma' \approx \gamma^{n/\ell}$ and $\varepsilon \approx q2^{-\ell}$.

This shows that there is no branch-and-bound search-to-decision reduction that performs significantly better than the greedy reduction for *all* functions – even if it has an estimation oracle that works for arbitrary subsets S of the domain. So, if we want to do better than the generic greedy approach, we must place some restrictions on our objective function f. (In particular, this shows that we cannot improve upon the search-to-decision reductions in [9, 16] without using some specific properties of the relevant lattice-based objective functions f.)

We therefore turn our attention to specific classes of functions that have additional structure. Specifically, we study branch-and-bound search-to-decision reductions for Max-Constraint Satisfaction Problems (Max-CSPs), and the Traveling Salesperson Problem (TSP). These problems are natural in this context in part because both of these problems are often solved using branch-and-bound techniques in practice, for finding either exact solutions

or approximate solutions. See, e.g., [3] for discussion of branch-and-bound algorithms for Max-CSPs (specifically MAX-k-SAT), and [4] for discussion of branch-and-bound algorithms for TSP.

In the case of Max-CSPs, we show a strong lower bound for any "reasonable" set of constraints $\mathcal{F}$. (See Section 6.) In this introduction, we discuss only the application of our more general result to MAX-3-SAT for simplicity.

Our result holds for *partial assignment queries*. That is, our oracle h_I for an instance I takes as input a partial assignment w to the n input variables and outputs a γ-approximation to the maximal number of constraints in the instance I satisfied by any assignment v that matches w.[5] This notion of a partial assignment arises naturally in this context (e.g., in practical branch-and-bound algorithms for MAX–CSP).

Before we state our result, we note that it is trivial to find an assignment to a MAX-3-SAT instance that satisfies roughly a $7/8 - o(1)$ fraction of the constraints. Indeed, a random assignment suffices with high probability. So, we can easily output an assignment that satisfies at least $7/8 - o(1)$ times as many clauses as an optimal assignment (with high probability, using an algorithm that is actually independent of the instance). Our lower bound shows that a branch-and-bound reduction cannot achieve an approximation factor of better than $7/8 + o(1)$, even with a very good estimation oracle with approximation factor $\beta = 1 - o(1)$. (Here, since we have switched from minimization to maximization, we have switched to oracles h satisfying

$$\beta \max_{x \in S} f(x) \leq h_f(x) \leq \max_{x \in S} f(x)$$

for some $\beta \leq 1$.) In other words, no branch-and-bound reduction performs significantly better than the algorithm that simply outputs a uniformly random assignment, even if the estimation oracle is extremely powerful.

▶ **Theorem 2** (Lower bound for MAX-3-SAT. See Section 6 for a more general result.)**.** *There exists a distribution over* MAX-3-SAT *instances I such that for every $\beta < 1$, there is a β-approximate estimation oracle h_I such that for any oracle algorithm $\mathcal{A}$ making at most q queries to h_I,*

$$\Pr[v^* \leftarrow \mathcal{A}^{h_f}() \ : \ \mathsf{SAT}_I(v^*) \geq \delta \max_{v \in \{0,1\}^n} \mathsf{SAT}_I(v)] \leq \varepsilon$$

where $\delta = 7/8 + o(1)$ and $\varepsilon \approx q2^{-(1-\beta)^2 n}$.

Finally, we study (non-metric) TSP. Here, we consider a natural class of oracles that work for *subtour* queries. That is, they take as input a path $(v_1, \ldots, v_k)$ of (distinct) vertices in the input graph, and they output a γ-approximation to the minimal value of a tour that contains the path $(v_1, \ldots, v_k)$. This naturally captures branch-and-bound reductions that, e.g., "build a path one edge at a time." We prove the following lower bound.

▶ **Theorem 3** (Lower bound for TSP. See Section 5.)**.** *For every $\gamma > 1$ and positive integer $\ell \ll n$, there exists a distribution G of TSP instances and a γ-approximate estimation oracle h_G such that for any oracle algorithm $\mathcal{A}$ making at most q oracle queries,*

$$\Pr[c^* \leftarrow \mathcal{A}^{h_G}() \ : \ w_G(c^*) \leq \gamma' \cdot \min_c w_G(c)] \leq \varepsilon \ ,$$

where $\gamma' \approx (\gamma - 1)n/\ell$, $\varepsilon \approx qe^{-\ell}$, and $w_G(c)$ is the weight of the tour c in G.

---

[5] To formally represent this in our model, consider the function $f : \{0,1\}^n \to \mathbb{R}_{\geq 0}$ such that $f(v)$ is the number of clauses satisfied by an assignment v. Then, h_f takes as input subsets $S_w \subseteq \{0,1\}^n$ consisting of all assignments v that match a partial assignment w. E.g., $\{v = (v_1, \ldots, v_n) \in \{0,1\}^n \ : \ v_1 = 0, v_{82} = 1\}$. Of course, it is far more natural to simply consider an oracle h_I that takes w as input directly.

We note in passing that this lower bound can be viewed as a partial answer to a question posed by Bellare and Goldwasser [2], who asked whether a search-to-decision reduction was possible for approximate TSP, and particularly one that preserves the approximation factor. Theorem 3 rules out a large class of such reductions, even those achieving a significantly worse approximation factor than what was considered in [2]. But, we only rule this out in an oracle model.

Our lower bound for TSP is in some sense quite strong. For example, it shows that a polynomial-time branch-and-bound reduction cannot achieve an approximation factor $\gamma' < o(n/\log n)$, even with an estimation oracle that achieves a constant approximation factor $\gamma = O(1)$. However, it is not immediately obvious whether there is a nearly matching upper bound. E.g., the natural greedy approach achieves an approximation factor of roughly $\gamma' = \gamma^{n/k}$ using roughly n^k queries, which is quite far from our lower bound. Perhaps our lower bound can be improved?

As it happens, there *is* a nearly matching upper bound (specifically, a reduction that uses roughly n^t queries and achieves an approximation factor of roughly $\gamma n/t$). So, our lower bound is essentially tight in our model. But, the reduction achieving this upper bound is inefficient and therefore rather unsatisfying. In other words, while the reduction uses relatively few oracle queries, it performs additional computation that requires superpolynomial time, as described in Theorem 14.[6]

1.4 Other interpretations of our model

Above, we have presented our model in terms of branch-and-bound reductions from approximation problems to associated estimation problems. However, we note that there are at least two different interpretations of our model (and results) that are perhaps just as interesting.

Branch-and-bound algorithms

We of course named our model of branch-and-bound reductions in analogy with branch-and-bound *algorithms*. Such algorithms are ubiquitous in the study of optimization problems. (See, e.g., [14] and the references therein.) To view our model in terms of branch-and-bound algorithms, one simply needs to view the estimation oracle h_f as some efficiently computable *heuristic*. In other words, a branch-and-bound algorithm works by using some heuristic h_f to estimate the optimal value of f on various subsets until it has found an approximate optimizer of f.

Though branch-and-bound algorithms are very natural, they seem to be quite difficult to understand from a theoretical perspective. E.g., Nemhauser said "I have always wanted to prove a lower bound about the behavior of branch and bound, but I never could" [11]. (See also the third proposed research direction in [14].) Of course, part of the difficulty in proving such lower bounds is simply coming up with a model that is sufficiently strong to capture a wide class of branch-and-bound algorithms but weak enough to allow for provable lower bounds.

[6] The reduction first calls the oracle on all n^k subpaths $V = (v_1, \ldots, v_k)$ of length k. It then finds (in superpolynomial time) a way to combine subpaths $V_1, \ldots, V_{n/k}$ together into a tour while minimizing $h_G(V_1) + \cdots + h_G(V_{n/k})$. It is not hard to show that such a reduction achieves an approximation factor of essentially $\gamma n/k$.

One can view our results as progress towards that goal. In particular, we prove lower bounds for branch-and-bound algorithms with arbitrary estimation heuristics h_f. That is, any branch-and-bound algorithm that beats our lower bounds must use some property of the heuristic h_f that is stronger than the approximation guarantee

$$\min_{x \in S} f(x) \leq h_f(S) \leq \gamma \min_{x \in S} f(x) \ .$$

Approximation vs. estimation for data structure problems

Yet another interpretation of our results is in terms of *data structures* for optimization problems. For this interpretation, we focus on the problem of optimizing an arbitrary function with arbitrary subset queries, as this is most natural in this context.

Specifically, consider the following very natural and simple data structure problem. We are first given as input a function $f : \{0,1\}^n \to \mathbb{R}_{\geq 0}$ (or, equivalently, a list of 2^n numbers) and allowed arbitrary time to preprocess it into some data structure H (with some constraint on the size of H). Then, we receive as input some subset $S \subseteq \{0,1\}^n$. In the estimation version of this problem, our goal is to estimate $\min_{x \in S} f(x)$ using as few queries to H as possible (i.e., reading as few bits or words from H as possible). In the approximation version, our goal is to find an $x^* \in S$ such that $f(x^*)$ is as small as possible, relative to $\min_{x \in S} f(x)$ (again, using as few queries to H as possible).

It is then natural to ask whether there is a *black-box data-structure reduction* from approximation to estimation in this setting. Here, a black-box reduction means an algorithm that solves the approximation problem using *only* the estimates obtained from the data structure for the estimation problem – i.e., a branch-and-bound algorithm. Notice in particular that, in the setting of data structures, the number of queries made by such a reduction is the natural complexity measure.

Our lower bound on branch-and-bound reductions extends immediately to black-box data-structure reductions as well. Specifically, no black-box data-structure reduction making q estimate queries can achieve an approximation factor significantly better than $\gamma' \approx \gamma^{\log |S| / \log q}$.

1.5 Our techniques

Our main technical tool is a "useless oracle lemma." The idea is that "an oracle cannot be useful if most of its answers are predictable." While this lemma is quite general, we focus below on how it applies to our setting. (Very similar ideas are used in a different context in the literature on submodular optimization. See, e.g., [18, 17].)

Suppose that we can construct a distribution over functions f together with a γ-approximate estimation oracle $h_f : \mathcal{S} \to \mathbb{R}_{\geq 0}$ such that for any fixed query $S \in \mathcal{S}$, $h_f(S) = g(S)$ with high probability over f, where g is some *fixed* function that does not depend on f. E.g., our hard instance for arbitrary optimization has this property with $g(S) \approx \gamma^{n - \log |S|}$. Then, the useless oracle lemma tells us that "an algorithm with oracle access to h_f is essentially no more powerful than an algorithm with no access to any oracle that depends on f at all." The specific statement is of course quantitative and depends on the number of oracle queries and the probability that $h_f(S) \neq g(S)$. (See Section 3. We do not claim that this idea is original, though we do not know of prior work using it.)

With this tool in hand, our goal becomes to construct distributions of functions f together with oracles h_f such that (1) the oracle is "useless" in the sense described above; and (2) for every fixed $x \in \{0,1\}^n$, $f(x) \geq \gamma' \min_{x' \in \{0,1\}^n} f(x')$ with high probability over f (i.e., there is

no way to choose an x independently of f such that $f(x)$ is nearly optimal). This boils down to finding a distribution over functions f such that (1) the random variable $\min_{x\in S} f(x)$ is highly concentrated; and (2) this quantity tends to be much larger as $|S|$ becomes smaller. In particular, concentration of $\min_{x\in S} f(x)$ around some value $g(S)$ (independent of f) implies that we can set $h_f(S) \approx g(S)$. And, if $g(S)$ increases rapidly as $|S|$ becomes smaller, then the optimal value of f, $\min_{x\in\{0,1\}^n} f(x) \approx g(\{0,1\}^n)$ will in particular be much smaller than $g(\{x'\})$ for any fixed $x' \in \{0,1\}^n$, causing our final approximation factor to be large.

Below, we briefly describe the distributions that we use and some of the intuition behind them.

Arbitrary functions

For our lower bound against optimizing arbitrary functions (Theorem 1), we sample each value $f(x)$ independently (but not uniformly) from a distribution over the set $\{1, \gamma, \gamma^2 \dots, \gamma^{n/\ell}\}$, where 2^ℓ is the number of queries made by the algorithm. We choose our distribution so that any subset $S \subseteq \{0,1\}^n$ of size roughly $2^{\ell k}$ (1) contains at least one element with value either $\gamma^{n/k-1}$ or $\gamma^{n/k}$ with probability significantly larger than $1 - 2^{-\ell}$; and (2) does not contain an element with value less than $\gamma^{n/k-2}$, except with probability significantly less than $2^{-\ell}$. We can then set $h_f(S) = \gamma^{n/k}$, and we see that with high probability "any 2^ℓ queries made by the algorithm will reveal no information about the function f with high probability."

TSP

For the Traveling Salesperson Problem, we sample the weight of each edge independently to be either 1 or essentially γn, with equal probabilities. We then use the theory of random graphs to argue that with high probability the value of an optimal tour containing any subpath of length ℓ is highly concentrated around a value of roughly $\gamma\ell n/2 + n - \ell/2$.

CSPs

For ("reasonable") k-CSPs, we construct "a random instance with a random planted satisfying assignment." That is, we first sample a uniformly random assignment $P \sim [c]^n$.[7] We then repeatedly sample a k-tuple $T = (t_1, \dots, t_k) \in [n]^k$ and add to our CSP a random constraint on the variables $x_{t_1}, \dots, x_{t_k}$ that is satisfied by our planted assignment $P_{t_1}, \dots, P_{t_k}$. We do this $m = O(n)$ times to generate our CSP. (For simplicity, we are ignoring the possibility that there might not exist any such constraint. In Section 6, we handle this carefully.)

Then, for any partial assignment w, we study $\rho_w(P)$, which is the maximum over all assignments $v \in [c]^n$ extending w of the probability that v satisfies a constraint sampled as above. It is easy to see via the Chernoff-Hoeffding bound that, for a fixed planted assignment P, the maximum over all such v of the actual number of satisfied constraints in our instance will be heavily concentrated around $\rho_w(P) \cdot m$. The difficult step in our analysis is then to prove that $\rho_w(P)$ is itself highly concentrated around its expectation $\mathbb{E}_P[\rho_w(P)]$. This follows from a careful application of McDiarmid's inequality.

[7] Throughout this paper, for $x \in \mathbb{N}$, $[x]$ denotes the set $\{1, 2, \dots, x\}$.

2 Preliminaries

Given an event E and a random variable X, we will write $\Pr[E \mid X]$ for the random variable whose value in case $X = x$ is $\Pr[E \mid X = x]$. We will need the following concentration inequalities.

▶ **Lemma 4** (Chernoff-Hoeffding bound [8]). *Suppose $X_1, \ldots, X_n$ are independent random variables, and let $X := \sum_{i=1}^{n} X_i$. Then for any $0 < \delta < 1$,*

$$\Pr\left[|X - \mathbb{E}[X]| \geq \delta\, \mathbb{E}[X]\right] \leq 2\exp(-\delta^2\, \mathbb{E}[X]/3).$$

▶ **Lemma 5** (McDiarmid's inequality [12]). *Suppose $X_1, \ldots, X_n$ are independent random variables, $c_1, \ldots, c_n$ are real numbers, and $f : \mathbb{R}^n \to \mathbb{R}$ is a function with the property that for all $i \in [n]$ and $x_i, x_i' \in \mathbb{R}$, $|f(x_1, \ldots, x_i, \ldots, x_n) - f(x_1, \ldots, x_i', \ldots, x_n)| \leq c_i$. Then for all $t > 0$,*

$$\Pr\left[|f(X_1, \ldots, X_n) - \mathbb{E}[f(X_1, \ldots, X_n)]| \geq t\right] \leq 2\exp\left(\frac{-t^2}{2\sum_{i=1}^{n} c_i^2}\right).$$

3 Useless Oracles

We now present the technical tool that we will use for all of our lower bounds, which we call the useless oracle lemma. (The authors do not claim that this result is original, though they do not know of any prior work using a similar idea.)

To understand the lemma and its relation to branch-and-bound algorithms, suppose that we sample the objective function $f : D \to \mathbb{R}_{\geq 0}$ that our algorithm is meant to optimize from some distribution, and then choose our heuristic h_f according to f. And, recall that such an algorithm's "only access to this objective function f" is via oracle access to this heuristic h_f, which satisfies $\min_{x \in S} f(x) \leq h_f(S) \leq \gamma \min_{x \in S} f(x)$ for every S in some collection of subsets $\mathcal{S}$ of the domain D.

Now, imagine that instead of giving our algorithm oracle access to h_f, we give it oracle access to some *other* function g, which is fixed (i.e., not a random variable) and therefore independent of our choice of f. Notice that this is a rather cruel thing to do, since now our algorithm cannot possibly hope to glean any information about f from its oracle queries. So, at least intuitively, it should be quite easy to prove lower bounds against such algorithms.

The useless oracle lemma provides conditions that allow us to effectively replace the oracle $\mathcal{O} = h_f$ with the "useless" oracle g. In particular, it says that if (1) h_f and g have the property that $\Pr[h_f(S) = g(S)]$ is large for all *fixed* S, and (2) our algorithm does not make too many oracle queries; then the output of our algorithm is nearly the same whether it is given access to h_f or to g.

▶ **Lemma 6** (The useless oracle lemma). *Let $g : \mathcal{S} \to R$ be a* fixed *function, and let $\mathcal{O} : \mathcal{S} \to R$ be a distribution over oracles such that for all $S \in \mathcal{S}$,*

$$\Pr[\mathcal{O}(S) \neq g(S)] \leq p\,.$$

Then, for any oracle algorithm $\mathcal{A}^{\mathcal{O}}$ making at most q queries to $\mathcal{O}$, the statistical distance between $\mathcal{A}^{\mathcal{O}}()$ and $\mathcal{A}^{g}()$ is at most qp.

Proof. Let $S_1, \ldots, S_q \in \mathcal{S}$ be the sequence of oracle queries made by $\mathcal{A}^{\mathcal{O}}$. Notice that the S_i are random variables, and that S_i might depend on $\mathcal{O}(S_j)$ for $j < i$ (which is what makes the lemma non-trivial). Let E_i be the event that there exists a $j \leq i$ such that $\mathcal{O}(S_j) \neq g(S_j)$. It suffices to prove that $\Pr[E_q] \leq qp$ (because the two distributions are identical if we condition on $\neg E_q$).

Notice that

$$\Pr[E_q] = \Pr[E_{q-1}] + \Pr[\mathcal{O}(S_q) \neq g(S_q) \text{ and } \neg E_{q-1}] \,.$$

Let $S'_1, \ldots, S'_q$ be the sequence of oracle queries made by $\mathcal{A}^g$ (as opposed to $\mathcal{A}^{\mathcal{O}}$), and notice that S'_i is independent of $\mathcal{O}$ by definition. In particular, this implies that $\Pr[\mathcal{O}(S'_i) \neq g(S'_i)] \leq p$. Therefore,

$$\Pr[\mathcal{O}(S_q) \neq g(S_q) \text{ and } \neg E_{q-1}] = \Pr[\mathcal{O}(S'_q) \neq g(S'_q) \text{ and } \neg E_{q-1}] \leq \Pr[\mathcal{O}(S'_q) \neq g(S'_q)] \leq p \,.$$

It follows that $\Pr[E_q] \leq p + \Pr[E_{q-1}]$, which implies the result when combined with the trivial fact that $\Pr[E_1] \leq p$. ◀

▶ **Corollary 7.** *Let $f \sim \mathcal{D}$ be sampled over some distribution of functions $f : D \to \mathbb{R}_{\geq 0}$, and let $\mathcal{S} \subseteq 2^D$ be a collection of subsets of D such that $D \in \mathcal{S}$ and $\{x\} \in \mathcal{S}$ for all $x \in D$. Suppose that there exists some fixed function $g : \mathcal{S} \to \mathbb{R}_{\geq 0}$ such that*

$$\Pr[g(S)/\gamma \leq \min_{x \in S} f(x) \leq g(S)] \geq 1 - p$$

for all $S \subseteq \mathcal{S}$ and some $p > 0$, $\gamma \geq 1$.

Then, there exists a γ-approximate heuristic $h_f : \mathcal{S} \to \mathbb{R}_{\geq 0}$ for f such that for any algorithm $\mathcal{A}$ making at most q oracle queries to h_f,

$$\Pr[x^* \leftarrow \mathcal{A}^{h_f}(),\ f(x^*) < \gamma' \min_{x \in D} f(x)] \leq (q+2)p \,,$$

where

$$\gamma' := \gamma^{-1} \cdot \min_{x \in D} \frac{g(\{x\})}{g(D)} \,.$$

Proof. We define

$$h_f(S) := \begin{cases} g(S) & g(S)/\gamma \leq \min_{x \in S} f(x) \leq g(S) \\ \min_{x \in S} f(x) & \text{otherwise.} \end{cases}$$

In other words, $h_f(S)$ is a γ-approximate heuristic that agrees with $g(S)$ whenever it is possible for a γ-approximate heuristic to do so (and when this is not possible, it simply outputs the exact minimal value).

Notice that by assumption

$$\Pr_{f \sim D}[h_f(S) \neq g(S)] \leq p \,.$$

We may therefore apply the useless oracle lemma with $\mathcal{O} = h_f$ to show that $\mathcal{A}^{h_f}()$ is within statistical distance qp of $\mathcal{A}^g()$. So, it suffices to show that

$$\Pr_{f \sim \mathcal{D}}[x^* \leftarrow \mathcal{A}^g(),\ f(x^*) \leq \gamma' \min_{x \in D} f(x)] \leq 2p \,.$$

Indeed, since g is independent of f (as it is a fixed function), we have that for any $r \geq 0$,

$$\Pr_{f\sim\mathcal{D}}[x^* \leftarrow \mathcal{A}^g(),\ f(x^*) \leq r] = \sum_{x\in D} \Pr[x^* \leftarrow \mathcal{A}^g(),\ x^* = x] \cdot \Pr_{f\sim\mathcal{D}}[f(x) \leq r] \leq \max_{x\in D} \Pr_{f\sim\mathcal{D}}[f(x) \leq r] .$$

By assumption, for any $x \in D$,

$$\Pr[f(x) < \min_{x'\in D} g(\{x'\})/\gamma] \leq p ,$$

and similarly,

$$\Pr[\min_{x'\in D} f(x') > g(D)] \leq p .$$

Therefore, for any $x \in D$,

$$\Pr[f(x) < \gamma' \min_{x'\in D} f(x')] \leq \Pr[f(x) < \min_{x'\in D} g(\{x'\})/\gamma] + \Pr[\min_{x'\in D} f(x') > g(D)] \leq 2p ,$$

and the result follows. ◀

4 A Generic Optimization Problem

▶ **Theorem 8.** *For any n, any $\gamma \geq 1$, and any integer $1 \leq \alpha \leq n$, there exists a distribution over functions $f : \{0,1\}^n \to \mathbb{R}_{\geq 0}$ such that for any integer $1 \leq k \leq \alpha$ and any $S \subseteq \{0,1\}^n$ with $2^{(k-1)n/\alpha} \leq |S| \leq 2^{kn/\alpha}$,*

$$\Pr_f[\gamma^{\alpha-k+1} \leq \min_{x\in S} f(x) \leq \gamma^{\alpha-k+2}] \geq 1 - 4(n/\alpha)2^{-n/\alpha} .$$

Proof. For each $x \in \{0,1\}^n$ and integer $1 \leq i \leq \alpha$, we set $f(x) = \gamma^i$ independently with probability p_i, where $p_i := \delta \cdot 2^{in/\alpha - n}$ for $i = 1, \ldots, \alpha - 1$ and $p_\alpha := 1 - p_1 - p_2 - \cdots - p_{\alpha-1}$, with $\delta := n/\alpha \cdot 2^{-n/\alpha} < 1$.

Notice that

$$p_\alpha = 1 - \delta \cdot \sum_{i=1}^{\alpha-1} 2^{in/\alpha - n} \geq 1 - \delta .$$

In particular, this is non-negative, so that this is in fact a valid probability distribution. We have

$$\Pr_f[\gamma^{\alpha-k+1} \leq \min_{x\in S} f(x) \leq \gamma^{\alpha-k+2}] = 1 - \Pr[\min_{x\in S} f(x) < \gamma^{\alpha-k+1}] - \Pr[\min_{x\in S} f(x) > \gamma^{\alpha-k+2}] .$$

We wish to argue that each of the probabilities on the right-hand side are smaller than, say, 2δ. First, we see that, for the non-trivial case $k > 2$,

$$\Pr[\min_{x\in S} f(x) > \gamma^{\alpha-k+2}] \leq (1 - p_{\alpha-k+2})^{|S|} \leq \left(e^{-p_{\alpha-k+2}}\right)^{2^{(k-1)n/\alpha}} = e^{-\delta 2^{n/\alpha}} \leq \delta ,$$

where the second inequality uses the fact that $1 - x \leq e^{-x}$. Second,

$$\Pr[\min_{x\in S} f(x) < \gamma^{\alpha-k+1}] \leq |S| \cdot \Pr[f(x) \leq \gamma^{\alpha-k}] \leq 2^{kn/\alpha} \cdot \delta \cdot \sum_{i=1}^{\alpha-k} 2^{in/\alpha - n} \leq 2\delta ,$$

as needed. ◀

▶ **Corollary 9.** *For any integer n, any $1 \le \ell \le n/3$, and any $\gamma \ge 1$, there exists a distribution of functions $f : \{0,1\}^n \to \mathbb{R}_{\ge 0}$ and a γ-approximate heuristic $h_f : 2^{\{0,1\}^n} \to \mathbb{R}_{\ge 0}$ for f such that for any algorithm $\mathcal{A}^{h_f}$ making at most q oracle queries to h_f,*

$$\Pr[x^* \leftarrow \mathcal{A}^{h_f}(),\ f(x) < \gamma' \cdot \min_{x \in \{0,1\}^n} f(x)] \le (q+2) \cdot \ell 2^{-\ell+3}$$

where $\gamma' := \gamma^{\lfloor n/\ell \rfloor - 1}$.

Proof. We apply Corollary 7 to the choice of f from Theorem 8, with $g(S) := \gamma^{\alpha - k + 2}$ for the unique integer $1 \le k \le \alpha$ satisfying $2^{(k-1)n/\alpha} \le |S| < 2^{kn/\alpha}$ where $\alpha := \lfloor n/\ell \rfloor$ and $g(\{0,1\}^n) := \gamma$. Notice in particular that $g(\{x\})/g(\{0,1\}^n) = \gamma^\alpha$. ◀

The following proposition shows that the lower bound of Corollary 9 is essentially tight.

▶ **Proposition 10.** *For all integers $n \ge \ell \ge 1$, there exists an algorithm $\mathcal{A}^h$ such that for all functions $f : \{0,1\}^n \to \mathbb{R}_{\ge 0}$, all $\gamma \ge 1$, and all γ-approximate heuristics $h : 2^{\{0,1\}^n} \to \mathbb{R}_{\ge 0}$ for f, $\mathcal{A}^h$ makes at most $\lceil n/\ell \rceil \cdot 2^\ell$ queries to h, and outputs x^* with $f(x^*) \le \gamma^{\lceil n/\ell \rceil} \min_{x \in S} f(x)$.*

Proof. The algorithm $\mathcal{A}^h$ is a natural greedy algorithm. It first partitions the domain $S^0 = \{0,1\}^n$ into $k := 2^\ell$ subsets $S_1^0, \dots, S_k^0$ and queries h on each subset. It sets $S^1 = S_i^0$ for some S_i^0 of minimum h value; that is, $h(S_i^0) = \min_{1 \le j \le k} h(S_j^0)$. It then repeats this process, partitioning S^1 itself into k subsets $S_1^1, \dots, S_k^1$ of size $2^{n-2\ell}$, querying h on each one, setting S^2 to be any subset with minimum h value, and so on. After $\lceil n/\ell \rceil - 1$ iterations, it finds a subset $S^{\lceil n/\ell \rceil - 1}$ with fewer than k elements. It then queries h on each singleton subset of $S^{\lceil n/\ell \rceil - 1}$ to find a singleton subset $S^{\lceil n/\ell \rceil} =: \{x^*\}$ of minimum h value, and outputs x^*.

It is clear that $\mathcal{A}^h$ makes at most $\lceil n/\ell \rceil \cdot 2^\ell$ queries to h. Moreover, we trivially have $f(x^*) = \min_{x \in S^{\lceil n/\ell \rceil}} f(x)$. And for each i, by the fact that h is a γ-approximate heuristic for f, we have that

$$\begin{aligned}
\min_{x \in S^{i+1}} f(x) &\le h(S^{i+1}) \\
&= \min_{1 \le j \le k} h(S_j^i) \\
&\le \min_{1 \le j \le k} \gamma \cdot \min_{x \in S_j^i} f(x) \\
&\le \gamma \min_{x \in S^i} f(x).
\end{aligned}$$

It is therefore clear by induction that the output x^* satisfies $f(x^*) \le \gamma^{\lceil n/\ell \rceil} \min_{x \in S} f(x)$. ◀

In particular, when $n = \mathrm{poly}(\ell)$, the greedy algorithm achieves $\gamma' = \gamma^{\tilde{O}(n/\log q)}$, and by Corollary 9, this is the best approximation factor achievable (up to lower-order terms) by any oracle algorithm that succeeds with constant probability.

5 The Traveling Salesperson Problem

We consider undirected weighted complete graphs on n vertices with no self-loops where all edge weights are non-negative. For an edge $e = \{i, j\}$ of a graph G, $w(e)$ and $w(i, j)$ denote the weight of e.

We write C_n for the set of all Hamiltonian cycles in a complete n-vertex graph G. The weight of a cycle c is the sum of the weights of its edges: $w(c) = \sum_{i=1}^n w(c_i, c_{i+1 \bmod n})$. For $0 \le k \le n-1$, let P_n^k be the set of all length-k simple paths on an n-vertex graph:

$$P_n^k = \{(v_0, \dots, v_k) \mid v_0, \dots, v_k \in [n], v_0, \dots, v_k \text{ all distinct}\}.$$

By $\mathrm{OPT}(G)$ we denote the weight of an optimal traveling salesperson (TSP) cycle in G, and for a path $p \in P_n^k$ by $\mathrm{OPT}(G,p)$ we denote the minimum weight of a TSP cycle in G containing p.

We will use the following result that says that a random graph contains a Hamiltonian cycle with all but negligible probability.

▶ **Lemma 11** (E.g., [1]). *A uniform random undirected graph with n vertices contains a Hamiltonian cycle with probability $\geq 1 - 2^{-n+o(n)}$.*

We define a "useless oracle" for TSP in Theorem 12. Namely, for every approximation factor $\gamma > 1$ and every probability of error ($\approx e^{-t}$), we give a distribution of graphs such that with high probability, the weight of an optimal Hamiltonian cycle extending a path of length k essentially does not depend on the path, but only on its length k. Intuitively, an approximate TSP oracle is useless for this distribution of graphs because (with high probability) it reveals no information about the input graph.

▶ **Theorem 12** (TSP useless oracle). *For all $\gamma > 1$ and $1 \leq t \leq \delta n/2$ where $\delta = (\gamma-1)/(\gamma+1)$, there exists a distribution $\mathcal{G}$ over n-vertex graphs, and values $v_0, \ldots, v_{n-1}$ such that*

- *$v_0 \leq n/\gamma$ and $v_{n-1} \geq (\gamma-1)n^2/(5t)$,*
- *for all $0 \leq k \leq n-1$ and all paths $p \in P_n^k$,*

$$\Pr_{G\sim\mathcal{G}}\left[\mathrm{OPT}(G,p) \in [v_k,\ \gamma v_k]\right] \geq 1 - O(e^{-\delta^2 t/30}).$$

For reasons of space, we defer the proofs of this and subsequent theorems to the appendix. They can also be found in the full version [7].

We are now ready to prove the main result of this section showing an essentially tight bound on search to decision reductions with an approximate TSP oracle.

▶ **Definition 13** (TSP oracle). *A function h_G is a γ-approximate TSP estimation oracle for $G \in \mathcal{G}_n$ if for all $0 \leq k \leq n-1$, for all $p \in P_n^k$,*

$$\mathrm{OPT}(G,p) \leq h_G(p) \leq \gamma\,\mathrm{OPT}(G,p).$$

▶ **Theorem 14** (TSP oracle bounds). *For every $\gamma > 1$ and positive integer $\ell \leq \delta^3 n/20$ for $\delta = (\gamma-1)/(\gamma+1)$, there exists a distribution $\mathcal{G}$ over n-vertex graphs G and a γ-approximate TSP estimation oracle h_G for G such that for any algorithm $\mathcal{A}^{h_G}$ making at most q queries to h_G,*

$$\Pr[c \leftarrow \mathcal{A}^{h_G}() \ :\ w(c) \leq \gamma' \cdot \mathrm{OPT}(G)] \leq \varepsilon,$$

where $\gamma' := (\gamma-1)\delta^2 n/(150\ell)$ and $\varepsilon := O\big(q \cdot e^{-\ell}\big)$.

Furthermore, for every $\gamma \geq 1$ and positive integer ℓ, there exists an algorithm $\mathcal{A}^{h_G}$ making at most $\binom{n}{\ell} \cdot \ell! \leq n^\ell$ queries to a γ-approximate TSP estimation oracle h_G that computes a $\gamma n/(\ell-1)$-approximation to TSP.

In particular, using q queries, one can achieve $\gamma' = \gamma n/(\log q - 1)$, but no algorithm that succeeds with constant probability can do better than $\gamma' = O((\gamma-1)n/\log q))$.

6 Constraint Satisfaction Problems

Informally, a constraint satisfaction problem (CSP) consists of constraints applied to variables; the goal is to find an assignment of values to variables that satisfies all or most of the constraints. For example, graph coloring is a CSP, where each edge corresponds to the constraint that the endpoints have different colors.

A CSP is specified by a non-empty family of constraint functions $\mathcal{F}$. Each constraint function $f \in \mathcal{F}$ is a function $f : [c]^k \to \{0,1\}$, where the alphabet size c and arity k are fixed. An assignment assigns a value from $[c]$ to each of the n variables $x_1, \ldots, x_n$. Formally, we represent this by a function $v : [n] \to [c]$. An assignment v satisfies a constraint $C = (f, (j_1, \ldots, j_k))$, written $v \models C$, if $f(v(x_{j_1}), \ldots, v(x_{j_k})) = 1$. We write $\mathsf{CSP}(\mathcal{F})$ for the CSP specified by $\mathcal{F}$; an instance $I \in \mathsf{CSP}(\mathcal{F})$ is simply a collection of constraints. For simplicity, we allow duplicate constraints. As is standard, we say I is satisfiable if there is an assignment v that satisfies all constraints of I. For example, the classical 3-SAT problem is $\mathsf{CSP}(\mathcal{F}_{3-\mathrm{SAT}})$, where $\mathcal{F}_{3-\mathrm{SAT}}$ is the family of constraint functions defined by disjunctive clauses (e.g., $f(x_1, x_2, x_3) = \neg x_1 \vee x_2 \vee x_3$).

We are interested in an approximation version of the constraint satisfaction problem, namely $\mathsf{MAX\text{–}CSP}(\mathcal{F})$ [10]. $\mathsf{MAX\text{–}CSP}(\mathcal{F})$ is the computational problem whose instances I are collections of $m(I)$ constraints on variables $x_1, \ldots, x_n$, and the objective is to find an assignment v to these variables that maximizes the number of satisfied constraints.

A *partial assignment* assigns values to a subset of the n variables. Formally, a partial assignment is represented by a partial function $w : [n] \to [c]$. An assignment v *extends* w if v agrees with w on all variables to which w assigns a value. Define $\mathsf{SAT}(I)$ to be the maximum number of satisfied constraints over all assignments. Define $\mathsf{SAT}_I(w)$ in the same way, except the maximum is taken over only the assignments extending w. In the special case where w is a total assignment, this is simply the number of constraints satisfied (unsatisfied) by that assignment. Similar to before, for $\beta < 1$, we define a function h_I to be a β-heuristic if $\beta\, \mathsf{SAT}_I(w) \le h_I(w) \le \mathsf{SAT}_I(w)$.

To state our lower bound for Max-CSPs in full generality, we must first define the hard distribution $\mathcal{D}_s$ over instances $I \in \mathsf{CSP}(\mathcal{F})$. $\mathcal{D}_s$ is defined by the following sampling process. All sampling steps are done uniformly at random. First we sample a "planted" assignment P. Then, for each of s steps $i = 1, 2, \ldots, s$, we sample an ordered tuple $\left(x_{J_1^{(i)}}, \ldots, x_{J_k^{(i)}}\right)$ of k distinct variables, and sample a constraint function F_i that is satisfied by the input $\left(P\left(x_{J_1^{(i)}}\right), \ldots, P\left(x_{J_k^{(i)}}\right)\right)$. If there is no such $F_i \in \mathcal{F}$, we write $C_i = \mathsf{NULL}$ to indicate that we do not add a constraint; otherwise, C_i is the constraint $(F_i, (J_1^{(i)}, \ldots, J_k^{(i)}))$. The sampled instance $I \sim \mathcal{D}_s$ consists of all non-NULL constraints C_i. Given an assignment v, it will be convenient to write $v \models i$ to mean that $C_i \neq \mathsf{NULL}$ and $v \models C_i$.

Let $a(\mathcal{F}) := \liminf_{n\to\infty} \min_v \Pr[v \text{ does not satisfy } C_i \mid C_i \neq \mathsf{NULL}]$. That is, $a(\mathcal{F})$ is (close to, for sufficiently large n) the minimal achievable expected fraction of unsatisfied constraints (in an instance drawn from $\mathcal{D}_s$) over all fixed assignments v. Our lower bound roughly states that even for satisfiable instances, it is hard to satisfy much more than a $(1 - a(\mathcal{F}))$ fraction of constraints. Although the definition of $a(\mathcal{F})$ looks complicated, it is actually a fairly natural measure of the hardness of choosing a fixed assignment to satisfy a random constraint from $\mathcal{F}$. For example, it is not hard to see that $a(\mathcal{F}_{3-\mathrm{SAT}}) = 1/8$, and $a(\mathcal{F}_{\mathrm{k-LIN}}) = 1/2$. These are exactly the expected fraction of random constraints not satisfied by any fixed assignment. We will write a for $a(\mathcal{F})$ when $\mathcal{F}$ is clear from context.

Say that a constraint family $\mathcal{F}$ is *constant satisfiable* (resp. *constant unsatisfiable*) if there is $b \in [c]$ for which every $f \in \mathcal{F}$ has $f(b, \ldots, b) = 1$ (resp. $f(b, \ldots, b) = 0$).

▶ **Theorem 15.** *For all constraint families $\mathcal{F}$ that are not constant unsatisfiable, there is $r > 0$ such that for all $0 < \varepsilon < 1$, there are $(1-\varepsilon)$-approximate estimation oracles h_I such that for all oracle algorithms $\mathcal{A}^{h_I}$ making at most q queries to h_I, letting $I \sim \mathcal{D}_s$ and $v \leftarrow \mathcal{A}^{h_I}()$, for sufficiently large n,*

$$\Pr[\mathsf{SAT}_I(v)/\mathsf{SAT}_I \ge (1+\delta)(1-a)] \le q \exp(-(\delta^2 + \varepsilon^2) \cdot r \cdot n),$$

where $s = 1000k^2 \log(c) n/\varepsilon^2$.

In particular, for any small $\delta > 0$, for sufficiently large n, an arbitrary assignment satisfies a $1 - a - \delta$ fraction of constraints with overwhelming probability, but no algorithm can satisfy a $1 - a + \delta$ fraction of constraints unless it makes exponentially many queries.

A few additional remarks are in order. The theorem is vacuous for trivially satisfiable constraint families $\mathcal{F}$, since the assignment v mapping every variable to b satisfies all constraints, which implies $a(\mathcal{F}) = 0$. But by the same logic, no non-trivial lower bound on the approximation ratio is possible for such families. Fortunately, if $\mathcal{F}$ is not trivially satisfiable, $a(\mathcal{F})$ is strictly positive. To see this, fix an assignment v, and let b be the majority value of v. By assumption, there is $f^* \in \mathcal{F}$ such that $f^*(b, \dots, b) = 0$. Sample a random constraint $(f, (j_1, \dots, j_k))$. Independent of $(j_1, \dots, j_k)$, over the random choice of P, we have that $f^*(P(x_{j_1}), \dots, P(x_{j_k})) = 1$ with probability at least $1/c^k$. Conditional on this, f is chosen to be f^* with probability at least $1/2^{c^k}$. Hence f is chosen to be f^* with probability at least $1/c^k \cdot 1/2^{c^k}$. Suppose $n \geq ck$. For any assignment v, we have $v(x_{j_1}) = \cdots = v(x_{j_k}) = b$ with probability (over the random choice of $j_1, \dots, j_k$ independent of P and f^*) at least $\Pi_{i=0}^{k-1}(n/c - i)/(n - i) \geq \Pi_{i=0}^{k-1}(k - i)/(ck - i)$ (the last expression corresponds to the case of $n = ck$ and balanced v). Thus, for $n \geq ck$, the random constraint is unsatisfied with probability at least $1/c^k \cdot 1/2^{c^k} \cdot \Pi_{i=0}^{k-1}(k - i)/(ck - i) > 0$; it follows $a(\mathcal{F}) > 0$.

The trivial unsatisfiability condition is slightly less natural; however, some similar condition is necessary for lower bounds. Consider the problem of 3-coloring a graph to maximize the number of edges with one green endpoint and one blue endpoint. This is clearly a CSP. Starting with all nodes colored red, one can color two nodes blue and green and use a heuristic to determine if there is an edge between them. Proceeding in this way, using $O(n^2)$ queries, one can recover the whole graph and (using unbounded computation) recover the optimal coloring. So, in our model, we cannot hope to rule out a branch-and-bound algorithm for such a CSP. This CSP is ruled out by trivial unsatisfiability, because coloring all nodes red makes the only constraint in the family (the blue-green constraint) output 0.

References

1 Yahav Alon and Michael Krivelevich. Random graph's Hamiltonicity is strongly tied to its minimum degree. *The Electronic Journal of Combinatorics*, pages 1–30, 2020. 14

2 Mihir Bellare and Shafi Goldwasser. The complexity of decision versus search. *SIAM Journal on Computing*, 23(1):97–119, 1994. 3, 7

3 Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh. *Handbook of satisfiability*, volume 336. IOS press, 2021. 6

4 William J. Cook. *In pursuit of the traveling salesman.* Princeton University Press, 2011. 6

5 Uriel Feige. On estimation algorithms vs approximation algorithms. In *FSTTCS*, 2008. 3

6 Uriel Feige and Shlomo Jozeph. Separation between estimation and approximation. In *ITCS*, 2015. 3

7 Alexander Golovnev, Siyao Guo, Spencer Peters, and Noah Stephens-Davidowitz. The (im)possibility of simple search-to-decision reductions for approximation problems, 2022. `arXiv:TR22-141`. 1, 14

8 Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58:13–30, 1963. 10

9 Gengran Hu and Yanbin Pan. Improvements on reductions among different variants of SVP and CVP. In *WISA*, 2013. 5

10 Sanjeev Khanna, Madhu Sudan, Luca Trevisan, and David P Williamson. The approximability of constraint satisfaction problems. *SIAM Journal on Computing*, 30(6):1863–1920, 2001. 15

11 Richard J. Lipton and Kenneth W. Regan. Branch and bound—Why does it work? https://rjlipton.wpcomstaging.com/2012/12/19/branch-and-bound-why-does-it-work/, December 2012. 7

12 Colin McDiarmid. Concentration. In *Probabilistic methods for algorithmic discrete mathematics*, pages 195–248. Springer, 1998. 10

13 Daniele Micciancio. Efficient reductions among lattice problems. In *SODA*, 2008. 4

14 David R. Morrison, Sheldon H. Jacobson, Jason J. Sauppe, and Edward C. Sewell. Branch-and-bound algorithms: A survey of recent advances in searching, branching, and pruning. *Discrete Optimization*, 19:79–102, February 2016. doi:10.1016/j.disopt.2016.01.005. 2, 7

15 Noah Stephens-Davidowitz. Dimension-preserving reductions between lattice problems. Unpublished manuscript, 2015. URL: http://noahsd.com/latticeproblems.pdf. 4

16 Noah Stephens-Davidowitz. Search-to-decision reductions for lattice problems with approximation factors (slightly) greater than one. In *APPROX*, 2016. 4, 5

17 Zoya Svitkina and Lisa Fleischer. Submodular approximation: Sampling-based algorithms and lower bounds. *SIAM Journal on Computing*, 40(6), 2011. 8

18 Jan Vondrák. Symmetry and approximability of submodular maximization problems. In *FOCS*, 2009. 8

A Omitted Proofs

A.1 Proof of Theorem 12

Proof. $\mathcal{G}$ is defined by independently setting the weight of each edge e as follows:

$$w(e) = \begin{cases} 1 & \text{w.p. } 1/2\,, \\ \omega := 1 + (\gamma - 1)n/t & \text{w.p. } 1/2\,. \end{cases}$$

The following is the definition of v_k:

$$v_k = \begin{cases} n/\gamma & \text{if } k = 0\,, \\ n & \text{if } 1 \le k \le t\,, \\ k(\omega+1)/(\gamma+1) + n - k & \text{if } t < k < n - t\,, \\ \big(k/2 + n/(\gamma+1)\big)(\omega+1)/2 & \text{if } k \ge n - t\,. \end{cases}$$

The definitions of v_0 and v_{n-1} satisfy the first item of the theorem statement as $v_0 = n/\gamma$, and

$$v_{n-1} \ge (n-1)(\omega+1)/4 \ge (n-1)(\gamma-1)n/(4t) \ge (\gamma-1)n^2/(5t)\,.$$

It remains to bound from above $\Pr_{G\sim\mathcal{G}}\left[\mathrm{OPT}(G,p) \notin [v_k, \gamma v_k]\right]$ for every fixed $p \in P_n^k$. By the Chernoff-Hoeffding bound (Lemma 4), for every $\varepsilon \in (0,1)$, p contains from $(1-\varepsilon)k/2$ to $(1+\varepsilon)k/2$ edges of weight ω with probability at least $1 - 2e^{-\varepsilon^2 k/6}$. Therefore,

$$\Pr_{G\sim\mathcal{G}}\left[w(p) \in [(1-\varepsilon)k(\omega+1)/2,\ (1+\varepsilon)k(\omega+1)/2]\right] \ge 1 - O\big(e^{-\varepsilon^2 k/6}\big)\,. \tag{2}$$

Let $G \sim \mathcal{G}$, and let G' be the graph consisting of the vertices of G not belonging to the path p, and the edges of G of weight 1. Note that G' is a uniformly random unweighted graph with $n-k-1$ vertices. By Lemma 11, G' contains a Hamiltonian cycle with probability $1 - 2^{-(n-k-1)(1-o(1))}$. With probability at least $1 - (3/4)^{n-k-1}$ the endpoints of p are connected by edges of weight 1 to a pair of consecutive points of the Hamiltonian cycle

in G'. Thus, with probability at least $1 - O\big((3/4)^{n-k}\big)$, p can be extended to a TSP cycle in G by taking a Hamiltonian path in G', removing an edge from it, and connecting the two endpoints to the endpoints of p by edges of weight 1. Hence,

$$\Pr_{G\sim\mathcal{G}}[\mathrm{OPT}(G,p) = w(p) + n - k] \geq 1 - O\big((3/4)^{n-k}\big)\,. \tag{3}$$

Now we consider the following four cases.

- $k = 0$. For the distribution $\mathcal{G}$, Lemma 11 implies that for every $p \in P_n^0$, $\mathrm{OPT}(G,p) = \mathrm{OPT}(G) = n$ with probability $1 - 2^{-n+o(n)} \geq 1 - O(e^{-\delta^2 t/30})$.
- $1 \leq k \leq t$. For each path $p \in P_n^k$, $\mathrm{OPT}(G,p) \geq n = v_k$, and, by (3), with probability at least $1 - O\big((3/4)^{n-k}\big) \geq 1 - O\big((3/4)^{n/2}\big) \geq 1 - O(e^{-\delta^2 t/30})$, we have that

  $$\mathrm{OPT}(G,p) = w(p) + n - k \leq k\omega + n - k \leq \gamma n = \gamma v_k\,.$$

- $t < k < n - t$. From (3), with probability $1 - O\big((3/4)^t\big)$, $\mathrm{OPT}(G,p) = w(p) + n - k$. By setting $\varepsilon = \delta$, from (2), with probability at least $1 - O(e^{-\delta^2 t/6})$, $w(p) \in [k(\omega+1)/(\gamma+1),\ \gamma k(\omega+1)/(\gamma+1)]$. Together these bounds give us that

  $$\mathrm{OPT}(G,p) = w(p) + n - k \in [v_k, \gamma v_k]$$

  with probability at least $1 - O\big((3/4)^t\big) - O(e^{-\delta^2 t/6}) \geq 1 - O(e^{-\delta^2 t/30})$.
- $k \geq n - t$. From (2) with $\varepsilon = 1/2 - \frac{n}{k(\gamma+1)} \geq 1/2 - 2/(\gamma+3)$, with probability $1 - O(e^{-(1-4/(\gamma+3))^2 k/24}) \geq 1 - O(e^{-\delta^2 t/30})$ (where we used that $t \leq \delta n/2$ and $k \geq n(1-\delta/2)$), we have that

  $$\begin{aligned} w(p) &\in [(1-\varepsilon)k(\omega+1)/2,\ (1+\varepsilon)k(\omega+1)/2] \text{ and} \\ \mathrm{OPT}(G,p) &\in [w(p),\ w(p) + \omega(n-k)] \\ &\subseteq [(1-\varepsilon)k(\omega+1)/2,\ (\omega+1)(n-(1-\varepsilon)k/2)] \\ &= \big[\big(k/2 + n/(\gamma+1)\big)(\omega+1)/2,\ \big(2n - \big(k/2 + n/(\gamma+1)\big)\big)(\omega+1)/2\big] \\ &\subseteq [v_k, \gamma v_k] \end{aligned}$$

  for every $k \geq n - t \geq n(1-\delta) = 2n/(\gamma+1)$. ◀

A.2 Proof of Theorem 14

We prove the first part of the theorem using Corollary 7. For this, we first denote by $\mathcal{G}$ the distribution of graphs from Theorem 12, and by D the set of all cycles C_n. We define the distribution $\mathcal{D}$ of functions $f_G \colon D \to \mathbb{R}_{\geq 0}$ computing the length of a given cycle in $G \sim \mathcal{G}$. For a path $p \in P_n^k$, we denote by $S_p \subseteq C_n$ the set of cycles containing the path p, and we define

$$\mathcal{S} = \{S_p \colon p \in P_n^k,\ 0 \leq k \leq n-1\}\,.$$

It is easy to see that $C_n \in \mathcal{S}$ and for every cycle $c \in C_n$, $\{c\} \in \mathcal{S}$. Now for a set $S_p \in \mathcal{S}$ corresponding to a path $p \in P_n^k$, we define $g(S_p) = \gamma v_k$. By Theorem 12 with $t = 30\ell/\delta^2$, for each S_p, $\Pr[g(S_p)/\gamma \leq \min_{c\in S_p} f(c) \leq g(S_p)] \geq 1 - p$ for $p = O\big(e^{-\delta^2 t/30}\big) = O\big(e^{-\ell}\big)$. Now we apply Corollary 7 to our choices of f and g, and

$$\gamma' := \gamma^{-1} \cdot \min_{c\in D} \frac{g(\{c\})}{g(D)} \geq \gamma^{-1} \cdot v_{n-1}/v_0 \geq (\gamma-1)\delta^2 n/(150\ell)\,,$$

and have that

$$\Pr_{G\sim\mathcal{G}}[c \leftarrow \mathcal{A}^{h_G}(),\ w(c) \le \gamma' \operatorname{OPT}(G)] \le (q+2)\cdot O\big(e^{-\ell}\big) \le O\big(q\cdot e^{-\ell}\big)\,.$$

To prove the "furthermore" part, we consider the following algorithm. Let G be an input graph on n vertices, and let n be a multiple of $\ell-1$. The algorithm first queries $h_G(p)$ for each path $p \in P_n^{\ell-1}$ of length $\ell-1$. Then the algorithm returns a cycle $c=(c_0,\dots,c_{n-1}) \in C_n$ that minimizes the sum

$$H(c) := h_G(c_0,\dots,c_{\ell-1}) + h_G(c_{\ell-1},\dots,c_{2(\ell-1)}) + \ldots + h_G(c_{n+1-\ell},\dots,c_{n-1},c_0)\,.$$

It is easy to see that the algorithm makes $|P_n^{\ell-1}| = \binom{n}{\ell}\cdot \ell!$ queries to h_G.

Since for every path $p \in P_n^{\ell-1}$, $w(p) \le h_G(p)$, the weight of the resulting cycle does not exceed $H(c)$. Now it remains to show that $\min_{x\in C_n} H(x) \le \gamma n \operatorname{OPT}(G)/(\ell-1)$. To this end, consider an optimal TSP cycle $c' \in C_n$ in G. Since for every subpath $p \in P_n^{\ell-1}$ of c', $h_G(p) \le \gamma \operatorname{OPT}(G)$, we have that

$$H(c) \le H(c') \le \gamma \operatorname{OPT}\cdot n/(\ell-1)\,.$$

A.3 Proof of Theorem 15

First, in the following lemma, we show that the "optimal satisfaction probability" $\rho_w(P)$ is concentrated as a function of P.

▶ **Lemma 16.**

$$\Pr\left[|\rho_w(P) - \mathbb{E}[\rho_w(P)]| \ge (\varepsilon/10)\,\mathbb{E}[\rho_w(P)]\right] \le 2\exp\Big(-n\varepsilon^2\,\mathbb{E}[\rho_w(P)]^2/(200k^2)\Big).$$

Proof. Notice that for all assignments v, P and indices $j \in [n]$, letting E be the event $(j=J_1^{(i)}) \vee \cdots \vee (j=J_k^{(i)})$, we have

$$\rho_v(P) = \Pr[E]\cdot\Pr[v \models i \mid E,P] + \Pr[\overline{E}]\cdot\Pr[v\models i \mid \overline{E},P]$$

Observe that $\Pr[E] = k/n$, and $\Pr[v\models i \mid \overline{E},P]$ does not depend on $P(j)$. Thus, defining

$$v^{\oplus j} = \begin{cases} \neg v(x) & x = j \\ v(x) & x \ne j, \end{cases}$$

we have

$$|\rho_v(P) - \rho_v(P^{\oplus j})| \le k/n.$$

It follows that for all partial assignments w,

$$|\rho_w(P) - \rho_w(P^{\oplus j})| \le k/n.$$

Indeed, if no v improves by more than k/n in success probability, the maximum success probability over all v extending w cannot improve by more than k/n either. The desired result follows by McDiarmid's inequality (Lemma 5). ◀

It follows from the definition of $a = a(\mathcal{F})$ and the Chernoff-Hoeffding bound that there is a constant $r' > 0$ such that for all assignments v:

$$\Pr_{I\sim\mathcal{D}}[\mathsf{SAT}_I(v)/\mathsf{SAT}_I \ge (1+\delta)(1-a)] = \Pr_{I\sim\mathcal{D}}[\mathsf{SAT}_I(v) \ge (1+\delta)(1-a)m(I)] \le \exp(-r'\cdot\delta^2\cdot n).$$

(Informally, any fixed assignment is unlikely to satisfy much more than a $1-a$ fraction of constraints.)

Notice that, for $x, y > 0$, if $x \in [(1-\varepsilon/3)y, (1+\varepsilon/3)y]$, then $x \in [(1-\varepsilon)z, z]$, where $z = (1+\varepsilon/3)y$. Thus, by the useless oracle corollary (Corollary 7), Theorem 15 will follow immediately from the claim below.

▷ Claim 17. Given a partial assignment w, let $\rho_w(P) = \max_{v:v \text{ extends } w} \Pr[v \models i \mid P]$. There is a constant $r > 0$ such that

$$\Pr_{I\sim\mathcal{D}_m}\left[|\mathsf{SAT}_I(w) - m\,\mathbb{E}[\rho_w(P)]| \geq (\varepsilon/3)m\,\mathbb{E}[\rho_w(P)]\right] \leq \exp(-r\cdot\varepsilon^2\cdot n).$$

Proof. Fix a partial assignment w. We will argue that $\mathsf{SAT}_I(w)$ is concentrated conditional on P. Fixing P and an assignment v, $\mathsf{SAT}_I(v)$ is the sum of m independent coin tosses $\mathbb{1}(v \models i)$ with success probability $\rho_v(P)$. For the optimal v^* extending w, we have $\mathbb{E}[\mathsf{SAT}_I(v)] = m\rho_w(P)$, and the Chernoff-Hoeffding bound gives

$$\Pr\left[|\mathsf{SAT}_I(v^*) - m\rho_w(P)| \geq (\varepsilon/10)m\rho_w(P) \mid P\right] \leq 2\exp(-\varepsilon^2 m\rho_w(P)/600). \tag{4}$$

Moreover, for any v extending w, $\mathsf{SAT}_I(v)$ is the sum of m independent coin tosses with a smaller success probability $\rho_v(P) \leq \rho_w(P)$. Hence the upper bound of (4) holds, namely

$$\Pr[\mathsf{SAT}_I(v) \geq (1+\varepsilon/10)m\rho_w(P) \mid P] \leq \exp(-\varepsilon^2 m\rho_w(P)/600). \tag{5}$$

Recall that $\mathsf{SAT}_I(w)$ is the maximum over all v extending w of $\mathsf{SAT}_I(v)$. Inequality (4) is thus a high-probability lower bound on $\mathsf{SAT}_I(w)$, and (5) combined with a union bound over the (at most c^n) assignments v extending w gives a high-probability upper bound. Specifically, plugging in $m = \frac{1000k^2\log(c)n}{\varepsilon^2}$, we have

$$\begin{aligned}
&\Pr\left[|\mathsf{SAT}_I(w) - m\rho_w(P)| \geq (\varepsilon/10)m\rho_w(P) \mid P\right] \\
&\leq (c^n + 2)\exp(-\varepsilon^2 m\rho_w(P)/600) \\
&\leq \exp(-n\rho_w(P)/(100k^2)).
\end{aligned} \tag{6}$$

Third, combining Inequality (6) with Lemma 16 yields

$$\Pr\left[|\mathsf{SAT}_I(w) - m\,\mathbb{E}[\rho_w(P)]| \leq (\varepsilon/3)m\,\mathbb{E}[\rho_w(P)]\right] \leq 2\exp(-n\varepsilon^2\,\mathbb{E}[\rho_w(P)]^2/(200k^2)), \tag{7}$$

where we have used $\mathbb{E}[\rho_w(P)] \leq 1 \Rightarrow \mathbb{E}[\rho_w(P)]^2 \leq \mathbb{E}[\rho_w(P)]$.

To finish the proof, we show that $\mathbb{E}[\rho_w(P)]$ is bounded below by a constant independent of n and w, if $n \geq ck$. Since $\mathcal{F}$ is not trivially unsatisfiable, for all $b \in [c]$, there is $f^* \in \mathcal{F}$ with $f(b,\ldots,b) = 1$. As we argued before with trivial satisfiability, when a random constraint is sampled, it is non-NULL and has constraint function f^* with probability at least $1/c^k \cdot 1/2^{c^k}$. Letting the variable indices in the constraint be $j_1,\ldots,j_k$, again for any assignment v (for $n \geq ck$) we have $v(x_{j_1}) = \cdots = v(x_{j_k}) = b$ with probability at least $\Pi_{i=0}^{k-1}(k-i)/(ck-i)$ (again, the last expression corresponds to the case of $n = ck$ and balanced v). Thus the constraint is satisfied with probability at least $1/c^k \cdot 1/2^{c^k} \cdot \Pi_{i=0}^{k-1}(k-i)/(ck-i)$, as needed. ◁