Electronic Colloquium on Computational Complexity, Revision 1 of Report No. 65 (2022)

Streaming and Sketching Complexity of CSPs: A survey*

Madhu Sudan†

May 3, 2022

Abstract

In this survey we describe progress over the last decade or so in understanding the complexity of solving constraint satisfaction problems (CSPs) approximately in the streaming and sketching models of computation. After surveying some of the results we give some sketches of the proofs and in particular try to explain why there is a tight dichotomy result for sketching algorithms working in subpolynomial space regime.

Contents

*This paper accompanies an invited talk by the author at ICALP 2022. A version of this paper will appear in the proceedings of the same.

†School of Engineering and Applied Sciences, Harvard University, Cambridge, Massachusetts, USA. Supported in part by a Simons Investigator Award and NSF Awards CCF 1715187 and CCF 2152413. Email: madhu@cs.harvard.edu.

ISSN 1433-8092

1 Introduction

In this article we survey the current state of knowledge on the *approximability* of *constraint satisfaction problems (CSPs)* using small space *streaming* and *sketching* algorithms. We start by reviewing the definitions below before turning to the survey of results.

2 CSPs: What and Why

CSPs form an infinite class of optimization problems where the goal is to assign n variables values from a finite set while maximizing the number of constraints that can be satisfied, where each constraint looks locally at the assignment of a few variables to determine if it is satisfied on not. Different problems in the class differ based on which type of constraints are allowed. Different instances of the problem arise by applying the constraints to different subsets (or subsequences) of variables. Algorithms aim to compute, or approximate, the maximum, over all assignments, of the fraction of constraints that can be simultaneously satisfied. Resource restrictions on the algorithm (time, space, number of passes in the streaming setting) as well as the type of constraints allow to determine the level of approximability that is feasible. In this survey we describe our knowledge of the approximability of CSPs when restricted to streaming and sketching algorithms with limited space.

We start by describing CSPs more formally. For positive integer n we use $[n]$ to denote the set $\{1,\ldots,n\}$ and $\mathbb{Z}_n$ to denote the set $\{0,\ldots,n-1\}$. A CSP problem is described by positive integers k, q and a family of functions $\mathcal{F} \subseteq \{f : \mathbb{Z}_q^k \to \{0,1\}\}$. Since k, q are implicit in $\mathcal{F}$, we refer to this problem as $\mathsf{Max\text{-}CSP}(\mathcal{F})$. Given variables $X_1,\ldots,X_n$, an assignment to the variables is a sequence $\mathbf{a} = (a_1,\ldots,a_n) \in \mathbb{Z}_q^n$. A constraint C on these variables is given by a pair $(f,(j_1,\ldots,j_k))$ where the first element of the pair $f \in \mathcal{F}$ is the choice of the type of constraint and the second element is a sequence of k *distinct* indices with $j_i,\ldots,j_k \in [n]$. An assignment $\mathbf{a}$ satisfies $C = (f,(j_1,\ldots,j_k))$ if and only if $f(a_{j_1},\ldots,a_{j_k}) = 1$. We use $C(\mathbf{a})$ to denote the quantity $f(X_{j_1},\ldots,X_{j_k})$. An instance of $\mathsf{Max\text{-}CSP}(\mathcal{F})$ on n variables and m constraints is given by $\Psi = (C_1,\ldots,C_m)$ with C_i being a constraint on $X_1,\ldots,X_n$ for every $i \in [m]$. Given a assignment $\mathbf{a}$ to the variables, the value on the instance Ψ at $\mathbf{a}$, denoted $\mathsf{val}_\Psi(\mathbf{a})$, is the quantity $\frac{1}{m}\sum_{i\in[m]} C_i(\mathbf{a})$, i.e., the value is the fraction of constraints of Ψ that are satisfied by $\mathbf{a}$. The value of the instance val_Ψ is defined to be the maximum value over all assignments, i.e., $\mathsf{val}_\Psi = \max_{\mathbf{a}\in[q]^n}\{\mathsf{val}_\Psi(\mathbf{a})\}$. The goal of CSP optimization algorithms is to compute, or approximate, val_Ψ given Ψ.[1]

Example: We illustrate the definition with the example of the $\mathsf{Max\text{-}CUT}$ problem, where given an undirected graph on vertex set $[n]$, the goal is to find a "cut" $S \sqcup \overline{S} = [n]$ that maximizes the number of edges crossing the cut (i.e, with one endpoint each in S and $\overline{S}$). This problem is captured by $q = k = 2$ and the family $\mathcal{F} = \{\oplus\}$ where $\oplus(u,v) = u + v \pmod 2$. For instance if the input graph for $\mathsf{Max\text{-}CUT}$ is the 5-cycle, then the input instance Ψ_5 to $\mathsf{Max\text{-}CSP}(\mathcal{F})$ will have 5 variables (corresponding to the vertices) and 5 constraints $C_1,\ldots,C_5$ (corresponding to the edges) with $C_i = (\oplus,(i,i+1))$ for $i \in [4]$ and $C_5 = (\oplus,(5,1))$. An assignment $\mathbf{a} \in \mathbb{Z}_2^5$ can be equated with the cut $S_\mathbf{a} = \{i | a_i = 1\}$ and it can be verified that $\mathsf{val}_{\Psi_5}(\mathbf{a})$ equals the fraction of edges cut by $S_\mathbf{a}$.

Note that in the example above, we could get positive integer weighted graphs also since there is no requirement that the constraints themselves be distinct. For simplicity we will assume the length on the stream is polynomial in n so that the weights are also polynomial, though as we point

[1] Throughout this paper we assume that $\mathcal{F}$ does not include the all 0 function. Such a function corresponds to placing constraints that are never satisfiable. Inclusion of such constraints in the family does not change the complexity of any of the tasks we consider since these constraints are easy to ignore.

out later this restriction does not alter the complexity of the approximation problems much. But the graphs will not have any self-loops due to the "restriction" that the variables in a constraint have to be distinct. If one wished to consider the Max-CUT problem where self-loops are also allowed (though in the case of Max-CUT this would make no sense - since a self-loop can never be cut), then one could consider instances Max-CSP($\mathcal{F}'$) where $\mathcal{F}' = \{\oplus, F\}$ and $F(u,v) = \oplus(u,u)$ for all u, v. (So v is just a dummy variable and u is a variable which supplies both arguments to the cut function $\oplus$.) Thus while the requirement that the variables are distinct may appear as a restriction, it is not: For every family $\mathcal{F}$ there exists a family $\mathcal{F}'$ such that Max-CSP($\mathcal{F}'$) captures the Max-CSP($\mathcal{F}$) problem where variables are allowed to repeat.

We also remark that in some prior works in the Boolean setting, i.e., when $q = 2$, constraints may be applied to "literals", rather than "variables". We refer to these results as applying to Boolean CSPs over literals. In our setting we apply constraints only to variables. Again, our setting is more general than the setting of Boolean CSPs over literals in that for every family $\mathcal{F} \subseteq \{f : \mathbb{Z}_2^k \to \{0,1\}\}$ there is a family $\mathcal{F}'$ such that CSP with constraints from $\mathcal{F}$ applied to literals is the same problem as Max-CSP($\mathcal{F}'$). For example consider the Max-2-LIN problem, i.e., the CSP whose constraints are given by linear equations modulo 2 on two distinct variables. Max-2-LIN is the Boolean CSP over literals over the family $\mathcal{F} = \{\oplus\}$. But if we let $\mathcal{F}' = \{\oplus, \overline{\oplus}\}$ where $\overline{\oplus}(u,v) = u + v + 1 \pmod 2$, then Max-2-LIN = Max-CSP($\mathcal{F}'$). On the other hand Max-CSP($\mathcal{F}$) is the Max-CUT problem which can not be expressed as a Boolean CSP over literals. Thus our setting is strictly richer in expressibility.

2.1 Why study CSPs?

Before going on to giving more precise descriptions of the approximation versions of CSPs and models of streaming algorithms, we digress to comment on why study CSPs at all.

CSPs do capture a host of natural optimization problems: Some familiar names of problems include the Maximum Cut problem in (undirected) graphs, the Maximum Dicut problem in directed graphs, the Maximum q-colorability problem in graphs, the Unique Games problem, etc. Each one of these problems on their own right has probably been the topic of multiple papers, and the umbrella of CSPs unifies their study. That being said this reason on its own is not as compelling as some of the other reasons we describe next — after all the study of CSPs does exclude many other natural optimization problems including problems based on connectivity in graphs such as flow maximization or congestion minimization. It also excludes global considerations such as balanced cuts or sparse cuts; and of course there are a host of non-graph-theoretic problems.

To this author, the real reason to study CSPs is that they tend to allow for finite classification. The first such result dates back to Schaefer [Sch78] who studied the satisfiability of Boolean CSPS and showed that they exhibit a dichotomy. Feder and Vardi [FV98] explored the expressibility of different logics and arrived at a morally "broadest" set of problems ("Monotone Monadic SNP") that could potentially exhibit a dichotomy, and showed that this set of problems was essentially equivalent to CSPs over arbitrary finite alphabets. They posed the dichotomy of this class as an open question which was eventually resolved by Bulatov [Bul17] and Zhuk [Zhu20] after many years of sustained attack. Subsequent works extended such classification quests to other classes of problems including optimization and counting. (See Creignou, Khanna and Sudan [CKS01] for some of the early lines of work.) Many of these bodies are extensive, see e.g. the recent monograph by Cai and Chen [CC17] and references therein for vast explorations of counting problems. In optimization and approximation the work of Raghavendra [Rag08] gives a fine dichotomy, under the "Unique Games Conjecture", that inspires some of the streaming work we describe in this survey.

Finite classifications are interesting in that they highlight the generality of some algorithms. Even the weak classification of the approximability of CSPs pointed to the general utility of the randomized rounding and Max Flow algorithms [KSTW01]. The sharp characterizations in [Rag08] point to the power of semidefinite programming and specifically to the sum-of-squares framework of algorithms. One could also ask similar questions in the context of streaming: The dichotomy work presented in this survey does highlight the role of norm estimation algorithms in streaming optimization. Other algorithms might emerge with further exploration.

Finite classifications also point to interesting phenomena. For instance in the context of polynomial time approximability most natural problems have approximability in well separated bands of functions: constant factor approximations, polylogarithmic approximations, and polynomial approximations are common whereas very few have intermediate approximability, say to within a factor of $2^{\sqrt{\log n}}$. A finite classification implies this phenomenon - the entire infinite class of functions only shows finitely many distinct behaviors. A similar phenomenon again seems to occur with streaming algorithms — many problems have polylogarithmic space approximation algorithms while others require polynomially growing space. Intermediate complexity is rare. CSPs again seems to validate this separation, at least in the context of sketching algorithms, as we will see in the rest of this survey.

3 Preliminaries: Approximation and Streaming

We formalize some basic notions related to approximation problems and streaming algorithms. While a reader familiar with the notion might skip ahead we recommend they make sure they understand the notions (and notations) of: (i) trivial approximation algorithms, (ii) gapped optimization problems and the notation: (γ, β)-Max-CSP($\mathcal{F}$) (iii) approximability and approximation-resistance and (iv) sketching algorithms.

3.1 Approximating CSPs

Since solving Max-CSP($\mathcal{F}$) exactly can be quite hard for most $\mathcal{F}$ we turn often to algorithms that produce approximate solutions. We discuss some basic definitions regarding these in this section. All definitions restrict algorithms to come from some resource-bounded class $\mathcal{C}$. While we defer the discussion of the specific classes considered to later sections, here we consider definitions for a generic such class $\mathcal{C}$.

The most common notion of approximation is an α-approximation algorithm for some $\alpha \in [0,1]$. An α-approximation algorithm $\mathbf{ALG}$ for Max-CSP($\mathcal{F}$) is one that for every instance Ψ outputs a value $\mathbf{ALG}(\Psi)$ satisfying $\alpha \cdot \mathsf{val}_\Psi \leq \mathbf{ALG}(\Psi) \leq \mathsf{val}_\Psi$.[2] We say Max-CSP($\mathcal{F}$) is α-approximable in $\mathcal{C}$ if there exists $\mathbf{ALG} \in \mathcal{C}$ that is an α-approximation algorithm for Max-CSP($\mathcal{F}$).

A more refined notion of approximation that is more common in the literature proving non-existence of algorithms is associated with gapped problems. Given $0 \leq \beta < \gamma \leq 1$, we say that an algorithm $\mathbf{ALG}$ solves the "(γ, β)-approximation version of Max-CSP($\mathcal{F}$)", abbreviated (γ, β)-Max-CSP($\mathcal{F}$), if the following two conditions hold: (1) For every Ψ such that $\mathsf{val}_\Psi \geq \gamma$, we have $\mathbf{ALG}(\Psi) = 1$ and (2) For every Ψ such that $\mathsf{val}_\Psi \leq \beta$, we have $\mathbf{ALG}(\Psi) = 0$. We say that (γ, β)-Max-CSP($\mathcal{F}$) is solvable in $\mathcal{C}$ if there exists an $\mathbf{ALG} \in \mathcal{C}$ solving (γ, β)-Max-CSP($\mathcal{F}$).

Assuming $\mathcal{C}$ satisfies mild closure properties the latter notion roughly captures α-approximability precisely, while giving more detailed information. To see some flavor of the translation between

[2] Note that a small space streaming algorithm has very little hope of outputting a near-optimal solution which might take $\Omega(n)$ space to represent. So we require our algorithms only to output the value achieved by the optimal, or approximately-optimal, solution.

the two notions, suppose Max-CSP($\mathcal{F}$) is α-approximated by **ALG**. Now consider a pair γ, β with $\beta < \alpha\gamma$. Then the algorithm **ALG**$'$ that outputs **ALG**$'(\Psi) = 1$ if **ALG**$(\Psi) > \beta$ and 0 otherwise solves (γ, β)-Max-CSP($\mathcal{F}$). And if $\mathcal{C}$ satisfies the closure property that **ALG**$' \in \mathcal{C}$ whenever **ALG** $\in \mathcal{C}$ then it follows that α-approximability in $\mathcal{C}$ implies (γ, β)-Max-CSP($\mathcal{F}$) is solvable in $\mathcal{C}$ for every $\beta < \alpha\gamma$. A rough converse is also true, again assuming some (this time more stronger) closure properties of $\mathcal{C}$ that we leave unspecified: If for some α we have that for every $\gamma \in [0, 1]$, the (γ, β)-Max-CSP($\mathcal{F}$) is solvable in $\mathcal{C}$ for $\beta = \alpha\gamma$, then for every $\epsilon > 0$ we have that Max-CSP($\mathcal{F}$) is $\alpha - \epsilon$ approximable in $\mathcal{C}$. (See [CGSV21b, Proposition 2.21] for a detailed statement and proof.)

The discussion above explains why the study of (γ, β)-Max-CSP($\mathcal{F}$) (for every γ, β and $\mathcal{F}$) is at least as rich as the study of α-approximability of Max-CSP($\mathcal{F}$). But it can provide more detailed information. For instance researchers are often interested in approximating the value on satisfiable, or nearly satisfiable, instances. (See for instance, [DK13, BK12, KOT+12] for such works in the setting of $\mathcal{C}$ being all polynomial time algorithms.) We can understand these corner cases by focussing on $\gamma = 1$ or $\gamma \to 1$ and exploring the maximal β such that (γ, β)-Max-CSP($\mathcal{F}$) is solvable in $\mathcal{C}$. For instance recent results show that for every $\beta < 1$, $(1, \beta)$-Max-DICUT is solvable in $\mathcal{C}$ when $\mathcal{C}$ is the class of polylog space bounded sketching algorithms — a result that is not captured by the single parameter approximability of the problem.

Before concluding we also highlight what is a "non-trivial" approximation. For families $\mathcal{F}$ where every constraint has at least one satisfying assignment this notion is quite simple. We say that an algorithm that outputs a constant (independent of the input Ψ) is a trivial algorithm. Note that trivial algorithms are still legitimate approximation algorithms. For instance the algorithm that always outputs 1/2 is a 1/2-approximation for Max-CUT — this is so since no instance Ψ of Max-CUT has $\mathsf{val}_\Psi < 1/2$. We say that an approximation ratio is non-trivial if it is not achieved by a trivial algorithm. Similarly we say that a (γ, β)-approximation is non-trivial if $\gamma < 1$ and there exists an instance Ψ with $\mathsf{val}_\Psi \leq \beta$. To quantify this notion of non-triviality we define $\rho_{\min}(\mathcal{F})$ to be $\inf_{\Psi \text{ instance of Max-CSP}(\mathcal{F})}\{\mathsf{val}_\Psi\}$. We say that a Max-CSP($\mathcal{F}$) is *approximation resistant* to $\mathcal{C}$ if for every $\alpha > \rho_{\min}(\mathcal{F})$ no algorithm **ALG** $\in \mathcal{C}$ is an α-approximation to Max-CSP($\mathcal{F}$). Equivalently for every $\rho_{\min}(\mathcal{F}) < \beta < \gamma < 1$ it is the case that (γ, β)-Max-CSP($\mathcal{F}$) is not solvable in $\mathcal{C}$. We say Max-CSP($\mathcal{F}$) is *approximable* in $\mathcal{C}$ if it is not *approximation resistant* to $\mathcal{C}$.

In what follows we will describe works exploring the various approximation factors achievable for Max-CSP($\mathcal{F}$) for different $\mathcal{F}$ with streaming algorithms that have bounded space. We shall also explore some restrictions of streaming algorithms known as sketching algorithms. We introduce these terms below.

3.2 Streaming and Sketching algorithms

We consider the approximability of Max-CSP($\mathcal{F}$) when the input instance $\Psi = (C_1, \ldots, C_m)$ is presented as sequence of constraints to the approximating algorithm. The algorithm is restricted in the amount of space it is provided. We allow the algorithm to be randomized: in all upper bounds the algorithm will be expected to generate and store any randomness in the restricted space it is given, while in the lower bounds we will rule out algorithms that are a distribution over deterministic algorithms (and so strictly more general).

Formally a space $s(n)$-streaming algorithm for (γ, β)-Max-CSP($\mathcal{F}$) on n variables is given by a pair of functions (τ, ν) where $\Gamma : \{0,1\}^{s(n)} \times \Lambda_n \to \{0,1\}^{s(n)}$ is the state transition function and output function $\nu : \{0,1\}^{s(n)} \to \{0,1\}^{s(n)}$, where $\Lambda_n = \Lambda_n(\mathcal{F})$ denotes the set of all possible constraints of Max-CSP($\mathcal{F}$) on n variables.On input a stream $\boldsymbol{\sigma} = (C_1, \ldots, C_m) \in \Lambda_n^m$ the algorithm first computes the state $S(\boldsymbol{\sigma}) = S_m$ where $S_i = \Gamma(S_{i-1}, C_i)$ for $i \in [m]$ and $S_0 = 0$ in the deterministic case. It then outputs $\nu(S(\boldsymbol{\sigma}))$. A randomized streaming algorithm is the same except

that now the initial state S_0 is distributed uniformly randomly over $\{0,1\}^{s(n)}$.

In this paper we will also focus on a restriction of streaming algorithms known as sketching algorithms. To define this notion consider the concatenation of two streams $\boldsymbol{\sigma} \circ \boldsymbol{\tau}$. By definition of a streaming algorithm the final state $S(\boldsymbol{\sigma} \circ \boldsymbol{\tau})$ can be determined from $S(\boldsymbol{\sigma})$ and $\boldsymbol{\tau}$. A sketching algorithm is one that is restricted even further in that $S(\boldsymbol{\sigma} \circ \boldsymbol{\tau})$ can be determined from $S(\boldsymbol{\sigma})$ and $S(\boldsymbol{\tau})$, i.e., there exists a composer function $C : \{0,1\}^{s(n)} \times \{0,1\}^{s(n)} \to \{0,1\}^{s(n)}$ such that for every $\boldsymbol{\sigma}, \boldsymbol{\tau} \in \Lambda_n^*$, we have $S(\boldsymbol{\sigma} \circ \boldsymbol{\tau}) = C(S(\boldsymbol{\sigma}), S(\boldsymbol{\tau}))$.

Most common sketching algorithms are obtained from so called "linear-sketching algorithms" where $C \in \Gamma_n$ is viewed as a vector in $v \in \mathbb{R}^N$ for some large N, and a stream $(C_1, \ldots, C_m)$ represents the sum of the m corresponding vectors $v_1 + \cdots + v_m$. The sketch of a vector v is given by Av where $A \in \mathbb{R}^{N \times s}$ projects v down to some low-dimensional subspace. Ignoring bit precision issues this compresses large N dimensional inputs into small s dimensional sketches that end up giving significant information about the original input, surprisingly often. It is easy to see that such linear sketching algorithms indeed satisfy the definition of sketching.

In the rest of this article we describe the surprising effectiveness of sketching algorithms in approximating Max-CSP($\mathcal{F}$). We also describe matching lower bounds for sketching algorithms that often generalize also to give streaming lower bounds.

4 Results on Streaming CSPs

Prior to 2010, despite extensive work on streaming algorithms and lower bounds for other problems, there were no works covering CSPs. This was even noted at a workshop at Bertinoro in 2011 [IMNO11].

Lower bounds for Max-CUT. The first works focussed on lower bounds for the Max-CUT problem in independent works by Kogan and Krauthgamer [KK15] and Khanna, Kapralov and Sudan [KKS15]. The former showed that there existed $\alpha < 1$ such that α-approximating Max-CUT requires $\Omega(\sqrt{n})$-space. The latter showed the tighter result showing that for every $\alpha > 1/2$, α-approximating Max-CUT requires $\Omega(\sqrt{n})$-space in the streaming setting. In other words Max-CUT is approximation resistant to $o(\sqrt{n})$ space streaming algorithms. Subsequent works focussed on the space complexity and pushed it higher. Khanna, Kapralov, Sudan and Velingker [KKSV17] pushed the space requirement up to linear at the cost of a weaker approximation; specifically they showed that there exists $\alpha < 1$ such that α-approximating Max-CUT requires $\Omega(n)$-space. Finally, in a tour-de-force work, Kapralov and Krachun [KK19] settled the approximability of Max-CUT essentially completely by showing it is approximation resistant to $o(n)$ space streaming algorithms.

An aside is in order here: The input to a Max-CSP($\mathcal{F}$) has length $O(m \log n)$ and even if we forbid repeated constraints m can be as large as n^k. So, a priori one could imagine space complexities of streaming algorithms being much higher than $O(n)$. But a folklore observation shows that it suffices for the streaming algorithm to maintain a random sample of $O(n/\epsilon^2)$ constraints and the optimum value on the sampled constraints is a $(1-\epsilon)$ approximation to the optimum value on the input instance.[3] Since this sample of constraints takes only $O(n \log n)$ space to store and the optimal value for this sample can be computed in $O(n)$ space (though using exponential time), it follows that $O(n \log n)$ space suffices to get α-approximation for every Max-CSP($\mathcal{F}$) problem for every $\alpha < 1$. In particular, returning to the Max-CUT problem, the space lower bound from [KK19] is optimal to within a logarithmic factor.

[3] This observation also relies on the fact that the value of every instance is bounded away from 0, which in turn relies on the fact that 0 is not a constraint function in $\mathcal{F}$.

Upper bounds for Max-DICUT. While the lower bounds for Max-CUT are technically hard (especially [KK19]) arguably these results are not very surprising: A streaming algorithm with limited (say polylogarithmic) space seems hardly capable of understanding the global structure imposed by the many different constraints and understanding how well they can be satisfied simultaneously. Indeed the lower bounds of [KKS15] show that $o(\sqrt{n})$ space streaming algorithms can not distinguish random graphs from random bipartite graphs with a planted bipartition. In view of such limited power to understand the global structure it would not have surprised some researchers (notably this author) if every Max-CSP($\mathcal{F}$) problem had turned out to be approximation resistant to $o(n)$-space streaming algorithms. In other words — it was conceivable in 2015 that there were no non-trivial streaming algorithms for CSPs.

A striking paper of Guruswami, Velingker and Velusamy [GVV17] changed the picture by giving an elegant and simple algorithm for approximating Max-DICUT, the problem whose input is a directed graph on vertex set $[n]$ and the goal is to find a cut $S \sqcup \overline{S}$ that maximized the number of edges going from S to $\overline{S}$. (This problem is expressible as Max-CSP($\mathcal{F}$) for $\mathcal{F} = \{u \wedge \overline{v}\}$.) The "trivial" approximability of this problem is 1/4, but [GVV17] gave a non-trivial 0.4-approximation algorithm for this problem using polylogarithmic space.[4] The key insight to their algorithm is that one can estimate some non-trivial global information about the input by appealing to norm estimation algorithms that have been well explored in the sketching community. In particular their work relies on algorithms for estimating the ℓ_1 norm of a vector in the "turnstile" model which go back to the work of Indyk [Ind06]. As we will discuss later, this algorithm can be generalized arguably quite surprisingly to many other CSPs.

Tight bounds and classification of Boolean binary CSPs. While the lower bound for Max-CUT is obviously tight, the Max-DICUT approximability of .4 from [GVV17] was not known to be tight. From the $1/2+\epsilon$-inapproximability of Max-CUT one can deduce a $(1/2+\epsilon)$-inapproximability, for every $\epsilon > 0$, for Max-DICUT as well (by a reduction which maps every edge from an instance of Max-CUT to a pair of directed edges between the same vertices). Neither the algorithm nor the analysis appear tight. Indeed in a subsequent work, Chou, Golovnev and Velusamy [CGV20] managed to improve both the algorithm and the lower bound to get a tight approximability of 4/9 for Max-DICUT. Specifically they give a polylog space algorithm achieving this approximation ratio and also prove that no streaming algorithm with $o(\sqrt{n})$ space can do better! This tight result for Max-DICUT may appear accidental, but [CGV20] go further and classify the approximability of every Boolean (i.e., with $q = 2$) CSP on literals on binary constraints (i.e., $k = 2$). In doing so their work points to some remarkable phenomena: For every $\alpha \in [0, 1]$, every CSP in the finite, but nevertheless diverse, class they consider either is α-approximable in polylogarithmic space, or is not $\alpha - \epsilon$ approximable (for every $\epsilon > 0$) with $o(\sqrt{n})$ space. And in all cases the approximation algorithm uses the ℓ_1-norm approximator in a manner similar to [GVV17]. Together these results suggest a broader phenomenon explored and somewhat confirmed in the further work reported next.

Sketching complexity of CSPs. In joint work with Chou, Golovnev and Velusamy [CGSV21b] we give a dichotomy result for all (γ, β)-Max-CSP($\mathcal{F}$) (i.e., for every $k, q, \mathcal{F}$ and every $\gamma, \beta \in [0, 1]$) for $o(\sqrt{n})$-space sketching algorithms, as described below.

Theorem 4.1 ([CGSV21b, Theorem 1.1]). *For every $q, k \in \mathbb{N}$, $0 \leq \beta < \gamma \leq 1$, and every $\mathcal{F} \subseteq \{\mathbb{Z}_q^k \to \{0, 1\}\}$, one of the following two conditions holds: Either (γ, β)-Max-CSP($\mathcal{F}$) can be*

[4] Throughout this article we will not spell out the exponent in polylogarithmic terms though of course the original papers give more detailed answers.

solved by a polylogarithmic space sketching algorithm, or for every $\epsilon > 0$, every sketching algorithm for $(\gamma - \epsilon, \beta + \epsilon)$-Max-CSP($\mathcal{F}$) requires $\Omega(\sqrt{n})$-space. Furthermore there is a polynomial space algorithm that decides, given γ, β and $\mathcal{F}$, which of the two conditions holds.

A corollary to approximation resistance is the following: For every $\mathcal{F}$, either Max-CSP($\mathcal{F}$) is approximable by a polylogarithmic space sketching algorithm, or it is approximation resistant to $o(\sqrt{n})$-space sketching algorithms.[5] In the special case of $k = q = 2$ the lower bound above extends beyond sketching algorithms to all streaming algorithms ([CGSV21b, Theorem 1.3]). Put together these results subsume all previous works with the exception of the linear space lower bound for Max-CUT from [KK19]. Even in the case of $k = q = 2$ the dichotomy is more detailed than the one in [CGV20] in that it covers all CSPs, not just CSPs on literals, and it also talks about the solvability of all (γ, β)-Max-CSP($\mathcal{F}$) and not only the best approximation ratio. For example the results show that for every sufficiently small $\epsilon > 0$, $(1 - \epsilon, 1 - 2\epsilon)$-Max-DICUT is solvable by a polylogarithmic space sketching algorithm while $(1 - \epsilon, 1 - 2\epsilon + \delta)$-Max-DICUT requires $\Omega(\sqrt{n})$ space for every streaming algorithm for every $\delta > 0$. In particular it asserts that nearly satisfiable instances are detectable by small space sketching algorithms.

The sketching algorithms used for the positive result in Theorem 4.1 builds on the algorithm of [CGV20], which we refer to as a "bias-based algorithm" here. We will discuss that algorithm further later, but highlight one major difference. Rather than appealing to ℓ_1-norm estimation algorithms, the new algorithm appeals to a matrix norm estimation algorithm, this time from the work of Andoni, Krauthgamer and Onak [AKO11]. (Roughly the ℓ_1 norm given by $\|(x_1, \ldots, x_n)\|_1 = \max_{b_1,\ldots,b_n \in \{-1,+1\}} \sum_{i=1} b_i x_i$ optimizes over a Boolean domain. The matrix norm estimators allow us to optimize some problems over q-ary domains.)

While the theorem holds out the possibility that there are non-trivial approximation algorithms for (infinitely) many CSPs, this is not immediate from the theorem statement due to the lack of "explicitness" of the classification. Specifically there is no simple relationship that says given $\mathcal{F}$ what range of γ and β are "easy" (i.e., solvable in polylog space) and which ones are not. This is unfortunately inevitable. As $\mathcal{F}$ gets more complex the relationships do seem to get more complex. The results of [CGSV21b] show that γ and β are determined by optimizing some $O(q^k)$ real variable linear function over the reals subject to some degree k polynomial constraints. Even in the case of Max-DICUT this leads to some degree 2 polynomials in γ and β that determine the complexity. (See [CGSV21b, Example 1, Pages 21-23] for more details.) Nevertheless the conditions can be analyzed computationally, and in particular using the quantified theory of reals (using only the existential theory does not seem to suffice) to understand the complexity of (γ, β)-Max-CSP($\mathcal{F}$) for any given $\gamma, \beta, \mathcal{F}$.

Remarkably some subsequent work has managed to extract explicit results, even for infinite families of functions, by exploring the decision conditions arising from the proof of Theorem 4.1. For instance, Boyland, Hwang, Prasad, Singer and Velusamy [BHP+21], analyze the approximability of Max-kAND for every $k \in \mathbb{N}$ — the problem where constraints are the conjunctions of k-literals — and give an exact expression for the approximation ratio of Max-kAND. (They show Max-kAND is approximable to within a factor that roughly looks like $2^{-(k-1)}(1 - O(1/k))$ - see [BHP+21] for an exact expression.) In particular this gives an infinite subfamily of CSPs that is non-trivially approximable by the algorithm from [CGSV21b]. [BHP+21] also pin down the approximability of some other symmetric functions. Another work, by Chou, Golovnev, Shahrasbi, Sudan and Velusamy [CGS+22], also analyzes the sketching approximability of some linear threshold functions, giving some infinite families that are approximation-resistant to $o(\sqrt{n})$-space sketching algorithms

[5] This corollary is not immediate from the theorem statement, but uses some additional aspects of the proof. See [CGSV21b, Theorem 2.14] for details.

and other infinite families that are approximable by polylog space sketching algorithms.

Streaming Lower Bounds. While the classification essentially only rules out sketching algorithms using $o(\sqrt{n})$-space for the hard problems, for a broad class of problems it even rules out non-trivial streaming algorithms. In fact for all problems it pins down the polylogarithmic space approximability to within a factor of q — we expand on this later below, but first speak about broad classes of approximation resistant problems. We start with some definitions.

We say that a distribution $\mathcal{D}$ on $\mathbb{Z}_q^k$ is one-wise independent if for every $i \in [k]$ we have that when $X = (X_1, \ldots, X_k)$ is sampled according to $\mathcal{D}$, then X_i is distributed uniformly over $\mathbb{Z}_q$. We say that $f : \mathbb{Z}_q^k \to \{0,1\}$ supports one-wise independence if there is a one-wise independent distribution $\mathcal{D}$ supported on a subset of the satisfying assignments of f, i.e., if $\mathbf{a} \in \mathbb{Z}_q^k$ has positive probability under $\mathcal{D}$ then $f(\mathbf{a}) = 1$. We say that $\mathcal{F}$ *supports one-wise independence* if every function $f \in \mathcal{F}$ supports one-wise independence. We say that $\mathcal{F}$ *weakly supports one-wise independence* if there exists $\mathcal{F}' \subseteq \mathcal{F}$ such that $\rho_{\min}(\mathcal{F}') = \rho_{\min}(\mathcal{F})$ and $\mathcal{F}'$ supports one-wise independence.

Theorem 4.2 ([CGSV21b, Theorem 2.17])**.** *If $\mathcal{F}$ weakly supports one-wise independence, then $\mathcal{F}$ is approximation-resistant to $o(\sqrt{n})$-space streaming algorithms.*

Many natural families support one-wise independence. For readers familiar with some of these problems, we name some here without definitions: Max-Exact-kSAT for $k \geq 2$, Max-kOR, Max-CUT, Max-qColoring, Max-Unique-Games$_q$ to name a few. All of these problems turn out to be approximation-resistant by the above theorem.

Linear space lower bounds. Another direction of work has tried to extend the results of [KK19], i.e., $\Omega(n)$-space streaming lower bounds, to problems beyond Max-CUT. Here, we are far from a full understanding, but we do get approximation resistance for a (strict) subclass of families supporting one-wise independence. We define the families next.

For a function $f : \mathbb{Z}_q^k \to \{0,1\}$ and $\mathbf{a} \in \mathbb{Z}_q^k$ we define the *width of f at $\mathbf{a}$* to be $\omega_{\mathbf{a}}(f) = \frac{1}{q}|\{\theta \in \mathbb{Z}_q | f(\mathbf{a} + (\theta, \theta, \cdots, \theta)) = 1\}|$. We define the *width* of f to be the quantity $\omega(f) = \max_{\mathbf{a} \in \mathbb{Z}_q^k}\{\omega_{\mathbf{a}}(f)\}$, i.e. the maximum over $\mathbf{a}$ of the width of f at $\mathbf{a}$. (Roughly the set $L_{\mathbf{a}} = \{\mathbf{a} + (\theta, \theta, \cdots, \theta) | \theta \in \mathbb{Z}_q\}$ is a *line* through $\mathbb{Z}_q^k$ and $\omega_{\mathbf{a}}(f)$ measures the density of the intersection of this line with $f^{-1}(1)$, and the width of f is the widest such intersection.) We define the *width* of $\mathcal{F}$, denoted $\omega(\mathcal{F})$, to be the minimum over $f \in \mathcal{F}$ of the width of f. Note that $1/q \leq \omega(\mathcal{F}) \leq 1$ for every $\mathcal{F}$. Finally we say that $\mathcal{F}$ is *wide* if $\omega(\mathcal{F}) = 1$, i.e., the width is maximal.

A simple example of a wide family is the k-equality function $f_{k\mathsf{EQ}}$ where $f_{k\mathsf{EQ}}(u_1, \ldots, u_k) = 1$ if and only $u_1 = \cdots = u_k$. Note that every wide family supports one-wise independence. But there exist functions supporting one-wise independence that are not wide: For example $\oplus_3 : \mathbb{Z}_2^3 \to \{0,1\}$ given by $\oplus_3(a,b,c) = a + b + c \pmod 2$ supports one-wise independence but has width $1/2$.

The following theorem is shown in joint work with Chou, Golovnev, Velingker and Velusamy [CGS+21].

Theorem 4.3 ([CGS+21, Theorem 1.1])**.** *For every wide family $\mathcal{F}$, Max-CSP($\mathcal{F}$) is approximation-resistant to $o(n)$-space streaming algorithms.*

We will not cover any aspects of the proof of this theorem in this article, except to say that it builds on the proof of [KK19] following exactly the same sequence of steps, while replacing every step in their proof with ingredients needed to handle k-ary functions over non-Boolean alphabets. While the class of functions covered by this theorem is even smaller than the set covered by Theorem 4.2, it suffices to imply the following theorem which pins down the approximability of every Max-CSP($\mathcal{F}$) to within a factor of q.

Theorem 4.4 ([CGS+21, Theorem 4.3]). *For every family $\mathcal{F}$ and every $\epsilon > 0$, $(\omega(\mathcal{F})-\epsilon, \rho(\mathcal{F})+\epsilon)$-Max-CSP($\mathcal{F}$) requires $\Omega(n)$ space for every streaming algorithm. Consequently, for every $\mathcal{F}$ the largest α for which Max-CSP($\mathcal{F}$) is α-approximable by a $o(n)$-space streaming algorithm satisfies $\alpha \in \left[\rho_{\min}(\mathcal{F}), \frac{\rho_{\min}(\mathcal{F})}{\omega(\mathcal{F})}\right]$.*

We remark that while Theorem 4.4 immediately implies Theorem 4.3, the proof in [CGS+21] essentially derives the former from the latter using simple arguments.

4.1 Aside: Ordering CSPs

Before turning to some of the technical ingredients in the proofs we take a brief detour to cover an application of the results described above to a somewhat different class of optimization problems called ordering CSPs. We describe this class informally first: Recall that the solution space of the standard CSPs (the ones we work with in the rest of this paper) comes from a product set, namely an n-tuple of variables $(X_1, \ldots, X_n)$ takes values from $\mathbb{Z}_q \times \mathbb{Z}_q \times \cdots \times \mathbb{Z}_q$. A variation of this theme considers the setting where the variables need to be ordered, i.e., the $(X_1, \ldots, X_n)$ take on values from $\mathrm{Sym}_n = \{\pi : [n] \to [n] \mid \pi \text{ is one-to-one}\}$. (I.e., $X_i = \pi(i)$ where π is a permutation.) The natural notion of local constraints on ordering problems pick sequences of k distinct variables out of the n variables (as in standard CSPs) and look at the ordering from Sym_k induced by these k variables and constrain them. Thus a constraint function in ordering CSPs is given by $\Pi : \mathrm{Sym}_k \to \{0,1\}$ and constraint families are a set of constraint functions. Thus for every k and every family $\mathcal{F} \subseteq \{\Pi : \mathrm{Sym}_k \to \{0,1\}\}$ we get an ordering CSP, denoted Max-OCSP($\mathcal{F}$). (Note that unlike in standard CSPs, there is no notion of an alphabet or q in the case of ordering CSPs.)

Two examples of ordering CSPs include the Maximum Acyclic Subgraph (MAS) problem and the Betweenness problem. The former asks, given a directed graph G, to find the largest acyclic subgraph in it. This problem is captured as Max-OCSP($\{<\}$) where $<: \mathrm{Sym}_2 \to \{0,1\}$ satisfies $<(\pi) = 1$ if and only if $\pi(1) < \pi(2)$. By placing the constraint $<(i,j)$ for every directed edge (i,j) in a graph G, we get an Max-OCSP($<$) instance that exactly captures the MAS instance. Betweenness is the ordering problem where constraints are given by a triple of variables and require that the ordering place the middle variable between the first and third (though allowing either of the first or the third to be the higher ranked variable). Once again it can be naturally formulated as an ordering CSP.

With ordering CSPs again, one can ask what is the trivial approximability of an ordering CSP and when can a ordering CSP be solved non-trivially. Both questions turn out to have simple answers though somewhat disappointing ones from the algorithmic point of view. Note that a random ordering satisfies a constraint Π with probability $\rho(\Pi) \stackrel{\text{def}}{=} \frac{1}{k!} \cdot |\Pi^{-1}(1)|$. Letting $\rho(\mathcal{F}) = \min_{\Pi \in \mathcal{F}}\{\rho(\Pi)\}$ we get that every instance of Max-OCSP($\mathcal{F}$) has value at least $\rho = \rho(\mathcal{F})$, and thus ρ-approximation is trivial. It turns out that there are no algorithms (that can do better for any $\mathcal{F}$ running in $o(n)$-space), as shown in the following theorem from joint work with Singer and Velusamy [SSV21].

Theorem 4.5. *For every k, every family $\mathcal{F} \subseteq \{\Pi : \mathrm{Sym}_k \to \{0,1\}\}$ and every $\epsilon > 0$, every streaming $(\rho(\mathcal{F}) + \epsilon)$-approximation algorithm for Max-OCSP($\mathcal{F}$) requires $\Omega(n)$ space.*

5 Some ideas behind the proofs

5.1 The $\Omega(\sqrt{n})$ space lower bound for Max-CUT

We start with the lower bound from [KKS15] on the Max-CUT problem. We start with some basic ideas about lower bounds. Lower bounds in streaming are typically "distributional". To prove a lower bound on (γ, β)-Max-CSP($\mathcal{F}$) for some $\gamma, \beta, \mathcal{F}$, for every sufficiently large n we construct two distributions of instances on n variables – the **YES** and **NO** distributions. The **YES** distributions are supported with probability $1 - o(1)$ on instances from the set $\Gamma = \{\Psi | \mathsf{val}_\Psi \geq \gamma\}$. Similarly, the **NO** distributions are supported with probability $1 - o(1)$ on instances from the set $B = \{\Psi | \mathsf{val}_\Psi \leq \beta\}$. In the case of Max-CUT we will thus consider a **YES** distribution supported (with probability one) on bipartite graphs, and **NO** instances will have cut value at most $1/2 + o(1)$ (with probability $1 - o(1)$). The goal is to prove that for any space s algorithm **ALG** with $s = o(\sqrt{n})$ the distribution of the final state of **ALG** in the **YES** and **NO** cases are very close in total variation distance. (For distributions $\mathcal{D}, \mathcal{D}'$ supported on some set Ω the total variation distance, denoted $\|\mathcal{D} - \mathcal{D}'\|_{\text{tv}}$, is the quantity $\frac{1}{2}\sum_{\omega \in \Omega} |\mathcal{D}(\omega) - \mathcal{D}'(\omega)|$.) Since the inputs are random, it suffices to consider deterministic $s(n)$-space bounded algorithms.

Both distributions are parameterized by two constants: a small $\alpha \in (0, 1)$ and large, but constant, integer T. The graphs are defined on vertex set $[n]$ and have roughly $(\alpha/2) \cdot T \cdot n$ edges. These edges come as the union of T matchings $M'_1, \ldots, M'_T$, each with roughly $\alpha n/2$ edges.In the **NO** distribution these matchings will just be uniform matchings of the right size (we will get to the exact distribution of size shortly). In the **YES** distribution a random cut of $[n]$ is chosen by picking a vector $\mathbf{x} \in \{0, 1\}^n$ uniformly at random and letting the cut be $\{i | x_i = 1\}$. The matchings $M_1, \ldots, M_t$ are uniform subject to the condition that every matched edge crosses the cut. The lower bound is proved by a "hybrid argument" involving T steps. For $t \in \{0, \ldots, T\}$ let S^Y_t denote the state of **ALG** after seeing the first t matchings from the **YES** distribution, and similarly let S^N_t denote the state of **ALG** after the first t matchings from the **NO** distribution. By definition we have $S^Y_0 = S^N_0$. The key step is to prove that for every t,

$$\|S^Y_t - S^N_t\|_{\text{tv}} \text{ is small assuming } \|S^Y_{t-1} - S^N_{t-1}\|_{\text{tv}} \text{ is small,} \tag{5.1}$$

and to use this result inductively to conclude $\|S^Y_T - S^N_T\|_{\text{tv}}$ is small which shows that the two distributions are not distinguishable by small space algorithms. By construction the **YES** distribution is supported on bipartite graphs. If αT is sufficiently large then it can be argued by a standard Chernoff plus union bound that with probability $1 - o(1)$, a graph from the **NO** distribution also has value at most $1/2 + o(1)$ and together these suffice for the lower bound on Max-CUT. We thus turn to the proof of Eq. (5.1).

The upper bound works by designing two-party one way communication problem that captures the added distinguishability of **YES** from **NO** conditioned on knowing $S^Y_{t-1} \approx_d S^N_{t-1}$ (where $\approx_d$ indicates that the two random variables are close in terms of total variation distance). A rough abstraction of this problem is as follows: Alice, who knows $\mathbf{x}$ must send some information about it to Bob. This information may capture information such as S^Y_{t-1} and/or S^N_{t-1}, both of which may in principle depend on $\mathbf{x}$, but should be limited to $o(\sqrt{n})$ bits. Now Bob, who gets to see M_t which is either (in the **YES** case) a random matching crossing the cut given by $\mathbf{x}$ or (in the **NO** case) a random matching, must distinguish the two.

It turns out a problem very similar to this was already defined and studied in the literature. Specifically, Gavinksy, Kempe, Kerenedis, Raz and de Wolf [GKK+08] define the Boolean Hidden Matching (BHP) problem where Alice is given a uniform vector $\mathbf{x} \in \mathbb{Z}^n_2$ and Bob is given a matching $\widetilde{M}$ with $m = \alpha n$ edges on vertex set $[n]$ drawn uniformly among all such matchings, and a 0/1

labelling $\mathbf{w} \in \mathbb{Z}_2^m$ on the edges where in the **YES** case, the label w_e of an edge $e = (i, j)$ satisfies $w_e = x_i + x_j \pmod 2$, while in the **NO** case w_e's are uniformly random and independent. The goal of the communication is for Bob to distinguish the **YES** case from the **NO** case. [GKK+08] show that this problem requires $\Omega(\sqrt{n})$ bits of communication to achieve constant advantage in distinguishing. (The advantage of a protocol is the probability that the protocol outputs 1 in the **YES** distribution minus the probability it does so in the **NO** distribution. Specifically the [GKK+08] result shows that for every $\delta > 0$ there exists $\tau > 0$ such that for every $\alpha < 1/2$ and every sufficiently large n, every protocol that achieves advantage δ must communicate at least $\tau\sqrt{n}$ bits. These quantifiers are somewhat important as we will see below.)

To use the BHM lower bound from [GKK+08] we need to address two issues. First the input to Bob in the BHM problem is not the same as coming from the streaming problem. This problem is easy to deal with — Bob is getting more information in the BHM problem than in the motivating Max-CUT based problem, and this only makes the lower bound even stronger. Formally Bob can reduce an instance of BHM to the streaming inspired-problem by dropping all the edges e that have label $w_e = 0$. This gives Bob roughly $\alpha n/2$ edges (since each edge crosses the cut with probability roughly $1/2$ and these are roughly independent events) reducing exactly to the setting in the streaming-inspired problem.

The second and more important issue is that the BHM problem was only "roughly" motivated by the streaming problem above — we need a more careful and formal argument connecting the two. Formally we consider the random variables, S_t^Y, S_t^N and a hybrid variable $\widetilde{S}$, where $\widetilde{S}$ is the state of **ALG** on receiving $M_1, \ldots, M_{t-1}$ from the **YES** distribution and M_t from the **NO** distribution. The BHM lower bound immediately implies that $\widetilde{S} \approx_d S_t^Y$: The only difference between the two states is the tth input which comes from the **YES** distribution for S_t^Y and from the **NO** distribution for $\widetilde{S}$; and the setup of BHM allows Alice to generate and communicate S_{t-1}^Y to Bob allowing Bob to compute the final state and use **ALG** to distinguish them. To complement we also have $\|\widetilde{S} - S_t^N\|_{\text{tv}} \leq \|S_{t-1}^Y - S_{t-1}^N\|_{\text{tv}}$ by the data processing inequality: $\widetilde{S}$ is determined by S_{t-1}^Y and $M_t \sim \mathbf{NO}$ while S_t^N is determined from S_{t-1}^N and M_t. We stress a subtle point here: It is crucial that M_t is independent of S_{t-1}^Y and S_{t-1}^N for this inequality to be applicable, and this does hold in our case since the **NO** distribution is independent of $\mathbf{x}$ which is the only variable connecting the different matchings in the **YES** case. (This subtlety is the reason why extensions of this proof apply only to families supporting one-wise independence, or only give sketching lower bounds.)

We also comment briefly on the choice of various parameters such as α, T, ϵ (where our goal is to prove hardness of $(1, 1/2 + \epsilon)$-Max-CUT), δ (the advantage allowed in BHM) and τ (where the space lower bound is $\tau\sqrt{n}$). We want our bound to hold for every $\epsilon > 0$ so given ϵ, we first pick α small enough for the BHM lower bound to hold. In our case it holds for every $\alpha < 1/2$. Given this choice of α we pick T large enough so that a graph from the **NO** distribution with $\alpha T n$ edges is very likely not to have a Max-CUT of fractional size more that $1/2 + \epsilon$. Given this choice we pick δ small enough so that T applications of the hybrid argument still lead to negligible advantage in distinguishing **YES** from **NO**. Finally the τ we obtain is whatever is guaranteed by the BHM lower bound for this choice of δ.

Our eventual streaming and sketching lower bounds will extend the ideas from above, but we will return to those after describing the algorithms for Max-DICUT from [GVV17, CGV20].

5.2 Bias-based algorithms for Max-DICUT

The key ingredient in the algorithm of [GVV17] for Max-DICUT is the notion of the "bias" of a graph on vertex set $[n]$. For a vertex v in a directed graph, let in-deg(v) denote the number of incoming edges into v and let out-deg(v) denote the number of outgoing edges. Now define

$\mathsf{bias}(v) = \text{in-deg}(v) - \text{out-deg}(v)$, and define $\mathsf{bias}(G) = \frac{1}{2m}\sum_{v\in[n]} |\mathsf{bias}(v)|$. Thus if we term the vector $(\mathsf{bias}(v))_{v\in[n]} \in \mathbb{R}^n$ to be the bias-vector of the graph, then the bias of the graph is essentially the ℓ_1 norm of this vector up to normalization. As mentioned already, the ℓ_1-norm and hence the bias of a graph can be estimated arbitrarily well by a streaming algorithm presented with a stream of edges using an algorithm from [Ind06]. The key to the algorithms of [GVV17] and [CGV20] are inequalities relating the bias of a graph to the dicut value, that allow them to output lower bounds of the value of the dicut. (For uniformity we will only talk about the fractional value here and later and use val_G to denote this quantity.)

Note that by definition $0 \leq \mathsf{bias}(G) \leq 1$ for every graph G on m vertices. [GVV17] show that $\mathsf{val}_G \leq \frac{1+\mathsf{bias}(G)}{2}$. This inequality follows easily from the observation that every cut must leave at least $|\text{in-deg}(v) - \text{out-deg}(v)|$ of the edges incident to v uncut. Since every uncut edge may be counted twice by this process, we get a lower bound of $\frac{1}{2}\sum_v |\text{in-deg}(v) - \text{out-deg}(v)|$ on the number of uncut edges.

[GVV17] complement upper bound above with a lower bound: For every G we must have $\mathsf{val}_G \geq \mathsf{bias}(G)$. This is "constructive" (though not in streaming sublinear space) — the greedy cut which puts all vertices with positive bias on the sink side of the cut and the rest on the source side achieves this. (A simple argument to see is iterative: Remove directed cycles from the graph one at a time till we get a DAG. This does not alter the bias. Now remove maximal length directed paths - each such path contributes one to the non-normalized bias, and also contributes at least one edge to the dicut since by maximality the source of the path must have zero indegree and the sink must have zero out degree.)

Combining the two bounds above with the lower bound $\mathsf{val}_G \geq 1/4$ for every G gives a .4 approximation algorithm: The algorithm computes $\mathsf{bias}(G)$ and outputs $\max\{\mathsf{bias}(G), 1/4\}$. To improve on this [CGV20] give an improved lower bound on val_G when $\mathsf{bias}(G) \leq 1/3$. Their bound is also "constructive" - they consider a random dicut where each vertex of positive bias is placed on the sink side with probability $1/2 + \delta$ independently (for some parameter δ that we will optimize later). Remaining vertices are placed on the sink side with probability $1/2 - \delta$ independently. They analyze the cut produced by this rounding after optimizing over δ and use the expected size as an additional lower bound. We won't reproduce their bound or analysis here, but only comment that the analysis involves optimizing degree two rational functions in δ. This already gives them a 4/9 approximation algorithm.

The choice of a single rounding probability for all vertices in the graph is somewhat surprising. (This probability may depend on the graph and bias, but once the graph is fixed all vertices get rounded with the same probability.) It seems like a choice made for ease of analysis - optimizing a single variable δ is easier than optimizing n variables! One could nevertheless ask — could we have done better with more careful choices? The surprising result from [CGV20] is that this won't help and indeed no $o(\sqrt{n})$-space algorithm can improve on the bound above! So somehow $\mathsf{bias}(G)$ is the right quantity to compute, and rounding independently with the same probability for all vertices (upto the choice of the preferred side) is the right algorithm!

5.3 The framework of [CGSV21b]

To extend the algorithm of the previous section to problems beyond Max-DICUT, we need to understand what are notions of bias of a variable and of the whole instance for Max-CSP($\mathcal{F}$) for general $\mathcal{F}$. (In this discussion we will assume $\mathcal{F}$ has a single function f though extensions to more functions are straightforward.) Recall that in the Max-DICUT problem constraints arrive as pairs (i, j) where the edge goes from vertex i to vertex j. Thus the in-degree of a vertex could be abstract as the number of constraints in which it is the second variable, and out-degree as the number of

constraints where it is the first variable. We use this to motivate a new notion of bias of a variable X_i, denoted $\mathsf{d\text{-}bias}(i)$ (for detailed bias): This will be a k dimensional vector whose jth coordinate $\mathsf{d\text{-}bias}(i)_j$ records the number of constraints in which X_i appears as the jth variable in a constraint. Considering all the biases of all vertices gives us an $n \times k$ matrix $B = B(\Psi)$ with $B(i,j) = \mathsf{d\text{-}bias}(i)_j$ that "represents" an instance Ψ.

The main idea in [CGSV21b] can roughly be captured as follows:

> *If there exists $t \in \mathbb{N}$ and instances Ψ_g and Ψ_b on t variables with $B(\Psi_g) = B(\Psi_b)$ such that $\mathsf{val}_{\Psi_g} \geq \gamma$ and $\mathsf{val}_{\Psi_b} \leq \beta$ then Max-CSP($\mathcal{F}$) can not be solved in $o(\sqrt{n})$ space by a sketching algorithm. Else it can be solved by a polylogarithmic space linear sketching algorithm.*

A priori neither statement should be obvious and we will give some idea below as to why they are true. Furthermore even if the statements are true it is not clear how to decide which of the two conditions hold (since a priori one may have to enumerate over all n and all pairs of instances to determine if the condition is true). It turns out all the issues get answered rather nicely jointly. It turns out that it suffices to consider (weighted) instances on kq variables to answer the final question, and studying the space of these instances also leads to the algorithms and the lower bounds.

Below we elaborate on this and in particular why it suffices to consider instances on a finite number of variables. We work with the simpler setting of Boolean CSPs (so $q = 2$) on literals, i.e., when constraints can be applied to variables as well as their negations. We note that the resulting setting ($|\mathcal{F}| = 1$, $q = 2$ and constraints on literals) is the case considered in a preliminary work [CGSV21a], whereas the more general result comes from [CGSV21b]. While the latter is a stronger result, the former offers more intuition into the proofs.

In this setting of constraints over Boolean literals, we show we only need to consider instances involving k variables — and we explain how this happens. Suppose there are two instances on n variables: Ψ_g with $\mathsf{val}_{\Psi_g} \geq \gamma$ and Ψ_b with $\mathsf{val}_{\Psi_b} \leq \beta$ satisfying $B(\Psi_g) = B(\Psi_b)$. We show how to simplify the two CSPs. From now onwards it will be convenient to think of a weighted CSP instance as being a distribution on constraints - where a constraint is chosen with probability proportional to its weight. Now, since we are considering Boolean CSPs on *literals* we can flip variables as necessary (by flipping literals in all constraints) till we get that 1^n is the assignment achieving $\mathsf{val}_{\Psi_g}(1^n) \geq \gamma$. To preserve $B(\Psi_g) = B(\Psi_b)$ we flip variables in Ψ_g and Ψ_b together. Note that this flipping preserves $\mathsf{val}_{\Psi_b} \leq \beta$. Next we observe that we can assume Ψ_g and Ψ_b are symmetric under permutations: I.e. if some constraint $C(X_1, \ldots, C_n)$ appears in Ψ_g with some probability p then for every permutation $\pi : [n] \to [n]$ the constraint $C(X_{\pi(1)}, \ldots, X_{\pi(n)})$ also appears in Ψ_g with the same probability p. (We can convert any Ψ to a Ψ' satisfying this feature as follows: To pick a random constraint of Ψ', pick a random constraint C of Ψ and a uniformly random permutation and let the constraint produced by $C(X_{\pi(1)}, \ldots, X_{\pi(n)})$.) This transformation preserves $\mathsf{val}_{\Psi_g} \geq \gamma$ since 1^n, the assignment achieving the maximum value is fixed under permutations. We also have that Ψ_b is closed under permutations since the (empty!) set of assignments that achieves value greater than β is also closed under permutations. The fact that Ψ_g and Ψ_b are symmetric under permutations make them very simple: All that determines these instances is the distribution supported on $\mathbb{Z}_2^k$ indicating the pattern of negations of the k variables in a randomly chosen constraint. The names of the variables are no longer relevant — since they are just a uniformly random sequence of k distinct variables! Suppose $\mathcal{D}_Y$ represents the distribution on $\mathbb{Z}_2^k$ given by Ψ_g and $\mathcal{D}_N$ the distribution given by Ψ_b. We now study these distributions further and they will lead us to the answers to the three issues raised earlier.

The space of $\mathcal{D}_Y$ and $\mathcal{D}_N$. We can go back from distributions $\mathcal{D}$ over $\mathbb{Z}_2^k$ to instances $\Psi_{\mathcal{D}}$ of $\mathsf{Max\text{-}CSP}(\mathcal{F})$ on k variables $X_1, \ldots, X_k$ as follows: A random constraint of $\Psi_{\mathcal{D}}$ is of the form $f(X_1 \oplus b_1, \ldots, X_k \oplus b_k)$ where $\mathbf{b} = (b_1, \ldots, b_k) \sim \mathcal{D}$. Now the fact that $\mathcal{D}_Y$ came from an instance Ψ_g of value at least γ implies that the all 1's assignment satisfies $\Psi_{\mathcal{D}_Y}$. The fact that $B(\Psi_g) = B(\Psi_b)$ implies that $\mathcal{D}_Y$ and $\mathcal{D}_N$ have the same marginals. It remains to interpret the implication that $\Psi_b \leq \beta$: We stress that it does not mean $\Psi_{\mathcal{D}_N}$ has value less than β - indeed $\Psi_{\mathcal{D}_N}$ can have value much larger than that or even γ! The implication turns out to be exactly the following: "For every $p \in [0,1]$ if $X_1, \ldots, X_k$ are assigned values identically and independently according to $\mathsf{Bern}(p)$ (i.e., they take values in $\mathbb{Z}_2$ with $\Pr[X_i = 1] = p$), then the expected value $\mathsf{val}_{\Psi_{\mathcal{D}_N}}(X_1, \ldots, X_k) \leq \beta$." I.e., no identical and independent probabilistic assignment to the variables satisfies many constraints.

It turn out we can now capture these considerations on $\mathcal{D}_Y$ and $\mathcal{D}_N$ in a nice mathematical framework and that will lead to matching algorithms and lower bounds. Note that a distribution on $\mathbb{Z}_2^k$ can be viewed as a vector in $\mathbb{R}^{2^k}$ in a natural way, and the space of all distributions is a convex set in $\mathbb{R}^{2^k}$. Now let $S_\gamma^Y(f)$ denote the subset of this set representing distributions $\mathcal{D}$ such that $\mathsf{val}_{\Psi_{\mathcal{D}}}(1^k) \geq \gamma$. Similarly let $S_\beta^N(f)$ denote the subset of distributions $\mathcal{D}$ such that for every $p \in [0,1]$, $\mathbb{E}_{\mathbf{b} \in \mathsf{Bern}(p)^k}[\mathsf{val}_{\Psi_{\mathcal{D}}}(\mathbf{b})] \leq \beta$. Both these sets are convex sets! (In particular for every p, the constraint $\mathbb{E}_{\mathbf{b} \in \mathsf{Bern}(p)^k}[\mathsf{val}_{\Psi_{\mathcal{D}}}(\mathbf{b})] \leq \beta$ is a linear constraint on $\mathcal{D}$, though we have infinitely many such constraints.) By construction the two sets are disjoint for $\beta < \gamma$, but they may still contain distributions with matching marginals! To see this we may project these two sets to their marginals: So let $K_\gamma^Y(f) \subseteq \mathbb{R}^k$ be the set of marginals of all distributions in $S_\gamma^Y(f)$ and similarly let $K_\beta^N(f)$ be the marginals of $S_\beta^N(f)$. The discussion thus far has reduced the question: "Do there exist n and instances Ψ_1 and Ψ_2 on n variables with $B(\Psi_1) = B(\Psi_2)$ such that $\mathsf{val}_{\Psi_1} \geq \gamma$ and $\mathsf{val}_{\Psi_2} \leq \beta$?" to the much simpler and finite dimensional question "Do $K_\gamma^Y(f)$ and $K_\beta^N(f)$ intersect?". (An affirmative answer to one question implies an affirmative answer to the other.)

Before turning to show why this leads to algorithms or lower bounds we first point out that the question of the intersection of these two sets is decidable. Specifically the intersection question can be posed as polynomial inequalities in $2^k + 1$ variables (2^k from $\mathcal{D}$ and one from p) of degree at most $k + 1$ with one variable (p) being universally quantified and the rest being existentially quantified. Results in the quantified theory of reals [BPR06] easily show how to decide this question in space polynomial in the input size, which in our case is roughly 2^k to represent the function f (and whatever else is needed to specify γ and β).

Sketching lower bound when $K_\gamma^Y(f) \cap K_\beta^N(f) \neq \emptyset$. It turns out that the existence of two distributions with matching marginals is the crux of the $\mathsf{Max\text{-}CUT}$ lower bound of [KKS15] and so extending to other settings is a reasonable hope. Specifically the $\mathsf{Max\text{-}CUT}$ lower bound relies on $\mathcal{D}_Y = \mathsf{Unif}(\{00, 11\})$ and $\mathcal{D}_N = \mathsf{Unif}(\mathbb{Z}_2^2)$. To extend to other problems and distributions, we use the same approach of dividing a long stream of constraints into T substreams of length αn. A communication problem captures the additional information gained by a substream while a hybrid argument combines the information gained from the substreams. Both steps turn out to be different though and we elaborate on them below.

The BHM problem could be interpreted as arising from the associated distributions above in two different ways. In both Alice gets $\mathbf{x} \in \mathbb{Z}_2^n$ and Bob's first input is a matching on $[n]$, which specifies potential constraints: Bob's second input can be interpreted in two ways: (1) For each constraint, he gets information on whether $\mathbf{x}$ satisfies the constraint or not, (2) Using the fact that $\mathcal{D}_Y$ is uniform on a subgroup of $\mathbb{Z}_2^2$, Bob gets input on which coset the variables in the constraint come from. The first interpretation doesn't seem naturally amenable to the lower bound techniques which seem more tailored to understanding inputs that are uniform in the **NO** case. The second

interpretation seems restricted to groups and cosets and in particular does not seem to support $\mathcal{D}_Y$ not being uniform on a set, leave alone a subgroup. However it is possible to extend this approach beyond such algebraic settings and this is what is done in [CGSV21b]. To do so they introduce the $(\mathcal{D}_Y, \mathcal{D}_N)$-Randomized Mask Detection Problem (RMD) which is again a distribution distinguishability problem in the one-way communication setting: Here Alice gets a vector $\mathbf{x} \in \mathbb{Z}_2^n$ and Bob gets a k hypermatching with $m = \alpha n$ edges. Additionally Bob gets a vector $\mathbf{w} \in \mathbb{Z}_2^{km}$, or equivalently, one vector in $\mathbb{Z}_2^k$ associated with each hyperedge of the matching. In the **YES** case this vector associated with a hyperedge is the labels of $\mathbf{x}$ restricted to the vertices incident to the hyperedge masked (i.e., xor-ed, or summed in $\mathbb{Z}_2$) by a vector $\mathbf{b} \in \mathbb{Z}_2^k$ drawn according to $\mathcal{D}_Y$. Each mask vector $\mathbf{b}$ is drawn independently for every hyperedge. The **NO** distribution is similar with the difference that now $\mathbf{b} \sim \mathcal{D}_N$ independently for each edge.

[CGSV21b] give a $\Omega(\sqrt{n})$ communication lower bound for this problem to achieve any constant advantage (this time for $\alpha < 1/k$). The lower bound works in two parts. First the extend the proof from [GKK+08] to general k in the setting where $\mathcal{D}_N$ is uniform on $\mathbb{Z}_2^k$. (As noted above this setting seems amenable to their proof technique). The second part of the proof shows how to use the first part to show hardness of RMD on distributions $\mathcal{D}_1$ and $\mathcal{D}_2$ that differ in a "simple" way (in particular they differ in probabilities of at most four structured points in their support). They then complement this by showing that one can move from every $\mathcal{D}_Y$ to every $\mathcal{D}_N$ (with matching marginals) using a finite number of steps (as a function of q and k) where each step creates a "simple" difference in the sense above. A series of triangle inequalities now shows that $(\mathcal{D}_Y, \mathcal{D}_N)$-RMD is also indistinguishable to $o(\sqrt{n})$-communication protocols.

To convert the RMD lower bound into a lower bound on $\mathsf{Max\text{-}CSP}(\mathcal{F})$ we first need to interpret the RMD inputs as constraints of a $\mathsf{Max\text{-}CSP}(\mathcal{F})$ problem, and then to prove that combining T substreams preserves indistinguishability by streaming algorithms. The first step is natural: We apply constraints so that the hidden vector $\mathbf{x}$ is expected to satisfy γ fraction of the constraints in the **YES** case: Specifically if a hyperedge gives Bob the information $\mathbf{x}|_S + \mathbf{b}$ corresponding to the restriction of $\mathbf{x}$ to some sequence S of k variables masked by $\mathbf{b}$, then the resulting constraint negates literals according to $\mathbf{x}|_S + \mathbf{b}$, so that after the negations are applied, the input to the constraint is $\mathbf{b}$ which, by the condition that $\mathcal{D}_Y \in S_\gamma^Y(f)$ is expected to satisfy the constraint with probability γ. Similarly in the **NO** case every assignment is expected to satsify the constraint with probability at most β. Taking sufficiently many constraints (i.e., $\alpha T \to \infty$) allows us to apply Chernoff bounds and the union bound to conclude tight bounds on the value of the resulting CSPs in the **YES** and **NO** case.

The tricky part turns out to be the combination. When $\mathcal{D}_N$ is uniform, the same hybrid argument as in the $\mathsf{Max\text{-}CUT}$ case works and using this twice we conclude that if $\mathcal{D}_Y$ and $\mathcal{D}_N$ have uniform marginals then the resulting CSP instances are indistinguishable by $o(\sqrt{n})$ space streaming algorithms. [CGSV21b] also cover some slight extensions that allow them to cover hardness of $\mathsf{Max\text{-}DICUT}$ where the underlying distributions do not have uniform marginals. But for general $\mathcal{D}_Y$ and $\mathcal{D}_N$ with non-uniform marginals the method truly breaks down and produces instances of $\mathsf{Max\text{-}CSP}(f)$ that are distinguishable by polylogspace algorithms as pointed out by [CKP+21]. However in such cases it is possible to show that no sketching algorithm can work. This relies on an easy reduction from RMD to a T-player simultaneous communication problem where T players each get inputs independently according to the distribution of Bob's input and then need to communicate short messages to a referee whose goal is to distinguish the inputs being all from the **YES** distribution or from the **NO** distribution. This yields the lower bound of Theorem 4.1.

A sketching algorithm when $K^Y_\gamma(f) \cap K^N_\beta(f) = \emptyset$. We now turn to the complementary result, giving a sketching algorithm when $K^Y_\gamma(f) \cap K^N_\beta(f) = \emptyset$. If the sets do not intersect then there must be a hyperplane in $\mathbb{R}^k$ separating them. Let this plane be given by $\lambda_1, \ldots, \lambda_k$ and τ so that $\{\mathbf{a} \in \mathbb{R}^k | \sum_{i \in [k]} \lambda_i a_i \geq \tau\}$ contains $K^Y_\gamma(f)$. Since $K^Y_\gamma(f)$ and $K^N_\beta(f)$ are closed sets if they are disjoint there must be a gap separating them and so we also have $\theta > 0$ so that $\{\mathbf{a} \in \mathbb{R}^k | \sum_{i \in [k]} \lambda_i a_i \leq \tau - \theta\}$ contains $K^N_\beta(f)$.

It is natural to think of λ_i as representing a preference that the ith variable in this constraint has for taking the value 1 with higher λ_i's representing higher preferences. When the ith variable in a constraint is negated we let $-\lambda_i$ capture its preference. These preferences allow aggregation across constraints and yield the definition: For $j \in [n]$, let $\mathsf{bias}(\Psi, j)$ be the sum of the appropriate λ values over all constraints that variable j participates in. Define $\mathsf{bias}(\Psi) = \frac{1}{m} \sum_{j=1}^n |\mathsf{bias}(\Psi, j)|$. (We note that these definitions extend the Max-DICUT notions exactly). $\mathsf{bias}(\Psi)$ can be estimated as previously by appealing to ℓ_1 norm estimation algorithms. The algorithm for (γ, β)-Max-CSP($\mathcal{F}$) now reports **YES** if and only if $\mathsf{bias}(\Psi) \geq \tau - \theta/2$. This algorithm turns out to be correct, using analysis ideas that follow in a straightforward way from the construction of the convex sets. In case the reader wonders where the ℓ_1 estimator is suggested in the construction of the convex sets, this happens in the step where we passed from a general Ψ_1 and Ψ_2 with matching detailed bias matrix B, to assuming 1^n achieves the maximum value of Ψ_1. The computational challenge behind this vertex is to compute the maximal satisfying assignment, flipping literals of Ψ_1 according to this, and then computing its bias. This step is achieved computationally by the ℓ_1 norm estimation algorithm!

The general case. Up to now we focussed on the simpler case of $\mathcal{F} = \{f\}$, $q = 2$ and constraints being applied to literals. It turns out that this is the exact setting considered in the early version [CGSV21a]. The extension to the general case, where $\mathcal{F}$ is not a singleton, constraints are applied only to variables, and q is general, appears in [CGSV21b]. The extensions do manage to work out with no surprises (at least no unpleasant ones).

The elimination of the need to work with literals is the conceptually hard step but works out by working with qk variables which is different from k even in the Boolean case. Roughly our simplified picture of the extremal examples Ψ_1 and Ψ_2, uses two variables for each of the k positions in constraint that a variable can appear in: one corresponding to the unnegated variable X and one to the negated variable. To extend to general q we now use q variables per coordinate $i \in [k]$ — with variable $X_{i,\sigma}$ roughly capturing the "literal" $X_i + \sigma \pmod q$. The extension to larger sets $\mathcal{F}$ is simple, we augment the detailed-bias information as well as the marginals to include information about which function $f \in \mathcal{F}$ the constraint is working with. This leads to sets $S^Y_\gamma(f)$, $S^N_\beta(f)$ that are extended to capture distributions over $\mathcal{F} \times \mathbb{Z}_q^k$ and the marginals $K^Y_\gamma(f)$, $K^N_\beta(f)$ now project to $\mathcal{F} \times [k] \times \mathbb{Z}_q$ dimensions. Somewhat surprisingly both the algorithm and the lower bounds extend to this setting with the ℓ_1 norm estimator replaced by an $\|\cdot\|_{1,\infty}$-norm estimator[6] of [AKO11]. Deciding intersection of the two sets reduces to quantified systems with 2 alternations, and roughly $|\mathcal{F}|q^k$ variables and degree k. Perhaps the most complex part of the extension is the extension of the streaming lower bounds which work with two variants of the RMD problem. Also the reduction of the communication problems to the streaming problems is a bit delicate due to the absence of literals but works out in the end. We omit the many details, referring the reader to the original paper [CGSV21b] for those.

[6] For matrix M with rows indexed by $[n]$ and columns by $\mathbb{Z}_q$ the $(1,\infty)$-norm is given by $\|M\|_{1,\infty} = \sum_{i=1}^n \max_{j \in \mathbb{Z}_q} |M_{ij}|$.

6 Future directions

One can hope for many possible extensions to the dichotomy reported in Theorem 4.1. Perhaps the dichotomy extends as is to streaming algorithms (i.e., beyond sketching), perhaps even for linear space, perhaps even for randomly ordered streams, and maybe even for multipass algorithms. Unfortunately, while several extensions are still possible, the clean dichotomy does seem to fray quite a bit for each possible extension.

At the moment it is still plausible that the dichotomy extends as is to all $o(\sqrt{n})$ space streaming algorithms though there is no strong evidence in either direction. For space beyond $\sqrt{n}$ there do seem to be a number of new candidate algorithms so our expectation would be that the current dichotomy won't hold and there may exist more than two classes of problems. We note that we don't have concrete theorems proving this though. For randomly ordered streams as well as multipass algorithms there seem to be new algorithms in polylogarithmic space. This is the subject of an upcoming work by the author with Saxena, Singer and Velusamy. Finally the multipass setting seems to be the most challenging for the lower bounds. Here some remarkable works [AKSY20, AN21], have shown strong space lower bounds but for progressively weak approximations. Here an interesting challenge is to establish a tight lower bound for any non-trivial CSP for arbitrarily large (but constant) number of passes.

Acknowledgements

I want to thank Santhoshini Velusamy for introducing me to this wonderful line of research. I want to thank her and all other co-authors — Chi-Ning Chou, Sasha Golovnev, Noah Singer, Raghuvansh Saxena, Amirbehshad Shahrasbi and Ameya Velingker — for the collaborations and discussions which informed this survey. They also caught numerous errors and gave suggestions that have hopefully made this survey more readable. I want to thank the organizers of ICALP 2022 for inviting me to write this survey, and to David Woodruff in particular for the additional encouragement and nudges that were crucial to get me going.

References

[AKO11] Alexandr Andoni, Robert Krauthgamer, and Krzysztof Onak. Streaming Algorithms via Precision Sampling. In *2011 IEEE 52nd Annual Symposium on Foundations of Computer Science (FOCS 2011, Palm Springs, CA, USA, October 23-25, 2011)*, pages 363–372, October 2011.

[AKSY20] 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, November 2020.

[AN21] Sepehr Assadi and Vishvajeet N. Graph streaming lower bounds for parameter estimation and property testing via a streaming XOR lemma. In Samir Khuller and Virginia Vassilevska Williams, editors, *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21-25, 2021*, pages 612–625. ACM, 2021.

[BHP+21] Joanna Boyland, Michael Hwang, Tarun Prasad, Noah Singer, and Santhoshini Velusamy. Closed-form expressions for the sketching approximability of (some) symmetric Boolean CSPs. *CoRR*, abs/2112.06319, 2021.

[BK12] Libor Barto and Marcin Kozik. Robust satisfiability of constraint satisfaction problems. In Howard J. Karloff and Toniann Pitassi, editors, *Proceedings of the 44th Symposium on Theory of Computing Conference, STOC 2012, New York, NY, USA, May 19 - 22, 2012*, pages 931–940. ACM, 2012.

[BPR06] Saugata Basu, Richard Pollack, and Marie-Françoise Roy. *Algorithms in Real Algebraic Geometry*. Springer, 2006.

[Bul17] Andrei A. Bulatov. A Dichotomy Theorem for Nonuniform CSPs. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS 2017, Berkeley, CA, USA, October 15-17, 2017)*, pages 319–330, October 2017.

[CC17] Jin-Yi Cai and Xi Chen. *Complexity Dichotomies for Counting Problems: Volume 1, Boolean Domain*. Cambridge University Press, 2017.

[CGS+21] Chi-Ning Chou, Alexander Golovnev, Madhu Sudan, Ameya Velingker, and Santhoshini Velusamy. Linear space streaming lower bounds for approximating CSPs. *CoRR*, abs/2106.13078, 2021. Extended abstract in Proc. STOC 2022.

[CGS+22] Chi-Ning Chou, Alexander Golovnev, Amirbehshad Shahrasbi, Madhu Sudan, and Santhoshini Velusamy. Sketching approximability of (weak) monarchy predicates. Manuscript, May 2022.

[CGSV21a] Chi-Ning Chou, Alexander Golovnev, Madhu Sudan, and Santhoshini Velusamy. Approximability of all Boolean CSPs with linear sketches. *CoRR*, abs/2102.12351, 2021.

[CGSV21b] Chi-Ning Chou, Alexander Golovnev, Madhu Sudan, and Santhoshini Velusamy. Approximability of all finite CSPs with linear sketches. *CoRR*, abs/2105.01161, 2021. Extended abstract appears in Proc. FOCS 2021.

[CGV20] 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. IEEE Computer Society, November 2020.

[CKP+21] Lijie Chen, Gillat Kol, Dmitry Paramonov, Raghuvansh Saxena, Zhao Song, and Huacheng Yu. Personal communication, March 2021.

[CKS01] Nadia Creignou, Sanjeev Khanna, and Madhu Sudan. *Complexity Classifications of Boolean Constraint Satisfaction Problems*. Discrete Mathematics and Applications. Society for Industrial and Applied Mathematics, 2001.

[DK13] Víctor Dalmau and Andrei A. Krokhin. Robust satisfiability for CSPs: Hardness and algorithmic results. *ACM Trans. Comput. Theory*, 5(4):15:1–15:25, 2013.

[FV98] Tomás Feder and Moshe Y. Vardi. The computational structure of monotone monadic SNP and constraint satisfaction: A study through datalog and group theory. *SIAM J. Comput.*, 28(1):57–104, 1998.

[GKK+08] Dmitry Gavinsky, Julia Kempe, Iordanis Kerenidis, Ran Raz, and Ronald de Wolf. Exponential separation for one-way quantum communication complexity, with applications to cryptography. *SIAM Journal on Computing*, 38(5):1695–1708, December 2008. Conference version in STOC 2007.

[GVV17] 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. Schloss Dagstuhl — Leibniz-Zentrum für Informatik, August 2017.

[IMNO11] Piotr Indyk, Andrew McGregor, Ilan Newman, and Krzysztof Onak. Open problems in data streams, property testing, and related topics, June 2011. Compiled from IITK Workshop on Algorithms for Processing Massive Data Sets (2009) and Bertinoro Workshop on Sublinear Algorithms (2011).

[Ind06] Piotr Indyk. Stable distributions, pseudorandom generators, embeddings, and data stream computation. *Journal of the ACM*, 53(3):307–323, May 2006. Conference version in FOCS 2000.

[KK15] 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. Association for Computing Machinery, 2015.

[KK19] 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. Association for Computing Machinery, June 2019.

[KKS15] 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. Society for Industrial and Applied Mathematics, January 2015.

[KKSV17] 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. Society for Industrial and Applied Mathematics, January 2017.

[KOT+12] Gábor Kun, Ryan O'Donnell, Suguru Tamaki, Yuichi Yoshida, and Yuan Zhou. Linear programming, width-1 CSPs, and robust satisfaction. In Shafi Goldwasser, editor, *Innovations in Theoretical Computer Science 2012, Cambridge, MA, USA, January 8-10, 2012*, pages 484–495. ACM, 2012.

[KSTW01] Sanjeev Khanna, Madhu Sudan, Luca Trevisan, and David P. Williamson. The Approximability of Constraint Satisfaction Problems. *SIAM Journal on Computing*, 30(6):1863–1920, January 2001. Conference versions in STOC 1997 and CCC 1997.

[Rag08] Prasad Raghavendra. Optimal algorithms and inapproximability results for every CSP? In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing (STOC 2008, Victoria, BC, Canada, May 17-20, 2008)*, pages 245–254, 2008.

[Sch78] Thomas J. Schaefer. The complexity of satisfiability problems. In *Proceedings of the 10th Annual ACM Symposium on Theory of Computing (STOC 1978, San Diego, CA, USA, May 1-May 3, 1978)*, pages 216–226. Association for Computing Machinery, May 1978.

[SSV21] 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, August 16-18, 2021)*, volume 207 of *LIPIcs*, pages 17:1–17:19. Schloss Dagstuhl — Leibniz-Zentrum für Informatik, September 2021.

[Zhu20] Dmitriy Zhuk. A Proof of the CSP Dichotomy Conjecture. *Journal of the ACM*, 67(5):30:1–30:78, August 2020. Conference version in FOCS 2017.

ECCC ISSN 1433-8092

https://eccc.weizmann.ac.il