Streaming Complexity of Approximating Max 2CSP and Max Acyclic Subgraph

Venkatesan Guruswami[*1], **Ameya Velingker**[†2], **and Santhoshini Velusamy**[‡3]

1 **Computer Science Department, Carnegie Mellon University, Pittsburgh, PA, USA**
venkatg@cs.cmu.edu

2 **School of Computer and Communication Sciences, EPFL, Lausanne, Switzerland**
ameya.velingker@epfl.ch

3 **Department of Computer Science and Engineering, Indian Institute of Technology Madras, Chennai, India**
cs13b059@smail.iitm.ac.in

Abstract

We study the complexity of estimating the optimum value of a Boolean 2CSP (arity two constraint satisfaction problem) in the single-pass streaming setting, where the algorithm is presented the constraints in an arbitrary order. We give a streaming algorithm to estimate the optimum within a factor approaching 2/5 using logarithmic space, with high probability. This beats the trivial factor 1/4 estimate obtained by simply outputting 1/4th of the total number of constraints.

The inspiration for our work is a lower bound of Kapralov, Khanna, and Sudan (SODA '15) who showed that a similar trivial estimate (of factor 1/2) is the best one can do for Max CUT. This lower bound implies that beating a factor 1/2 for Max DICUT (a special case of Max 2CSP), in particular, to distinguish between the case when the optimum is $m/2$ versus when it is at most $(1/4 + \epsilon)m$, where m is the total number of edges, requires polynomial space. We complement this hardness result by showing that for DICUT, one can distinguish between the case in which the optimum exceeds $(1/2 + \epsilon)m$ and the case in which it is close to $m/4$.

We also prove that estimating the size of the maximum acyclic subgraph of a directed graph, when its edges are presented in a single-pass stream, within a factor better than 7/8 requires polynomial space.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems, G.1.6 Optimization, G.2.1 Combinatorics, G.2.2 Graph Theory

Keywords and phrases approximation algorithms, constraint satisfaction problems, optimization, hardness of approximation, maximum acyclic subgraph

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2017.8

1 Introduction

We are concerned with the ability of single-pass streaming algorithms to estimate the optimum value of constraint satisfaction problems (CSPs), focusing in particular on very

* Research supported in part by NSF grant CCF-1526092.
† Work done mostly while at CMU.
‡ Work done mostly while at CMU, Pittsburgh during S.N. Bose Scholars Program 2016 conducted by the Science and Engineering Board (SERB), Department of Science and Technology (DST), Govt. of India, the Indo-U.S. Science and Technology Forum (IUSSTF) and WINStep Forward.

© Venkatesan Guruswami, Ameya Velingker, and Santhoshini Velusamy;
licensed under Creative Commons License CC-BY

simple (Boolean, arity two) constraints. The impetus for our investigation is a striking lower bound result by Khanna, Kapralov, Sudan [16] for the problem of estimating the Max Cut in a graph, when the edges arrive one-by-one in a streaming fashion. There is a trivial factor 1/2-approximation for the problem using only $O(\log n)$ space, namely, count the number of edges and output half this value as the estimate for Max Cut value.[1] The authors of [16] showed that even with $\tilde{\Omega}(\sqrt{n})$ space, a single-pass streaming algorithm cannot achieve a factor $(1/2+\epsilon)$-approximation, for any constant $\epsilon > 0$.[2] The lower bound in fact holds even if the edges arrive in a random (as opposed to worst-case) order. A later work shows that obtaining a β-approximation, for some β bounded away 1 requires $\Omega(n)$ space [17]. In contrast, there are streaming algorithms producing a $(1-\epsilon)$-approximation in $\tilde{O}_\epsilon(n)$ space, by use of "cut sparsifiers" [2, 19].

1.1 Context: Approximation resistance of CSPs

The Max Cut problem is a particular, most basic form of CSP, with underlying constraints being of the form $x \neq y$. More generally, a CSP over domain D is specified by a template $\Lambda = \{P_1, \ldots, P_s\}$ of predicates $P_i : D^{a_i} \to \{0,1\}$ (a_i is the arity of P_i), and an instance of MaxCSP(Λ) is specified by a variable set V and a collection of "constraint tuples" (i, τ) with $i \in \{1, 2, \ldots, s\}$ denoting the type of constraint and $\tau \in V^{a_i}$ denoting the tuple of variables to which the constraint is applied. The goal is to find an assignment $\sigma : V \to D$ so that a maximum number of constraints are satisfied, where a constraint (i, τ) is satisfied by σ if $P_i(\sigma(\tau_1), \ldots, \sigma(\tau_{a_i})) = 1$ (in other words, setting the variables in the scope of this constraint according to assignment σ satisfies the predicate P_i). The maximum possible number of satisfied constraints is called the optimum value of the CSP instance. For most templates Λ, the Max CSP problem is NP-hard to solve optimally (see [20] for a dichotomy theorem for Max CSP classifying the rare easy cases). So there has been a lot of work on designing approximation algorithms. An absolutely trivial algorithm is the random assignment algorithm that ignores the instance structure, and simply assigns a random value to each variable. This achieves a α_Λ approximation for MaxCSP(Λ) where $\alpha_\Lambda = \min_i\{\mathbb{E}[P_i]\}$, with the expectation taken over a random input to P_i – we call α_Λ the *random assignment threshold*. Since the seminal work of Håstad [13] it has been established that for several interesting CSPs, it is NP-hard to beat the performance ratio of this trivial algorithm! Such CSPs are called approximation resistant in the literature (see, for instance, [8] and references therein). Already for arity 3, several important CSPs such as Max E3SAT and Max E3LIN (linear equations mod 2) are approximation resistant.

Thanks to semidefinite programming, for arity 2 CSPs, one can do better than the α_Λ factor [7, 14]. In particular, for (Boolean) Max 2CSP, where the domain is $D = \{0,1\}$ and Λ includes all predicates of arity 2, the seminal work of Goemans and Williamson [7], gave a factor 0.79607 algorithm (this ratio was further improved to 0.8593 in [4]).[3] The GW algorithm was a substantial improvement over the random assignment threshold of 1/4 which was also the best known algorithm for Max 2CSP at that time. For the specific case of Max Cut, Goemans and Williamson get the famous 0.87856 approximation factor, a vast

[1] We use numbers < 1 to designate the approximation ratio for the maximization problems we study: a factor γ approximation means the output estimate is at least γ times the optimum, and always at most the optimum.

[2] Throughout, we allow streaming algorithms to be randomized, and their estimate should satisfy the approximation guarantee with probability say 9/10.

[3] This guarantee is stated for the Max DICUT problem, which is a CSP with a single predicate $P(x,y) = \overline{x} \wedge y$, but in fact it holds for Max 2CSP in general.

improvement over the random assignment threshold of 1/2 (which was again the best known algorithm at that point).

The aforementioned Khanna, Kapralov, Sudan result [16], however, shows that in the streaming model, Max Cut is approximation resistant! Thus, streaming algorithms cannot non-trivially estimate the optimum of even the simplest CSP. This raises the question whether streaming algorithms operating in small space can non-trivially approximate (i.e., beat the random assignment threshold) for *some* CSP, or whether every CSP is approximation resistant in the streaming model.

1.2 Our results for Max 2CSP and Max DICUT

In this work, we give a factor 2/5 streaming algorithm for Max 2CSP that uses $O(\log n)$ space. In particular, this beats the random assignment threshold of 1/4.

▶ **Theorem 1.** *Fix any $\gamma > 0$. There is an efficient single-pass streaming algorithm that, given as input a Max 2CSP instance on n variables, with constraints arriving one-by-one in an arbitrary order, uses $O_\gamma(\log n)$ space and with probability at least 9/10 outputs an estimate in the range $[(2/5-\gamma)OPT, OPT]$, where OPT is the optimum value of the CSP instance.*

Any arity 2 Boolean predicate can be expressed as the disjunction of (at most 4) AND constraints, at most one of which can be satisfied by any assignment. By AND constraints, we mean one of the predicates $(x \wedge y)$, $(\overline{x} \wedge y)$, $(x \wedge \overline{y})$, and $(\overline{x} \wedge \overline{y})$ (that is, we take an AND of two *literals*). Any Max 2CSP instance can thus be mapped into a Max 2AND instance with the same optimum value. The above theorem therefore follows from our result about Max 2AND stated below (without loss of generality, in the rest of the paper, we only focus on Max 2AND and not Max 2CSP):

▶ **Theorem 2.** *Fix any $\gamma > 0$. There is an $O_\gamma(\log n)$ space single-pass streaming algorithm that can estimate the optimum value of a Max 2AND instance, whose AND constraints arrive in an arbitrary order, within a factor of $2/5-\gamma$ with probability at least 9/10. More specifically, on an instance with m constraints and optimum value OPT, the algorithm outputs a lower estimate on OPT which, with probability at least 9/10, lies in the range $[(2/5-\gamma)OPT, OPT]$.*

Our algorithm and analysis are simple and elementary, and are based on the combination of two observations. The first is that the bias of instance, which is sum over all variables of $|\mathrm{pos}_v - \mathrm{neg}_v|$ where pos_v (resp. neg_v) is the number of AND constraints in which v participates as a positive literal (resp. negated literal), is a good proxy for the optimum value when the optimum is large. The second is that the bias can be estimated efficiently in a streaming fashion via L_1 norm estimation of a vector under bounded dynamic updates of its coordinates.

Note that prior to 1994 there was *no* efficient algorithm known to approximate Max 2AND (or even the restricted Max DICUT problem) within a factor better than the random assignment threshold of 1/4. So, its simplicity notwithstanding, it is perhaps surprising that one can in fact have a low-space and time-efficient streaming algorithm that achieves a factor much better then 1/4.

Since the semidefinite programming based approximation for Max DICUT, many works have also given simpler algorithms that beat the factor 1/4. We mention some of them here. Trevisan used randomized rounding of a natural linear program to give a factor 1/2 algorithm [21]. Alimonti obtained a factor 1/3 approximation using local search [1]. Halperin and Zwick presented simple factor 2/5 and 9/20 algorithms based on some path removal ideas, and also a factor 1/2 algorithm (via a combinatorial method to find a half-integral LP

solution) [11]. (In Appendix A, we give a different proof of the half-integrality of the LP, and the associated (non-streaming) factor 1/2 algorithm.) Feige and Jozeph [5] give a very simple factor 2/5 algorithm for Max DICUT: take the greedy cut which sets variables whose out-degree is at least their in-degree to 0, and remaining to 1, and return the better of this cut and a uniformly random cut.

None of these algorithms seem to have an efficient streaming implementation. The closest to our algorithm is the greedy algorithm in [5], and we are able to get a streaming friendly estimate of the DICUT value by avoiding computation of the greedy cut, but instead the total bias of all vertices. Further our approach extends naturally to Max 2AND.

Hardness of factor $1/2 + \epsilon$ approximation. We do not know if the 2/5 approximation factor for Max 2AND is the best possible in the streaming model. However, one cannot achieve an approximation factor larger than 1/2. This is because, by a trivial reduction from the streaming lower bound for Max Cut in [16], we can deduce the following hardness even for the special case of Max DICUT.

▶ **Theorem 3.** *For any constant $\epsilon > 0$, a factor $(1/2+\epsilon)$ randomized streaming approximation algorithm for Max DICUT must use space $\tilde{\Omega}(\sqrt{n})$. Specifically, a randomized streaming algorithm that can decide, with success probability $9/10$, whether an m edge directed graph has a dicut of value at least $m/2$ or has no dicut of value $(1/4+\epsilon)m$, requires $\tilde{\Omega}(\sqrt{n})$ space.*

Complementary algorithmic result. We show the tightness of the above hardness result via the following algorithmic result for Max DICUT. The approach is again based on estimation of the bias of the graph: we prove that graphs whose dicut value is close to $m/4$ must have small bias, and graphs with dicut value noticeably larger than $m/2$ must have noticeable bias.

▶ **Theorem 4.** *There is a randomized streaming algorithm using $O(\log n)$ space that can, with probability $9/10$, distinguish between directed graphs with maximum dicut value more than $(1/2+8\epsilon)m$ from graphs with maximum dicut value at most $(1/4+\epsilon)m$ (where m is the number of edges), for any $\epsilon \in (0, 1/16)$.*

1.3 Streaming complexity of Maximum Acylic Subgraph

In the final part of the paper, we turn to another fundamental problem, *Maximum Acyclic Subgraph* (MAS): Given a directed graph $G = (V, E)$, find an acyclic subgraph with maximum possible number of edges. Equivalently, we want an ordering of the vertices in V so that a maximum number of arcs in E go forward. Note that this makes MAS also a kind of 2CSP, albeit over a large domain $D = \{1, 2, \ldots, |V|\}$ with constraints of the form $x < y$.

The trivial algorithm which orders elements randomly, or the deterministic algorithm that takes the better solution among an arbitrary ordering and its reversal, achieves a factor 1/2 approximation. Unlike 2CSPs over fixed domains, where there are algorithms that beat the random assignment threshold [3, 14], for MAS there is no known polynomial time factor $(1/2 + \epsilon)$ approximation algorithm. However, such an algorithm is ruled out under Khot's Unique Games Conjecture [9]. The best known NP-hardness for MAS seems to be for approximation factors exceeding 65/66 [18].

Motivated by this state of affairs, we investigate whether one can show better hardness results against the restricted model of single-pass streaming algorithms. Our ultimate goal here would be to show that getting a $(1/2 + \epsilon)$-approximation requires polynomial space (we conjecture this to be the case). In this work, we prove the following weaker result. The

proof proceeds via a reduction from the Boolean Hidden Matching problem, inspired by an analogous reduction for Max Cut from [16].

▶ **Theorem 5.** *Any randomized algorithm that, given a single pass over a stream of edges of an n-vertex directed graph G in arbitrary order, outputs a $(7/8+\epsilon)-$approximation to the MAS value of G with probability at least $3/4$, must use $\Omega_\epsilon(\sqrt{n})$ space.*

We note that the above hardness factor is much better than the currently best known NP-hardness. This raises a general theme of showing space lower bounds for approximation in the streaming model for problems that currently lack intractability results in the form of NP-hardness (or perhaps even Unique Games-hardness). In this broader context, one should of course take hardness results in the streaming model with a grain of salt – the streaming lower bound for Max Cut shows that streaming algorithms might be much weaker than polynomial time algorithms. Still, we view this direction as an interesting blend between approximation algorithms in general and constraint satisfaction in particular and streaming complexity, one that could nevertheless shed some new light on the core difficulty of problems such as MAS.

1.4 Open problems

We close this front matter by highlighting two natural open problems raised by our work.

1. What is the best approximation ratio achievable by a single-pass streaming algorithm with logarithmic space for Max 2CSP (or even the restricted Max DICUT)? The answer lies in the range $[2/5, 1/2]$. We suspect either $2/5$ or $1/2$ might be the right answer.
2. What is the best approximation ratio achievable by a single-pass streaming algorithm with logarithmic space for Maximum Acyclic Subgraph? The answer lies in the range $[1/2, 7/8]$. Here we conjecture that $1/2$ is the right answer.

2 Preliminaries

Max 2AND. We formally define the Max 2AND problem. An instance of the problem consists of a set of boolean variables $x_1, x_2, \ldots, x_n$, along with a set of clauses on these variables. Each clause consists of a conjunction of two literals, i.e., each clause is of one of the following forms, for some $i \neq j$: $x_i \wedge x_j$, $x_i \wedge \overline{x_j}$, or $\overline{x_i} \wedge \overline{x_j}$. The value of the instance is the maximum possible number of clauses that are satisfied for some setting of $x_1, x_2, \ldots, x_n$.

For each variable, it will be convenient to consider the number of constraints in which that variable appears as a literal in either positive or negative form. Thus, for any i, we define pos_i to be the number of constraints in which x_i appears non-negated, while we define neg_i to be the number of constraints in which x_i appears negated, i.e., as $\overline{x_i}$.

Special Case: Maximum Dicuts. One special case of the Max 2AND problem is the Max DICUT problem. We describe the Max DICUT in the terminology of graph theory below.

$G = (V, E)$ denotes a directed graph with vertex set V and edge set E, where $|V| = n$ and $|E| = m$. For any vertex $v \in V$, d_v, in_v, and out_v denote the overall degree, in-degree and out-degree of vertex v, respectively.

▶ **Definition 6.** A *dicut* is an ordered partition (A, B) of the vertex set of a directed graph into two disjoint subsets. The *dicut value* or *size* of the dicut is defined as the number of directed edges going from a vertex in A to a vertex in B.

▶ **Definition 7.** A *maximal dicut* (*Max DICUT*) of a directed graph $G = (V, E)$ is a dicut with the maximum dicut value.

Let (S, T) be an ordered partition of V and u be a vertex in V. Then, $E(S \to T)$ denotes the set of edges going from set S to set T, $E(u \to T)$ denotes the set of edges going from vertex u to vertices in set T, $E(S \to u)$ denotes the set of edges going from vertices in set S to vertex u, and $E(S \to S)$ denotes the edges with both endpoints inside the set S.

▶ Remark. Note that the Max DICUT problem can be viewed as a special case of the MAX 2AND problem in which each clause has exactly one positive literal and one negative literal. Vertices of the underlying graph correspond to boolean variables, and each directed edge from vertex i to vertex j corresponds to a clause of the form $x_i \wedge \overline{x_j}$. It is easy to see that a maximal dicut (A, B) in the graph terminology corresponds to the assignment of variables defined by $x_i = 1$ if vertex i is in set A, while $x_i = 0$ if vertex i is in set B.

▶ Remark. The value of any Max 2AND instance with m clauses is at least $\frac{m}{4}$, since a uniformly random assignment of boolean variables satisfies $\frac{m}{4}$ clauses on expectation.

▶ **Definition 8.** A randomized algorithm is said to give a $\alpha-$*approximation* to *Max 2AND* with *failure probability* δ (or *success probability* $1 - \delta$) if for any instance Ψ, it outputs a value in the interval $[\alpha d, d]$ with probability at least $1 - \delta$, where $\delta \in \left[0, \frac{1}{2}\right)$, $\alpha \in (0, 1)$, and d is the *Max 2AND* value of Ψ.

3 Single-Pass Streaming Complexity

Given a single pass over a stream of m constraints (in arbitrary order) of a Max 2AND instance Ψ over n variables, the problem is to estimate the Max 2AND value of Ψ using $O(\log n)$ space.

3.1 $(2/5 - \gamma)$-Approximation of Max 2AND

In analysing the Max 2AND value of an instance, it will be useful to consider a notion we call *bias*. Intuitively, for each vertex, we wish to compare the number of constraints in which x_i appears in positive form versus the number of constraints in which x_i appears in negated form. The following definition of bias captures this intuition.

▶ **Definition 9.** The *bias* of a Max 2AND instance Ψ on n variables, denoted bias_Ψ, is defined as

$$\text{bias}_\Psi = \sum_{i=1}^{n} |\text{pos}_i - \text{neg}_i|.$$

▶ Remark. $0 \leq \text{bias}_\Psi \leq 2m$.

Next, we prove a couple of theorems showing the relation between the bias of an instance and the Max 2AND value.

Intuitively, observe that if the bias of an instance is close to $2m$, then most variables x_i satisfy the property that most constraints involving x_i have the same literal on x_i (i.e., x_i appears in positive or negated form). Thus, it is reasonable to expect that in order to maximize the number of satisfied constraints, x_i should be set to the value that guarantees the truth of most of these literals. The following theorem essentially states that this is the case, and a bias close to $2m$ implies a Max 2AND value that is close to optimal, i.e., close to m.

▶ **Theorem 10.** *If the bias of an instance Ψ with n variables and m constraints is at least $(1-\epsilon)2m$, where $\epsilon \in [0,1]$, then the* Max 2AND *value of Ψ is at least $(1-\epsilon)m$.*

Proof. Assume that $\text{bias}_\Psi \geq (1-\epsilon)2m$. Now, consider the following greedy assignment $x_1 = x'_1, x_2 = x'_2, \ldots, x_n = x'_n$: For each i, we let $x'_i = 1$ if $\text{pos}_i \geq \text{neg}_i$, and $x'_i = 0$ otherwise. We claim that the number of constraints satisfied by this assignment is at least $(1-\epsilon)m$, which would imply that the Max 2AND value of Ψ is also at least $(1-\epsilon)m$.

Note that the number of unsatisfied constraints is at most

$$\sum_{i=1}^{n} \min\{\text{pos}_i, \text{neg}_i\}.$$

Thus, using the fact that

$$\begin{aligned}\text{bias}_\Psi &= \sum_{i=1}^{n} |\text{pos}_i - \text{neg}_i| \\ &= \sum_{i=1}^{n} (\text{pos}_i + \text{neg}_i) - 2\sum_{i=1}^{n} \min\{\text{pos}_i, \text{neg}_i\} \\ &= 2m - 2\sum_{i=1}^{n} \min\{\text{pos}_i, \text{neg}_i\}, \end{aligned} \tag{1}$$

we have that the number of satisfied constraints of the assignment is at least

$$m - \sum_{i=1}^{n} \min\{\text{pos}_i, \text{neg}_i\} \geq m - \frac{2m - \text{bias}_\Psi}{2} = \frac{\text{bias}_\Psi}{2} \geq (1-\epsilon)m\ ,$$

as desired. ◀

The following theorem essentially shows a converse statement, namely, that in order to have a near-optimal Max 2AND value, i.e., close to m, the bias needs to be close to $2m$.

▶ **Theorem 11.** *If the bias of a Max 2AND instance Ψ with n variables and m constraints is at most $(1-\epsilon)2m$, where $\epsilon \in [0,1]$, then its Max 2AND value is at most $\left(1 - \frac{\epsilon}{2}\right) m$.*

Proof. Consider an assignment $x_1 = x'_1, x_2 = x'_2, \ldots, x_n = x'_n$ that satisfies the maximum number of constraints of Ψ. Note that for any i, at least $\min\{\text{pos}_i, \text{neg}_i\}$ constraints involving x_i are not satisfied. Therefore, the total number of constraints of Ψ that are not satisfied is at least

$$\frac{\sum_{i=1}^{n} \min\{\text{pos}_i, \text{neg}_i\}}{2}.$$

By (1), it follows that the Max 2AND value of Ψ is at most

$$\begin{aligned} m - \frac{\sum_{i=1}^{n} \min\{\text{pos}_i, \text{neg}_i\}}{2} &= m - \frac{2m - \text{bias}_\Psi}{4} = \frac{m}{2} + \frac{\text{bias}_\Psi}{4} \\ &\leq \frac{m}{2} + \frac{(1-\epsilon)2m}{4} = \left(1 - \frac{\epsilon}{2}\right) m, \end{aligned}$$

as desired. ◀

The above theorems show us that the bias of an instance and its Max 2AND value are closely related. Thus, if we can compute the bias of an instance efficiently in the single-pass streaming setting, then we obtain a method to estimate its Max 2AND value.

Algorithm 1 A $(2/5-\gamma)$-approximation algorithm of Max 2AND in the single-pass streaming setting.

1: Input: A single pass over the m constraints of an instance Ψ over n variables $x_1, x_2, \ldots, x_n$, along with a parameter $\gamma < 2/5$ for desired closeness of approximation ratio.
2: Choose $\delta = 5\gamma/(4-5\gamma)$.
3: Compute the L_1 norm of the bias vector using the technique given in [15] (for an approximation within $1 \pm \delta$) to obtain $\widetilde{\text{bias}}_\Psi$.
4: **if** $\widetilde{\text{bias}}_\Psi \geq m/2$ **then**
5: **return** $\widetilde{\text{bias}}_\Psi/2(1+\delta)$
6: **else**
7: **return** $m/4$
8: **end if**

Let us define the *bias vector* of a Max 2AND instance Ψ with n variables and m constraints to be a vector with n components such that the i^{th} component is equal to $\text{pos}_i - \text{neg}_i$ for all $v \in V(G)$. Then, bias_Ψ is the L_1 norm of the bias vector. Each constraint $l_1 \wedge l_2$ that arrives in the stream changes the i^{th} and j^{th} components of the bias vector, where literal l_1 involves variable x_i and l_2 involves variable x_j. In particular, the arrival of the constraint increases the i^{th} component by 1 if l_1 is x_i while it decreases the component by 1 if l_1 is $\overline{x_i}$. Similarly, the j^{th} component increases by 1 if l_2 is x_j, while it decreases by 1 if l_2 is $\overline{x_j}$.

The following theorem given by Indyk in [15] shows that it is possible to compute the L_1 norm of a vector efficiently under bounded dynamic updates of its coordinates in the single-pass streaming setting.

▶ **Theorem 12** (from [15]). *Let S be a stream of data, where each chunk of data is of the form (i,a), $i \in [n]$ and $a \in \{-M \ldots M\}$, where M is a constant. The L_1 norm of the data defined by $L_1(S) = \|V(S)\|_1$, where $V(S)_i = \sum_{(i,a)\in S} a$ can be estimated by an algorithm that, given an arbitrary input stream S, outputs a quantity in the interval $[(1-\epsilon)L_1(S), (1+\epsilon)L_1(S)]$ with probability at least $9/10$, such that the algorithm uses only $O(\log n/\epsilon^2)$ words of storage.*

Theorem 12 can be adapted to our setting by converting each constraint of the form $x_i \wedge x_j$ to a data chunk $\{(i,1),(j,1)\}$, each constraint of the form $x_i \wedge \overline{x_j}$ to a data chunk $\{(i,1),(j,-1)\}$, and each constraint of the form $\overline{x_i} \wedge \overline{x_j}$ to a data chunk $\{(i,-1),(j,-1)\}$. This shows that we can compute the bias of a directed graph up to any constant precision with high probability.

We are now ready to show our main algorithmic result, namely, that one can obtain a 2/5-approximation to Max 2AND in the streaming model.

▶ **Theorem 13.** *Algorithm 1 is a $(2/5-\gamma)$-approximation algorithm of* Max 2AND *with success probability $9/10$ in the single-pass streaming setting.*

Proof. Note that if $\text{bias}_\Psi = (1/4+\epsilon)2m$, $\epsilon \in [\delta/2, 3/4]$, then by Theorem 10, the Max 2AND value Val of Ψ is at least $(1/4+\epsilon)m$ and by Theorem 11, Val is at most $(5/8+\epsilon/2)m$. Moreover, by Theorem 12, with probability at least $9/10$, the L_1 norm estimation subroutine of Algorithm 1 outputs an estimate $\widetilde{\text{bias}}_\Psi \in ((1-\delta)\text{bias}_\Psi, (1+\delta)\text{bias}_\Psi)$. Since

$$(1-\delta)\text{bias}_\Psi \geq (1-\delta)\left(\frac{1}{4}+\epsilon\right)2m \geq (1-\delta)\left(\frac{1}{4}+\frac{\delta}{2}\right)2m \geq \frac{m}{2}\ ,$$

Algorithm 1 returns $\widetilde{\text{bias}}_\Psi/2(1+\delta)$ in this case. Furthermore,

$$\frac{\widetilde{\text{bias}}_\Psi/2}{\text{Val}} \geq \frac{(1-\delta)\text{bias}_\Psi/2}{(5/8+\epsilon/2)m} = \frac{(1-\delta)(1/4+\epsilon)m}{(5/8+\epsilon/2)m} \geq \frac{2}{5}(1-\delta)$$

and

$$\frac{\widetilde{\text{bias}}_\Psi/2}{\text{Val}} \leq \frac{(1+\delta)\text{bias}_\Psi/2}{(1/4+\epsilon)m} \leq \frac{(1+\delta)(1/4+\epsilon)m}{(1/4+\epsilon)m} \leq 1+\delta\ .$$

Thus, we obtain an approximation ratio of

$$\frac{2}{5}\cdot\frac{1-\delta}{1+\delta} \geq \frac{2}{5} - \gamma.$$

by the choice of δ in the algorithm

Next, consider the case in which $\text{bias}_\Psi < (1/4+\delta/2)2m$. Then, note that the algorithm always outputs a value that is at least $m/4(1+\delta)$. Moreover, by Theorem 11, we have that $\text{Val} \leq (5/8+\delta/4)m$. Therefore, the approximation ratio in this case is

$$\frac{m/4(1+\delta)}{(5/8+\delta/4)m} = \frac{2}{5}\cdot\frac{1+\delta}{1+\frac{2\delta}{5}} \geq \frac{2}{5} - \gamma\ . \qquad \blacktriangleleft$$

3.2 Hardness of $(1/2+\epsilon)$-approximation and a complementary streaming algorithm for Max DICUT

Next, we consider the Max DICUT problem. We start by examining the regime in which the Max DICUT value of an instance is close to the lower bound of $m/4$, i.e., as suboptimal as possible. For the Max DICUT problem, we define the following notion of bias for a directed graph G.

▶ **Definition 14.** We define the *bias* of a directed graph $G=(V,E)$, denoted bias_G, as $\text{bias}_G = \sum_{v\in V}|\text{out}_v - \text{in}_v|$.

▶ Remark. Note that bias_G, as defined in Definition 14, gives the identical value as bias_Ψ for the corresponding MAX 2AND instance Ψ (see Remark 2 for the correspondence). We use the notion bias_G along with the graph formulation of Max DICUT for convenience in this section.

The following theorem shows that if the DICUT value of a Max DICUT instance is close to $m/4$, then the bias of the corresponding graph must be small.

▶ **Theorem 15.** *If the Max DICUT value of a directed graph $G=(V,E)$ is at most $\left(\frac{1}{4}+\epsilon\right)m$, where $\epsilon \in \left[0,\frac{1}{16}\right]$, then its bias is at most $32\epsilon m$.*

Proof. Let $G=(V,E)$ be a directed graph with Max DICUT value at most $\left(\frac{1}{4}+\epsilon\right)m$, where $\epsilon \in \left[0,\frac{3}{4}\right]$. Let (A,B) be a maximal dicut of G.

$$|E(A\to B)| \leq \left(\frac{1}{4}+\epsilon\right)m. \tag{2}$$

$$|E(B\to A)| \leq \left(\frac{1}{4}+\epsilon\right)m. \tag{3}$$

For every vertex $u\in A$, we have

$$|E(u\to B)| \geq |E(A\to u)|. \tag{4}$$

If there is a $u \in A$ that does not satisfy (4), then it can be moved to set B to give a dicut of larger size, which contradicts the fact that (A, B) is a maximal dicut of G. Similarly, for every vertex $v \in B$, we have

$$|E(A \to v)| \geq |E(v \to B)|. \tag{5}$$

The total number of edges in the graph is

$$|E(A \to A)| + |E(B \to B)| + |E(A \to B)| + |E(B \to A)| = m. \tag{6}$$

From (4) and (5), we have

$$\max(|E(A \to A)|, |E(B \to B)|) \leq |E(A \to B)| \leq \left(\frac{1}{4} + \epsilon\right) m. \tag{7}$$

From (2), (3), (6) and (7), we get

$$\min(|E(A \to A)|, |E(B \to A)|, |E(B \to B)|) \geq \left(\frac{1}{4} - 3\epsilon\right) m. \tag{8}$$

We will now obtain an upper bound on $\sum_{v \in B} |\text{out}_v - \text{in}_v|$.

$$\begin{aligned}\sum_{v \in B} |\text{out}_v - \text{in}_v| &= \sum_{v \in B} \big||E(v \to B)| + |E(v \to A)| - |E(B \to v)| - |E(A \to v)|\big| \\ &\leq \sum_{v \in B} (\big||E(v \to B)| - |E(A \to v)|\big| + \big||E(v \to A)| - |E(B \to v)|\big|).\end{aligned} \tag{9}$$

Using (2),(5) and (8), we get

$$\begin{aligned}\sum_{v \in B} \big||E(v \to B)| - |E(A \to v)|\big| &= \sum_{v \in B} |E(A \to v)| - \sum_{v \in B} |E(v \to B)| \\ &\leq \left(\frac{1}{4} + \epsilon\right) m - \left(\frac{1}{4} - 3\epsilon\right) m \\ &= 4\epsilon m.\end{aligned} \tag{10}$$

We call a vertex $v \in B$ "good" if

$$|E(B \to v)| > |E(v \to A)|.$$

and "bad" if it is not "good". Now, consider the ordered partition (B, A). If we move a "good" vertex from B to A, the dicut value of the resulting partition is larger than the dicut value of (B, A). We know that the size of any dicut in G is at most $\left(\frac{1}{4} + \epsilon\right) m$. From (8), we can infer that the increase in the dicut value by moving a "good" vertex cannot exceed $4\epsilon m$. Let B_{g} denote the set of all "good" vertices in B. We have

$$\sum_{v \in B_{\text{g}}} (|E(B \to v)| - |E(v \to A)|) \leq 4\epsilon m. \tag{11}$$

Let B_{b} denote the set of all "bad" vertices in B. From (8) and (11), we get

$$\begin{aligned}\sum_{v \in B_{\text{b}}} |E(B \to v)| &\geq \left(\frac{1}{4} - 3\epsilon\right) m - \sum_{v \in B_{\text{g}}} |E(B \to v)| \\ &\geq \left(\frac{1}{4} - 3\epsilon\right) m - 4\epsilon m - \sum_{v \in B_{\text{g}}} |E(v \to A)| \\ &= \left(\frac{1}{4} - 7\epsilon\right) m - \sum_{v \in B_{\text{g}}} |E(v \to A)|.\end{aligned} \tag{12}$$

Using (12), we get

$$\begin{aligned}\sum_{v\in B_{\mathrm{b}}}\left(|E(v\to A)|-|E(B\to v)|\right) &\le \sum_{v\in B}|E(v\to A)|-\left(\frac{1}{4}-7\epsilon\right)m\\ &= |E(B\to A)|-\left(\frac{1}{4}-7\epsilon\right)m\\ &\le 8\epsilon m. \end{aligned} \tag{13}$$

From (9), (10), (11) and (13), we have

$$\sum_{v\in B}|\mathrm{out}_v-\mathrm{in}_v|\le 16\epsilon m.$$

Using similar arguments, we can conclude that

$$\sum_{v\in A}|\mathrm{out}_v-\mathrm{in}_v|\le 16\epsilon m.$$

Hence, the bias of G is at most $32\epsilon m$. ◀

▶ **Corollary 16.** *If the* Max DICUT *value of a directed graph $G=(V,E)$ is $\frac{m}{4}$, then its bias is* 0*, i.e.,* $\mathrm{in}_v=\mathrm{out}_v\ \ \forall v\in V$.

▶ Remark. It is reasonable to expect a converse statement of Theorem 15 to hold, i.e., that a bias close to zero implies a Max DICUT value that is close to $m/4$. However, it turns out that such a statement is not true. For example, consider the following instance of Max DICUT: Let $G(V,E)$ be an undirected perfect matching on $2n$ vertices. Using G, construct a directed graph $G'(V,E')$ by adding directed edges $u\to v,\ \ v\to u$ to E' for each undirected edge (u,v) in E. The Max DICUT value of G' is n, which is much larger than $m/4=2n/4=n/2$. However, the bias of G' is 0.

We now state the non-existence of a better-than-1/2−approximation algorithm for Max DICUT when the constraints of an instance arrive one-by-one in random order in the single-pass streaming setting.

Kapralov et al. gave a lower bound for approximating MAXCUT in the single-pass streaming setting in [16]. By showing a simple reduction from MAXCUT to Max DICUT, we observe that the same lower bound applies to Max DICUT. The following theorem is taken from [16].

▶ **Theorem 17** (from [16])**.** *Let $\epsilon>0$ be a constant. Let $G=(V,E)$, $|V|=n$, $|E|=m$ be an undirected graph. Any randomized algorithm that, given a single pass over a stream of edges of G presented in random order, outputs a $(1/2+\epsilon)$−approximation to the value of the maximum cut in G with probability at least $9/10$ over its internal randomness must use $\tilde{\Omega}(\sqrt{n})$ space.*

The reduction from MAXCUT to Max DICUT is as follows. Given any undirected graph G, convert it into a directed graph G' by adding two directed edges $u\to v$ and $v\to u$ for every edge $(u,v)\in E(G)$. Observe that G has a cut of size k if and only if G' has a dicut of size k. Therefore, the MAXCUT value of G is equal to the Max DICUT value of G'. Note that this reduction can be done on the fly and does not require any additional storage. Thus, we have the following theorem.

▶ **Theorem 18.** *Let $\epsilon > 0$ be a constant. Let $G = (V, E)$, $|V| = n$, $|E| = m$ be a directed graph. Any randomized algorithm that, given a single pass over a stream of edges of G presented in random order, outputs a $(1/2 + \epsilon)-$approximation to the* Max DICUT *value of G with probability at least $9/10$ over its internal randomness must use $\tilde{\Omega}(\sqrt{n})$ space.*

▶ Remark. Since an instance of Max DICUT can be viewed as a special instance of Max 2AND in which all clauses have one positive literal and one negative literal, the aforementioned theorem also precludes the existence of a randomized streaming algorithm that approximates the *Max 2AND* value of an instance to a factor of $1/2 + \epsilon$ without using $\tilde{\Omega}(\sqrt{n})$ space.

To prove Theorem 17, [16] showed that no $o(\sqrt{n})$ algorithm can distinguish between distributions D^Y and D^N, where graphs drawn from D^Y have MAXCUT value m, while graphs drawn from D^N have MAXCUT value at most $(1/2 + \epsilon)m$ for any $\epsilon \in [0, 1/2]$. By applying the reduction from MAXCUT to Max DICUT (the number of edges is doubled in the reduced graph), we can conclude than no $o(\sqrt{n})$ algorithm can distinguish between directed graphs with Max DICUT value $m/2$ and graphs with Max DICUT value at most $(1/4 + \epsilon)m$ for any $\epsilon \in [0, 1/4]$.

▶ **Theorem 19.** *Given a single pass over the edges of a directed graph G in any order, by computing the bias of G we can distinguish between directed graphs with* Max DICUT *value more than $(1/2 + 8\epsilon)m$ and graphs with* Max DICUT *value at most $(1/4 + \epsilon)m$ for any $\epsilon \in (0, 1/16)$, with success probability $9/10$.*

Proof. If the Max DICUT value of a graph is at most $(\frac{1}{4} + \epsilon)m$, then using Theorem 15 we can conclude that its bias is at most $32\epsilon m$. If the Max DICUT value of a graph is more than $(1 - \delta)m$, then using the contrapositive result of Theorem 11, we can conclude that its bias is more than $(1 - 2\delta)2m$. By substituting $\delta = 1/2 - 8\epsilon$, we get that the bias of the graph is more than $32\epsilon m$. Thus, we can distinguish the two graphs by computing their bias values using the L_1 sampling method. ◀

Note that while [16] implied that no $o(\sqrt{n})$ algorithm can distinguish between directed graphs with Max DICUT value $m/2$ and graphs with Max DICUT value at most $(1/4 + \epsilon)m$ for any $\epsilon \in [0, 1/4]$, Theorem 19 shows that it is possible to distinguish between directed graphs with Max DICUT value $(1/2 + 0.0001)m$ and graphs with Max DICUT value at most $(1/4 + 0.00001)m$ in the single-pass streaming setting using a $O(\log n)$ algorithm that computes the bias of a graph.

4 Maximum Acyclic Subgraph

▶ **Definition 20.** The Maximum Acyclic Subgraph (MAS) value of a directed graph $G = (V, E)$ is the size of the largest acyclic subgraph of G, where we define the size of a graph to be the number of edges in it.

Given a single pass over a stream of edges of a directed graph $G = (V, E)$, we are interested in computing an approximate estimate of the MAS value of G using logarithmic space.

▶ **Definition 21.** A randomized algorithm is said to give a $\alpha-$approximation to MAS for some $\alpha \in (0, 1)$ if for any input $G = (V, E)$, it outputs a value in the interval $[\alpha d, d]$ with probability at least $9/10$, where d is the MAS value of G.

In this section, we show the non-existence of a better-than-$7/8-$approximation algorithm to MAS in the low-space single-pass streaming setting. We show this via a reduction from

Figure 1 Edge set E_1.

Boolean Hidden Matching (BHM), a two party one-way communication problem. A strong communication lower bound for BHM was given by Gavinsky et al. in [6]. In [16], Kapralov et al. showed the hardness of approximating MAXCUT using the BHM problem and its extension given by Verbin and Yu in [22]. Inspired by this, here we adapt this approach to show hardness of approximating MAS.

▶ **Definition 22.** The Boolean Hidden Matching (*BHM*) problem is a communication complexity problem in which Alice gets a Boolean vector $x \in \{0,1\}^n$ and Bob gets an undirected perfect matching M on n vertices and a Boolean vector w of length $n/2$, where we identify the perfect matching M with its Boolean edge incidence matrix of dimension $\frac{n}{2} \times n$. It is a promise problem in which Bob outputs **YES** when $Mx \oplus w = 0^{n/2}$ and outputs **NO** when $Mx \oplus w = 1^{n/2}$.

The following theorem was proved by Gavinsky et al. in [6].

▶ **Theorem 23.** *Any randomized one-way communication protocol for solving* BHM*, where Alice sends messages to Bob, that succeeds with probability at least* $9/10$ *has complexity* $\Omega(\sqrt{n})$.

We now use the above theorem to prove our streaming lower bound for approximating MAS.

▶ **Theorem 24.** *Let $\epsilon > 0$ be a constant. Let $G = (V, E)$, $|V| = n$, $|E| = m$ be a directed graph. Any randomized algorithm that, given a single pass over a stream of edges of G, outputs a $(7/8 + \epsilon)-$approximation to the MAS value of G with probability at least $9/10$ over its internal randomness must use $\Omega(\sqrt{n})$ space.*

Proof. Let **ALG** be a randomized algorithm that uses space c and gives a better-than-$7/8-$approximation to MAS in the single-pass streaming setting. We will show that **ALG** can be used to obtain a randomized one-way communication protocol for BHM with complexity c.

Let $x \in \{0,1\}^n$ be the vector that Alice receives. Alice creates edge set E_1, her part of the graph that will be given as input to **ALG**, as shown in Fig. 1. For each $i \in [n]$, she creates four vertices a_i, b_i, c_i and d_i. If $x_i = 0$, she adds edges $a_i \to b_i$ and $d_i \to c_i$ to E_1. Else, she adds edges $b_i \to a_i$ and $c_i \to d_i$ to E_1. She then treats E_1 as the first half of the stream of edges, runs **ALG** on E_1 and sends the state of **ALG** to Bob.

Bob constructs edge set E_2, his part of the graph, as shown in Fig. 2 and Fig. 3. Let M be the perfect matching on n vertices and w be the boolean vector of length $n/2$ that Bob receives. Let $M_i = (i_1, i_2)$ denote the $i-$th edge in the matching M (fix any ordering) and w_i denote the $i-$th coordinate of w, where $i \in [n/2]$. If $w_i = 0$, he adds edges $b_{i_1} \to a_{i_2}$, $b_{i_2} \to a_{i_1}$, $d_{i_1} \to c_{i_2}$ and $d_{i_2} \to c_{i_1}$ to E_2. Else, he adds edges $b_{i_1} \to b_{i_2}$, $a_{i_2} \to a_{i_1}$, $d_{i_1} \to d_{i_2}$ and $c_{i_2} \to c_{i_1}$ to E_2.

He treats E_2 as the second half of the stream and completes the execution of **ALG** on the stream starting from the state that Alice sent. The total number of edges in the graph is

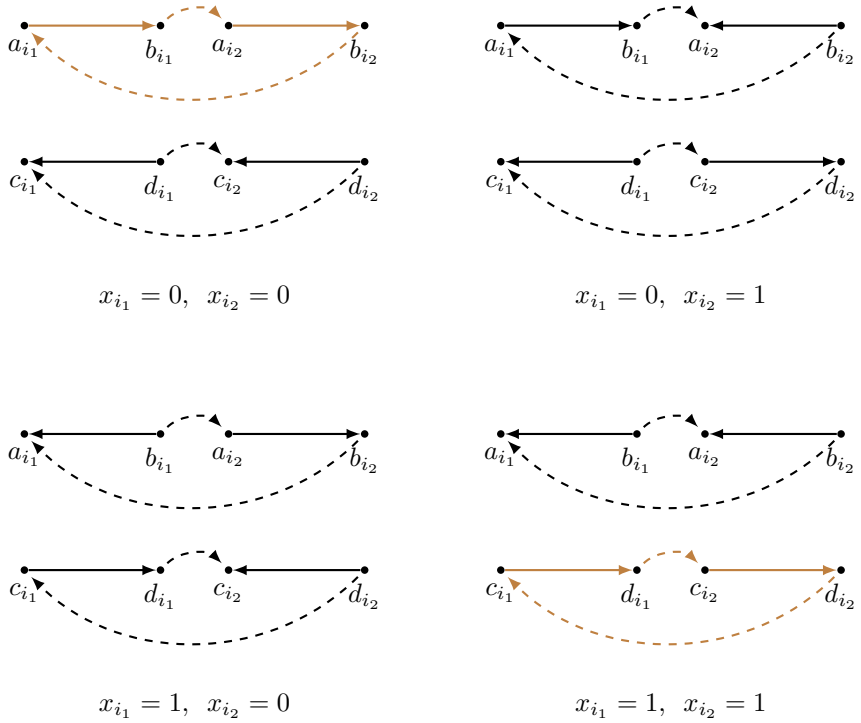

Figure 2 Edge set E_2 when $w_i = 0$ (Cycles are marked in brown).

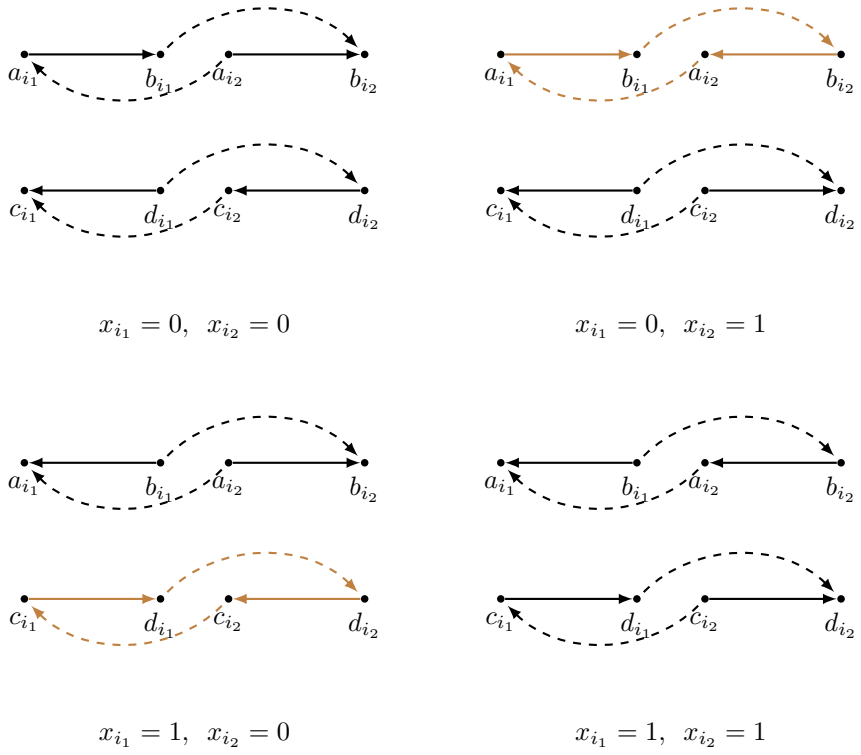

Figure 3 Edge set E_2 when $w_i = 1$ (Cycles are marked in brown).

$2n + 4(n/2) = 4n$ (Alice adds two edges for each coordinate of x and Bob adds four edges for each edge in M). If the MAS value output by **ALG** is greater than $7n/2$, then Bob outputs **NO**, else he outputs **YES**.

The correctness of the reduction can be shown in the following way.

$$Mx \oplus w = \begin{bmatrix} x_{1_1} \oplus x_{1_2} \oplus w_1 \\ \vdots \\ x_{i_1} \oplus x_{i_2} \oplus w_i \\ \vdots \\ x_{(n/2)_1} \oplus x_{(n/2)_2} \oplus w_{(n/2)} \end{bmatrix}.$$

As depicted in Fig. 2 and Fig. 3, we can observe that when $x_{i_1} \oplus x_{i_2} \oplus w_i = 0$, there is exactly one cycle and when $x_{i_1} \oplus x_{i_2} \oplus w_i = 1$, there are no cycles. Therefore, if $Mx \oplus w = 1^{n/2}$, then the graph is acyclic and the MAS value is $4n$. If $Mx \oplus w = 0^{n/2}$, then the graph contains exactly $n/2$ disjoint cycles (corresponding to each $i \in [n/2]$) and hence the MAS value is $7n/2$ (subtract one edge from each cycle to get the maximum acyclic subgraph). Since **ALG** gives a better-than-7/8−approximation to MAS, it outputs a MAS value greater than $7n/2$ if and only if $Mx \oplus w = 1^{n/2}$. Since the state sent by Alice to Bob (after executing the first half of the stream) contains at most c bits, the above protocol has randomized one-way communication complexity c with success probability at least 9/10. By using Theorem 23, we infer that $c = \Omega(\sqrt{n})$. ◀

References

1. Paola Alimonti. Non-oblivious local search for MAX 2-CSP with application to MAX DICUT. In *23rd International Workshop on Graph-Theoretic Concepts in Computer Science*, pages 2–14, 1997.
2. András A. Benczúr and David R. Karger. Approximating s-t minimum cuts in $\tilde{O}(n^2)$ time. *Proceedings of the 28th annual ACM symposium on Theory of computing*, pages 47–55, 1996.
3. Lars Engebretsen and Venkatesan Guruswami. Is constraint satisfaction over two variables always easy? *Random Structures and Algorithms*, 25(2):150–178, 2004.
4. Uriel Feige and Michel X. Goemans. Aproximating the value of two prover proof systems, with applications to MAX 2SAT and MAX DICUT. In *Third Israel Symposium on Theory of Computing and Systems (ISTCS)*, pages 182–189, 1995. doi:10.1109/ISTCS.1995.377033.
5. Uriel Feige and Shlomo Jozeph. Oblivious algorithms for the Maximum Directed Cut problem. *Algorithmica*, 71(2):409–428, 2015. doi:10.1007/s00453-013-9806-z.
6. Dmitry Gavinsky, Julia Kempe, Iordanis Kerenidis, Ran Raz, and Ronald de Wolf. Exponential separations for one-way quantum communication complexity, with applications to cryptography. In *Proceedings of the Thirty-ninth Annual ACM Symposium on Theory of Computing*, STOC'07, pages 516–525, New York, NY, USA, 2007. ACM. doi:10.1145/1250790.1250866.
7. Michel X. Goemans and David P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *J. ACM*, 42(6):1115–1145, 1995. doi:10.1145/227683.227684.
8. Venkatesan Guruswami and Euiwoong Lee. Towards a characterization of approximation resistance for symmetric CSPs. In *Proceedings of Approximation, Randomization, and Combinatorial Optimization: Algorithms and Techniques (APPROX/RANDOM)*, pages 305–322, 2015. doi:10.4230/LIPIcs.APPROX-RANDOM.2015.305.

9 Venkatesan Guruswami, Rajsekar Manokaran, and Prasad Raghavendra. Beating the random ordering is hard: Inapproximability of maximum acyclic subgraph. In *Proceedings of the 49th IEEE Symposium on Foundations of Computer Science*, pages 573–582, 2008.

10 Venkatesan Guruswami and Yuan Zhou. Tight bounds on the approximability of almost-satisfiable horn SAT and exact hitting set. *Theory of Computing*, 8(11):239–267, 2012. doi:10.4086/toc.2012.v008a011.

11 Eran Halperin and Uri Zwick. Combinatorial approximation algorithms for the maximum directed cut problem. In *Proceedings of the 12th Annual Symposium on Discrete Algorithms*, pages 1–7, 2001. URL: http://dl.acm.org/citation.cfm?id=365411.365412.

12 Eran Halperin and Uri Zwick. Combinatorial approximation algorithms for the maximum directed cut problem. In *Proceedings of the Twelfth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA'01, pages 1–7, Philadelphia, PA, USA, 2001. Society for Industrial and Applied Mathematics. URL: http://dl.acm.org/citation.cfm?id=365411.365412.

13 Johan Håstad. Some optimal inapproximability results. *J. ACM*, 48(4):798–859, 2001. doi:10.1145/502090.502098.

14 Johan Håstad. Every 2-CSP allows nontrivial approximation. *Computational Complexity*, 17(4):549–566, 2008. doi:10.1007/s00037-008-0256-y.

15 Piotr Indyk. Stable distributions, pseudorandom generators, embeddings, and data stream computation. *J. ACM*, 53(3):307–323, May 2006. doi:10.1145/1147954.1147955.

16 Michael Kapralov, Sanjeev Khanna, and Madhu Sudan. Streaming Lower Bounds for Approximating MAX-CUT. In *Proceedings of the Twenty-sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA'15, pages 1263–1282, Philadelphia, PA, USA, 2015. Society for Industrial and Applied Mathematics. URL: http://dl.acm.org/citation.cfm?id=2722129.2722213.

17 Michael Kapralov, Sanjeev Khanna, Madhu Sudan, and Ameya Velingker. $(1 + \Omega(1))$-approximation to MAX-CUT Requires Linear Space. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA'17, pages 1703–1722, Philadelphia, PA, USA, 2017. Society for Industrial and Applied Mathematics. URL: http://dl.acm.org/citation.cfm?id=3039686.3039798.

18 Alantha Newman. Approximating the maximum acyclic subgraph. Master's thesis, MIT, June 2000.

19 D. A. Spielman and N. Srivastava. Graph sparsification by effective resistances. *STOC*, pages 563–568, 2008.

20 Johan Thapper and Stanislav Zivny. The complexity of finite-valued CSPs. *J. ACM*, 63(4):37:1–37:33, 2016. doi:10.1145/2974019.

21 Luca Trevisan. Parallel approximation algorithms by positive linear programming. *Algorithmica*, 21(1):72–88, 1998. doi:10.1007/PL00009209.

22 Elad Verbin and Wei Yu. The streaming complexity of cycle counting, sorting by reversals, and other problems. In *Proceedings of the Twenty-second Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA'11, pages 11–25, Philadelphia, PA, USA, 2011. Society for Industrial and Applied Mathematics. URL: http://dl.acm.org/citation.cfm?id=2133036.2133038.

A Approximating Max DICUT Using LP Rounding

In this section, we give a deterministic $1/2-$approximation algorithm to solve the Max DICUT problem in polynomial time in the usual setting (not streaming).

We first look at the binary integer programming (BIP) formulation of the Max DICUT problem. Given a directed graph $G = (V, E)$, the objective is to obtain an ordered partition

(A, B) of V such that the number of edges not in $E(A \to B)$ is minimum.

$$\begin{aligned}
\text{minimize} \quad & \sum_{e(i,j)\in E} z_e. \\
\text{subject to} \quad & z_e \geq x_i, \qquad \forall e(i,j) \in E. \\
& z_e \geq 1 - x_j, \ \forall e(i,j) \in E. \\
& z_e, x_i \in \{0,1\}, \ \ \forall i \in V, \forall e \in E.
\end{aligned}$$

In the above BIP, $e(i,j)$ denotes a directed edge e from vertex i to vertex j. $x_i, \ \forall i \in V$ are indicator variables. If $x_i = 0$, then $i \in A$, else $i \in B$. z_e is constrained to be at least as large as $\max(x_i, 1 - x_j)$. Since the objective is to minimize $\sum_{e(i,j)\in E} z_e$, in any BIP solution, $z_e = \max(x_i, 1 - x_j)$. Thus, $z_e = 0$ if $e \in E(A \to B)$ and 1 otherwise. Hence, the optimal value of $\sum_{e(i,j)\in E} z_e$ gives the number of edges not in the Max DICUT of G.

It is hard to get an exact solution for the above BIP in the worst case. Therefore, we consider below its LP relaxation to approximately estimate the number of edges not in the Max DICUT.

$$\begin{aligned}
\text{minimize} \quad & \sum_{e(i,j)\in E} z_e. \\
\text{subject to} \quad & z_e \geq x_i, \qquad \forall e(i,j) \in E. \\
& z_e \geq 1 - x_j, \ \forall e(i,j) \in E. \\
& z_e, x_i \in [0,1], \ \ \forall i \in V, \forall e \in E.
\end{aligned}$$

Any optimum solution to the above LP assigns $z_e = \max(x_i, 1 - x_j)$. Thus $\{x_i\}$ are the independent variables and we denote any solution f to the above LP by $f = \{x_i\}$. The existence of a half-integral optimal solution to the LP and a combinatorial algorithm to obtain it was given by Halperin and Zwick in [12]. In this paper, we present an alternate algorithm to obtain a half-integral optimal solution to the LP, adapted from [10].

▶ **Theorem 25** (Half Integrality). *There is a polynomial-time algorithm that given a optimal solution $f = \{x_i\}$ to the above LP, converts f into another optimal solution $f^* = \{x_i^*\}$ such that each x_i^* is half-integral, i.e., $x_i^* \in \{0, 1, 1/2\}$, and Val$(f^*) \leq$ Val(f).*

Proof. Algorithm 2 takes as input the above LP formulation and one of the solutions $f = \{x_i\}$, and outputs the desired f^*. At a high level, the algorithm iteratively moves the LP variables that are not half integral to half integral values. We need to prove that the algorithm terminates in polynomial number of iterations and in each iteration, it creates a valid LP solution whose objective value is at most the previous objective value.

Algorithm 2 always maintains a valid solution f to the LP (i.e., all x_i's are in the range $[0, 1]$). We first prove that it terminates within a polynomial number of iterations. Consider the set $W_f = \{0 < x < 1/2 : \exists i \in V \mid x = x_i \text{ or } x = 1 - x_i\}$. In each loop, the algorithm picks a p from W_f. At the end of the loop, p is removed from W_f and no new element is added. Thus, after a linear number of steps $W_f = \emptyset$ and the loop terminates.

We now prove that the objective value of the LP does not increase after each iteration. Define $f^{(t)} = \{x_i^{(t)} = t\}_{i\in S} \cup \{x_i^{(t)} = 1 - t\}_{i\in S'} \cup \{x_i^{(t)} = x_i\}_{i\in V\setminus(S\cup S')}$ for $t \in [a, b]$. Observe that $p \in [a, b]$. If we can show that Val$(f^{(t)})$ is a linear function for $t \in [a, b]$, it proves that $\min(\text{Val}(f^{(a)}), \text{Val}(f^{(b)})) \leq \text{Val}(f^{(p)}) = \text{Val}(f)$. To prove linearity of Val$(f^{(t)})$, we only need to show that $g_{ij}(t) = \max(x_i^{(t)}, 1 - x_j^{(t)})$ is linear for $t \in [a, b], \ \forall e(i,j) \in E$. We prove each case separately for $t \in [a, b]$.

Algorithm 2 Round any LP solution $f = \{x_i\}$ to a half-integral solution f^*, with $\text{Val}(f^*) \leq \text{Val}(f)$.

while $\exists i \in V : x_i \notin \{0, 1, 1/2\}$ **do**
  choose $k \in v$, such that $x_k \notin \{0, 1, 1/2\}$ (arbitrarily)
  if $x_k < 1/2$ **then**
    $p \leftarrow x_k$
  else
    $p \leftarrow 1 - x_k$
  end if
  $S \leftarrow \{i : x_i = p\}, S' \leftarrow \{i : x_i = 1 - p\}$
  $a \leftarrow \max\{x_i : x_i < p, 1 - x_i : x_i > 1 - p, 0\}$
  $b \leftarrow \min\{x_i : x_i > p, 1 - x_i : x_i < 1 - p, 1/2\}$
  $f^{(a)} \leftarrow \{x_i^{(a)} = a\}_{i \in S} \cup \{x_i^{(a)} = 1 - a\}_{i \in S'} \cup \{x_i^{(a)} = x_i\}_{i \in V \setminus (S \cup S')}$
  $f^{(b)} \leftarrow \{x_i^{(b)} = b\}_{i \in S} \cup \{x_i^{(b)} = 1 - b\}_{i \in S'} \cup \{x_i^{(b)} = x_i\}_{i \in V \setminus (S \cup S')}$
  if $\text{Val}(f^{(a)}) \leq \text{Val}(f^{(b)})$ **then**
    $f \leftarrow f^{(a)}$
  else
    $f \leftarrow f^{(b)}$
  end if
end while
return f (as f^*)

- If $i, j \in V \setminus (S \cup S')$, $g_{ij}(t)$ is a constant function.
- If $i \in S, j \in S'$, $g_{ij}(t) = t$.
- If $i \in S', j \in S$, $g_{ij}(t) = 1 - t$.
- If $i, j \in S$ (or $i, j \in S'$), $g(t) = \max(t, 1 - t)$. Since $a, b \leq \frac{1}{2}$ and $t \in [a, b]$, $g_{ij}(t) = 1 - t$.
- If $i \in S, j \in V \setminus (S \cup S')$, $g(t) = \max(t, 1 - x_j)$. If we plot all the x_i's and $1 - x_i'$s on the $[0, 1]$ line, a is the maximum value less than p and b is the minimum value greater than p. $\forall i \in V \setminus (S \cup S'), x_i \notin (a, b)$ and $1 - x_i \notin (a, b)$.Thus, depending on x_j, $g_{ij}(t)$ is either a constant or a linear function.
- If $i \in S', j \in V \setminus (S \cup S')$ (or $i \in V \setminus (S \cup S'), j \in S \cup S'$), we can show that $g_{ij}(t)$ is linear by using the same argument as above. ◀

▶ **Lemma 26.** *If the optimum value of the LP is at most ϵm, then the* Max DICUT *value of the corresponding graph is at least $\left(1 - \frac{3}{2}\epsilon\right) m$.*

Proof. Using Algorithm 2, we obtain a half-integral solution to the LP relaxation in polynomial time. This solution partitions the vertex set into three subsets. Let $A = \{i : x_i = 0\}$, $B = \{i : x_i = 1\}$ and $U = \{i : x_i = 1/2\}$. The solution assigns $z_e = 0$ for $e \in E(A \to B)$, $z_e = 1/2$ for $e \in E(U \to B) \cup E(A \to U) \cup E(U \to U)$ and $z_e = 1$ otherwise. If we round off each variable in U to either 0 or 1 with probability $1/2$, on expectation, at least half of $\{z_e\}$ that are assigned value $1/2$ currently become 0. This implies that there exists a rounding r which makes at most half of $\{z_e\}$ with value $1/2$ become 1 after that.

Since the LP optimum is at most ϵm, the number of $\{z_e\}$ that take value $1/2$ are at most $2\epsilon m$. After rounding r, the LP solution looks similar to the BIP solution. The increase in the objective value is at most $\frac{1}{2} \times 2\epsilon m \times \frac{1}{2} = \frac{\epsilon}{2} m$. Thus, the Max DICUT value of the graph is at least $\left(1 - \frac{3}{2}\epsilon\right) m$. ◀

Algorithm 3 A deterministic 1/2−approximation algorithm of Max DICUT.

Input: A directed graph $G = (V, E)$.
Solve the LP relaxation of the Max DICUT problem for G. Let t be the corresponding optimum value.
if $t \leq m/2$ **then**
    return $(m - 3t/2)$
else
    return $m/4$
end if

▶ **Theorem 27.** *Algorithm 3 is a deterministic polynomial time* 1/2−*approximation algorithm of* Max DICUT.

Proof. The running time of Algorithm 3 follows from the fact that any LP can be solved in deterministic polynomial time. If t is the optimum value returned by the LP relaxation, then the BIP optimum value is at least t. This implies that the Max DICUT value of the corresponding graph is at most $m - t$. Lemma 26 implies that the Max DICUT value is at least $(m - 3t/2)$. When $t \leq m/2$, the algorithm returns $(m - 3t/2)$ as the Max DICUT value. In this case, the approximation ratio is $(m - 3t/2)/(m - t) \geq 1/2$. When $t > m/2$, the Max DICUT value is at most $m/2$. Since the algorithm outputs $m/4$ in this case, the approximation ratio is 1/2. ◀