Oblivious Algorithms for the Max-kAND Problem

Noah G. Singer
Department of Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA

Abstract

Motivated by recent works on streaming algorithms for constraint satisfaction problems (CSPs), we define and analyze *oblivious algorithms* for the Max-kAND problem. This is a class of simple, combinatorial algorithms which round each variable with probability depending only on a quantity called the variable's *bias*. Our definition generalizes a class of algorithms defined by Feige and Jozeph (Algorithmica '15) for Max-DICUT, a special case of Max-2AND.

For each oblivious algorithm, we design a so-called *factor-revealing* linear program (LP) which captures its worst-case instance, generalizing one of Feige and Jozeph for Max-DICUT. Then, departing from their work, we perform a fully explicit analysis of these (infinitely many!) LPs. In particular, we show that for all k, oblivious algorithms for Max-kAND provably outperform a special subclass of algorithms we call "superoblivious" algorithms.

Our result has implications for streaming algorithms: Generalizing the result for Max-DICUT of Saxena, Singer, Sudan, and Velusamy (SODA'23), we prove that certain separation results hold between streaming models for *infinitely many* CSPs: for *every* k, $O(\log n)$-space sketching algorithms for Max-kAND known to be optimal in $o(\sqrt{n})$-space can be beaten in (a) $O(\log n)$-space under a random-ordering assumption, and (b) $O(n^{1-1/k}D^{1/k})$ space under a maximum-degree-D assumption. Even in the previously-known case of Max-DICUT, our analytic proof gives a fuller, computer-free picture of these separation results.

2012 ACM Subject Classification Theory of computation → Approximation algorithms analysis; Theory of computation → Streaming, sublinear and near linear time algorithms; Theory of computation → Discrete optimization

Keywords and phrases streaming algorithm, approximation algorithm, constraint satisfaction problem (CSP), factor-revealing linear program

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2023.15

Category APPROX

Related Version *Full Version with Proofs*: `https://arxiv.org/abs/2305.04438`

Supplementary Material *Software (Source Code)*: `https://github.com/singerng/oblivious-csps` archived at `swh:1:dir:0662828b8cd21f298e6a1163db40e7b7d2253825`

Funding This material is based upon work supported by the National Science Foundation Graduate Research Fellowship Program under Grant No. DGE2140739. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the National Science Foundation.

Acknowledgements I would like to thank Madhu Sudan and Santhoshini Velusamy for generous feedback and comments on the manuscript; Pravesh Kothari and Peter Manohar for helpful discussions; and anonymous reviewers at APPROX whose feedback helped improve the presentation in the paper.

1 Introduction

In this work, we study a restricted but natural class of randomized algorithms called *oblivious algorithms* for a family of *constraint satisfaction problems (CSPs)* called Max-kAND for $k \geq 2$. In the Max-kAND problem, an algorithm is presented with a list of m constraints on

n Boolean variables; each constraint is a disjunction on k literals (e.g., $x_1 \wedge \neg x_4 \wedge \neg x_5$ for the case of Max-3AND); and the goal is to satisfy the highest possible fraction of constraints. We begin by introducing these problems and algorithms informally and discussing the context and motivation for our work.

1.1 Background

Context on Max-kAND

Constraints of Max-kAND are maximally "precise" among all Boolean CSPs: each constraint specifies *exactly* what its k variables must be assigned to in order for that constraint to be satisfied.[1] Numerous works have developed algorithms for Max-kAND [43, 22, 23, 8] as well as NP-hardness-of-approximation results [44, 42, 34, 15, 35]; we now know that $\Theta(k/2^k)$-approximations are the best achievable in polynomial time assuming P $\neq$ NP [8, 35].

Further attention has been devoted to important special cases of Max-kAND. One particularly important example is the Max-DICUT problem, a special case of Max-2AND where each constraint is of the form "$x \wedge \neg y$". Max-DICUT can be viewed alternatively as a directed graph optimization problem, where the goal is to find a directed cut (S, T) maximizing the number of edges (s, t) such that $s \in S$ and $t \in T$. Approximation algorithms for Max-DICUT (and sometimes Max-2AND) were developed in [18, 16, 33, 30], and its hardness-of-approximation was studied in [24]. Max-3AND was studied in [47, 45].

The importance of Max-kAND, the extensive work on its polynomial-time approximability, and the "preciseness" of its constraints have inspired significant study on its approximability in restricted algorithmic settings. For instance, Trevisan [43] showed that the natural linear programming (LP) relaxation for Max-kAND beats the trivial (uniformly random rounding) algorithm's approximation ratio by a factor of 2, and that this LP's "nice" structure allows it to be solved (approximately) by distributed algorithms. Max-DICUT in particular has been studied extensively in various restrictive algorithmic frameworks and models, including "combinatorial" algorithms [21], spectral partitioning algorithms [46], local search algorithms [1, 2], parallel algorithms [6], near-linear time algorithms [40], and online algorithms [5].

Oblivious algorithms for Max-kAND

In this paper, we define a restricted class of algorithms for Max-kAND called *oblivious algorithms* (which were previously studied in the case of Max-DICUT by Feige and Jozeph [17]). Oblivious algorithms fall into the paradigm of *probabilistic rounding* algorithms. These algorithms somehow choose, for each variable v, a probability $p_v \in [0, 1]$, and then assign x_v to 1 w.p. p_v and 0 w.p. $1 - p_v$ (independently of all other variables); the goal is then to choose these probabilities efficiently while ensuring that the resulting assignment satisfies many of the constraints in expectation.

Informally, oblivious algorithms are probabilistic rounding algorithms which choose rounding probabilities for each variable v in a "very local" way. It is simplest to define these algorithms, as in [17], in the case of (the graph-theoretic view of) Max-DICUT. In this context, oblivious algorithms choose a rounding probability for each vertex depending only its *bias*. In a directed graph G, the bias of a vertex v is a real number in the interval $[-1, +1]$,

[1] In particular, as observed by Trevisan [43], an *arbitrary* Boolean predicate ϕ of arity k with r satisfying assignments can be converted to r "disjoint" applications of Max-kAND constraints; this transformation makes an instance of Max-CSP(ϕ) into an instance of Max-kAND and drops the value by a factor exactly r. In turn, this means that algorithms for Max-kAND can approximate the acceptance probability of k-bit probabalistically checkable proofs (PCP) verifiers.

and is defined as the difference between v's in- and out-degrees divided by its total degree. (Note that the in- and out-degrees of a vertex completely describe what the vertex can "see" about the graph around it by exploring paths of length 1.) The natural generalization of bias to a variable in Max-kAND is the difference between its number of positive and negative appearances in the instance, divided by the total number of appearances,[2] and oblivious algorithms again are those which round each variable depending only on its bias.

In this paper, we also define a class of algorithms called *superoblivious algorithms*. These are a subclass of oblivious algorithms which round each variable depending only on the *sign* of the bias (i.e., positive, negative, or zero) and not the magnitude; for instance, they ignore the distinction between variables of bias +0.9 (95% of appearances are positive) and bias +0.1 (55% of appearances are positive). A principal focus of this paper is showing that for all k, oblivious algorithms can achieve strictly better approximation ratios vs. superoblivious algorithms for Max-kAND, and we motivate this focus in the following section.

Max-kAND and streaming algorithms

Max-kAND and its special case Max-DICUT have recently received heavy attention in one particular family of models, namely, those of *streaming algorithms* [20, 14, 7, 37, 36].[3] In these models, the algorithm has highly bounded space (relative to n, the number of variables in the instance) and takes one or more passes over the list of constraints in the instance before outputting an estimate for the value of the instance. It turns out that certain separation results established between streaming models in [36, 37] are closely related to the distinction between oblivious and superoblivious algorithms described in the previous subsection, and to illuminate this connection, we pause to give a quick recap of these works on streaming algorithms for CSPs.

Chou, Golovnev, and Velusamy [14] designed a $O(\log n)$-space streaming algorithm which (arbitrarily-close-to-)4/9-approximates the value of instances of Max-DICUT, and showed that this algorithm is optimal in $o(\sqrt{n})$ space. Chou, Golovnev, Sudan, and Velusamy [12, 13] vastly generalized this "dichotomy" result to hold for all CSPs (even over non-Boolean alphabets): They showed that for every CSP predicate ϕ, there is a ratio α^*, below which there are $O(\log n)$-space approximation algorithms, and above which there are no $o(\sqrt{n})$-space algorithms.[4]

The recent works of Saxena, Singer, Sudan, and Velusamy [37, 36] demonstrated that Max-DICUT exhibits a streaming-complexity-theoretic phenomenon previously unbeknownst to any CSP: It admits approximation algorithms in certain streaming regimes which beat the optimal $o(\sqrt{n})$-space approximation ratio of 4/9 [14]. Specifically, better-than-4/9-approximations are achievable either by (i) using $\widetilde{O}(\sqrt{n})$ space or (ii) assuming that the input stream of constraints is randomly ordered. These works relied heavily on the investigation by Feige and Jozeph [17] of oblivious algorithms for Max-DICUT. Indeed, Feige and Jozeph [17] constructed a $0.483 > 4/9$-approximation oblivious algorithm for Max-DICUT, and the main thrust of [37, 36] is to use stronger streaming models to "implement" the [17] algorithm by sampling random edges and calculating the biases of their endpoints, thereby beating the 4/9-approximation of [14].

---

[2] Or, more generally in weighted instances, the total weight of the clauses in which it appears positively vs. those in which it appears negatively.

[3] Other works in the broader area of approximation algorithms for CSPs in streaming models include [3, 4, 29, 26, 27, 9, 10, 11, 19, 27, 28, 39]. See also Sudan's survey [41] and the author's undergraduate thesis [38].

[4] Technical note: The lower bounds in [12, 13] only hold in general for a special class of streaming algorithms called sketching algorithms; we ignore this point in the exposition.

The principal motivation for the current work is extending the results of [37, 36] to Max-kAND for all k: That is, showing that tweaking the streaming model can lead to improved approximations vs. the model studied in the dichotomy theorem of [12]. It is not difficult to show that oblivious algorithms can be "implemented" in these modified streaming settings, analogously to the Max-DICUT case. Thus, letting α_k^* denote the $O(\log n)$-vs.-$\Omega(\sqrt{n})$-space dichotomy threshold for Max-kAND, the goal becomes to show that oblivious algorithms beat the ratio α_k^*. A recent work of Boyland, Hwang, Prasad, Singer, and Velusamy [7] studied exactly this ratio, and showed that for each k, the optimal algorithm for Max-kAND essentially "implements" a superoblivious algorithm. (Note that *a priori* one cannot simulate a superoblivious algorithm in $O(\log n)$ space; the algorithm in [7] instead measures some quantitative properties of the input instance which turn out to relate to the behavior of a certain superoblivious algorithm.) In particular, the threshold ratio α_k^* equals the ratio achieved by *some* superoblivious algorithm. Thus, to prove the separation results, it suffices to show that oblivious algorithms strictly outperform superoblivious algorithms (for all k); as mentioned above, this is the main focus of the current work.

Linear programming as a tool for analyzing approximation algorithms

The key technical ingredient in the current work is the definition and analysis of a so-called *factor-revealing linear program (LP)*. In such an LP, feasible solutions encode instances of the problem at hand (i.e., Max-kAND); when we fix an algorithm in the designated class (i.e., an oblivious algorithm), the objective function "reveals" the approximation ratio the algorithm achieves on any given instance. Similar programs were first studied in depth for facility location problems by Jain, Mahdian, Markakis, Saberi, and Vazirani [25], and have been examined in other contexts such as online bipartite matching [31]. Our LP is a generalization of the one developed by Feige and Jozeph [17] for Max-DICUT.

To show that there is a 0.483-approximation oblivious algorithm for Max-DICUT, Feige and Jozeph [17] used computer analysis of their LP; as mentioned above, the fact that $0.483 > 4/9$ – that is, oblivious algorithms outperform superoblivious algorithms for Max-DICUT – in turn powered the works of [37, 36] in establishing that there are improved approximations in stronger streaming models. However, this result of [17] was used as a black box. We believe that our approach, which contrasts between "superoblivious" and more general "oblivious" algorithms and attacks the corresponding LPs from an analytical perspective, gives a more natural and systematic explanation for why Max-DICUT admits these improved approximations – and implies that Max-kAND does as well, for all k.

1.2 Results

Next, we turn to statements of our results. In our notation, an oblivious algorithm for Max-kAND is denoted $\texttt{Obl}_k^{\mathbf{t},\mathbf{p}}$, where $\mathbf{t}$ and $\mathbf{p}$ are, respectively, a *bias partition* which splits the space of possible biases (i.e., the interval $[-1,+1]$) into discrete intervals, and a *rounding vector* $\mathbf{p}$ specifying a probability with which to round variables for each of these intervals. We denote by $\alpha(\texttt{Obl}_k^{\mathbf{t},\mathbf{p}})$ the approximation ratio achieved by this algorithm. (See Section 2 below for formal definitions of these objects.)

To state the theorems properly, we first define some relevant quantities which first arose in the context of small-space sketching algorithms for Max-kAND in the work of Boyland *et al.* [7]. We define

$$\gamma_k \stackrel{\text{def}}{=} \begin{cases} \frac{1}{k} & k \text{ odd} \\ \frac{1}{k+1} & k \text{ even,} \end{cases} \tag{1}$$

and

$$p_k^* \stackrel{\text{def}}{=} \frac{1}{2}(1+\gamma_k) \tag{2}$$

and

$$\alpha_k^* \stackrel{\text{def}}{=} 2\cdot(p_k^*(1-p_k^*))^{\lfloor k/2\rfloor}. \tag{3}$$

That is, for even k, $\alpha_k^* = 2^{-(k-1)}(1-1/(k+1))^{k/2}(1+1/(k+1))^{k/2}$, and for odd k, $\alpha_k^* = 2^{-(k-1)}(1-1/k)^{(k-1)/2}(1+1/k)^{(k-1)/2}$. In particular, at $k=2$, we have $p_k^* = 2/3$ and $\alpha_k^* = 4/9$.

Our first theorem states that the optimal superoblivious algorithm for Max-kAND achieves ratio α_k^*, and that this algorithm rounds with probability p_k^*:

▶ **Theorem 1** (Characterization for superoblivious algorithms). *For every $k \geq 2$, there is a unique superoblivious algorithm $\mathtt{Obl}_k^{\mathbf{t},\mathbf{p}}$ achieving ratio $\alpha(\mathtt{Obl}_k^{\mathbf{t},\mathbf{p}}) = \alpha_k^*$, and all other superoblivious perform strictly worse. (In particular, for $\mathbf{t} = (0,1)$, every rounding vector $\mathbf{p} = (p)$ satisfies $\alpha(\mathtt{Obl}_k^{\mathbf{t},\mathbf{p}}) \leq \alpha_k^*$, with equality if and only if $p = p_k^*$.)*

Our main theorem then states that one can improve over these superoblivious algorithms using other oblivious algorithms, and indeed, it suffices to consider only slight "perturbations" of the optimal superoblivious algorithms:

▶ **Theorem 2** (Main theorem: Better oblivious algorithms). *For every $k \geq 2$, there exists a bias partition $\mathbf{t}$ and a rounding vector $\mathbf{p}$ such that the oblivious algorithm $\mathtt{Obl}_k^{\mathbf{t},\mathbf{p}}$ achieves $\alpha(\mathtt{Obl}_k^{\mathbf{t},\mathbf{p}}) \gneq \alpha_k^*$. (In particular, there exists $\epsilon^* > 0$ such that for all $0 < \epsilon \leq \epsilon^*$, there exists $0 < \delta < 1$ such that $\mathbf{t} = (\delta, 1)$ and $\mathbf{p} = (p_k^* + \epsilon)$ satisfy $\alpha(\mathtt{Obl}_k^{\mathbf{t},\mathbf{p}}) \gneq \alpha_k^*$.)*

These theorems are both proven by analyzing the dual of a certain natural "factor-revealing" linear program. Arguably, this "dual" perspective systematizes the *ad hoc* analyses of small-space sketching algorithms for Max-DICUT in [14] and for Max-kAND in [7]. Indeed, the analysis in those paper examined certain systems of linear inequalities using "elementary" reasoning (i.e., taking nonnegative linear combinations), and it is exactly this type of reasoning which is captured by the technology of dual linear programs. Our perspective also sheds direct light on why oblivious algorithms outperform the 4/9 ratio for Max-2AND was key to the streaming separations established for Max-DICUT by Saxena *et al.* [37, 36].

Indeed, our results also have the following implications for streaming algorithms, generalizing connections established by Saxena, Singer, Sudan, and Velusamy [37] for the special case of Max-DICUT. We say an instance Ψ of Max-kAND is in *input form* if it is unweighted (i.e., the weight of every clause is 1), though multiple copies of the same clause are allowed. These instances will be the input to our algorithms though this is essentially without loss of generality as general instances can be "rounded" to such instances via standard arguments.

▶ **Theorem 3** (Random-order streaming algorithm). *For all $k \geq 2$, there exists $\alpha > \alpha_k^*$ such that for all $\epsilon > 0$, the following holds. There is an $O(\log n)$-space streaming algorithm which, for every instance Ψ of* Max-kAND *in input form with n variables and* $\mathrm{poly}(n)$ *clauses, given as input Ψ's clauses in a* randomly-ordered *stream, outputs an $(\alpha-\epsilon)$-approximation to the* Max-kAND *value of Ψ with probability* 99/100.

▶ **Theorem 4** (Bounded-degree streaming algorithm). *For all $k \geq 2$, there exists $\alpha > \alpha_k^*$ such that for all $\epsilon > 0$, the following holds: For all $D \geq 2$, there is an $O(D^{1/k}n^{1-1/k}\log n/\epsilon^{2/k})$-space streaming algorithm which, for every instance Ψ of input form with n variables and* maximum degree $\leq D$ *(i.e., every variable is contained in $\leq D$ clauses), given as input Ψ's clauses in an* adversarially-ordered *stream, outputs an $(\alpha_k^*+\epsilon)$-approximation to the* Max-kAND *value of Ψ with probability* 99/100.

Table 1 The table below displays concrete approximation ratio for Max-kAND, for $k \in \{2, \ldots, 5\}$. In the second column, we write the trivial upper bound of $2^{-(k-1)}$ on the approximation ratio of all oblivious algorithms (and more generally all "local" algorithms) for Max-kAND (see Observation 7 below). In the third column, we write α_k^*, the approximation ratio achieved by the best superoblivious algorithm (and also by the best $o(\sqrt{n})$-space sketching algorithm [7]). In the fourth column, we highlight that the only previously known CSP for which oblivious algorithms outperformed superoblivious algorithms was Max-2AND (from [17]). In the fifth, we include approximation ratios from our "perturbed superoblivious" algorithms, as in Theorem 2 with $\delta = 0.01$ and $\epsilon = 0.001$. Finally, in the sixth column, we report ratios achieved by much more complex algorithms which we constructed. These algorithms are parametrized by triples (ℓ, x, y), where ℓ specifies the number of bias classes, and x, y specify a rounding vector. ℓ is chosen such that the number of variables, which is roughly $(2\ell)^k$ (see Section 3.1 below), is in at most the hundreds of thousands, in order for the LP solver to run in a reasonable amount of time. The bias partition is a uniform partition of $[0, 1]$ into ℓ intervals and, imitating the algorithm in [17, Proof of Theorem 1.3], our rounding vector are "two-piece piecewise-linear functions": the first part of the vector, up to bias x, interpolates linearly between probability $\frac{1}{2}$ and y, and the second part interpolates linearly between probability y and 1. The values in the last two columns were calculated with a Python script and the LP solver `glpk`; the code is available online at `https://github.com/singerng/oblivious-csps`. For the final column, the parameters (x, y) were chosen using a grid search; solving the final LPs took 1 hour, 56 minutes on a 2021 Macbook Pro.

k	Upper bound	Superobl.	Prev.	New, pert.	New, piecewise lin.
2	$1/2 = 0.5$	$4/9 \approx 0.4444$	0.4835 [17]	0.4457	0.4844 @ $(200, 0.5, 1.0)$
3	$1/4 = 0.25$	$2/9 \approx 0.2222$		0.2226	0.2417 @ $(30, 0.7, 1.0)$
4	$1/8 = 0.125$	$72/625 = 0.1152$		0.1157	0.1188 @ $(11, 0.8, 0.8)$
5	$1/16 = 0.0625$	$36/625 \approx 0.0576$		0.0578	0.0589 @ $(7, 0.95, 0.8)$

Both of these results are interesting because, as shown by Boyland *et al.* [7] (analyzing families of algorithms and lower bounds due to Chou *et al.* [12]), there are $O(\log n)$-space streaming algorithms which output (arbitrarily close to) α_k^*-approximations for adversarially-ordered streams, and this is the best achievable ratio in $o(\sqrt{n})$ space.[5] Therefore, our results show that by relaxing either the adversarial-ordering assumption or the space bound, one can achieve better algorithms (the latter under a bounded-degree assumption). Analogous results to Theorems 3 and 4 were obtained for Max-DICUT by Saxena, Singer, Sudan, and Velusamy [36] for the special case of Max-DICUT, there contrasting with algorithms and lower bounds due to Chou, Golovnev, and Velusamy [14].

We also include some explicit improved approximation ratios calculated using computer search and LP solvers in Table 1.

1.3 Technical overview

The first main technical step in the paper is to develop, for each oblivious algorithm $\texttt{Obl}_k^{\mathbf{t},\mathbf{p}}$ (defined by a bias partition $\mathbf{t}$ and a rounding vector $\mathbf{p}$), a linear program (LP) which characterizes the approximation ratio of $\texttt{Obl}_k^{\mathbf{t},\mathbf{p}}$; this LP is contained in Lemma 11 below, and is a generalization of the LP developed for oblivious algorithms in Max-DICUT in [17]. The LP has a simple structure: Each feasible solution corresponds to a certain family of instances of Max-kAND on which $\texttt{Obl}_k^{\mathbf{t},\mathbf{p}}$ produces the same approximation to the Max-kAND

[5] Again, this lower bound is currently only known to hold for a subclass of streaming algorithms called *sketching* algorithms, but the algorithm in Theorem 4 appears to be such an algorithm.

value, and the objective equals this approximation value. In particular, we will assign to each clause in an instance Ψ of Max-kAND a "pattern" based on the biases of and negations on the variables, and use the observation that the probability that any particular clause is satisfied depends only on its pattern.

Next, we formulate the dual LP for this original "primal" LP (see Lemma 12 below). Here is where we benefit massively from the fact that the performance of oblivious algorithms is captured by a linear program, because by the magic of LP duality, it is possible to constructively show that oblivious algorithms perform *well*: While feasible solutions to the *primal* LP *upper-bound* the ratio achieved by an oblivious algorithm $\texttt{Obl}_k^{\mathbf{t},\mathbf{p}}$, feasible solutions to the *dual* LP *lower-bound* the ratio! In other words, to prove that an oblivious algorithm performs well on *all* instances, it suffices to construct a *single* feasible dual solution.

To prove Theorem 2, we now want to compare the dual LP for superoblivious algorithms and their "perturbations", and show that we can get "improved" feasible solutions in the latter case. It turns out that in this setting, the primal LP has $O(k^5)$ variables and only 7 inequality constraints; therefore the dual LP has 7 variables and $O(k^5)$ inequality constraints. We make the crucial observation that in the superoblivious case there is an optimal dual solution which is *sparse*: It is supported on only 3 variables. Since this dual solution is so simple, we can analytically prove its feasibility for all k. The statement of the kind of dual solution we are seeking is in Lemma 13 below.

And moreover, this inequality will show that in the superoblivious case, when we plug in our special solution, all but 6 of the $O(k^5)$ dual constraints have slack! Thus, for very small values of δ, it will be sufficient to slightly perturb this special solution in a way that makes these 6 "core" constraints strictly satisfied. This involves careful analysis based on certain elementary inequalities, using simple inequalities such as that $\frac{1-\epsilon}{1-k\epsilon} > \frac{1+\epsilon}{1+k\epsilon}$ for all $\epsilon > 0$ and $k > 1$. The analysis outlined in this paragraph is omitted from this version of the paper; see the full version.

1.4 Future questions

Streaming algorithms

We hypothesize that the bounded-degree assumption in Theorem 4 can be relaxed to give an $\widetilde{O}(n^{1-1/k})$-space algorithm for *all* instances (in input form with poly(n) clauses). Specifically, Saxena, Singer, Sudan, and Velusamy [36] developed sketching techniques enabling such a guarantee for Max-DICUT, the bounded-degree counterpart being provided by their earlier work [37]; perhaps these ideas can be extended to Max-kAND.

More CSPs

It would also be interesting to extend the framework in this paper to more CSPs, both other Boolean CSPs and to CSPs over larger alphabets. To the best of our knowledge, it is even plausible that every CSP which admits nontrivial $O(\log n)$-space sketching algorithms (as analyzed in [13]) also admits "oblivious-style" approximation algorithms, which in turn yield better sublinear-space streaming algorithms. A good starting point here would be to analyze symmetric Boolean CSPs, since in that setting we know that all CSPs which do not support one-wise distributions of satisfying assignments admit such nontrivial sketching algorithms (see [12, Proposition 2.10]); this class includes Max-kAND, for which we developed such results in this paper, but we could hope for improved "oblivious-style" algorithms for other such CSPs, e.g. symmetric threshold functions.

"Uniform" hard instances

In the special case of Max-DICUT, Feige and Jozeph [17] constructed what might be called a *uniformly hard* instance of Max-DICUT: For this single instance, *every* oblivious algorithm achieves a ratio less than than 0.49. (This strengthens the $\frac{1}{2}$ bound from a "trivial" instance, a single bidirected edge; see also Observation 7 below.) It would be interesting to construct similar instances for Max-kAND, $k \geq 3$, especially if such this construction could be made analytic. We note that such an object corresponds to a feasible solution to the linear program in Lemma 11 for which *every* choice of rounding vector has objective strictly less than $\frac{1}{2}$, and therefore for Max-kAND, any proof would require certifying that a certain degree-k polynomial is bounded below $\frac{1}{2}$ over $p \in [0, 1]$.

An optimal rounding curve?

To construct an 0.483-approximate oblivious algorithm for Max-DICUT, Feige and Jozeph [17] rounded vertices using (a discretization of) a sigmoid-shaped piecewise-linear function: This function rounds vertices with bias $b \in (0, 1]$ to 1 with probability

$$p(b) = \begin{cases} \frac{1}{2} + b & 0 \leq b \leq \frac{1}{2} \\ 1 & \frac{1}{2} \leq b \leq 1 \end{cases}.$$

But is it possible to analytically calculate the optimal rounding function (and is it unique)? Given any discretization of biases into intervals, one could in principle enumerate all basic feasible solutions to the LP, and then calculate the best rounding vector; that is, each rounding vector will induce an objective function for the LP, and the best rounding vector maximizes the minimum objective over all basic feasible solution. Towards this, it might be helpful to get a handle on the vertices of this LP's polytope, and whether there is some simple way to enumerate them.

Outline

We define Max-kAND and oblivious algorithms formally in Section 2 and develop the linear-programming characterization for the approximation ratio of oblivious algorithms, and some other basic tools, in Section 3. We begin analyzing the dual LP to prove Theorem 2 in Section 4. The remainder of the proof of Theorem 2, and the proofs of the remaining theorems, are omitted; see the full version.

2 Definitions: Max-kAND and oblivious algorithms

We now give formal definitions for the Max-kAND problem and for oblivious algorithms.

The Max-kAND problem

For the remainder of the paper, we adopt a (nonstandard) convention which views variables in Max-kAND as taking $\{-1, +1\}$ values (as opposed to $\{0, 1\}$); this is for notational convenience in defining bias and similar concepts.

▶ **Definition 5** (Max-kAND). *An* instance *of the* Max-kAND *problem on n variables is given by a sequence of constraints $C_1, \ldots, C_m$, with $C_j = (V_j^+, V_j^-, w_j)$, consisting of "positive variables" $V_j^+ \subseteq [n]$ and "negative variables" $V_j^- \subseteq [n]$ with $V_j^+ \cap V_j^- = \emptyset$ and $|V_j^+ \cup V_j^-| = k$, and a weight $w_j \geq 0$. An* assignment *for this problem is given by $(\mathbf{x}) = (x_1, \ldots, x_n) \in \{\pm 1\}^n$, and the* value *of this assignment is*

$$\mathsf{val}_\Psi(\mathbf{x}) \stackrel{\text{def}}{=} \frac{\sum_{j=1}^m \mathbb{1}[x_v = +1\ \forall v \in V_j^+ \wedge x_v = -1\ \forall v \in V_j^-] \cdot w_j}{\sum_{j=1}^m w_j}.$$

The value *of the instance* Ψ *is*

$$\mathsf{val}_\Psi \stackrel{\text{def}}{=} \max_{\mathbf{x} \in \{\pm 1\}^n} \mathsf{val}_\Psi(\mathbf{x}).$$

Next, towards defining the bias of a variable in an instance, for any variable $v \in [n]$, we define its *positive* and *negative weight*:

$$w_\Psi^+(v) \stackrel{\text{def}}{=} \sum_{j=1}^m \mathbb{1}[v \in V_j^+] \cdot w_j \text{ and } w_\Psi^-(v) \stackrel{\text{def}}{=} \sum_{j=1}^m \mathbb{1}[v \in V_j^-] \cdot w_j. \tag{4}$$

Bias

Then, we define the *bias* of a variable as:[6]

$$\mathsf{bias}_\Psi(v) \stackrel{\text{def}}{=} \frac{w_\Psi^+(v) - w_\Psi^-(v)}{w_\Psi^+(v) + w_\Psi^-(v)}. \tag{5}$$

Bias partitions

Next, we define notations for a partition of the space of possible biases $[-1, +1]$ into $L = 2\ell+1$ intervals labeled by $\{-\ell, \ldots, +\ell\}$. The data of such a partition is a "bias partition" vector $\mathbf{t} = (t_0, \ldots, t_\ell)$ with $0 \le t_0 < \cdots < t_{+\ell} = 1$. Each such vector $\mathbf{t}$ defines a set of intervals $\mathsf{Int}_{-\ell}^{\mathbf{t}}, \ldots, \mathsf{Int}_{+\ell}^{\mathbf{t}}$ such that each real number in $[-1, +1]$ belongs to exactly one interval. Specifically, we define:

- $\mathsf{Int}_0^{\mathbf{t}} := [-t_0, +t_0]$, and
- $\mathsf{Int}_{-i}^{\mathbf{t}} := [-t_i, -t_{i-1})$ and $\mathsf{Int}_{+i}^{\mathbf{t}} := (+t_{i-1}, t_i]$ for each $i \in \{1, \ldots, \ell\}$.

Our choice of which ends of these intervals are open and which are closed is an arbitrary convention; the only important property of the decomposition of $[-1, +1]$ into intervals is that it is *symmetric*.

We also let t_i^+ and t_i^- denote the upper and lower bounds on $\mathsf{Int}_i^{\mathbf{t}}$, respectively, i.e., $t_i^+ = \sup \mathsf{Int}_i^{\mathbf{t}}$ and $t_i^- = \inf \mathsf{Int}_i^{\mathbf{t}}$, so that e.g., $t_i^+ = t_i$ for $i \ge 0$, whereas $t_i^+ = t_{-(i+1)}$ for $i < 0$.

Oblivious algorithms

Given a partition of $[-1, +1]$ into bias class intervals (defined by some vector $\mathbf{t}$ as in the previous paragraph), we now define an algorithm which randomly rounds each variable of an Max-kAND instance according to which interval its bias falls into. The data of such a rounding scheme is a *rounding vector* $\mathbf{p} = (p_1, \ldots, p_\ell)$ of probabilities; given this vector, a vertex with bias in $\mathsf{Int}_{+i}^{\mathbf{t}}$ is rounded to 1 with probability p_i.

More precisely, we define an oblivious algorithm as follows:

[6] Throughout the paper, we assume every variable appears in at least one constraint, and therefore that $w_\Psi^+(v) + w_\Psi^-(v) > 0$. (This is WLOG, since variables appearing in no constraints can be ignored for the purposes of Max-kAND.)

▶ **Definition 6** (Oblivious algorithm for Max-kAND). *Let $L = 2\ell + 1 \geq 3$ be an odd integer. Let $\mathbf{t} = (t_0, \ldots, t_\ell)$ be a bias partition and $\mathbf{p} = (p_1, \ldots, p_\ell)$ a rounding vector. For any $k \geq 2$, the oblivious algorithm $\mathtt{Obl}_k^{\mathbf{t},\mathbf{p}}$ for* Max-kAND *behaves as follows: Given an instance Ψ, for each variable $v \in \{1, \ldots, n\}$ independently:*

- *If $\mathsf{bias}_\Psi(v) \in \mathsf{Int}_0^{\mathbf{t}}$, assign $x_v \mapsto 1$ w.p. $\frac{1}{2}$, $x_v \mapsto -1$ w.p. $\frac{1}{2}$.*
- *If $\mathsf{bias}_\Psi(v) \in \mathsf{Int}_{+i}^{\mathbf{t}}$ for $i \in \{1, \ldots, \ell\}$, assign $x_v \mapsto 1$ w.p. p_i, $x_v \mapsto -1$ w.p. $1 - p_i$.*
- *If $\mathsf{bias}_\Psi(v) \in \mathsf{Int}_{-i}^{\mathbf{t}}$ for $i \in \{1, \ldots, \ell\}$, assign $x_v \mapsto 1$ w.p. $1 - p_i$, $x_v \mapsto -1$ w.p. p_i.*

We denote by $\mathtt{Obl}_k^{\mathbf{t},\mathbf{p}}(\Psi)$ the expected value of the assignment produced by this algorithm,[7] *and by*

$$\alpha(\mathtt{Obl}_k^{\mathbf{t},\mathbf{p}}) \stackrel{\text{def}}{=} \inf_{\Psi} \frac{\mathtt{Obl}_k^{\mathbf{t},\mathbf{p}}(\Psi)}{\mathsf{val}_\Psi}$$

the approximation ratio achieved by this algorithm.

In the simplest interesting case, we have $\ell = 1$, $\mathbf{t} = (0, 1)$, and $\mathbf{p} = (p)$. These algorithms, which we call *superoblivious* algorithms, ignore the *magnitude* of the bias of each variable, rounding only based on sign: E.g., negatively-biased variables are rounded to 1 w.p. $1 - p$.

▶ **Observation 7** (A "symmetric" hard instance). *There is a simple lower-bound construction which shows that* no *oblivious algorithm for* Max-kAND *can achieve a ratio better than $2^{-(k-1)}$. (Note that the constant α_k^* defined above, which according to Theorem 1 is the optimal ratio of superoblivious algorithms, equals this upper bound times a "discounting" factor.) Consider any k and the instance with two equally weighted constraints $C^+ = (+1, \ldots, +1), (1, \ldots, k)$ and $C^- = (-1, \ldots, -1), (1, \ldots, k)$; that is, the two constraints want $x_1, \ldots, x_k$ to be all-$(+1)$'s and all-(-1)'s, respectively. Every variable has bias zero so it will be rounded uniformly by every oblivious algorithm, yielding value 2^{-k}, while the "greedy" all-$(+1)$'s (or all-(-1)'s) assignment achieves value $\frac{1}{2}$. (Indeed, this "lower bound" holds for any class of algorithms which cannot "break the symmetry" between these two greedy assignments.) Thus, for* every *oblivious algorithm $\mathtt{Obl}_k^{\mathbf{t},\mathbf{p}}$, $\alpha(\mathtt{Obl}_k^{\mathbf{t},\mathbf{p}}) \leq 2^{-(k-1)}$.*

▶ **Observation 8** (Generalizing oblivious algorithms). *There are a few natural ways to generalize our definition of oblivious algorithms:*

- *We could consider rounding functions which are not "antisymmetric". That is, we could round variables with bias $+b$ and variables with bias $-b$ with probabilities which are not complementary. In particular, for $b = 0$, we could round bias-0 variables to 1 with probability $p \neq \frac{1}{2}$. However, on the instance described in the previous observation, such an algorithm outputs an assignment of value $\frac{1}{2}p^k + \frac{1}{2}(1-p)^k$, and this value is uniquely maximized at $p = \frac{1}{2}$.*
- *We could use "general rounding functions", that is, arbitrary functions mapping each bias value to a rounding probability, to round each vertex. Under this "rounding functions" view, the above definition via intervals corresponds to "step fucntions" where the domain is broken up into finitely many intervals on which the function is constant. However, the focus of the current work is analyzing the approximation ratio of oblivious algorithms via linear programming, and such "continuous" algorithms would not be (directly) amenable to analysis via linear programming.*

[7] We abuse this notation and often think of $\mathtt{Obl}_k^{\mathbf{t},\mathbf{p}}(\Psi)$ as the *output* of the oblivious algorithm, i.e., we think of the oblivious algorithm's goal as outputting a (scalar) estimate of the value of the instance; this holds especially in the context of streaming algorithms.

- *We could use a "max" algorithm, taking an ensemble of oblivious algorithms and outputting the best assignment produced by any of them.*

For Max-2AND*, Feige and Jozeph [17] showed that even allowing all these generalizations simultaneously stills fall short of achieving the ultimate goal of a $\frac{1}{2}$-approximation, and they described a specific instance witnessing this.*

3 The linear-programming framework for oblivious algorithms

In this section, we develop a linear program which captures the "worst-case instance" for any oblivious algorithm, and therefore can be used to calculate the approximation ratio (Lemma 11), as well as the corresponding dual linear program (Lemma 12). These will be applied to bound the approximation ratios of certain oblivious algorithms in the following sections.

3.1 Clause patterns

Let Ptn_k^L denote the set of vectors $\mathbf{c} = (\mathbf{c}^+, \mathbf{c}^-) = (c^+_{-\ell}, \dots, c^+_{+\ell}, c^-_{-\ell}, \dots, c^-_{-\ell})$ whose entries are natural numbers and sum to k. These are useful because they describe each particular clause from the perspective of an L-class oblivious algorithms. In particular, given a clause C, we denote its *pattern* $\mathsf{ptn}^{\mathbf{t}}(C) = (c^+_{-\ell}, \dots, c^+_{+\ell}, c^-_{-\ell}, \dots, c^-_{-\ell}) \in \mathsf{Ptn}_k^L$ where c_i^+ and c_i^- denote the number of positive and negative literals in C whose variables have bias class i, respectively, for each $i \in \{-\ell, \dots, +\ell\}$. That is, e.g.,

$$c_i^+ = |\{v \in V_j^+ : \mathsf{bias}_\Psi(v) \in \mathsf{Int}_i^{\mathbf{t}}\}|.$$

Now, for any rounding vector $\mathbf{p} = (p_1, \dots, p_\ell)$, we define

$$\mathsf{prob}^{\mathbf{p}}(\mathbf{c}) = 2^{-(c_0^+ + c_0^-)} \prod_{i=1}^{\ell} p_i^{c^+_{+i} + c^-_{-i}} (1-p_i)^{c^-_{+i} + c^+_{-i}} \tag{6}$$

for each $\mathbf{c} \in \mathsf{Ptn}_k^L$.[8] Then we have:

▷ Claim 9. Let Ψ be an instance of Max-kAND with clauses $C_1, \dots, C_m$ with weights $w_1, \dots, w_m$, respectively. Then

$$\mathtt{Obl}_k^{\mathbf{t},\mathbf{p}}(\Psi) = \frac{\sum_{j=1}^m \mathsf{prob}^{\mathbf{p}}(\mathsf{ptn}^{\mathbf{t}}(C_j)) \cdot w_j}{\sum_{j=1}^m w_j}.$$

Proof. By linearity of expectation, it suffices to show that each clause C_j is satisfied w.p. $\mathsf{prob}^{\mathbf{p}}(\mathsf{ptn}^{\mathbf{t}}(C_j))$. We can rewrite

$$\mathsf{prob}^{\mathbf{p}}(\mathbf{c}) = 2^{-c_0^+} 2^{-c_0^-} \prod_{i=1}^{\ell} p_i^{c^+_{+i}} p_i^{c^-_{-i}} (1-p_i)^{c^-_{+i}} (1-p_i)^{c^+_{-i}}.$$

Recalling that each variable is assigned independently, and the clause is satisfied iff each literal is, the above expression precisely represents the probability that the clause is satisfied. (E.g., if there is a negative literal whose variable has bias class $+i$, this literal is satisfied with probability $1-p_i$; the number of such factors in the probability is c^-_{+i}.) ◁

We observe that $|\mathsf{Ptn}_k^L| = \binom{k+2L-1}{2L-1}$ by the "stars-and-bars" formula. For instance, if $L = 3$ (as will be the case in the explicit analysis in the following sections), we have $|\mathsf{Ptn}_k^L| = O(k^5)$.

[8] In this expression we adopt the convention $0^0 = 1$, i.e., if $p_i = 0$ but $c^-_{+i} + c^+_{-i} = 0$ then we ignore the factor 0.

3.2 The factor-revealing linear program

We denote by $\mathsf{PosPtn}_k^L \subseteq \mathsf{Ptn}_k^L$ the space of clause patterns without negations, i.e., $\mathbf{c}$ such that $c^-_{-\ell} = \cdots = c^-_{+\ell} = 0$. For two vectors $\mathbf{x} = (x_1, \ldots, x_n), \mathbf{y} = (y_1, \ldots, y_n) \in \mathbb{R}^n$, let $\mathbf{x} \odot \mathbf{y} = (x_1 y_1, \ldots, x_n y_n)$ denote their entrywise product. To design the linear program, we will need the following useful proposition:

▶ **Proposition 10** (Flipping). *Let Ψ be an instance of* Max-kAND*, and for any assignment $\mathbf{y} = (y_1, \ldots, y_n) \in \{\pm 1\}^n$, let $\mathsf{flip}^{\mathbf{y}}(\Psi)$ denote the instance of* Max-kAND *where we "flip" the variables v with $y_v = -1$; that is, each clause $C_j = (V_j^+, V_j^-, w_j)$ in Ψ becomes a clause $D_j = (U_j^+, U_j^-, w_j)$ where $U_j^+ = \{v \in V_j^+ : y_v = +1\} \cup \{v \in V_j^- : y_v = -1\}$ and $U_j^- = \{v \in V_j^+ : y_v = -1\} \cup \{v \in V_j^- : y_v = +1\}$. Then:*

- *For every assignment $\mathbf{x} \in \{\pm 1\}^n$, $\mathsf{val}_\Psi(\mathbf{x}) = \mathsf{val}_{\mathsf{flip}^{\mathbf{y}}(\Psi)}(\mathbf{x} \odot \mathbf{y})$.*
- *In particular, if $\mathbf{x}$ is an optimal assignment to Ψ, then $\mathbf{x} \odot \mathbf{y}$ is an optimal assignment to $\mathsf{flip}^{\mathbf{y}}(\Psi)$.*
- $\mathtt{Obl}_k^{\mathbf{t},\mathbf{p}}(\Psi) = \mathtt{Obl}_k^{\mathbf{t},\mathbf{p}}(\mathsf{flip}^{\mathbf{y}}(\Psi))$.

Proof. Follows immediately from definitions. ◀

▶ **Lemma 11** (Primal characterization). *For every bias partition $\mathbf{t} = (t_0, \ldots, t_\ell)$ and rounding vector $\mathbf{p} = (p_1, \ldots, p_\ell)$, the approximation ratio $\alpha(\mathtt{Obl}_k^{\mathbf{t},\mathbf{p}})$ achieved by $\mathtt{Obl}_k^{\mathbf{t},\mathbf{p}}$ equals the value of the following linear program:*

$$\begin{cases} \text{minimize} & \displaystyle\sum_{\mathbf{c} \in \mathsf{Ptn}_k^L} \mathsf{prob}^{\mathbf{p}}(\mathbf{c}) \cdot W(\mathbf{c}) & \\ \text{s.t.} & W(\mathbf{c}) \geq 0 & \forall \mathbf{c} \in \mathsf{Ptn}_k^L \\ & \displaystyle\sum_{\mathbf{c} \in \mathsf{PosPtn}_k^L} W(\mathbf{c}) = 1 & \\ & t_i^-(W^+(i) + W^-(i)) \leq W^+(i) - W^-(i) & \forall i \in \{-\ell, \ldots, +\ell\} \\ & W^+(i) - W^-(i) \leq t_i^+(W^+(i) + W^-(i)) & \forall i \in \{-\ell, \ldots, +\ell\} \end{cases}$$

where we define the linear functions

$$W^+(i) = \sum_{\mathbf{c} \in \mathsf{Ptn}_k^L} c_i^+ W(\mathbf{c}) \text{ and } W^-(i) = \sum_{\mathbf{c} \in \mathsf{Ptn}_k^L} c_i^- W(\mathbf{c}).$$

Proof. Let α_{alg} denote the approximation ratio of $\mathtt{Obl}_k^{\mathbf{t},\mathbf{p}}$, and α_{LP} the minimum value of the linear program. This proof generalizes [17, Proof of Theorem 1.2].

$\alpha_{\text{LP}} \leq \alpha_{\text{alg}}$. We show that for every instance Ψ of Max-kAND, there is a feasible LP solution $\{W(\mathbf{c})\}_{\mathbf{c} \in \mathsf{Ptn}_k^L}$ of objective value $\mathtt{Obl}_k^{\mathbf{t},\mathbf{p}}(\Psi)/\mathsf{val}_\Psi$.

Towards this claim, by Proposition 10, we can assume WLOG that the all-(+1)'s assignment is optimal for Ψ. Also, we assume WLOG by rescaling that Ψ has total weight $\frac{1}{\mathsf{val}_\Psi}$, i.e., $\sum_{j=1}^m w_j = \frac{1}{\mathsf{val}_\Psi}$. Now, let $C_1, \ldots, C_m$ denote the constraints of Ψ, and let $W(\mathbf{c}) := \sum_{j=1}^m \mathbb{1}[\mathsf{ptn}^{\mathbf{t}}(C_j) = \mathbf{c}] w_j$. We claim that $\{W(\mathbf{c})\}_{\mathbf{c} \in \mathsf{Ptn}_k^L}$ is feasible and has objective value $\sum_{\mathbf{c} \in \mathsf{Ptn}_k^L} \mathsf{prob}^{\mathbf{p}}(\mathbf{c}) W(\mathbf{c}) = \mathtt{Obl}_k^{\mathbf{t},\mathbf{p}}(\Psi)/\mathsf{val}_\Psi$.

First, we check feasibility. Clearly all $W(\mathbf{c})$'s are nonnegative. Next, we have

$$\begin{aligned}
\mathsf{val}_\Psi &= \mathsf{val}_\Psi(+\mathbf{1}) && \text{(all-(+1)'s is optimal)}\\
&= \frac{\sum_{j=1}^m w_j \mathbb{1}[|V_j^-| = 0]}{\sum_{j=1}^m w_j} && \text{(def. of } \mathsf{val}_\Psi(+\mathbf{1})\text{)}\\
&= \frac{\sum_{\mathbf{c}\in\mathsf{PosPtn}_k^L} W(\mathbf{c})}{1/\mathsf{val}_\Psi} && \text{(def. of } \mathsf{PosPtn}_k^L \text{ and total weight assumption)}
\end{aligned}$$

which rearranges to $\sum_{\mathbf{c}\in\mathsf{PosPtn}_k^L} W(\mathbf{c}) = 1$.

Now, recall the definitions of $\mathsf{bias}_\Psi, w_\Psi^+, w_\Psi^-$ from Section 2. Fix a bias class $i \in \{-\ell, \dots, +\ell\}$. For any variable v with bias class i, we have $\mathsf{bias}_\Psi(v) \in \mathsf{Int}_i^{\mathbf{t}}$, so $t_i^- \le \mathsf{bias}_\Psi(v) \le t_i^+$, so multiplying through by $w_\Psi^+(v) + w_\Psi^-(v)$, we get

$$t_i^-(w_\Psi^+(v) + w_\Psi^-(v)) \le w_\Psi^+(v) - w_\Psi^-(v) \le t_i^+(w_\Psi^+(v) + w_\Psi^-(v)).$$

Letting $\mathcal{V}_i$ denote the set of all variables in Ψ with bias class i, we can sum over these equations to get

$$t_i^- \sum_{v\in\mathcal{V}_i}(w_\Psi^+(v) + w_\Psi^-(v)) \le \sum_{v\in\mathcal{V}_i}(w_\Psi^+(v) - w_\Psi^-(v)) \le t_i^+ \sum_{v\in\mathcal{V}_i}(w_\Psi^+(v) + w_\Psi^-(v)).$$

We claim that

$$W^+(i) = \sum_{v\in\mathcal{V}} w_\Psi^+(v),$$

and similarly $W^-(i) = \sum_{v\in\mathcal{V}} w_\Psi^-(v)$. These equalities imply that $W(\cdot)$ satisfies the feasibility constraints, and it remains to prove them. Now recall $w_\Psi^+(v) = \sum_{j=1}^m \mathbb{1}[v \in V_j^+]w_j$; therefore,

$$\sum_{v\in\mathcal{V}} w_\Psi^+(v) = \sum_{j=1}^m \sum_{v\in V_j^+} \mathbb{1}[\mathsf{bias}_\Psi(v) \in \mathsf{Int}_i^{\mathbf{t}}]w_j,$$

and j-th term in this sum is precisely c_i^+ where $\mathbf{c} = (\mathbf{c}^+, \mathbf{c}^-) = \mathsf{ptn}^{\mathbf{t}}(C_j)$. The proof for $W^-(i)$ is similar.

Finally, by Claim 9 and our assumption $\sum_{j=1}^m w_j = 1/\mathsf{val}_\Psi$, we have

$$\mathtt{Obl}_k^{\mathbf{t},\mathbf{p}}(\Psi) = \frac{\sum_{j=1}^m w_j \mathsf{prob}^{\mathbf{p}}(\mathsf{ptn}^{\mathbf{t}}(C_j))}{\sum_{j=1}^m w_j} = \frac{\sum_{\mathbf{c}\in\mathsf{Ptn}_k^L} W(\mathbf{c})\cdot\mathsf{prob}^{\mathbf{p}}(\mathbf{c})}{1/\mathsf{val}_\Psi},$$

which rearranges to $\mathtt{Obl}_k^{\mathbf{t},\mathbf{p}}(\Psi)/\mathsf{val}_\Psi = \sum_{\mathbf{c}\in\mathsf{Ptn}_k^L} W(\mathbf{c})\cdot\mathsf{prob}^{\mathbf{p}}(\mathbf{c})$, as desired.

$\alpha_{\mathrm{LP}} \ge \alpha_{\mathrm{alg}}$. This argument is essentially converse to the former argument, but there are two technical issues: (i) the linear program does not encode strict inequality constraints, while an oblivious algorithm needs to (in the sense that e.g., if $t_0 = 0$, then the algorithm rounds vertices with bias 0 and bias $+\epsilon$ differently), and (ii) since an Max-kAND constraint cannot use a variable twice, we might need many variables with the same bias in the instance we create. We omit the proof here; see the full version. ◀

▶ **Lemma 12** (Dual characterization). *For every bias partition $\mathbf{t} = (t_0, \ldots, t_\ell)$ and rounding vector $\mathbf{p} = (p_1, \ldots, p_\ell)$, the approximation ratio $\alpha(\mathtt{Obl}_k^{\mathbf{t},\mathbf{p}})$ achieved by $\mathtt{Obl}_k^{\mathbf{t},\mathbf{p}}$ equals the value of the following linear program:*

$$\begin{cases} \max & z & & \\ \text{s.t.} & \mathbb{1}[\mathbf{c} \in \mathsf{PosPtn}_k^L] \cdot z & & \\ & + \sum_{i=-\ell}^{+\ell} \left(((1-t_i^-)c_i^+ - (t_i^-+1)c_i^-)y_i^- + ((t_i^+-1)c_i^+ + (1+t_i^+)c_i^-)y_i^+\right) \le \mathsf{prob}^{\mathbf{p}}(\mathbf{c}) & \forall \mathbf{c} \in \mathsf{Ptn}_k^L \\ & y_i^- \ge 0 & \forall i \in \{-\ell, \ldots, +\ell\} \\ & y_i^+ \ge 0 & \forall i \in \{-\ell, \ldots, +\ell\} \end{cases}$$

Proof. To place the primal LP (from Lemma 11) in a more standard form, we rewrite the primal inequality $t_i^-(W^+(i)+W^-(i)) \le W^+(i)-W^-(i)$ as $(t_i^--1)W^+(i)+(t_i^-+1)W^-(i) \le 0$; expanding the definitions of $W^+(i)$ and $W^-(i)$, this is equivalent to $(t_i^--1)\sum_{\mathbf{c}\in\mathsf{Ptn}_k^L} c_i^+W(\mathbf{c})+(t_i^-+1)\sum_{\mathbf{c}\in\mathsf{Ptn}_k^L} c_i^-W(\mathbf{c}) \le 0$. Similarly, the inequality $W^+(i)-W^-(i) \le t_i^+(W^+(i)+W^-(i))$ becomes $(1-t_i^+)\sum_{\mathbf{c}\in\mathsf{Ptn}_k^L} c_i^+W(\mathbf{c}) - (1+t_i^+)\sum_{\mathbf{c}\in\mathsf{Ptn}_k^L} c_i^-W(\mathbf{c}) \le 0$. Therefore, the primal LP is equivalent to the following standard-form LP:

$$\begin{cases} \text{minimize} & \sum_{\mathbf{c}\in\mathsf{Ptn}_k^L} \mathsf{prob}^{\mathbf{p}}(\mathbf{c}) \cdot W(\mathbf{c}) & & \\ \text{s.t.} & W(\mathbf{c}) \ge 0 & \forall \mathbf{c} \in \mathsf{Ptn}_k^L \\ & \sum_{\mathbf{c}\in\mathsf{PosPtn}_k^L} W(\mathbf{c}) = 1 & & \\ & (1-t_i^+)\sum_{\mathbf{c}\in\mathsf{Ptn}_k^L} c_i^+W(\mathbf{c}) - (1+t_i^+)\sum_{\mathbf{c}\in\mathsf{Ptn}_k^L} c_i^-W(\mathbf{c}) \le 0 & \forall i \in \{-\ell, \ldots, +\ell\} \\ & (t_i^- - 1)\sum_{\mathbf{c}\in\mathsf{Ptn}_k^L} c_i^+W(\mathbf{c}) + (t_i^-+1)\sum_{\mathbf{c}\in\mathsf{Ptn}_k^L} c_i^-W(\mathbf{c}) \le 0 & \forall i \in \{-\ell, \ldots, +\ell\} \end{cases}$$

By LP duality, the above LP has the same value as its dual LP, which is the LP in the hypothesis.[9] ◀

4 Proving Theorem 2 by analyzing "dual slack"

In this section, we outline the first step towards proving Theorem 2 by constructing dual solutions which witness lower bounds on the approximation ratio of oblivious algorithms. This first step is the following lemma, which gives a clean sufficient condition for a lower bound on the approximation ratio by constructing a certain sparse dual solution and applying the dual program (Lemma 12):

[9] See e.g. [32, p. 85]). One has to be careful with the signs, since our primal LP is a *minimization* LP. Instead, we can consider the LP which maximizes $-\sum_{\mathbf{c}\in\mathsf{Ptn}_k^L} \mathsf{prob}^{\mathbf{p}}(\mathbf{c}) \cdot W(\mathbf{c})$ (whose output is the negation of our desired output).
Applying duality to this LP gives one which minimizes z' such that $\mathbb{1}[\mathbf{c} \in \mathsf{PosPtn}_k^L] \cdot z' + \sum_{i=-\ell}^{\ell} \left(((1-t_i^-)c_i^+ - (t_i^-+1)c_i^-)y_i^- + ((t_i^+-1)c_i^+ + (1+t_i^+)c_i^-)y_i^+\right) \ge -\mathsf{prob}^{\mathbf{p}}(\mathbf{c})$. Transforming to a maximization problem equivalent to our original LP (since we had a negation!), we maximize $-z'$ such that $\mathbb{1}[\mathbf{c} \in \mathsf{PosPtn}_k^L] \cdot z' + \sum_{i=-\ell}^{\ell} \left(((1-t_i^-)c_i^+ - (t_i^-+1)c_i^-)y_i^- + ((t_i^+-1)c_i^+ + (1+t_i^+)c_i^-)y_i^+\right) \ge -\mathsf{prob}^{\mathbf{p}}(\mathbf{c})$. Finally, we negate both sides of this inequality, and use the bijective transformation $z = -z'$.

▶ **Lemma 13** (Sufficient conditions for good approximations). *For every $k \geq 2 \in \mathbb{N}$, $0 \leq \gamma, \delta \leq 1$, let $\mathbf{t} = (\delta, 1)$ and $\mathbf{p} = (\frac{1}{2}(1+\gamma))$. The algorithm $\mathtt{Obl}_k^{\mathbf{t},\mathbf{p}}$ has approximation ratio $\alpha(\mathtt{Obl}_k^{\mathbf{t},\mathbf{p}}) \geq 2^{-(k-1)}\beta$ if the following statement holds: There exist $X, Y \geq 0$ such that:*

$$\begin{cases} (1+\delta)\left(1-\dfrac{i+j}{k}\right)Y + (1-\delta)\dfrac{j}{k}X \leq \beta^{-1}(1-\gamma)^i(1+\gamma)^j & \forall i,j \in \mathbb{N}, i+j \leq k \\ 2-(1-\delta)\left(1-\dfrac{i+j}{k}\right)Y - (1+\delta)\dfrac{i}{k}X \leq \beta^{-1}(1-\gamma)^i(1+\gamma)^j & \forall i,j \in \mathbb{N}, i+j \leq k \end{cases}$$

Proof. Consider applying the dual characterization of the approximation ratio (Lemma 12) with the solution $z = 2\beta/2^k$, $y^+_{-1} = X\beta/(k2^k)$, $y^+_0 = Y\beta/(k2^k)$, and $y^+_{+1} = y^-_{-1} = y^-_0 = y^-_{+1} = 0$; it is sufficient to show that this solution is feasible. Note that $t^+_{-1} = -\delta$ and $t^+_0 = \delta$. Thus, the feasibility constraints in Lemma 12 become

$$\frac{\beta}{2^k}\left(\mathbb{1}[\mathbf{c} \in \mathsf{PosPtn}_k^L]\cdot 2 + ((-\delta-1)c^+_{-1} + (1-\delta)c^-_{-1})\frac{X}{k} + ((\delta-1)c^+_0 + (1+\delta)c^-_0)\frac{Y}{k}\right) \leq \mathsf{prob}^{\mathbf{p}}(\mathbf{c}) \quad \forall \mathbf{c} \in \mathsf{Ptn}_k^L. \quad (7)$$

By Equation (6), and since $c^+_{-1} + c^+_0 + c^+_{+1} + c^-_{-1} + c^-_0 + c^-_{+1} = k$, we have

$$\mathsf{prob}^{\mathbf{p}}(\mathbf{c}) = \left(\frac{1}{2}-\frac{\gamma}{2}\right)^{c^+_{-1}+c^-_{+1}}\left(\frac{1}{2}\right)^{c^+_0+c^-_0}\left(\frac{1}{2}+\frac{\gamma}{2}\right)^{c^+_{+1}+c^-_{-1}} = 2^{-k}(1-\gamma)^{c^+_{-1}+c^-_{+1}}(1+\gamma)^{c^+_{+1}+c^-_{-1}}.$$

Thus, dividing through by $\beta/2^k$, Equation (7) becomes

$$\mathbb{1}[\mathbf{c} \in \mathsf{PosPtn}_k^L]\cdot 2 + ((1-\delta)c^-_{-1} - (1+\delta)c^+_{-1})\frac{X}{k} + ((1+\delta)c^-_0 - (1-\delta)c^+_0)\frac{Y}{k} \leq \beta^{-1}(1-\gamma)^{c^+_{-1}+c^-_{+1}}(1+\gamma)^{c^+_{+1}+c^-_{-1}} \quad \forall \mathbf{c} \in \mathsf{Ptn}_k^L. \quad (8)$$

Finally, we claim that Equation (8) is implied by the hypothesis. Indeed, we consider two cases. First, if $\mathbf{c} \in \mathsf{PosPtn}_k^L$, then $c^-_{-1} = c^-_0 = c^-_{+1} = 0$ and $c^+_0 = k - c^+_{-1} - c^+_{+1}$, so Equation (8) becomes

$$2 - (1+\delta)\frac{c^+_{-1}}{k}X - (1-\delta)\frac{k-c^+_{-1}-c^+_{+1}}{k}Y \leq \beta^{-1}(1-\gamma)^{c^+_{-1}}(1+\gamma)^{c^+_{+1}}.$$

This is precisely the second hypothesized inequality, for $c^+_{-1} = i, c^+_{+1} = j$. On the other hand, if $\mathbf{c} \notin \mathsf{PosPtn}_k^L$, then we observe that replacing $(c^+_{-1}, c^+_0, c^+_{+1}, c^-_{-1}, c^-_0, c^-_{+1}) \mapsto (0,0,0,c^-_{-1} + c^+_{-1}, c^-_0 + c^+_0, c^-_{+1} + c^+_{+1})$ fixes the RHS of Equation (8), while only increasing the LHS; thus, it suffices to prove Equation (8) only in this extreme case. Hence, we can assume $c^-_0 = k - c^-_{-1} - c^-_{+1}$, so Equation (8) becomes

$$(1-\delta)\frac{c^-_{-1}}{k}X + (1+\delta)\frac{k-c^-_{-1}-c^-_{+1}}{k}Y \leq \beta^{-1}(1-\gamma)^{c^-_{+1}}(1+\gamma)^{c^-_{-1}},$$

which is precisely the first assumed condition for $c^-_{-1} = j, c^-_{+1} = i$. ◀

▶ Remark 14. We chose the specific family of dual solutions used in the proof of Lemma 13 by inspecting an LP solver's output for $k = 2$ and $k = 3$. Our investigation also suggests that this solution is unique in a certain sense: In the simplest case of $k = 2$ and $\delta = \gamma = 0$, it appears that *every* optimal feasible solution requires $y^+_{-1} = 2/9$, and further, the only solution with only two nonzero y entries sets $y^+_0 = 1/9$.

References

1 Paola Alimonti. New local search approximation techniques for maximum generalized satisfiability problems. *Information Processing Letters*, 57(3):151–158, February 1996. Conference version in CIAC 1994. doi:10.1016/0020-0190(95)00196-4.

2 Paola Alimonti. Non-oblivious local search for MAX 2-CCSP with application to MAX DICUT. In Rolf H. Möhring, editor, *Graph-Theoretic Concepts in Computer Science*, Lecture Notes in Computer Science, pages 2–14. Springer, 1997. doi:10.1007/BFb0024483.

3 Sepehr Assadi, Gillat Kol, Raghuvansh R. Saxena, and Huacheng Yu. Multi-Pass Graph Streaming Lower Bounds for Cycle Counting, MAX-CUT, Matching Size, and Other Problems. In *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS 2020, Virtual, November 16-19, 2020)*, pages 354–364, virtual, November 2020. doi:10.1109/FOCS46700.2020.00041.

4 Sepehr Assadi and Vishvajeet N. Graph streaming lower bounds for parameter estimation and property testing via a streaming XOR lemma. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing (STOC 2021, Virtual, June 21-25, 2021)*, pages 612–625, virtual, June 2021. Association for Computing Machinery. doi:10.1145/3406325.3451110.

5 Amotz Bar-Noy and Michael Lampis. Online maximum directed cut. *Journal of Combinatorial Optimization*, 24(1):52–64, July 2012. Conference version in ISAAC 2009. doi:10.1007/s10878-010-9318-6.

6 Nico Bertram, Jonas Ellert, and Johannes Fischer. A Parallel Framework for Approximate Max-Dicut in Partitionable Graphs. In Christian Schulz and Bora Uçar, editors, *20th International Symposium on Experimental Algorithms (SEA 2022, Heidelberg, Germany, July 25-27, 2022)*, volume 233 of *LIPIcs*, pages 10:1–10:15, Heidelberg, Germany, 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.SEA.2022.10.

7 Joanna Boyland, Michael Hwang, Tarun Prasad, Noah Singer, and Santhoshini Velusamy. On sketching approximations for symmetric Boolean CSPs. In Amit Chakrabarti and Chaitanya Swamy, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX 2022, Virtual, September 19-21, 2022)*, volume 245 of *LIPIcs*, pages 38:1–38:23, virtual, July 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.APPROX/RANDOM.2022.38.

8 Moses Charikar, Konstantin Makarychev, and Yury Makarychev. Near-optimal algorithms for maximum constraint satisfaction problems. *ACM Transactions on Algorithms*, 5(3):1–14, July 2009. Conference version in SODA 2007. doi:10.1145/1541885.1541893.

9 Lijie Chen, Gillat Kol, Dmitry Paramonov, Raghuvansh Saxena, Zhao Song, and Huacheng Yu. Towards Multi-Pass Streaming Lower Bounds for Optimal Approximation of Max-Cut. In *Proceedings of the 2023 Annual ACM-SIAM Symposium on Discrete Algorithms*, 2023.

10 Chi-Ning Chou, Alexander Golovnev, Amirbehshad Shahrasbi, Madhu Sudan, and Santhoshini Velusamy. Sketching Approximability of (Weak) Monarchy Predicates. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX 2022, Virtual, September 19-21, 2022)*, volume 245 of *LIPIcs*, pages 35:1–35:17, virtual, 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.APPROX/RANDOM.2022.35.

11 Chi-Ning Chou, Alexander Golovnev, Madhu Sudan, Ameya Velingker, and Santhoshini Velusamy. Linear Space Streaming Lower Bounds for Approximating CSPs. In *Proceedings of the 54th Annual ACM Symposium on Theory of Computing (STOC 2022, Rome, Italy, June 20-24, 2022)*, Rome, Italy, 2022. doi:10.1145/3519935.3519983.

12 Chi-Ning Chou, Alexander Golovnev, Madhu Sudan, and Santhoshini Velusamy. Approximability of all Boolean CSPs with linear sketches. *arXiv*, February 2021. arXiv:2102.12351v7.

13 Chi-Ning Chou, Alexander Golovnev, Madhu Sudan, and Santhoshini Velusamy. Approximability of all finite CSPs with linear sketches. In *Proceedings of the 62nd Annual IEEE Symposium on Foundations of Computer Science (FOCS 2021, Denver, CO, USA, February 7-10, 2022)*, Denver, CO, USA, 2021. IEEE Computer Society. doi:10.1109/FOCS52979.2021.00117.

14 Chi-Ning Chou, Alexander Golovnev, and Santhoshini Velusamy. Optimal Streaming Approximations for all Boolean Max-2CSPs and Max-kSAT. In *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS 2020, Virtual, November 16-19, 2020)*, pages 330–341, virtual, November 2020. IEEE Computer Society. doi:10.1109/FOCS46700.2020.00039.

15 Lars Engebretsen and Jonas Holmerin. More efficient queries in PCPs for NP and improved approximation hardness of maximum CSP. *Random Structures and Algorithms*, 33(4):497–514, December 2008. Conference version in STACS 2005. doi:10.1002/rsa.20226.

16 Uriel Feige and Michel X. Goemans. Approximating the value of two prover proof systems, with applications to MAX 2SAT and MAX DICUT. In *Proceedings of the 3rd Israel Symposium on the Theory of Computing and Systems (ISTCS 2003, January 4-6, 1995)*, pages 182–189. IEEE Computer Society, 1995. doi:10.1109/ISTCS.1995.377033.

17 Uriel Feige and Shlomo Jozeph. Oblivious Algorithms for the Maximum Directed Cut Problem. *Algorithmica*, 71(2):409–428, February 2015. doi:10.1007/s00453-013-9806-z.

18 Michel X. Goemans and David P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM*, 42(6):1115–1145, November 1995. Conference version in STOC 1994. doi:10.1145/227683.227684.

19 Venkatesan Guruswami and Runzhou Tao. Streaming Hardness of Unique Games. In Dimitris Achlioptas and László A. Végh, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX 2019, Cambridge, MA, USA, September 20-22, 2019)*, volume 145 of *LIPIcs*, pages 5:1–5:12, Cambridge, MA, USA, September 2019. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.APPROX-RANDOM.2019.5.

20 Venkatesan Guruswami, Ameya Velingker, and Santhoshini Velusamy. Streaming Complexity of Approximating Max 2CSP and Max Acyclic Subgraph. In Klaus Jansen, José D. P. Rolim, David Williamson, and Santosh S. Vempala, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX 2017, Berkeley, CA, USA, August 16-18, 2017)*, volume 81 of *LIPIcs*, pages 8:1–8:19, Berkeley, CA, USA, August 2017. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.APPROX-RANDOM.2017.8.

21 Eran Halperin and Uri Zwick. Combinatorial approximation algorithms for the maximum directed cut problem. In *Proceedings of the 12th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2001, Washington, DC, USA, January 7-9, 2001)*, pages 1–7, Washington, DC, USA, 2001.

22 Gustav Hast. Approximating Max kCSP Using Random Restrictions. In Klaus Jansen, Sanjeev Khanna, José D. P. Rolim, and Dana Ron, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX 2004, Cambridge, MA, USA, August 22-24, 2004)*, volume 3122 of *LNCS*, pages 151–162, Cambridge, MA, USA, 2004. Springer. doi:10.1007/978-3-540-27821-4_14.

23 Gustav Hast. Approximating Max kCSP – Outperforming a Random Assignment with Almost a Linear Factor. In Luís Caires, Giuseppe F. Italiano, Luís Monteiro, Catuscia Palamidessi, and Moti Yung, editors, *Automata, Languages and Programming (ICALP 2005, July 11-15, 2005)*, volume 3580 of *LNCS*, pages 956–968. Springer, 2005. doi:10.1007/11523468_77.

24 Johan Håstad. Some optimal inapproximability results. *Journal of the ACM*, 48(4):798–859, 2001. doi:10.1145/502090.502098.

25 Kamal Jain, Mohammad Mahdian, Evangelos Markakis, Amin Saberi, and Vijay V. Vazirani. Greedy facility location algorithms analyzed using dual fitting with factor-revealing LP. *Journal of the ACM*, 50(6):795–824, November 2003. doi:10.1145/950620.950621.

26 Michael Kapralov, Sanjeev Khanna, and Madhu Sudan. Streaming lower bounds for approximating MAX-CUT. In *Proceedings of the 26th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2015, San Diego, California, USA, January 4-6, 2015)*, pages 1263–1282, San Diego, California, USA, January 2015. Society for Industrial and Applied Mathematics. doi:10.1137/1.9781611973730.84.

27 Michael Kapralov, Sanjeev Khanna, Madhu Sudan, and Ameya Velingker. $(1 + \omega(1))$-approximation to MAX-CUT requires linear space. In *Proceedings of the 28th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2017, Barcelona, Spain, January 16-19, 2017)*, pages 1703–1722, Barcelona, Spain, January 2017. Society for Industrial and Applied Mathematics. doi:10.5555/3039686.3039798.

28 Michael Kapralov and Dmitry Krachun. An optimal space lower bound for approximating MAX-CUT. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing (STOC 2019, Phoenix, AZ, USA, June 23-26, 2019)*, pages 277–288, Phoenix, AZ, USA, June 2019. Association for Computing Machinery. doi:10.1145/3313276.3316364.

29 Dmitry Kogan and Robert Krauthgamer. Sketching cuts in graphs and hypergraphs. In *Proceedings of the 6th Annual Conference on Innovations in Theoretical Computer Science (ITCS 2015, Rehovot, Israel, January 11-13, 2015)*, pages 367–376, Rehovot, Israel, 2015. Association for Computing Machinery. doi:10.1145/2688073.2688093.

30 Michael Lewin, Dror Livnat, and Uri Zwick. Improved Rounding Techniques for the MAX 2-SAT and MAX DI-CUT Problems. In William J. Cook and Andreas S. Schulz, editors, *Integer Programming and Combinatorial Optimization*, pages 67–82, 2002. doi:10.1007/3-540-47867-1_6.

31 Mohammad Mahdian and Qiqi Yan. Online bipartite matching with random arrivals: An approach based on strongly factor-revealing LPs. In *Proceedings of the 43rd Annual ACM Symposium on Theory of Computing (STOC 2011, San Jose, CA, USA, June 6-8, 2011)*, pages 597–606, San Jose, CA, USA, June 2011. Association for Computing Machinery. doi:10.1145/1993636.1993716.

32 Jiří Matoušek and Bernd Gärtner. *Understanding and Using Linear Programming*. Universitext. Springer, Berlin; New York, 2007.

33 Shiro Matuura and Tomomi Matsui. 0.863-Approximation Algorithm for MAX DICUT. In Michel Goemans, Klaus Jansen, José D. P. Rolim, and Luca Trevisan, editors, *Approximation, Randomization, and Combinatorial Optimization: Algorithms and Techniques (APPROX 2001, Berkeley, CA, USA, August 18-20, 2001)*, volume 2129 of *LNCS*, pages 138–146, Berlin, Heidelberg, 2001. Springer. doi:10.1007/3-540-44666-4_17.

34 Alex Samorodnitsky and Luca Trevisan. A PCP characterization of NP with optimal amortized query complexity. In *Proceedings of the 32nd Annual ACM Symposium on Theory of Computing (STOC 2000, Portland, OR, USA, May 21-23, 2000)*, pages 191–199, Portland, OR, USA, 2000. Association for Computing Machinery. doi:10.1145/335305.335329.

35 Alex Samorodnitsky and Luca Trevisan. Gowers Uniformity, Influence of Variables, and PCPs. *SIAM Journal on Computing*, 39(1):323–360, January 2009. Conference version in STOC 2006. doi:10.1137/070681612.

36 Raghuvansh R. Saxena, Noah Singer, Madhu Sudan, and Santhoshini Velusamy. Improved streaming algorithms for Maximum Directed Cut via smoothed snapshots. In *63rd Annual Symposium on Foundations of Computer Science*, Santa Cruz, CA, USA, 2023. IEEE Computing Society. To appear.

37 Raghuvansh R. Saxena, Noah Singer, Madhu Sudan, and Santhoshini Velusamy. Streaming complexity of CSPs with randomly ordered constraints. In *Proceedings of the 2023 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2023, Florence, Italy, January 22-25, 2023)*, Florence, Italy, January 2023.

38 Noah Singer. *On Streaming Approximation Algorithms for Constraint Satisfaction Problems*. Undergraduate thesis, Harvard University, Cambridge, MA, March 2022.

39 Noah Singer, Madhu Sudan, and Santhoshini Velusamy. Streaming approximation resistance of every ordering CSP. In Mary Wootters and Laura Sanità, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX 2021, Virtual, August 16-18, 2021)*, volume 207 of *LIPIcs*, pages 17:1–17:19, virtual, July 2021. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.APPROX/RANDOM.2021.17.

40 David Steurer. Fast SDP Algorithms for Constraint Satisfaction Problems. In *Proceedings of the 21st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2010, Austin, TX, USA, January 17-19, 2010)*, pages 684–697, Austin, TX, USA, January 2010. Society for Industrial and Applied Mathematics. doi:10.1137/1.9781611973075.56.

41 Madhu Sudan. Streaming and Sketching Complexity of CSPs: A survey (Invited Talk). In Mikołaj Bojańczyk, Emanuela Merelli, and David P. Woodruff, editors, *49th International Colloquium on Automata, Languages, and Programming*, volume 229 of *LIPIcs*, pages 5:1–5:20. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.ICALP.2022.5.

42 Madhu Sudan and Luca Trevisan. Probabilistically checkable proofs with low amortized query complexity. In *Proceedings of the 39th Annual Symposium on Foundations of Computer Science (SFCS 1998, Palo Alto, CA, USA, November 8-11, 1998)*, pages 18–27, Palo Alto, CA, USA, 1998. IEEE Computer Society. doi:10.1109/SFCS.1998.743425.

43 Luca Trevisan. Parallel Approximation Algorithms by Positive Linear Programming. *Algorithmica*, 21(1):72–88, May 1998. doi:10.1007/PL00009209.

44 Luca Trevisan. Recycling queries in PCPs and in linearity tests. In *Proceedings of the 30th Annual ACM Symposium on Theory of Computing (STOC 1998, Dallas, Texas, USA, May 24-26, 1998)*, pages 299–308, Dallas, Texas, USA, 1998. Association for Computing Machinery. doi:10.1145/276698.276769.

45 Luca Trevisan, Gregory B. Sorkin, Madhu Sudan, and David P. Williamson. Gadgets, Approximation, and Linear Programming. *SIAM Journal on Computing*, 29(6):2074–2097, January 2000. Conference version in FOCS 1996. doi:10.1137/S0097539797328847.

46 Zhenning Zhang, Donglei Du, Chenchen Wu, Dachuan Xu, and Dongmei Zhang. A spectral partitioning algorithm for maximum directed cut problem. *Journal of Combinatorial Optimization*, 42(3):373–395, October 2021. Conference version in COCOA 2017. doi:10.1007/s10878-018-0369-4.

47 Uri Zwick. Approximation algorithms for constraint satisfaction problems involving at most three variables per constraint. In *Proceedings of the 9th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 1998, San Francisco, CA, USA, January 25-27, 1998)*, pages 201–210, San Francisco, CA, USA, 1998. Association for Computing Machinery. doi:10.5555/314613.314701.