SIAM J. DISCRETE MATH. © 2017 Society for Industrial and Applied Mathematics
Vol. 31, No. 2, pp. 1263–1276

SPARSIFICATION OF TWO-VARIABLE VALUED CONSTRAINT SATISFACTION PROBLEMS[*]

ARNOLD FILTSER[†] AND ROBERT KRAUTHGAMER[‡]

Abstract. A valued constraint satisfaction problem (VCSP) instance (V, Π, w) is a set of variables V with a set of constraints Π weighted by w. Given a VCSP instance, we are interested in a reweighted subinstance $(V, \Pi' \subset \Pi, w')$ that preserves the value of the given instance (under every assignment to the variables) within factor $1 \pm \epsilon$. A well-studied special case is cut sparsification in graphs, which has found various applications. We show that a VCSP instance consisting of a single boolean predicate $P(x, y)$ (e.g., for cut, $P = \mathsf{XOR}$) can be sparsified into $O(|V|/\epsilon^2)$ constraints iff the number of inputs that satisfy P is anything but one (i.e., $|P^{-1}(1)| \neq 1$). Furthermore, this sparsity bound is tight unless P is a relatively trivial predicate. We conclude that also systems of 2SAT (or 2LIN) constraints can be sparsified.

Key words. valued constraint satisfaction problem, cut sparsification, boolean predicates, MAX-CSP

AMS subject classifications. 68Q25, 68W25

DOI. 10.1137/15M1046186

1. Introduction.

The seminal work of Benczúr and Karger [4] showed that every edge-weighted undirected graph $G = (V, E, w)$ admits cut sparsification within factor $(1+\epsilon)$ using $O(\epsilon^{-2} n \log n)$ edges, where we denote throughout $n = |V|$. To state it more precisely, assume that edge weights are always non negative and let $\mathsf{Cut}_G(S)$ denote the total weight of edges in G that have exactly one endpoint in S. Then for every such G and $\epsilon \in (0, 1)$, there is a reweighted subgraph $G_\epsilon = (V, E_\epsilon \subseteq E, w_\epsilon)$ with $|E_\epsilon| \leq O(\epsilon^{-2} n \log n)$ edges such that

$$(1) \qquad \forall S \subset V, \qquad \mathsf{Cut}_{G_\epsilon}(S) \in (1 \pm \epsilon) \cdot \mathsf{Cut}_G(S),$$

and moreover, such G_ϵ can be computed efficiently.

This sparsification methodology turned out to be very influential. The original motivation was to speed up algorithms for cut problems—one can compute a cut sparsifier of the input graph and then solve an optimization problem on the sparsifier—and indeed this has been a tremendously effective approach; see, e.g., [4, 5, 10, 14, 12]. Another application of this remarkable notion is to reduce space requirements, either when storing the graph or in streaming algorithms [1]. In fact, followup work offered several refinements, improvements, and extensions (such as to spectral sparsification or to cuts in hypergraphs, which in turn have more applications); see, e.g., [16, 17, 15, 7, 8, 9, 13, 3, 11]. The current bound for cut sparsification is $O(n/\epsilon^2)$ edges, proved by Batson, Spielman, and Srivastava [3], and it is known to be tight [2].

We study the analogous problem of sparsifying constraint satisfaction problems (CSPs), which was raised in [11, section 4] and goes as follows. Given a set of

[*]Received by the editors October 30, 2015; accepted for publication (in revised form) January 27, 2017; published electronically June 22, 2017. A preliminary version is available at arXiv:1509.01844.
http://www.siam.org/journals/sidma/31-2/M104618.html

Funding: The first author was partially supported by the Lynn and William Frankel Center for Computer Sciences. The second author's work was supported in part by Israel Science Foundation grant 897/13 and US-Israel BSF grant 2010418.

[†]Ben-Gurion University of the Negev, Beer-Sheva 8410501, Israel (arnoldf@cs.bgu.ac.il).
[‡]Weizmann Institute of Science, Rehovot 76100, Israel (robert.krauthgamer@weizmann.ac.il).

constraints on n variables, the goal is to construct a sparse subinstance that has approximately the same value as the original instance under *every possible assignment*; see section 2 for a formal definition. Such sparsification of CSPs can be used to reduce storage space and running time of many algorithms.

We restrict our attention to two-variable constraints (i.e., of arity 2) over boolean domain (i.e., alphabet of size 2). To simplify matters even further we shall start with the case where all the constraints use the same predicate $P : \{0,1\}^2 \rightarrow \{0,1\}$. This restricted case of CSP sparsification already generalizes cut sparsification—simply represent every vertex $v \in V$ by a variable x_v and every edge $(v,u) \in E$ by the constraint $x_v \neq x_u$.

Observe that such CSPs also capture other interesting graph problems, such as the *uncut edges* (using the predicate $x_v = x_u$), *covered edges* (using the predicate $x_v \vee x_u$), or *directed-cut edges* (using the predicate $x_v \wedge \neg x_u$). Even though these graph problems are well-known and extensively studied, we are not aware of any sparsification results for them, and at a first glance such sparsification may even seem surprising, because these problems do not have the combinatorial structure exploited by [4] (a bound on the number of approximately minimum cuts) or the linear-algebraic description used by [15, 3] (as quadratic forms over Laplacian matrices).

Results. For CSPs consisting of a single predicate $P : \{0,1\}^2 \rightarrow \{0,1\}$, we show in Theorem 3.7 that a $(1+\epsilon)$-sparsifier of size $O(n/\epsilon^2)$ always exists iff $|P^{-1}(1)| \neq 1$ (i.e., P has 0, 2, 3, or 4 satisfying inputs). Observe that the latter condition includes the two graphical examples above uncut edges and covered edges but excludes directed-cut edges. We further show in Theorem 4.1 that our sparsity bound above is tight, except for some relatively trivial predicates P. We then build on our sparsification result in section 5 to obtain $(1+\epsilon)$-sparsifiers for other CSPs, including 2SAT (which uses four predicate types) and 2LIN (which uses two predicate types).

Finally, we explore future directions, such as more general predicates and a generalization of the sparsification paradigm to sketching schemes. In particular, we see that the above dichotomy according to number of satisfying inputs to the predicate extends to sketching.

2. Two-variable boolean predicates and digraphs.

A *predicate* is a function $\mathsf{P} : \{0,1\}^2 \rightarrow \{0,1\}$ (recall we restrict ourselves throughout to two variables and a boolean domain). Given a set of variables V, a *constraint* $\langle (v,u), \mathsf{P} \rangle$ consists of a predicate P and an ordered pair (v,u) of variables from V. For an assignment $A : V \rightarrow \{0,1\}$, we say that A *satisfies* the constraint whenever $\mathsf{P}(A(v), A(u)) = 1$. A valued constraint satisfaction problem (VCSP) instance $\mathcal{I}$ is a triple (V, Π, w), where V is a set of variables, Π is a set of constraints over V (each of the form $\pi_i = \langle (v_i, u_i), p_i \rangle$), and $w : \Pi \rightarrow \mathbb{R}_+$ is a weight function. The *value* of an assignment $A : V \rightarrow \{0,1\}$ is the total weight of the satisfied constraints, i.e.,

$$\mathrm{Val}_{\mathcal{I}}(A) := \sum_{\pi_i \in \Pi} w(\pi_i) \cdot p_i(A(v_i), A(u_i)).$$

For $\epsilon \in (0,1)$, an *ϵ-sparsifier* of $\mathcal{I}$ is a (reweighted) subinstance $\mathcal{I}_\epsilon = (V, \Pi_\epsilon \subseteq \Pi, w_\epsilon)$ where

$$\forall A : V \rightarrow \{0,1\}, \qquad \mathrm{Val}_{\mathcal{I}_\epsilon}(A) \in (1 \pm \epsilon) \cdot \mathrm{Val}_{\mathcal{I}}(A).$$

The goal is to minimize the number of constraints, i.e., $|\Pi_\epsilon|$. There are 16 different predicates $\mathsf{P} : \{0,1\}^2 \rightarrow \{0,1\}$, which are listed in Table 1 with names for easy reference.

TABLE 1

All possible predicates $\mathsf{P} : \{0,1\}^2 \to \{0,1\}$*, where blank cells denote value* 0*. Predicates* $0x, x0, x1, 1x$ *are determined by a single variable. Predicates* $01, \mathsf{Dicut}, \overline{10}, \overline{01}$ *are satisfied by a single assignment or all but a single one.*

x_1	x_2	$\vec{0}$	nOr	01	$0x$	Dicut	$x0$	Cut	nAnd	And	unCut	$x1$	$\overline{10}$	$1x$	$\overline{01}$	Or	$\vec{1}$
0	0		1		1		1		1		1		1		1		1
0	1			1	1			1	1			1	1			1	1
1	0					1	1	1	1					1	1	1	1
1	1									1	1	1	1	1	1	1	1

We first focus on the case where all the constraints in Π use the same predicate P[1], in which case we can represent the VCSP $\mathcal{I}$ by an edge-weighted digraph $G^{\mathcal{I}} = (V, E, w)$. Each variable in V is represented by a vertex, and each constraint over the pair (v, u) will be represented by a directed edge from v to u, with the same weight as the constraint (formally, $E = \{(v, u) \mid (\langle v, u\rangle, \mathsf{P}) \in \Pi\}$, and abusing notation set edge weights $w(v, u) = w(\langle(v, u), P\rangle))$. This transformation preserves all the information about the VCSP and allows us to make reductions between VCSPs with different predicates P as their sole predicate.

Given a digraph G, a predicate P and a subset $S \subseteq V$, define

$$\mathsf{P}_G(S) := \sum_{(v,u)\in E} \mathsf{P}(\mathbf{1}_S(v), \mathbf{1}_S(u)) \cdot w((v, u)),$$

where $\mathbf{1}_S$ denotes the indicator function. For example, applying this definition to the cut predicate $\mathsf{Cut} : (x, y) \to \mathbf{1}_{\{x \neq y\}}$, we have

$$\mathsf{Cut}_G(S) = \sum_{(v,u)\in E} \mathsf{Cut}(\mathbf{1}_S(v), \mathbf{1}_S(u)) \cdot w((v, u)) = \sum_{(v,u)\in E} |\mathbf{1}_S(v) - \mathbf{1}_S(u)| \cdot w((v, u)),$$

which is just the total weight of the edges crossing the cut S. This matches the definition we gave in the introduction, except for the technical subtlety that G is now a directed graph, which makes no difference for symmetric predicates like Cut. We shall assume henceforth that G is directed.

We shall say that a subinstance G_ϵ is an *ϵ-P-sparsifier* of G if

$$\forall S \subseteq V, \qquad \mathsf{P}_{G_\epsilon}(S) \in (1 \pm \epsilon) \cdot \mathsf{P}_G(S).$$

Observe that given an assignment A for the variables V, we can set $S_A := \{u \mid A(u) = 1\}$. It then holds that $\mathrm{Val}_{\mathcal{I}}(A) = \mathsf{P}_{G^{\mathcal{I}}}(S_A)$, where $G^{\mathcal{I}}$ is the appropriate digraph for the VCSP. As there exists a bijection between such VCSPs and digraphs, we conclude as follows.

Observation 2.1. The existence of an ϵ-P-*sparsifier* $G_\epsilon = (V, E_\epsilon, w_\epsilon)$ for $G^{\mathcal{I}}$ implies the existence of an ϵ-sparsifier $\mathcal{I}_\epsilon$ for $\mathcal{I}$ with $|E_\epsilon|$ constraints.

Note that the converse is true as well, i.e., an ϵ-sparsifier for $\mathcal{I}$ implies the existence of an ϵ-P-sparsifier for $G_{\mathcal{I}}$ of size $|\Pi_\epsilon|$. From now on, we focus on finding an ϵ-P-sparsifier for an arbitrary digraph G (for different choices of the predicate P).

[1]The collection of predicates used in a VCSP is sometimes called its *signature*. In this paper we mainly deal with VCSPs whose signature is of size one.

3. A single predicate.

In this section we go over all the predicates $\mathsf{P} : \{0,1\}^2 \to \{0,1\}$ and classify them into sparsifiable and nonsparsifiable predicates; see Theorems 3.5, 3.6, and 3.7. For simplicity, we state our sparsification results as existential, but in fact all these sparsifiers can be computed in polynomial time.

Our main technique is a graph transformation, which is well-known but apparently only in very different contexts. On the face of it, it is not clear which predicates other than Cut do admit nontrivial sparsification. For example, the uncut edges in a graph do not satisfy a key property of cuts that was used in [4] for cut-sparsification (namely, a polynomial bound on the number of near-minimum cuts in a graph), and it is not clear a priori which edges must be included in every sparsifier (again in analogy with cuts, where all bridge edges must be retained), These deficiencies suggest that the edge-sampling approach, which is very effective for cuts [4, 15, 8], would fail for other predicates and may further be viewed as evidence for the impossibility of sparsification. Thus, we were surprised to find out that different predicates can all be analyzed using one simple graph transformation, which appears easy in retrospect and provides a unifying explanation.

In our classification, we appeal to two basic predicates, the first of which is Cut, which is already known to be sparsifiable.

THEOREM 3.1 (see [3]). *For every digraph G and parameter $\epsilon \in (0,1)$, there is an ϵ-Cut-sparsifier for G with $O\left(|V|/\epsilon^2\right)$ edges.*

Our second basic predicate is the predicate And, which behaves significantly differently. We call a digraph $G = (V,E)$ *strongly asymmetric* if for every $(v,u) \in E$ it holds that $(u,v) \notin E$.

THEOREM 3.2. *For every strongly asymmetric digraph $G = (V,E,w)$ with strictly positive weights and $\epsilon \in (0,1)$, every ϵ-And-sparsifier $G_\epsilon = (V,E_\epsilon,w_\epsilon)$ must satisfy $E_\epsilon = E$.*

Proof. Let $G_\epsilon = (V,E_\epsilon,w_\epsilon)$ be such a sparsifier, i.e., for every $S \subseteq V$ it holds that $\mathsf{And}_{G_\epsilon}(S) \in (1 \pm \epsilon) \cdot \mathsf{And}_G(S)$. Then for every $e = (v,u) \in E$ we must have $(v,u) \in E_\epsilon$, as otherwise for the set $S = \{v,u\}$ it will hold that $\mathsf{And}_{G_\epsilon}(\{v,u\}) = 0$ while $\mathsf{And}_G(\{v,u\}) = w(e) > 0$, a contradiction. ∎

Remark 3.3. For every digraph (which is not necessarily strongly asymmetric), the same proof shows that $|E_\epsilon| \geq \frac{1}{2}|E|$.

Remark 3.4. Our definition of an ϵ-P-sparsifier requires G_ϵ to be a subgraph of G, but we can state Theorem 3.2 in a more general way: For every digraph $G_\epsilon = (V,E_\epsilon,w_\epsilon)$ (not necessarily a subgraph) such that every $S \subseteq V$ satisfies $\mathsf{And}_{G_\epsilon}(S) \in (1 \pm \epsilon) \cdot \mathsf{And}_G(S)$ necessarily E_ϵ agrees with E up to the directions of the edges.

Next, we show that every other predicate is similar either to Cut or to And in terms of sparsifability. We describe a reduction that will be useful to show both sparsifability and nonsparsifability. (This reduction is based on a well-known transformation of a given graph, called the "bipartite double cover" (see, e.g., [6]), although we are not aware of its use in the same way.) Let γ be a function that maps a digraph $G = (V,E,w)$ where $V = \{v_1, v_2, \ldots, v_n\}$ to a digraph $\gamma(G) = (V^\gamma, E^\gamma, w^\gamma)$ where $V^\gamma = \{v_{-n}, \ldots, v_{-1}, v_1, \ldots, v_n\}$, $E^\gamma = \{(v_i, v_{-j}) \mid (v_i, v_j) \in E\}$, $w^\gamma((v_i, v_{-j})) = w((v_i, v_j))$. For every subset $S \subseteq V$, we introduce the notation $-S := \{v_{-i} \mid v_i \in S\}$, $\bar{S} := \{v_i \mid v_i \in V \setminus S\}$ and $-\bar{S} := \{v_{-i} \mid v_i \in V \setminus S\}$. Figure 1 illustrates the effect of γ on an arbitrary set S.

THEOREM 3.5. *For every digraph $G = (V,E,w)$ and $\epsilon \in (0,1)$ there is a subdigraph G_ϵ with $O(|V|/\epsilon^2)$ edges such that for every predicate $\mathsf{P} \in$*

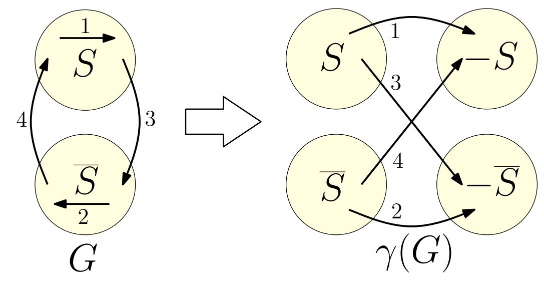

FIG. 1. *The mapping γ applied on G and its effect on an arbitrary $S \subseteq V$. For example, an edge from $v_i \in S$ to $v_j \in \bar{S}$ is represented by an arrow of type 3 and becomes in $\gamma(G)$ an edge from $v_i \in S$ to $v_{-j} \in -\bar{S}$.*

$\{\mathit{Cut}, \mathit{unCut}, \mathit{Or}, \mathit{nAnd}, \overline{10}, \overline{01}, x0, x1, 0x, 1x, \vec{1}, \vec{0}\}$, *the digraph G_ϵ is an ϵ-P-*sparsifier *of G. (Note that G_ϵ does not depend on P.)*

Proof. Given G and ϵ, first construct $\gamma(G)$ as above. Next, apply Theorem 3.1 to obtain for $\gamma(G)$ a cut sparsifier $\gamma(G)_\epsilon = (V^\gamma, E^\gamma_\epsilon \subseteq E^\epsilon, w^\gamma_\epsilon)$, which contains $O(|V^\gamma|/\epsilon^2) = O(|V|/\epsilon^2)$ edges. Now construct a digraph $G_\epsilon = (V, E_\epsilon, w_\epsilon)$ where $E_\epsilon = \{(v_i, v_j) \mid (v_i, v_{-j}) \in E^\gamma_\epsilon\}$ and $w_\epsilon(v_i, v_j) = w^\gamma_\epsilon(v_i, v_{-j})$. Observe that $\gamma(G_\epsilon) = \gamma(G)_\epsilon$, i.e., if we apply γ on G_ϵ we get exactly $\gamma(G)_\epsilon$.

Now suppose that for a predicate P, there is a function $f_P : 2^V \to 2^{V^\gamma}$ such that for every digraph H on the vertex set V, it holds that

$$\forall S \subset V, \qquad \mathsf{P}_H(S) = \mathsf{Cut}_{\gamma(H)}(f_P(S)). \tag{2}$$

Then we could apply (2) twice, first to G_ϵ and then to G, and obtain that

$$\forall S \subset V, \qquad \mathsf{P}_{G_\epsilon}(S) = \mathsf{Cut}_{\gamma(G)_\epsilon}(f_P(S)) \in (1 \pm \epsilon) \cdot \mathsf{Cut}_{\gamma(G)}(f_P(S)) = (1 \pm \epsilon) \cdot \mathsf{P}_G(S).$$

Hence, the existence of such a function f_P implies that G_ϵ is an ϵ-P-sparsifier. And indeed, we can show such f_P for some predicates P, as follows:

- $f_{\mathsf{unCut}}(S) = S \cup -\bar{S}$;
- $f_{\mathsf{Cut}}(S) = S \cup -S$;
- $f_{0x}(S) = \bar{S}$;
- $f_{x0}(S) = -\bar{S}$;
- $f_{x1}(S) = -S$;
- $f_{1x}(S) = S$;
- $f_{\vec{1}}(S) = S \cup \bar{S}$; and
- $f_{\vec{0}}(S) = \emptyset$.

To verify that $f_{\mathsf{unCut}}(S) = S \cup -\bar{S}$ satisfies Equation 2, i.e., that $\mathsf{unCut}_H(S) = \mathsf{Cut}_{\gamma(H)}(S \cup \bar{S})$, observe that both sides consist exactly of the edges of types 1 and 2 in Figure 1. The other predicates can be easily verified similarly, which completes the proof for all $\mathsf{P} \in \{\mathsf{Cut}, \mathsf{unCut}, 0x, x0, x1, 1x, \vec{1}, \vec{0}\}$.

To show that G_ϵ is a sparsifier also for predicates $\mathsf{P} \in \{\mathsf{Or}, \mathsf{nAnd}, \overline{10}, \overline{01}\}$ we need a slightly more general argument. Suppose that for a predicate P, there are functions $f^1_P, f^2_P, f^3_P : 2^V \to 2^{V^\gamma}$ such that for every digraph H on the vertex set V,

$$\mathsf{P}_H(S) = \tfrac{1}{2}\left[\mathsf{Cut}_{\gamma(H)}(f^1_P(S)) + \mathsf{Cut}_{\gamma(H)}(f^2_P(S)) + \mathsf{Cut}_{\gamma(H)}(f^3_P(S))\right]. \tag{3}$$

Then we could apply (3) twice, first to G_ϵ and then to G, and obtain that

$$\begin{aligned}\mathsf{P}_{G_\epsilon}(S) &= \tfrac{1}{2}\left[\mathsf{Cut}_{\gamma(G)_\epsilon}(f_P^1(S)) + \mathsf{Cut}_{\gamma(G)_\epsilon}(f_P^2(S)) + \mathsf{Cut}_{\gamma(G)_\epsilon}(f_P^3(S))\right]\\ &\in (1\pm\epsilon)\cdot\tfrac{1}{2}\left[\mathsf{Cut}_{\gamma(G)}(f_P^1(S)) + \mathsf{Cut}_{\gamma(G)}(f_P^2(S)) + \mathsf{Cut}_{\gamma(G)}(f_P^3(S))\right]\\ &= (1\pm\epsilon)\cdot\mathsf{P}_G(S).\end{aligned}$$

Hence, the existence of three such functions will imply that G_ϵ is an ϵ-P-sparsifier. And indeed, we let

- $f^1_{\mathsf{Or}}(S) = S$, $f^2_{\mathsf{Or}}(S) = -S$, $f^3_{\mathsf{Or}}(S) = S \cup -S$;
- $f^1_{\mathsf{nAnd}}(S) = \bar{S}$, $f^2_{\mathsf{nAnd}}(S) = -\bar{S}$, $f^3_{\mathsf{nAnd}}(S) = \bar{S} \cup -\bar{S}$;
- $f^1_{\overline{10}}(S) = \bar{S}$, $f^2_{\overline{10}}(S) = -S$, $f^3_{\overline{10}}(S) = \bar{S} \cup -S$; and
- $f^1_{\overline{01}}(S) = S$, $f^2_{\overline{01}}(S) = -\bar{S}$, $f^3_{\overline{01}}(S) = S \cup -\bar{S}$.

To verify that $f^1_{\mathsf{Or}}, f^2_{\mathsf{Or}}, f^3_{\mathsf{Or}}$ satisfies (3), observe that both sides consist exactly of the edges of types $1, 3, 4$ in Figure 1. The other predicates can be easily verified similarly, which completes the proof for all $\mathsf{P} \in \{\mathsf{Or}, \mathsf{nAnd}, \overline{10}, \overline{01}\}$. ∎

Next, we use γ for a reduction from And to all the remaining predicates. In particular it will imply their "resistance to sparsification."

THEOREM 3.6. *Given parameters n and $m \le \binom{n}{2}$, there is a digraph $G = (V, E, w)$ with $2n$ vertices and m edges such that for every $\epsilon \in (0,1)$ and every predicate $\mathsf{P} \in \{\mathsf{nOr}, 01, \mathsf{Dicut}, \mathsf{And}\}$, for every ϵ-P-sparsifier $G_\epsilon = (V, E_\epsilon, w_\epsilon)$ of G it holds that that $E_\epsilon = E$. (Note that G does not depend on P.)*

Proof. Let $G = (V, E, w)$ be an arbitrary strongly asymmetric digraph with n vertices, m edges, and strictly positive weights. Let $\gamma(G)$ be the digraph constructed by our reduction. Note that $\gamma(G)$ consist of $2n$ vertices and m edges. $\gamma(G)$ will be the digraph for which we will prove the theorem.

Fix some predicate P. Let $\gamma(G)_\epsilon = (V^\gamma, E^\gamma_\epsilon \subseteq E_\epsilon, w^\gamma_\epsilon)$ be some ϵ-P-sparsifier for $\gamma(G)$. Let $G_\epsilon = (V, E_\epsilon, w_\epsilon)$ be a digraph where $E_\epsilon = \{(v_i, v_j) \mid (v_i, v_{-j}) \in E^\gamma_\epsilon\}$ and $w_\epsilon((v_i, v_j)) = w^\gamma_\epsilon((v_i, v_{-j}))$. Note that $\gamma(G_\epsilon) = \gamma(G)_\epsilon$.

Now suppose that there is a function $f_P : 2^V \to 2^{V^\gamma}$ such that for every digraph H on the vertex set V, it holds that

$$\forall S \subset V, \qquad \mathsf{And}_H(S) = \mathsf{P}_{\gamma(H)}(f_P(S)). \tag{4}$$

Then we could apply (4) twice, first to G_ϵ and then to G, and obtain that

$$\forall S \subset V, \qquad \mathsf{And}_{G_\epsilon}(S) = \mathsf{P}_{\gamma(G)_\epsilon}(f_P(S)) \in (1\pm\epsilon)\cdot \mathsf{P}_{\gamma(G)}(f_P(S)) = (1\pm\epsilon)\cdot\mathsf{And}_G(S).$$

Hence, assuming such a function f exists, G_ϵ is an ϵ-And-sparsifier for G. According to Theorem 3.2, necessarily $E_\epsilon = E$, and in particular $E^\gamma_\epsilon = E^\gamma$.

Hence, the existence of such functions f_P for all $\mathsf{P} \in \{\mathsf{nOr}, 01, \mathsf{Dicut}, \mathsf{And}\}$ will imply our theorem. And indeed, we let

- $f_{And}(S) = S \cup -S$;
- $f_{nOr}(S) = \bar{S} \cup -\bar{S}$;
- $f_{Dicut}(S) = S \cup -\bar{S}$; and
- $f_{01}(S) = \bar{S} \cup -S$.

To verify that $f_{Dicut}(S) = S \cup -\bar{S}$ satisfies (4), observe that both sides consist exactly of the edges of type 1 in Figure 1. The other predicates can be easily verified similarly. ∎

We conclude our main theorem, which basically puts together Theorems 3.5 and 3.6.

THEOREM 3.7. *Let P be a binary predicate, and let $\epsilon \in (0,1)$ be some parameter.*

- *If P has a single "1" in its truth table, then there exist a VCSP $\mathcal{I} = (V, \Pi, w)$ with a single predicate P such that every ϵ-P-sparsifier of $\mathcal{I}$ will have $\Omega(|V|^2)$ constraints.*
- *If P does not has a single "1" in its truth table, then for every VCSP $\mathcal{I} = (V, \Pi, w)$ with single predicate P, there exists an ϵ-P-sparsifier with $O\left(|V|/\epsilon^2\right)$ constraints.*

4. Lower bounds (for a single predicate). In this section we will show that Theorem 3.5 is tight. More precisely, we will show that for every $\mathsf{P} \in \{\mathsf{Cut}, \mathsf{unCut}, \mathsf{Or}, \mathsf{nAnd}, \overline{10}, \overline{01}\}$, there exists an n-vertex graph G such that every ϵ-P-sparsifier G_ϵ of G must contain $\Omega(n/\epsilon^2)$ edges.[2] The first step was done by [2], who showed that Theorem 3.1 is tight, i.e., for every n and $\epsilon \in (1/\sqrt{n}, 1)$, there exists an n-vertex graph G such that every ϵ-Cut-sparsifier G_ϵ of G must contain $\Omega(n/\epsilon^2)$ edges. Using our reduction γ in a similar manner to Theorem 3.5, this lower bound can be extended to unCut based on the fact that $\mathsf{Cut}_G(S) = \mathsf{unCut}_{\gamma(G)}\left(S \cup -\bar{S}\right)$. However, γ fails to extend the lower bound to predicates with three 1's in their truth table. To this end, we will define sketching schemes, a variation of sparsification where the goal is to maintain the approximate value of every assignment using a small data structure, possibly without any combinatorial structure; see the definition below. We will use a lower bound on the sketch-size of Cut from [2] to prove the lower bound on the number of edges in a sparsifier (and also on the sketch-size) for Or. The extension to other predicates with three 1's in their truth table is straightforward using γ. Sketching is interesting on its own, and we have further discussion and lower bounds regarding sketching in section 6.3.

Formally, a *sketching scheme* (or a *sketch* in short) is a pair of algorithms $(\mathrm{sk}, \mathrm{est})$. Given a weighted digraph $G = (V, E, w)$ and a predicate P, algorithm sk returns a string sk_G (intuitively, a short encoding of the instance). Given $\mathrm{sk}_\mathcal{I}$ and a subset $S \subseteq V$, algorithm est returns a value (without looking at G) that estimates $\mathsf{P}_G(S)$. We say that it is an ϵ-P-*sketching-scheme* if for every digraph G, and for every subset $S \subseteq V$, $\mathrm{est}(\mathrm{sk}_G, S) \in (1 \pm \epsilon) \cdot \mathsf{P}_G(S)$. The *sketch-size* is $\max_G |\mathrm{sk}_G|$, the maximum length of the encoding string over all the digraphs with n variables, often measured in bits. *sk* might be probabilistic algorithm, but for our purposes it is enough to think only about the deterministic case. Note that an algorithm for constructing ϵ-sparsifiers always provides an ϵ-sketching-scheme, where the sketch-size is asymptotically equal to the number of constraints in the constructed sparsifiers when measured in machine words (and up to logarithmic factors when measured in bits). Sparsification is advantageous over general sketching as it preserves the combinatorial structure of the problem. Nevertheless, one may be interested in constructing sketches as they may potentially require significantly smaller storage.

THEOREM 4.1. *Fix a predicate $\mathsf{P} \in \{\mathsf{Cut}, \mathsf{unCut}, \mathsf{Or}, \mathsf{nAnd}, \overline{10}\}$, an integer n, and $\epsilon \in (1/\sqrt{n}, 1)$. The sketch-size of every ϵ-P-sketching-scheme on n variables is $\Omega(n/\epsilon^2)$. Moreover, there is an n-vertex digraph G, such that every ϵ-P-sparsifier of G has $\Omega(n/\epsilon^2)$ edges.*

[2]The other predicates $\{x0, x1, 0x, 1x, \vec{1}, \vec{0}\}$ are kind of trivial in the sense of sparsification. $\vec{0}$ sparsified by the empty graph. $\vec{1}$ can be sparsified using a single edge. $\{x0, x1, 0x, 1x\}$ could be sparsified using n edges.

Proof. We follow the line-of-proof of Theorems 2.3 and 2.4 in [2]. Specifically, they show that the sketch-size of every ϵ-Cut-sketching-scheme is $\Omega(n/\epsilon^2)$ bits, by proving that a certain family $\mathcal{F}$ of n-vertex graphs is hard to sketch and consequently to sparsify. By similar arguments to Theorem 3.5, this lower bound easily extends to unCut. Indeed, recall that $\mathsf{Cut}_G(S) = \mathsf{unCut}_{\gamma(G)}\left(S \cup -\bar{S}\right)$, and thus a ϵ-unCut-sparsifier (or sketch) for $\gamma(G)$ yields an ϵ-Cut-sparsifier (or sketch) for G with the same number of edges (size).

Once we prove the lower bound for predicate Or, a reduction from Or using γ will extend it also to nAnd, $\overline{10}$ and $\overline{01}$, because

$$\mathsf{Or}_G(S) = \mathsf{nAnd}_{\gamma(G)}(\bar{S} \cup -\bar{S}) = \overline{01}_{\gamma(G)}(S \cup -\bar{S}) = \overline{10}_{\gamma(G)}(\bar{S} \cup -S). \tag{5}$$

We will thus focus on the predicate Or. As it is a symmetric predicate, we can work with graphs rather then digraphs. The main observation in our proof is that for every undirected graph $G = (V, E, w)$, if $\deg_G(v)$ denotes the degree of vertex v, then

$$\forall S \subset V, \qquad \mathsf{Cut}_G(S) = 2 \cdot \mathsf{Or}_G(S) - \sum_{v \in S} \deg_G(v). \tag{6}$$

The graph family $\mathcal{F}$ consists of graphs G constructed as follows. Let $s_1, \dots, s_{n/2} \in \{0,1\}^{1/\epsilon^2}$ be balanced $1/\epsilon^2$ bit-strings (i.e., each s_i has normalized Hamming weight exactly $1/2$), and let the graph G be a disjoint union of the graphs $\{G_j \mid j \in [\epsilon^2 n/2]\}$, where each G_j is a bipartite graph, whose two sides, each of size $1/\epsilon^2$, are denoted $L(G_j)$ and $R(G_j)$. The edges of G are determined by $s_1, \dots, s_{n/2}$, where each bit string s_i is indicates the adjacency between vertex $i \in \cup_j L(G_j)$ and the vertices in the respective $R(G_j)$. They further observe (in the proof of [2, Theorem 2.4]) that the lower bound holds even if the sketching scheme is relaxed as follows:

1. The estimation is required only for cut queries contained in a single G_j, namely, cut queries $S \cup T$, where $S \subset L(G_j)$ and $T \subset R(G_j)$ for the same j.
2. The estimation achieves additive error μ/ϵ^3, where $\mu = 10^{-4}$ (instead of multiplicative error $1 \pm \epsilon$).

To prove a sketch-size lower bound for a $(\mu\epsilon)$-Or-sketching-scheme $(\mathrm{sk}^{\mathsf{Or}}, \mathrm{est}^{\mathsf{Or}})$, we assume it has sketch-size $s = s(n, \epsilon)$ bits and use it to construct a Cut-sketching-scheme $(\mathrm{sk}^{\mathsf{Cut}}, \mathrm{est}^{\mathsf{Cut}})$ that achieves the estimation properties 1 and 2 on graphs of the aforementioned form and has sketch-size $s + 2n\log(1/\epsilon)$ bits. Then by [2], this sketch-size must be $\Omega(n/\epsilon^2)$, and we conclude that $s = \Omega(n/\epsilon^2)$ as required.

Given a graph $G \in \mathcal{F}$, let $\mathrm{sk}_G^{\mathsf{Cut}}$ be a concatenation of $\mathrm{sk}_G^{\mathsf{Or}}$ and a list of all vertex degrees in G. The degrees in G are bounded by $1/\epsilon^2$, hence the size of $\mathrm{sk}_G^{\mathsf{Cut}}$ is indeed $s + 2n\log(1/\epsilon)$ bits. Given a cut query $S \cup T$ contained in some G_j, define the estimation algorithm (which we now construct for Cut) to be

$$\mathrm{est}^{\mathsf{Cut}}(\mathrm{sk}_G^{\mathsf{Cut}}, S \cup T) := 2 \cdot \mathrm{est}^{\mathsf{Or}}(\mathrm{sk}_G^{\mathsf{Or}}, S \cup T) - \sum_{v \in S \cup T} \deg_G(v). \tag{7}$$

Let us analyze the error of this estimate. First, observe that as in each G_j there are precisely $\frac{1}{2\epsilon^4}$ edges, $\mathsf{Or}_G(S \cup T) \le \frac{1}{2\epsilon^4}$, and thus

$$\mathrm{est}^{\mathsf{Or}}(\mathrm{sk}_G^{\mathsf{Or}}, S \cup T) \in (1 \pm \mu\epsilon) \cdot \mathsf{Or}_G(S \cup T) \subseteq \mathsf{Or}_G(S \cup T) \pm \frac{\mu}{2\epsilon^3}\ .$$

Plugging this estimate into (7) and then recalling our initial observation (6), we obtain as desired

$$\begin{aligned}\mathrm{est}^{\mathsf{Cut}}(\mathrm{sk}_G^{\mathsf{Cut}}, S \cup T) &\in 2 \cdot \mathsf{Or}_G(S \cup T) \pm \frac{\mu}{\epsilon^3} - \sum_{v \in S \cup T} \deg_G(v) \\ &= \mathsf{Cut}_G(S \cup T) \pm \frac{\mu}{\epsilon^3} \ .\end{aligned}$$

To prove a lower bound on the size of an Or-sparsifier, we follow the argument in [2, Theorem 2.4], which shows that given an ϵ-Cut-sparsifier G_ϵ with $s = s(n, \epsilon)$ edges for a graph $G \in \mathcal{F}$, there is a Cut-sparsifier G_μ of G_ϵ, with additive error $\mu/2\epsilon^3$, such that G_μ has only integer weights and henceforth can be encoded using $O(s(\mu^{-2} + \log(\epsilon^{-2}n/s)))$ bits. In fact, there is nothing special here about Cut. The same proof will work (with the same properties) for predicate Or, assuming a sparsifier is required to be a subgraph (to remove this restriction, just erase all the edges between G_j to G_i for $i \neq j$, which adds only a small additive error).

Now suppose that every graph G of the form specified above admits a $\frac{\mu}{2}\epsilon$-Or-sparsifier G_ϵ with s edges. Then as explained above (about repeating the argument of [2]) there is a graph G_μ that sparsifies G_ϵ with additive error $\mu/2\epsilon^3$ and can be encoded by a string $\mathcal{I}_G$ of size $O(s \log(\epsilon^{-2}n/s))$ bits (recall that μ is a constant). Use it to construct a Cut-sketching-scheme with additive error μ/ϵ^3 as follows. Given the graph G, set $\mathrm{sk}_G^{\mathsf{Cut}}$ to be the concatenation of $\mathcal{I}_G$ and a list of the degrees of all the vertices in G. Then $|\mathcal{I}_G| = O(s \log(\epsilon^{-2}n/s)) + 2n \log(1/\epsilon)$. For a cut query $S \cup T$ contained in some G_j, define the estimation algorithm (using the Or sparsifier) to be

$$\mathrm{est}^{\mathsf{Cut}}(\mathrm{sk}_G^{\mathsf{Cut}}, S \cup T) := 2 \cdot \mathsf{Or}_{G_\mu}(S \cup T) - \sum_{v \in S \cup T} \deg_G(v).$$

Then we can again analyze it by plugging the above error bounds and then using (6),

$$\begin{aligned}\mathrm{est}^{\mathsf{Cut}}(\mathrm{sk}_G^{\mathsf{Cut}}, S \cup T) &\in 2 \cdot \mathsf{Or}_{G_\epsilon}(S \cup T) \pm \frac{\mu}{2\epsilon^3} - \sum_{v \in S \cup T} \deg_G(v) \\ &\in 2 \cdot \mathsf{Or}_G(S \cup T) \pm \frac{\mu}{\epsilon^3} - \sum_{v \in S \cup T} \deg_G(v) \\ &= \mathsf{Cut}_G(S \cup T) \pm \frac{\mu}{\epsilon^3} \ .\end{aligned}$$

By [2], the sketch-size must be $|\mathcal{I}_G| = \Omega(n/\epsilon^2)$, hence $s = \Omega(n/\epsilon^2)$ (for at least one graph $G \in \mathcal{F}$) as required. ∎

5. Multiple predicates and applications.

5. Multiple predicates and applications. In this section we extend Theorem 3.5 to VCSPs using multiple types of predicates. In particular, we prove sparsifability for some classical problems. Again, our sparsification results are stated as existential bounds, but these sparsifiers can actually be computed in polynomial time.

THEOREM 5.1. *For every $\epsilon \in (0, 1)$ and a VCSP (V, Π, w) whose constraints $\langle (v, u), P \rangle \in \Pi$ all satisfy $P \notin \{\mathit{nOr}, 01, \mathit{Dicut}, \mathit{And}\}$, there exists an ϵ-sparsifier for $\mathcal{I}$ with $O(|V|/\epsilon^2)$ constraints.*

This bound is tight, according to Theorem 4.1. We prove it by a straightforward application of Theorem 3.5. Partition $\mathcal{I}$ to disjoint VCSPs according to the predicates in the constraints, and then for each sub-VCSP find an ϵ-sparsifier using Theorem 3.5. The union of this sparsifiers is an ϵ-sparsifier for $\mathcal{I}$. A formal proof follows.

Proof of Theorem 5.1. For each predicate P, let $\Pi^P = \{\pi \in \Pi \mid \pi = \langle (v,u), \mathsf{P} \rangle\}$. Note that $\{\Pi^P\}$ forms a partition of Π. For each P, let $\mathcal{I}^P = (V, \Pi^P, w^P)$, where w^P is the restriction of w to Π^P. Let $\mathcal{I}^P_\epsilon = (V, \Pi^P_\epsilon, w^P_\epsilon)$ be an ϵ-P-sparsifier for $\mathcal{I}^P$ with $|\Pi^P_\epsilon| = O(|V|/\epsilon^2)$ constraints according to Theorem 3.5 (recall that $\mathsf{P} \notin \{\mathsf{nOr}, 01, \mathsf{Dicut}, \mathsf{And}\}$). Set $\mathcal{I}_\epsilon = (V, \Pi_\epsilon, w_\epsilon)$, $\Pi_\epsilon = \bigcup_P \Pi^P_\epsilon$ and $w_\epsilon = \bigcup_P w^P_\epsilon$. For every assignment A,

$$\begin{aligned}
\mathrm{Val}_{\mathcal{I}_\epsilon}(A) &= \sum_{\pi_i \in \Pi_\epsilon} w_\epsilon(\pi_i) \cdot p_i(A(v_i), A(u_i)) \\
&= \sum_{\mathsf{P}} \sum_{\pi_i \in \Pi^P_\epsilon} w^P_\epsilon(\pi_i) \cdot \mathsf{P}(A(v_i), A(u_i)) \\
&\in (1 \pm \epsilon) \cdot \sum_{P} \sum_{\pi_i \in \Pi^P} w^P(\pi_i) \cdot \mathsf{P}(A(v_i), A(u_i)) \\
&= (1 \pm \epsilon) \cdot \sum_{\pi_i \in \Pi} w(\pi_i) \cdot p_i(A(v_i), A(u_i)) \\
&= (1 \pm \epsilon) \cdot \mathrm{Val}_{\mathcal{I}}(A),
\end{aligned}$$

and note that indeed $|\Pi_\epsilon| \le O(n/\epsilon^2)$. ∎

$\mathsf{2SAT}$ (boolean satisfiability problem over constraints with two variables) can be viewed as a VCSP which uses only the predicates Or, nAnd, $\overline{10}$, and $\overline{01}$. By Theorem 5.1, for every $\mathsf{2SAT}$ formula Φ over n variables, and for every $\epsilon \in (0,1)$, there is a sub-formula Φ_ϵ with $O(n/\epsilon^2)$ clauses, such that Φ and Φ_ϵ have the same value for every assignment up to factor $1+\epsilon$.[3]

$\mathsf{2LIN}$ is a system of linear equations (modulo 2), where each equation contains two variables and has a nonnegative weight. Notice that the equation $x + y = 1$ is a constraint using the Cut predicate, while the equation $x + y = 0$ is a constraint using the unCut predicate. By Theorem 5.1, if n denotes the number of variables, then for every $\epsilon \in (0,1)$ we can construct a sparsifier with only $O(n/\epsilon^2)$ equations (i.e., a reweighted subset of equations, such that on every assignment it agrees with the original system up to factor $1+\epsilon$).

We note that by our lower bound (Theorem 4.1), there are instances of $\mathsf{2SAT}$ ($\mathsf{2LIN}$) for which every ϵ-sparsifier must contain $\Omega(n/\epsilon^2)$ clauses (equations).

6. Further directions.

Based on the past experience of cut sparsification in graphs—which has been extremely successful in terms of techniques, applications, extensions, and mathematical connections—we expect VCSP sparsification to have many benefits. A challenging direction is to identify which predicates admit sparsification, and our results make the first strides in this direction.

We now discuss potential extensions to our results in the previous sections (which characterize two-variable predicates over a boolean alphabet). We first consider predicates with more variables, and in particular show sparsification for k-SAT formulas, in section 6.1. We then consider predicates with large alphabets in section 6.2, showing in particular a sparsifier construction for k-Cut and that linear equations (modulo $k \ge 3$) are not sparsifiable. We also consider sketching schemes; notably we discuss a looser sketching model called *for-each* in section 6.3. Finally, we study *spectral* sparsification for unCut, a notion that preserves some algebraic properties in addition to the "uncuts" in section 6.4.

[3] We use here the version of $\mathsf{2SAT}$ where each clause has weight and every assignment has value, rather than the version when we only ask whether there is an assignment that satisfies all the clauses.

6.1. Predicates over more variables and k-SAT. It is natural to ask for the best bounds on the size of ϵ-P-sparsifiers for different predicates $\mathsf{P} : \{0,1\}^k \to \{0,1\}$. A first step toward answering this question was already done by [11].

THEOREM 6.1 (see [11]). *For every hypergraph $H = (V,E,w)$ with hyperedges containing at most r vertices, and $\epsilon \in (0,1)$, there is a reweighted subhypergraph H_ϵ with $O(n(r + \log n)/\epsilon^2)$ hyperedges such that*

$$\forall S \subseteq V, \quad \mathit{Cut}_{H_\epsilon}(S) \in (1 \pm \epsilon) \cdot \mathit{Cut}_H(S).$$

Here we say that a hyperedge e is *cut* by S if $S \cap e \notin \{\emptyset, e\}$ (i.e., not all the vertices in e are in the same side). Observe that Cut is equivalent to the predicate NAE (not all equal). In particular Theorem 6.1 implies that for every VCSP using only NAE, there is an ϵ-sparsifier with $O(n(r + \log n)/\epsilon^2)$ constraints.

A k-SAT is essentially a VCSP that uses only predicates with a single 0 in their truth table. Kogan and Krauthgamer [11] use Theorem 6.1 to construct an ϵ-sketching-scheme with sketch-size $\tilde{O}(nk/\epsilon^2)$ for k-SAT formulas (i.e., only for VCSPs of this particular form). We observe that their sketching scheme can be further used to construct an ϵ-sparsfier, as follows.

First, recall how the sketching scheme of [11] works. Given a k-SAT formula $\Phi = (V, \mathcal{C}, w)$ (variables, clauses, weight over $\mathcal{C}$), construct a hypergraph H on vertex set $V \cup -V \cup \{f\}$. We associate the literal v_i with vertex v_i, associate the literal $\neg v_i$ with vertex v_{-i}, and use f to represent the "false." Each clause becomes a hyperedge consisting of f and (the vertices associated with) the literals in $\mathcal{C}$ (for example, $v_5 \vee \neg v_7 \vee v_{12}$ becomes $\{f, v_5, v_{-7}, v_{12}\}$). Observe that given a truth assignment $A : V \to \{0,1\}$, if we define $S_A := \{u \mid A(u) = 0\}$, then $\mathrm{Val}_\Phi(A) = \mathsf{Cut}_H(S_A \cup \{f\})$, and using Theorem 6.1 this provides a sketching scheme. Moreover, given an ϵ-Cut-sparsifier H_ϵ for H, let Φ_ϵ be the formula which has only the clauses associated with edges that "survived" the sparsification, with the same weight. Notice that for every assignment A,

$$\mathrm{Val}_{\Phi_\epsilon}(A) = \mathsf{Cut}_{H_\epsilon}(S_A \cup \{f\}) \in (1 \pm \epsilon) \cdot \mathsf{Cut}_H(S_A \cup \{f\}) = (1 \pm \epsilon) \cdot \mathrm{Val}_\Phi(A).$$

THEOREM 6.2. *Given k-SAT formula Φ over n variables and parameter $\epsilon \in (0,1)$, there is an ϵ-sparsifier subformula ϕ_ϵ with $O(n(k + \log n)/\epsilon^2)$ clauses.*

In contrast, we are not aware of any nontrivial sparsification result for the parity predicate (on $k \geq 3$ boolean variables), and this remains an interesting open problem.

6.2. Predicates over larger alphabets. Our results deal only with predicates that get two input values in $\{0,1\}$. A natural generalization is to sparsify a VCSP that uses a predicate over an alphabet of size k, i.e., $\mathsf{P} : [k] \times [k] \to \{0,1\}$, where $[k] := \{0, 1, \ldots, k-1\}$. One predicate that we can easily sparsify is NE (not-equal), which is satisfied if the two constrained variables are assigned different values. Indeed, in the graphs language, this is called a k-Cut, where the value of a partition $(S_0, \ldots, S_{k-1})$ of the vertices is the total weight of all edges with endpoints in different parts. It turns out that the ϵ-Cut-sparsifier is in particular an ϵ-k-Cut-sparsifier, using the following well-known double-counting argument:

$$\begin{aligned}\mathsf{k\text{-}Cut}_{G_\epsilon}(S_0,\ldots,S_{k-1}) &= \frac{1}{2}\cdot\left[\mathsf{Cut}_{G_\epsilon}\left(S_0,\overline{S_0}\right)+\cdots+\mathsf{Cut}_{G_\epsilon}\left(S_{k-1},\overline{S_{k-1}}\right)\right]\\ &\in (1\pm\epsilon)\cdot\frac{1}{2}\cdot\left[\mathsf{Cut}_{G}\left(S_0,\overline{S_0}\right)+\cdots+\mathsf{Cut}_{G}\left(S_{k-1},\overline{S_{k-1}}\right)\right]\\ &= (1\pm\epsilon)\cdot\mathsf{k\text{-}Cut}_G(S_0,\ldots,S_{k-1}).\end{aligned}$$

In contrast, linear equation predicates are nonsparsifiable for alphabet $[k]$ of size $k \geq 3$. Specifically, for $a \in [k]$, let the predicate Sum_a be satisfied by $x, y \in [k]$ iff $x + y = a \pmod k$. Then for every positively weighted digraph $G = (V, E, w)$, and every $\epsilon \in (0, 1)$, $a \in [k]$, every Sum_a-ϵ-sparsifier $G_\epsilon = (V, E_\epsilon, w_\epsilon)$ of G must have $E = E_\epsilon$. The argument is similar to the proof of Theorem 3.2. Assume for contradiction there exist $e \in E \setminus E_\epsilon$. Choose $x, y, z \in [k]$ that satisfy $x + y = a$, however the three sums $z + x$, $z + y$, $z + z$ are all not equal to a (modulo k); this is clearly possible for $k \geq 4$ and easily verified by case analysis for $k = 3$. Consider an assignment where the endpoints of e have values x and y, respectively, and all other vertices have value z. Under this assignment, the value of G is $w(e) > 0$, while the value of G_ϵ is zero, a contradiction.

6.3. Sketching. In Theorem 4.1 we showed that for every predicate $\mathsf{P} \in \{\mathsf{Cut}, \mathsf{unCut}, \mathsf{Or}, \mathsf{nAnd}, \overline{10}\}$, the sketch-size of every ϵ-P-sketching-scheme is $\Omega(n/\epsilon^2)$.

Let us now address predicates with a single 1 in their truth table. In the spirit of the proof of Theorem 3.2, given encoding sk_G by an ϵ-And-sketching-scheme we can completely restore the graph G. As there are $2^{\binom{n}{2}}$ different graphs, the sketch-size of every ϵ-And-sketching-scheme is at least $\Omega(n^2)$ bits. Imitating the proof of Theorem 3.6, we can extend this lower bound to Dicut, 01, and 10.

For-each sketches. In order to reduce storage space of a sketch, one might weaken the requirements even further and allow the sketch to give a good approximation only with high probability. A *for-each sketching scheme* is a pair of algorithms $(\mathrm{sk}, \mathrm{est})$; algorithm sk is a randomized algorithm that given a graph G returns a string sk_G, whose distribution we denote by $\mathcal{D}_G$; algorithm est is given such a string sk_G and a subset $S \subseteq V$ and returns (deterministically) a value $\mathrm{est}(\mathrm{sk}_G, S)$. We say that it is an (ϵ, δ)-*P-sketching-scheme* if

$$\forall G = (V, E, w), \forall S \subseteq V, \quad \Pr_{\mathrm{sk}_G \in \mathcal{D}_G}\left[\mathrm{est}(\mathrm{sk}_G, S) \in (1 \pm \epsilon)\cdot \mathsf{P}_G(S)\right] \geq 1 - \delta\ .$$

In [2], it was showed that if we consider n-vertex graphs with weights only in the range $[1, W]$, then there is an $(\epsilon, 1/\mathrm{poly}(n))$-$\mathsf{Cut}$-sketching-scheme with sketch-size $\tilde{O}\left(n\epsilon^{-1}\cdot\log\log W\right)$ bits. Imitating Theorem 3.5, we can construct $(\epsilon, 1/\mathrm{poly}(n))$-$\mathsf{P}$-sketching-scheme with the same sketch-size for every predicate P whose truth table does not have a single 1 (and weights restricted to the range $[1, W]$). A nearly matching lower bound by [2] shows that for every $\epsilon \in (2/n, 1/2)$, every $(\epsilon, 1/10)$-Cut-sketching-scheme must have sketch-size $\Omega(n/\epsilon)$. Using γ, this lower bound can be extended to unCut. This technique does not work for predicates with three 1's in their truth table. Fortunately, we can duplicate the proof of [2] while replacing Cut by Or and using the fact that for every two vertices v, u in the graph G, it holds that $\mathsf{Or}(\{v\}) + \mathsf{Or}(\{u\}) - \mathsf{Or}(\{v, u\}) = \mathbf{1}_{\{\{u,v\}\in E\}}$. We omit the details of this straightforward argument. A reduction from Or using γ and (5) will extend the lower bound also to nAnd, $\overline{10}$ and $\overline{01}$.

Given a sketch sk_G (i.e., one sample from distribution $\mathcal{D}_G$) which encodes an (ϵ, δ)-And-sketching-scheme, one can reconstruct every edge of G (every bit of the adjacency matrix) with constant probability. Standard information-theoretical arguments (indexing problem) imply that the sketch-size of every (ϵ, δ)-And-sketching-scheme is $\Omega(n^2)$ bits. Using γ we can extend this lower bound to Dicut, 01 and 10.

6.4. unCut spectral sparsifiers. Given an undirected n-vertex graph $G = (V, E, w)$, the Laplacian matrix is defined as $L_G = D_G - A_G$, where A_G is the adjacency matrix (i.e., $A_{i,j} = w_{i,j} = w(\{v_i, v_j\})$) and D_G is a diagonal matrix of degrees (i.e., $D_{i,i} = \sum_{j \neq i} w_{i,j}$ and for $i \neq j$, $D_{i,j} = 0$). For every $x \in \mathbb{R}^n$ it holds that $x^t L_G x = \sum_{\{v_i,v_j\} \in E} w_{i,j} \cdot (x_i - x_j)^2$. In particular, for $\mathbf{1}_S$ the indicator vector of some subset $S \subseteq V$ it holds that $\mathbf{1}_S^t L_G \mathbf{1}_S = \mathsf{Cut}_G(S)$. A subgraph H of G is called an *ϵ-spectral-sparsifier* of G if

$$\forall x \in \mathbb{R}^n, \quad x^t L_H x \in (1 \pm \epsilon) \cdot x^t L_G x \ .$$

Note that an ϵ-spectral-sparsifier is in particular an ϵ-Cut-sparsifier. Nonetheless, spectral sparsifiers preserve additional properties such as the eigenvalues of the Laplacian matrix (approximately). Batson, Spielman, and Srivastava [3] showed that every graph admits an ϵ-spectral-sparsifier with $O(n/\epsilon^2)$ edges.

DEFINITION 6.3. *Given a graph G, we call $U_G = (D_G + A_G)$ the* negated Laplacian *of G. Given a subset $S \subseteq V$, let $\phi_S \in \mathbb{R}^n$ be a vector such that $\phi_{S,i} = 1$ if $v_i \in S$ and $\phi_{S,i} = -1$ otherwise.*

One can verify that for arbitrary $x \in \mathbb{R}^n$,

$$x^t U_G x = \sum_{i<j} w_{i,j} \cdot (x_i + x_j)^2 \ .$$

In particular, for every subset $S \subseteq V$, it holds that

$$\phi_S^t U_G \phi_S = 4 \cdot \mathsf{unCut}_G(S) \ .$$

Next, we will show how we can use U_G to construct an unCut-sparsifier G_ϵ (in an alternative way to Theorem 3.5) such that U_{G_ϵ} has (approximately) the same eigenvalues as U_G. A matrix $M \in \mathbb{R}^{n \times n}$ is called *balanced symmetric diagonally dominant* (*BSDD*) if $M = M^t$ and for every index i, $M_{i,i} = \sum_{j \neq i} |M_{i,j}|$. Note that L_G and U_G are both BSDD. A matrix M' is *governed* by M if whenever $M'_{i,j} \neq 0$, also $M_{i,j} \neq 0$ and has the same sign. Note that if H is a subgraph of G, then U_H is governed by U_G. A matrix M' is called an *ϵ-spectral-sparsifier* of M if M' is governed by M and

$$\forall x \in \mathbb{R}^n, \quad x^t M' x \in (1 \pm \epsilon) \cdot x^t M x \ .$$

The following was implicitly shown in [2].

THEOREM 6.4 (see [2]). *Given BSDD matrix $M \in \mathbb{R}^{n \times n}$ and parameter $\epsilon \in (0, 1)$, there is an ϵ-spectral-sparsifier M' for M, where M' is BSDD matrix with $O(n/\epsilon^2)$ nonzero entries.*

Fix a graph G and parameter ϵ; according to Theorem 6.4, there is a BSDD balanced matrix H with $O(n/\epsilon^2)$ nonzero entries, which is a ϵ-spectral-sparsifier for U_G. Moreover, H is governed by U_G. These properties define a graph G_ϵ such that $U_{G_\epsilon} = H$. In particular G_ϵ is an ϵ-unCut-sparsifier of G with $O(n/\epsilon^2)$ edges.

REFERENCES

[1] K. J. Ahn and S. Guha, *Graph sparsification in the semi-streaming model*, in 36th International Colloquium on Automata, Languages and Programming, Lecture Notes in Comput. Sci. 5556, Springer-Verlag, Berlin, 2009, pp. 328–338, https://doi.org/10.1007/978-3-642-02930-1_27.

[2] A. Andoni, J. Chen, R. Krauthgamer, B. Qin, D. P. Woodruff, and Q. Zhang, *On sketching quadratic forms*, in Proceedings of ITCS'16, ACM, 2016, pp. 311–319, https://doi.org/10.1145/2840728.2840753.

[3] J. D. Batson, D. A. Spielman, and N. Srivastava, *Twice-Ramanujan sparsifiers*, SIAM Rev., 56 (2014), pp. 315–334, https://doi.org/10.1137/130949117.

[4] A. A. Benczúr and D. R. Karger, *Approximating* s-t *minimum cuts in* $\tilde{O}(n^2)$ *time*, in Proceedings of the 28th Annual ACM Symposium on Theory of Computing, ACM, 1996, pp. 47–55, https://doi.org/10.1145/237814.237827.

[5] A. A. Benczúr and D. R. Karger, *Randomized Approximation Schemes for Cuts and Flows in Capacitated Graphs*, CoRR cs.DS/0207078, https://arXiv.org/abs/cs/0207078, 2002.

[6] R. A. Brualdi, F. Harary, and Z. Miller, *Bigraphs versus digraphs via matrices*, J. Graph Theory, 4 (1980), pp. 51–73, https://doi.org/10.1002/jgt.3190040107.

[7] M. K. de Carli Silva, N. J. A. Harvey, and C. M. Sato, *Sparse Sums of Positive Semidefinite Matrices*, CoRR abs/1107.0088, https://arXiv.org/abs/1107.0088, 2011.

[8] W. S. Fung, R. Hariharan, N. J. Harvey, and D. Panigrahi, *A general framework for graph sparsification*, in Proceedings of the 43rd Annual ACM Symposium on Theory of Computing, ACM, 2011, pp. 71–80, https://doi.org/10.1145/1993636.1993647.

[9] M. Kapralov and R. Panigrahy, *Spectral sparsification via random spanners*, in Proceedings of the 3rd Innovations in Theoretical Computer Science Conference, ACM, 2012, pp. 393–398, https://doi.org/10.1145/2090236.2090267.

[10] D. R. Karger and M. S. Levine, *Random sampling in residual graphs*, in Proceedings of the Symposium on Theory of Computing, 2002, pp. 63–66.

[11] D. Kogan and R. Krauthgamer, *Sketching cuts in graphs and hypergraphs*, in Proceedings of the Conference on Innovations in Theoretical Computer Science, ACM, 2015, pp. 367–376, https://doi.org/10.1145/2688073.2688093.

[12] A. Madry, *Fast approximation algorithms for cut-based problems in undirected graphs*, in Proceedings of the Symposium on Foundations of Computer Science, IEEE, 2010, pp. 245–254.

[13] I. Newman and Y. Rabinovich, *On multiplicative* λ*-approximations and some geometric applications*, SIAM J. Comput., 42 (2013), pp. 855–883, https://doi.org/10.1137/100801809.

[14] J. Sherman, *Breaking the multicommodity flow barrier for* $O(\sqrt{\log n})$*-approximations to sparsest cut*, in Proceedings of the Symposium on Foundations of Computer Science, 2009, pp. 363–372.

[15] D. A. Spielman and N. Srivastava, *Graph sparsification by effective resistances*, SIAM J. Comput., 40 (2011), pp. 1913–1926, https://doi.org/10.1137/080734029.

[16] D. A. Spielman and S.-H. Teng, *Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems*, in Proceedings of the 36th Annual ACM Symposium on Theory of Computing, ACM, 2004, pp. 81–90, https://doi.org/10.1145/1007352.1007372.

[17] D. A. Spielman and S.-H. Teng, *Spectral sparsification of graphs*, SIAM J. Comput., 40 (2011), pp. 981–1025, https://doi.org/10.1137/08074489X.