*Todo list

Going for Speed: Sublinear Algorithms for Dense r-CSPs

Grigory Yaroslavtsev *
University of Pennsylvania
grigory@grigory.us

July 27, 2014

Abstract

We give new sublinear and parallel algorithms for the extensively studied problem of approximating n-variable r-CSPs (constraint satisfaction problems with constraints of arity r) up to an additive error $O(\epsilon n^r)$. The running time of our algorithms is $O\left(\frac{n}{\epsilon^2}\right) + 2^{O\left(\frac{1}{\epsilon^2}\right)}$ for Boolean r-CSPs and $O\left(\frac{k^4 n}{\epsilon^2}\right) + 2^{O\left(\frac{\log k}{\epsilon^2}\right)}$ for r-CSPs with constraints on variables over an alphabet of size k. For any constant k this gives optimal dependence on n in the running time unconditionally, while the exponent in the dependence on $1/\epsilon$ is polynomially close to the lower bound under the exponential-time hypothesis, which is $2^{\Omega(1/\sqrt{\epsilon})}$.

For Max-Cut this gives an exponential improvement in dependence on $1/\epsilon$ compared to the sublinear algorithms of Goldreich, Goldwasser and Ron (JACM'98) and a linear speedup in n compared to the algorithms of Mathieu and Schudy (SODA'08). For the maximization version of k-Correlation Clustering problem our running time is $O(k^4 n/\epsilon^2) + k^{O(1/\epsilon^2)}$, improving the previously best $nk^{O\left(\frac{1}{\epsilon^3}\log\frac{k}{\epsilon}\right)}$ by Guruswami and Giotis (SODA'06).

Corrected running time.

Corrected running time.

Rephrasing.

Corrected running time.

1 Introduction

Approximation algorithms for constraint satisfaction problems have received a lot of attention in the recent years. In particular, polynomial-time approximation schemes for dense instances have been developed using several different approaches [dlV96, GGR98, AKK99, FK99, AdlVKK03, dlVKKV05, dlVKM07, MS08, AFNS09, BHHS11, YZ14], including combinatorial sampling methods, subsampling from linear programming and semidefinite programming relaxations and rounding of hierarchies of linear programming relaxations. Notably, some of these algorithms run in sublinear time in the input size (e.g. [GGR98, AdlVKK03]). In particular, for Max-Cut, probably the most commonly studied CSP, a partition which gives a cut of size within ϵn^2 of the optimum can be constrained in linear in n time for fixed ϵ [GGR98].

In this paper we revisit the problem of constructing approximate solutions for instances of r-CSPs with additive error ϵn^r, focusing on algorithms which run in sublinear time. To the best of our knowledge, even for Max-Cut, the most basic 2-CSP, among all algorithms considered in the previous work only those of [GGR98] can be used to reconstruct the solution in sublinear time. We are not aware of any algorithms for general r-CSPs over large alphabets which achieve sublinear running time. Note that multiple algorithms exist (e.g. [AdlVKK03, MS08]) for approximating the cost of the optimum solution in sublinear (and even constant) time, however all these approaches take at least linear time to reconstruct the solution itself (see Table 1). The motivation for fast algorithms comes from applications of r-CSPs to areas such as clustering and graph partitioning. For example, the results of [GG06] can be interpreted as showing (implicitly) that Correlation Clustering into k clusters can be expressed as a 2-CSP problem over an alphabet of size k and hence algorithms for approximating CSPs can be directly applied to this problem.

Rephrashing

Rephrasing.

*This work was done while the author was supported by a postdoctoral fellowship at the Warren Center for Network and Data Sciences at the University of Pennsylvania and the Institute Postdoctoral Fellowship at the Brown University, ICERM.

1.1 Previous work

In Table 1 we compare the running times of our algorithms against those achieved prior to this work via three different approaches to designing PTASes for dense r-CSPs: combinatorial algorithms based on sampling, subsampling of mathematical relaxations and rounding of linear programming hierarchies. For any fixed k our algorithms give the best running time.

Rephrasing.

Combinatorial algorithms with sampling. Historically this is the first framework which led to development of PTASes for dense problems. For Max-Cut and some other graph problems sampling-based combinatorial algorithms were developed by de la Vega [dlV96], Goldreich, Goldwasser and Ron [GGR98] and Arora, Karger and Karpinski [AKK99]. Prior to our work, which falls into this category, fastest algorithms for dense problems were also obtained via this framework. Fastest of such algorithms were [GGR98] and [MS08], which have incomparable running times. Former work achieves sublinear dependence on the input size, while its dependence on ϵ is worse. Latter work gives algorithms with linear running time in the input size, while achieving better dependence on ϵ.

Efficiency of combinatorial algorithms based on sampling makes them appealing for applications to clustering and graph partitioning [GGR98, GG06]. Some of these algorithms [GGR98] are extremely easy to parallelize, while others [MS08] are inherently sequential. Our work falls in between these two extremes — the basic versions of our algorithms are sequential but we show how to transform them into parallel algorithms. We discuss parallel versions of our algorithms in Section 4.

Subsampling of mathematical relaxations. This approach was first suggested by [AdlVKK03], who studied the effect of subsampling on the value of the objective function of mathematical relaxations of dense NP-hard problems. It was further extended to more general settings by [BHHS11]. To the best of our knowledge, all algorithms obtained in this line of work run either in linear or polynomial time in the input size.

Rounding linear programming hierarchies. The most recent approach is based on rounding linear programming hierarchies. De la Vega and Kenyon-Matheiu [dlVKM07] showed that $O(1/\epsilon^2)$ rounds of Sherali-Adams hierarchy suffice to bring the integrality gap for dense instances of Max-Cut down to $(1+\epsilon)$. More generally, Yoshida and Zhou [YZ14] showed that $2^{O(r)} \log k/\epsilon^2$ rounds of Sherali-Adams suffice for general r-CSPs over an alphabet of size k.

1.2 Related work in streaming algorithms

We would like to highlight the connection between the line of work on sublinear time algorithms and related area of sublinear space streaming algorithms. Recently it has been shown that $(1 \pm \epsilon)$-approximate *cut sparsifiers* and *spectral sparsifiers* can be constructed in the streaming model in $\tilde{O}(n/\epsilon^2)$ space [AGM12, GKP12, KL13, KTM$^+$14]. This is almost optimal (see [BK96, ST11, SS11, BSS14]) and in particular implies that dense Max-Cut problem that we consider can be solved in almost optimal sublinear space in the streaming model, if there are no restrictions on the running time. The dependence on n in our algorithms for Max-Cut is sublinear and optimal, as is space in the streaming algorithms. It remains open whether both of these sublinear time and space bounds can be achieved by the same algorithm.

1.3 Our results and techniques

In order to introduce the main ideas we first develop a simpler version of our algorithm for Max-Cut in Section 2. Our main algorithm for Max-Cut follows uses a more general approach that we take in Section 3 and has the following guarantee.

Theorem 1.1. *There is an algorithm which approximates* Max-Cut *with additive error* $O(\epsilon n^2)$ *and runs in time* $O(n/\epsilon^2) + 2^{O(1/\epsilon^2)}$.

Running times of approximation schemes for dense r-CSPs				
	Max-Cut	Binary r-CSP ($r > 2$)	k-ary 2-CSP	k-ary r-CSP ($r > 2$)
Combinatorial algorithms with sampling				
[AKK99]	$n^{O(1/\epsilon^2)}$	—	—	—
[dlV96]	$O\left(n^2 \cdot 2^{1/\epsilon^{2+o(1)}}\right)$	—	—	—
[GGR98]	$O\left(\frac{n \log 1/\epsilon}{\epsilon^2}\right) + 2^{O\left(\frac{\log 1/\epsilon}{\epsilon^3}\right)}$	—	$O\left(\frac{n \log k/\epsilon}{\epsilon^2}\right) + 2^{O\left(\frac{\log^2 k/\epsilon}{\epsilon^3}\right)}$	—
[MS08]	$O(n^2) + 2^{O(1/\epsilon^2)}$	$O(n^r) + 2^{O(1/\epsilon^2)}$	$O(n^2) + 2^{O\left(\frac{\log k}{\epsilon^2}\right)}$	$O(n^r) + 2^{O\left(\frac{\log k}{\epsilon^2}\right)}$
Our work	$O\left(\frac{n}{\epsilon^2}\right) + 2^{O\left(\frac{1}{\epsilon^2}\right)}$	$O\left(\frac{n}{\epsilon^2}\right) + 2^{O\left(\frac{1}{\epsilon^2}\right)}$	$O\left(\frac{nk^4}{\epsilon^2}\right) + 2^{O\left(\frac{\log k}{\epsilon^2}\right)}$	$O\left(\frac{nk^4}{\epsilon^2}\right) + 2^{O\left(\frac{\log k}{\epsilon^2}\right)}$
Subsampling of mathematical relaxations				
[AdlVKK03]	$O(n^2) \cdot 2^{\tilde{O}(1/\epsilon^2)}$	$O(n^r) \cdot 2^{\tilde{O}(1/\epsilon^2)}$	—	—
Rounding linear programming hierarchies				
[dlVKM07]	$n^{O(1/\epsilon^2)}$	—	—	—
[YZ14]	$n^{O(1/\epsilon^2)}$	$n^{2^{O(r)}/\epsilon^2}$	$n^{O(\log k/\epsilon^2)}$	$n^{2^{O(r)} \log k/\epsilon^2}$

Table 1: Approximation algorithms with additive error ϵn^r.

The intuition behind our algorithms comes from [GGR98] and [MS08]. In particular, our algorithm for Max-Cut is inspired by [GGR98], who introduced a sampling-based technique for approximating dense instances of Max-Cut with sublinear running time. However, we depart from their approach, which partitions the graph into $\Theta(1/\epsilon)$ parts and then partitions each part based on an optimum partition of a sample of size $\Theta(1/\epsilon^2)$ drawn from the rest of the graph. Such an approach seems to inherently require the complexity to be $2^{\Omega(1/\epsilon^3)}$ due to the partitioning step. Instead, we use a bootstrapping scheme from the linear time algorithm of [MS08]. In a nutshell, an optimum solution on a primary sample of size $O(1/\epsilon^2)$ suffices to partition a large secondary sample of size $O(1/\epsilon^4)$ within an additive error $O(1/\epsilon^7)$ and then this approximate solution on the secondary sample can be further used to construct an approximate solution for the entire graph. The original algorithm of [MS08] takes $O(n^2)$ time, i.e. linear in the input size to convert the approximately optimal solution for the secondary sample into an approximately optimal solution for the entire graph. We show how to choose sampling rate at each step of the algorithm in order to maintain the same approximation guarantee in optimal sublinear time.

We introduce the main ideas in Section 2, where we describe a simplified version of our algorithm for Max-Cut, Algorithm 1. If the greedy step in this algorithm is performed using exact degrees then the analysis of this algorithm is given in [MS08]. While the idea of using sampling in the greedy step might seem natural, the details are quite involved. Indeed, a single greedy step can be easily seen to introduce only small error even if it is made based on approximate degrees. However, Algorithm 1 executes the greedy steps sequentially and degrees used to place the current vertex depend on the placement of the previous vertices. This may allow the errors the algorithm makes when placing vertices to affect degrees used when placing subsequent vertices and thus lead to amplification of errors. We use martingale-based analysis based on [MS08] to break down the error analysis in such a way that it can be done independently for every step (Lemma 2.2). It is crucial that this analysis still applies to the new martingale that we use, which tracks performance of an approximate greedy algorithm instead of an exact one. Given this, there are two sources of error in every step: the fact that we are using a greedy choice and the approximation used in this choice. In Lemma 2.3 we show that these two sources can be treated independently and analyze the error introduced by sampling in Proposition 2.4. To analyze the error introduced by the greedy choice we use the martingale-based argument of [MS08] adapted to our new martingale (Lemma 2.5). It is crucial that the argument is robust to the change in the definition of the martingale – we show this by reproducing the analysis. Finally, we give the overall analysis of the approximation given by Algorithm 1 in Lemma 2.6. In order to achieve the desired additive bound together with linear running time of the greedy step we choose the sample size

Extended discussion of new ideas as compared to [MS08] as per Comment 1.

to be time-dependent. Selection of the sampling rate is one of the most challenging parts of the analysis.

In Section 3 we give the analysis of our main algorithm, Algoirthm 2, which is a faster version of Algorithm 1 and also works for general r-CSPs. It introduces the bootstrapping step in order to reduce the running time. This proof follows a similar structure and hence all observations above apply. Again, we are able to break the analysis into steps and separate the error introduced by the greedy selection process and the approximation involved in it. Even though the details are more delicate since we now have to deal with the more general case of r-CSPs we are still able to adapt some of the technical lemmas of [MS08] under the new definitions that we use. This leaves us two main technical challenges to address: selection of the sample space and choice of the sampling rate sufficient to achieve the desired approximation. For Max-Cut the sampling space is easily determined to be the set of already placed neighbors of the vertex, which is currently being placed greedily. For general r-CSPs we show that an appropriate generalization is the set of all *critical constraints* (see Definition 3.1) containing currently processed variable. This choice is crucial for the analysis to be extendable to this case because it allows to bound the size of the sampling space at time t by t^{r-1}. It is important that the sampling space can be shown to be much smaller than the set of all constraints involving current variable, which may have size n^{r-1}. Together with a careful choice of time-dependent sampling rate, which generalizes the one we use in Algorithm 1, the bound on the size of the sampling space allows to achieve both the optimal bound on the running time and the additive approximation guarantee.

For general r-CSPs over an alphabet of size k we obtain the following results, which follow from the analysis of Algorithm 2.

Theorem 1.2. *There is an algorithm which approximates any r-CSP problem over alphabet of size k within additive error $O(\epsilon n^r)$ and runs in time $O\left(\frac{nk^4}{\epsilon^2}\right) + 2^{O\left(\frac{\log k}{\epsilon^2}\right)}$.*

Theorem 1.2 immediately implies algorithms with the same running time for the k-Correlation Clustering problem by taking $r = 2$.

Our last result is a lower bound, which is proved in Section A and complements the performance guarantees of our algorithms, implying a lower bound of $\Omega(n/\epsilon^2) + 2^{\Omega(1/\sqrt{\epsilon})}$ on the running time of any algorithm for Max-Cut, assuming ETH.

Theorem 1.3. *Any algorithm $\mathcal{A}$, which approximates* Max-Cut *within error ϵn^2 in the adjacency list model has to make at least $\Omega(n/\epsilon^2)$ queries to the edges of the graph. Every such algorithm also has to have at least $2^{\Omega(1/\sqrt{\epsilon})}$ running time assuming the exponential time hypothesis.*

This theorem justifies the fact that the running time of sublinear algorithms for r-CSPs is naturally divided into terms, the first one corresponding to the query complexity and the second one corresponding to the computational complexity of the problem. While the first term in our work is provably unconditionally almost tight by Theorem 1.3, we can only lower bound the second term conditionally since it corresponds to the computational complexity of the problem.

The computational lower bound in Theorem 1.3 is easy to show. Consider Max-Cut on instances of size $1/\sqrt{\epsilon}$. Assuming ETH, such instances can't be solved exactly in time $2^{o(1/\sqrt{\epsilon})}$. However, an additive error guarantee of ϵn^2 requires that such instances have to be solved exactly by our algorithm. The query complexity lower bound comes from the intuition that any PTAS for dense instances of Max-Cut has to estimate the degrees of at least a constant fraction of vertices up to an additive error ϵn, which requires a sample of size $\Omega(1/\epsilon^2)$ per vertex and yields the overall lower bound. However, the technical details of this proof are more involved and are given in Section A. We use Yao's principle and construct a random family of hard instances as follows. Let $V = V_0 \cup V_1 \cup V_2$, where $|V_0| = |V_1| = 4n/9$ and $|V_2| = n/9$. The vertex set $V_0 \cup V_1$ induces a complete bipartite graph $K_{4n/9,4n/9}$ with parts V_0 and V_1, which corresponds to a planted dense solution for Max-Cut. For each vertex $v \in V_2$ we randomly pick a side of this cut $r_v \in \{0, 1\}$ with probability $1/2$ each and add edges with probability $1/2 + \epsilon$ to each vertex on the side r_v and with probability $1/2$ to each vertex on the other side $1 - r_v$. Thus, in the optimum solution each vertex in $v \in V_2$ has to be placed on the side $1 - r_v$ of the cut. The intuition behind the lower bound is that even if the algorithm guesses the planted solution on $V_0 \cup V_1$ without any queries then in order to get an additive error $c\epsilon n^2$ for some sufficiently small c it still has to guess $1 - r_v$ with probability greater than $1/2$ for vertices

$v \in V_2$. This is impossible without sampling at least $\Omega(1/\epsilon^2)$ edges by a Chernoff-type lower bound against sampling algorithms [CEG95]. However, the technical details are more complicated for two reasons. First, an approximation algorithm can partition $V_0 \cup V_1$ differently than the optimum planted solution, which might simplify the task of guessing the optimal side for vertices in V_2. We show that unless the partitioning of $V_0 \cup V_1$ is sufficiently close to optimum the algorithm incurs an error of $\Omega(\epsilon n^2)$ on edges induced by these vertices alone. Second and more subtle issue is the way $V_0 \cup V_1$ is partitioned by the algorithm might depend on the edges adjacent to vertices in V_2. To address this we use a probabilistic argument, arguing by a union bound that for *every* partition of $V_0 \cup V_1$, which is sufficiently close to the optimum, placing vertices in V_2 optimally still requires $\Omega(n/\epsilon^2)$ queries.

2 Max-Cut

To illustrate the main ideas we first present Algorithm 1, which demonstrates how sampling can be used to speed up an approximate greedy algorithm.

Algorithm 1: Greedy PTAS with subsampling.

input : Graph $G(V, E)$, where $|V| = n$, parameter ϵ.

Pick a sample S of $t_0 = 1/\epsilon^2$ vertices uniformly at random without replacement
for *each of the* 2^{t_0} *possible partitions of* S *into two parts* **do**
  $S^{t_0} = S, t = t_0 + 1$
  for *each vertex* $v \in V \setminus S$ *in random order* **do**
    Pick a sample V^t of $s_t = O\left(\frac{n^{2/3}}{t^{2/3}\epsilon^2}\right)$ vertices uniformly at random without replacement from S^{t-1}.
    Assign v to the side of the cut, which maximizes the number of cut edges with respect to the current partition of V^t
    $S^t = S^{t-1} \cup \{v\}, t = t + 1$
Output the best cut over all iterations

Added time-dependent sampling rate as per Comment 4.

Theorem 2.1. *Algorithm 1 gives an additive* $O(\epsilon n^2)$*-approximation for* Max-Cut *in time* $O(n) \cdot 2^{O(1/\epsilon^2)}$

First, note that the total size of all samples is $\sum_{\tau=t_0}^{n} s_\tau = \frac{n^{2/3}}{\epsilon^2} \sum_{\tau=t_0}^{n} \tau^{-2/3} = O\left(\frac{n}{\epsilon^2}\right)$. Thus, the overall running time of the algorithm is $O\left(\frac{n}{\epsilon^2} 2^{1/\epsilon^2}\right)$ for all iterations of the loop[1].

New running time analysis with time-dependent samping rate + remark on possible sample sizes.

The sides of the cut are indexed by $i \in \{1, 2\}$. A cut is represented by a vector $x \in \{0, 1\}^{2n}$, where $x_{ui} = 1$ iff the vertex u is assigned a label i. Let A to denote a matrix with entries A_{i,u_1,j,u_2} defined as follows: $A_{i,u_1,j,u_2} = 1/2$ if $u_1 = u_2$ and $(i, j) \in E$ and $A_{i,u_1,j,u_2} = 0$ otherwise. Then we can express the objective function as minimization of a bilinear form $x^T Ax$. For a fixed iteration t of the algorithm we use A^t to denote the matrix sampled from A with columns corresponding to vertices in V^t being the same as the columns of A, while all other columns replaced be zeros. Formally $A^t_{i,u_1,j,u_2} = A_{i,u_1,j,u_2}$ if $j \in V^t$ and $A^t_{i,u_1,j,u_2} = 0$ otherwise.

We will track the solution obtained at time t using variables x^t_{ui} such that $x^t_{ui} = 1$ if at time t the vertex u is assigned label i and $x^t_{ui} = 0$ otherwise (either u is assigned a different label or not assigned a label at all). Let r_t denote the t-th vertex considered by the algorithm. Let S^t denote the set of vertices assigned by time t. Let x^* be the optimum cut. We denote the exact and approximate greedy choices at time t for each

[1]For this analysis of the running time as well as for the analysis of the approximation below (Lemma 2.6) any choice of sample size $s_t = \frac{n^\delta}{t^\delta \epsilon^2}$ would suffice. The choice of $\delta = 2/3$ allows to minimize the constant factor in the running time.

vertex u as $\tilde{g}^t_{ui}$ and g^t_{ui}, which are given as:

$$\tilde{g}^t_{ui} = \begin{cases} x^*_{ui}, \text{ if } t \le t_0, \\ 1, \text{ if } t > t_0, i = \arg\min_j A_{uj} x^{t-1} \\ 0, \text{ otherwise.} \end{cases} \qquad g^t_{ui} = \begin{cases} x^*_{ui}, \text{ if } t \le t_0, \\ 1, \text{ if } t > t_0, i = \arg\min_j A^t_{uj} x^{t-1} \\ 0, \text{ otherwise.} \end{cases}$$

By the definition of the greedy step we can write $g^t_{r_t} = x^t_{r_t} - x^{t-1}_{r_t}$.

A *fictitious cut* is defined using a set of auxiliary variables $\hat{x}^t_v$ such that $\hat{x}^t_v = x^t_v$ if $v \in \{r_1, \dots, r_t\}$ and $\hat{x}^t_v = \frac{1}{t}\sum_{\tau=1}^t g^\tau_v$ otherwise.

Lemma 2.2. *For every t it holds that:*

$$(\hat{x}^t)^T A \hat{x}^t - (\hat{x}^{t-1})^T A \hat{x}^{t-1} \le 2(\hat{x}^t - \hat{x}^{t-1})^T A \hat{x}^{t-1} + \frac{4n^2}{t^2}.$$

Proof. Observe that $(\hat{x}^t)^T A \hat{x}^t - (\hat{x}^{t-1})^T A \hat{x}^{t-1} = 2(\hat{x}^t - \hat{x}^{t-1})^T A \hat{x}^{t-1} + (\hat{x}^t - \hat{x}^{t-1})^T A (\hat{x}^t - \hat{x}^{t-1})$.

In order to bound the second term let's express the components of $\hat{x}^t - \hat{x}^{t-1}$. There are three cases:

1. **Case 1.** $u \in S^t, u \notin S^{t-1}$. In this case we have $\hat{x}^t_u = x^t_u = g^t_u$ by the definition of greedy. Also, $x^{t-1}_u = 0$ because u hasn't been assigned yet at time $t-1$. Thus, $\hat{x}^t_u - \hat{x}^{t-1}_u = g^t_u - \hat{x}^{t-1}_u$.
2. **Case 2.** $u \notin S^t$. In this case we have $\hat{x}^t_u = \frac{1}{t}\sum_{\tau=1}^t g^\tau_u$ and $\hat{x}^{t-1}_u = \frac{1}{t-1}\sum_{\tau=1}^{t-1} g^\tau_u$ We have $\hat{x}^t_u - \hat{x}^{t-1}_u = \frac{1}{t} g^t_u - \frac{1}{t(t-1)}\sum_{\tau=1}^{t-1} g^\tau_u = \frac{1}{t}\cdot(g^t_u - \hat{x}^{t-1}_u)$.
3. **Case 3.** $u \in S^{t-1}$. In this case $\hat{x}^t_u - \hat{x}^{t-1}_u = 0$ because $\hat{x}^t_u = \hat{x}^{t-1}_u = x^{t-1}_u$.

Thus, we have that $|\hat{x}^t - \hat{x}^{t-1}|_1 = \sum_u |\hat{x}^t_u - \hat{x}^{t-1}_u|_1 \le 2 + \frac{2}{t}(n-t) = 2\frac{n}{t}$. This implies that $(\hat{x}^t - \hat{x}^{t-1})^T A(\hat{x}^t - \hat{x}^{t-1}) \le \max_{ij} A_{ij} \cdot |\hat{x}^t - \hat{x}^{t-1}|_1^2 \le 4\frac{n^2}{t^2}$, completing the proof. ∎

We denote $q^t = \hat{x}^t - \frac{n}{t}x^t$.

Lemma 2.3. *For all $t \ge t_0$ it holds that:*

$$\mathbb{E}\left[(\hat{x}^t)^T A \hat{x}^t - (\hat{x}^{t-1})^T A \hat{x}^{t-1}\right] \le \frac{4n^2}{t^2} + \frac{4n^2}{(t-1)\sqrt{s_t}} + \frac{2n}{t(n-t+1)}\mathbb{E}\left[\left|Aq^{t-1}\right|_1\right]$$

Proof. We first show the following auxiliary statement.

Proposition 2.4. *For every $u \notin S^{t-1}$ it holds that $\mathbb{E}\left[(g^t_{ui} - \hat{x}^{t-1}_{ui})^T A x^{t-1}\right] \le \frac{2t}{\sqrt{s_t}}$.*

Changed to reflect time-dependent sampling rate.

Proof. We have

$$(g^t_{ui} - \hat{x}^{t-1}_{ui})^T A x^{t-1} = (g^t_{ui} - \tilde{g}^t_{ui}) A x^{t-1} + (\tilde{g}^t_{ui} - \hat{x}^{t-1}_{ui})^T A x^{t-1} \le (g^t_{ui} - \tilde{g}^t_{ui}) A x^{t-1}$$

where the inequality follows from the fact that the optimal greedy choice $\tilde{g}^t$ at step t minimizes $\tilde{g}^t_{ui} A x^{t-1}$ for every u over the choice of i and thus $(\tilde{g}^t_{ui} - \hat{x}^{t-1}_{ui})^T A x^{t-1} \le 0$. W.l.o.g we assume that the optimal greedy choice is $\tilde{g}^t_{u1} = 1$, i.e. the the vertex u is assigned label $i = 1$. Then for the term $(g^t_{ui} - \tilde{g}^t_{ui}) A x^{t-1}$ we have:

$$(g^t_{ui} - \tilde{g}^t_{ui}) A x^{t-1} = \begin{cases} \tilde{N}^t_{u1} - \tilde{N}^t_{u2}, & \text{if } N^t_{u1} < N^t_{u2} \\ 0, & \text{if } N^t_{u1} \ge N^t_{u2}, \end{cases}$$

where $\tilde{N}^t_{ui}$ and N^t_{ui} denote the number of neighbors of u with label i in S^{t-1} and V^{t-1} respectively. By a union bound, we have:

$$\Pr[N^t_{u1} < N^t_{u2}] \le \Pr\left[N^t_{u1} < \frac{s_t}{t-1}\frac{\tilde{N}^t_{u1} + \tilde{N}^t_{u2}}{2}\right] + \Pr\left[N^t_{u2} > \frac{s_t}{t-1}\frac{\tilde{N}^t_{u1} + \tilde{N}^t_{u2}}{2}\right]$$

We introduce notation $\alpha = \frac{\tilde{N}^t_{u1} - \tilde{N}^t_{u2}}{2}$. Since $\mathbb{E}[N^t_{u1}] = \frac{s_t}{t-1}\tilde{N}^t_{u1}$ by an additive Hoeffding bound we have:

$$\Pr\left[N^t_{u1} < \frac{s_t}{t-1}\tilde{N}^t_{u1} - \alpha\frac{s_t}{t-1}\right] \leq e^{-\frac{2\alpha^2 s_t}{(t-1)}}.$$

Bounding the second term similarly we get that $\Pr[N^t_{u1} < N^t_{u2}] \leq 2e^{-\frac{2\alpha^2 s_t}{(t-1)^2}}$. Thus, $\mathbb{E}[(g^t_{ui} - \tilde{g}^t_{ui})Ax^{t-1}] \leq 4\alpha e^{-\frac{2\alpha^2 s_t}{(t-1)^2}}$. Taking the derivative of the latter expression with respect to α we observe that its maximum is achieved if $\alpha^2 = \frac{(t-1)^2}{4s_t}$. This implies that $\mathbb{E}[(g^t_{ui} - \tilde{g}^t_{ui})Ax^{t-1}] \leq \frac{2(t-1)}{\sqrt{s_t}}e^{-1/2} \leq \frac{2t}{\sqrt{s_t}}$.

By Lemma 2.2 it suffices to bound $\mathbb{E}\left[(\hat{x}^t - \hat{x}^{t-1})^T A\hat{x}^{t-1}\right]$. We have:

$$(\hat{x}^t - \hat{x}^{t-1})^T A\hat{x}^{t-1} = \frac{n}{t-1}(\hat{x}^t - \hat{x}^{t-1})^T Ax^{t-1} + (\hat{x}^t - \hat{x}^{t-1})^T Aq^{t-1}.$$

We bound the first term using Proposition 2.4. If $u \notin S^{t-1}$, but $u \in S^t$ then $\hat{x}^t_u - \hat{x}^{t-1}_u = g^t_u - \hat{x}^{t-1}_u$. Hence, $\mathbb{E}[(\hat{x}^t_{ui} - \hat{x}^{t-1}_{ui})Ax^{t-1}] = \frac{2t}{\sqrt{s_t}}$. If $u \notin S^t$ then $\hat{x}^t_u - \hat{x}^{t-1}_u = \frac{1}{t}\left(g^t_u - \hat{x}^{t-1}_u\right)$ and $\mathbb{E}[(\hat{x}^t_{ui} - \hat{x}^{t-1}_{ui})Ax^{t-1}] = \frac{2}{\sqrt{s_t}}$. Since the total number of such vertices is $n - t$ we have:

$$\mathbb{E}\left[(\hat{x}^t - \hat{x}^{t-1})^T Ax^{t-1}\right] = \frac{4t}{\sqrt{s_t}} + \frac{4(n-t)}{\sqrt{s_t}} = \frac{4n}{\sqrt{s_t}}.$$

Thus, the first term is bounded by $\frac{4n^2}{(t-1)\sqrt{s_t}}$ as desired.

Now we bound the second term. Taking expectation we have:

$$\mathbb{E}\left[(\hat{x}^t - \hat{x}^{t-1})^T Aq^{t-1}\right] = \mathbb{E}_{S^{t-1}}\left[\mathbb{E}_{r_t}\left[(\hat{x}^t - \hat{x}^{t-1})^T Aq^{t-1}|S^{t-1}\right]\right]$$

$$= \mathbb{E}_{S^{t-1}}\left[\sum_v \mathbb{E}_{r_t}\left[(\hat{x}^t_v - \hat{x}^{t-1}_v)A_v q^{t-1}|S^{t-1}\right]\right] = \mathbb{E}_{S^{t-1}}\left[\sum_{v,i} A_{v,i}q^{t-1}\mathbb{E}_{r_t}\left[(\hat{x}^t_{v,i} - \hat{x}^{t-1}_{v,i})|S^{t-1}\right]\right]$$

$$\geq \mathbb{E}_{S^{t-1}}\left[-\sum_v \left|A_v q^{t-1}\right|_1 \mathbb{E}_{r_t}\left[\left|\hat{x}^t_v - \hat{x}^{t-1}_v\right|_1 |S^{t-1}\right]\right] \geq \mathbb{E}_{S^{t-1}}\left[-\frac{2n}{t(n-t+1)}\sum_v \left|A_v q^{t-1}\right|_1\right] = -\frac{2n}{t(n-t+1)}\mathbb{E}\left[\left|Aq^{t-1}\right|_1\right]$$

where in the second inequality we bound $\mathbb{E}_{r_t}\left[\left|\hat{x}^t_v - \hat{x}^{t-1}_v\right| |_1 S^{t-1}\right]$ as follows:

$$\mathbb{E}_{r_t}\left[\left|\hat{x}^t_v - \hat{x}^{t-1}_v\right|_1 |S^{t-1}\right] = \Pr[r_t = v|S^{t-1}] \cdot |g^t_v - \hat{x}^{t-1}_v|_1 + \Pr[r_t \neq v|S^{t-1}] \cdot |g^t_v - \hat{x}^{t-1}_v|_1/t$$
$$= \cdot|g^t_v - \hat{x}^{t-1}_v|_1\left(\frac{1}{n-t+1} + \frac{1}{t}\left(1 - \frac{1}{n-t+1}\right)\right) = \frac{n}{t(n-t+1)}|g^t_v - \hat{x}^{t-1}_v|_1 \leq \frac{2n}{t(n-t+1)}. \quad \blacksquare$$

The following lemma is proved in Appendix B.1.

Moved the proof to the appendix.

Lemma 2.5. *For all $t \geq t_0$ it holds that $\mathbb{E}\left[|Aq^t|_1\right] = O\left(n(n-t)\left(\frac{1}{\sqrt{t}} + \frac{1}{\epsilon t}\right)\right) = O\left(\frac{n(n-t)}{\sqrt{t}}\right)$*

Lemma 2.6. *For all $t \geq t_0$ it holds that $\mathbb{E}\left[(\hat{x}^t)^T A\hat{x}^t\right] - \mathbb{E}\left[(\hat{x}^{t_0})^T A\hat{x}^{t_0}\right] \leq O(\epsilon n^2)$.*

Proof. The proof follows from Lemma 2.3 and Lemma 2.5. Using Lemma 2.3 applied for τ from t_0 to t:

$$\mathbb{E}\left[(\hat{x}^t)^T A\hat{x}^t - (\hat{x}^{t_0})^T A\hat{x}^{t_0}\right] \leq \sum_{\tau=t_0+1}^{t}\frac{4n^2}{\tau^2} + \sum_{\tau=t_0+1}^{t}\frac{4n^2}{(\tau-1)\sqrt{s_\tau}} + \sum_{\tau=t_0+1}^{t}\frac{2n}{\tau(n-\tau+1)}\mathbb{E}\left[\left|Aq^{\tau-1}\right|_1\right]$$
$$\leq O(\epsilon^2 n^2) + \sum_{\tau=t_0+1}^{t}\frac{4n^2}{(\tau-1)\sqrt{s_\tau}} + \sum_{\tau=t_0+1}^{t} O\left(\frac{n^2}{\tau^{3/2}}\right) \leq O(\epsilon^2 n^2) + O(\epsilon n^2) + \sum_{\tau=t_0+1}^{t} O\left(\frac{n^2}{\tau^{3/2}}\right)$$
$$\leq O(\epsilon n^2) + O\left(\frac{n^2}{\sqrt{t_0}}\right) \leq O(\epsilon n^2),$$

where the second inequality is by Lemma 2.5, the third is because $\sum_{\tau=t_0+1}^{t} \frac{4n^2}{(\tau-1)\sqrt{s_\tau}} \leq \frac{\epsilon}{n^{1/3}} \sum_{\tau=t_0}^{n} \tau^{-2/3} = O(\epsilon n)$ and the fourth is by $\sum_{\tau=t_0+1}^{n} \frac{1}{\tau^{3/2}} \leq \int_{\tau=t_0}^{\infty} \frac{1}{\tau^{3/2}} d\tau = O\left(\frac{1}{\sqrt{t_0}}\right)$. ∎

Time-dependent sampling rate in the analysis of the approxiamtion guarantee.

Finally, we are ready to prove Theorem 2.1.

Proof of Theorem 2.1. Follows from Lemma 2.6 applied to $t = n$ since $\hat{x}^{t_0} = x^*$ is the optimal cut. ∎

3 Fast algorithm for r-CSPs

In order to generalize our algorithms to general r-CSPs we need to introduce a notion of a critical constraint.

Definition 3.1 (Critical r-tuples and constraints)**.** *For a given partial assignment S and a variable v_t an r-tuple $(i_1, \ldots, i_r)$ is critical, if $i_1 = t$ and $i_2, \ldots, i_r \in S$. A constraint is critical if it is defined on a set of variables, whose indices form a critical r-tuple, and is not satisfied by S.*

Changed the definition.

Algorithm 2: Fast Greedy PTAS with subsampling.

input : A k-ary r-CSP instance over n variables, parameter ϵ.

1 Pick a sample S_1 of $t_1 = O(\log^2 k/\epsilon^4)$ variables uniformly at random without replacement
2 Pick a sample $S_0 \subseteq S_1$ of $t_0 = 1/\epsilon^2$ variables uniformly at random without replacement
3 **for** *each of the k^{t_0} possible assignments of values to variables in S_0* **do**
4 $S_0^{t_0} = S_0, t = t_0 + 1, s_t = O\left(\frac{n^{2/3}k^4}{t^{2/3}\epsilon^2}\right)$.
5 **for** *each variable $v \in S_1 \setminus S_0$ in random order* **do**
6 Pick a sample V^t of s_t critical r-tuples uniformly without replacement from the set of all critical r-tuples for v and assignment S^{t-1}.
7 Assign variable v the value maximizing the number of satisfied critical constraints in V^t.
8 $S_0^t = S_0^{t-1} \cup \{v\}, t = t + 1$
9 Assign values to variables in S_1 according to the best assignment w found over all iterations in line 3
10 **for** *each variable $v \in V \setminus S_1$ in random order* **do**
11 $S_1^{t_1} = S_1$, $t = t_1 + 1, s_t = O\left(\frac{n^{2/3}k^4}{t^{2/3}\epsilon^2}\right)$.
12 Pick a sample V^t of s_t critical r-tuples uniformly without replacement from the set of all critical r-tuples for v and assignment S^{t-1}.
13 Assign variable v the value maximizing the number of satisfied critical constraints in V^t.
14 $S_1^t = S_1^{t-1} \cup \{v\}, t = t + 1$
15 Output the assignment constructed in the loop on line 10

Changed the description of the sampling process in the algorithm as per Comment 6 and introduced time-dependent sample size.

The assignment of values to the variables is represented by a vector $x \in \{0,1\}^{nk}$, where $x_{ui} = 1$ iff the variable u is assigned value i. The objective function can be written as a multilinear function:

$$A(x^{(1)}, \ldots, x^{(r)}) = \sum_{\substack{1 \leq u_1, \ldots, u_r \leq n \\ 1 \leq i_1, \ldots, i_r \leq k}} A_{u_1, i_1, \ldots, u_r, i_r} x^{(1)}_{u_1, i_1} \ldots, x^{(r)}_{u_r, i_r},$$

where A is an nk-dimensional array symmetric under permutation of the r indices (u_j, i_j). The variables x^t_{ui} and $\hat{x}^t_{ui}$, are defined as in Section 2, except that we now call them assignments instead of cuts as before. Random variable r_t corresponds to the variable chosen at random at step t. Random variable g^t denote the optimum greedy choice of the assignment for this variable with respect to all its critical constraints in S^{t-1}.

Random variable $\tilde{g}^t$ denotes the optimum such greedy choice but only with respect to constraints in V^t. We also define an array A^t sampled from A at iteration t of the algorithm as follows:

$$A^t_{u_1,i_1,u_2,i_2,\ldots,u_r,i_r} = \begin{cases} A_{u_1,i_1,u_2,i_2,\ldots,u_r,i_r}, \text{ if}(u_1 = r_t, u_2, u_3, \ldots, u_r) \in V_t, \text{i.e. the constraint is critical.} \\ 0, \text{ otherwise.} \end{cases}$$

We also introduce notation an nk-dimensional vector $A(\cdot, x, \ldots, x)$ with components defined as $A(\cdot, x, \ldots, x)_{ui} = A(e_{ui}, x, \ldots, x)$.

Recall that $q^t = \hat{x}^t - \frac{n}{t}x^t$. We use notation S_1^t to denote the set of variables, which are assigned values after iteration t of the loop on line 10 of Algorithm 2.

Lemma 3.1. *(Analog of Lemma 2.3) For every $t \geq t_1$ it holds that:*

$$\mathbb{E}\left[A(\hat{x}^t, \ldots, \hat{x}^t) - A(\hat{x}^{t-1}, \ldots, \hat{x}^{t-1})\right] \leq \frac{2^{r+2}n^r}{t^r} + \frac{2k^2n^r}{t\sqrt{s_t}} + \frac{2n}{t(n-t+1)}\mathbb{E}\left[|A(\cdot, q^{t-1}, \ldots, q^{t-1})|_1\right]$$

Corrected statement as per Comment 2.

Proof. First we prove the following proposition, which generalizes Proposition 2.4.

Proposition 3.2. *For every $u \notin S_1^{t-1}$ it holds that $\mathbb{E}\left[A(g^t_{ui} - \hat{x}^{t-1}_{ui}, x^{t-1}, \ldots, x^{t-1})\right] \leq \frac{2kt^{r-1}}{\sqrt{s_t}}$*

Proof. The proof generalizes the proof of Proposition 2.4. There are two differences: we now work with an alphabet of size $k > 2$ and have an instance of an r-CSP instead of an instance of Max-Cut. The size of the alphabet can be taken care of by a union bound, which introduces an extra factor of k in the result. To handle larger arity of the CSP recall the definition of a *critical constraint* (Definition 3.1). It captures the intuition that the hardest type of r-CSP predicates for us are predicates r-LIN, i.e. parities on at most r variables which can't be satisfied until all variables are assigned values (a special case 2-LIN corresponds to Max-Cut). The key observation is that at time t there can be at most t^{r-1} critical constraints, which have r_t as the unassigned variable since there are at most that many cricical r-tuples. Using these two observations the proof follows the lines of the proof of Proposition 2.4 with t replaced by t^{r-1}, which is used as a new bound on the size of the set we sample from. ∎

The rest of the proof of Lemma 3.1 is given in Appendix B.3. ∎

Proof added.

Lemma 3.3. *(Analog of Lemma 2.5) For every t and $\sigma = O\left(\frac{n^r}{\sqrt{t}}\sqrt{\frac{n-t}{n}}\right)$ it holds that:*

$$\mathbb{E}\left[|A(\cdot, q^t, \ldots, q^t)|_1\right] = O(\sigma).$$

Proof.

Lemma 3.4. *For every $t \geq t_0$ it holds that:*

$$\mathbb{E}\left[A(x^t, \ldots, x^t) - A(\hat{x}^{t_0}, \ldots, \hat{x}^{t_0})\right] \leq O(\epsilon n^r).$$

Proof. By Lemma 3.1 and Lemma 3.3 we have:

$$\mathbb{E}\left[A(x^t, \ldots, x^t) - A(\hat{x}^{t_0}, \ldots, \hat{x}^{t_0})\right] \leq O\left(\sum_{\tau=t_0}^{t} \frac{2^{r+2}n^r}{\tau^r} + \sum_{\tau=t_0}^{t} \frac{2k^2n^r}{\tau\sqrt{s_\tau}} + \sum_{\tau=t_0}^{t} \frac{2n}{\tau(n-\tau+1)} \frac{n^r}{\sqrt{\tau}}\sqrt{\frac{n-\tau}{n}}\right)$$

$$= O\left(\epsilon n^r + 2k^2n^r \int_{t_0}^{t} \frac{1}{\tau\sqrt{s_\tau}}d\tau + n^r \sum_{\tau=t_0}^{n/2} \frac{1}{\tau^{3/2}} + n^{r+1/2} \sum_{n/2}^{n} \frac{\sqrt{n-\tau}}{\tau^{3/2}(n-\tau+1)}\right)$$

$$= O\left(\epsilon n^r + \epsilon n^r + \frac{n^r}{\sqrt{t_0}} + n^{r-1} \int_1^{n/2} \frac{1}{\sqrt{z}}dz\right) = O(\epsilon n^r),$$

where the first equality follows by direct calculations, the third inequality uses the definition $s_\tau = O\left(\frac{n^{2/3}k^4}{\tau^{2/3}\epsilon^2}\right)$ and the last two equalities follow by direct calculations . ∎

Modified calculations

We use notation $x^t_{(y)}$, $\hat{x}^t_{(y)}$ and $q^t_{(y)}$ to denote the corresponding variables at time t in the greedy process starting from the assignment y instead of the optimum assignment x^* as before. Let Y be the set of all $k^{t^0} = 2^{O(\log k/\epsilon^2)}$ possible assignments of variables the set S_0 that the Algorithm 2 tries in the line 3.

The proof of the following lemma is given in Appendix B.2.

Lemma 3.5. *For every $t \geq t_0$ and $\sigma = O\left(\frac{n^{r-1}}{\sqrt{t}}\sqrt{\frac{n-t}{n}}\right)$ it holds that:*

$$\mathbb{E}\left[\max_{y\in Y}\left|A(\cdot, q^t_{(y)}, \ldots, q^t_{(y)})\right|_1\right] = O\left(\frac{\sigma \log k}{\epsilon}\right),$$
$$\mathbb{E}\left[\max_{y\in Y}\left|A(q^t_{(y)}, q^t_{(y)}, \ldots, q^t_{(y)})\right|\right] = O\left(\frac{\sigma \log k}{\epsilon}\right)$$

Now we are ready to complete the analysis of Algorithm 2, giving the proof of Theorem 1.2

Proof of Theorem 1.2. Let w denote the best assignment found over all iterations of the loop in line 3, which is used as a seed assignment to the vertices in S_1 for the second loop in line 10. We have:

$$\mathbb{E}\left[A\left(\hat{x}^n_w, \hat{x}^n_w, \ldots, \hat{x}^n_w\right)\right] =$$
$$\mathbb{E}\left[A\left(\hat{x}^{t_1}_{(w)}, \hat{x}^{t_1}_{(w)}, \ldots, \hat{x}^{t_1}_{(w)}\right)\right] + \sum_{t_1 \leq t \leq n} \mathbb{E}\left[A\left(\hat{x}^t_{(w)}, \hat{x}^t_{(w)}, \ldots, \hat{x}^t_{(w)}\right) - A\left(\hat{x}^{t-1}_{(w)}, \hat{x}^{t-1}_{(w)}, \ldots, \hat{x}^{t-1}_{(w)}\right)\right] \quad (1)$$

By Lemma 3.1 for each term in the sum in (1) we have:

$$\mathbb{E}\left[A\left(\hat{x}^t_{(w)}, \hat{x}^t_{(w)}, \ldots, \hat{x}^t_{(w)}\right) - A\left(\hat{x}^{t-1}_{(w)}, \hat{x}^{t-1}_{(w)}, \ldots, \hat{x}^{t-1}_{(w)}\right)\right] \leq \frac{2^{r+2}n^r}{t^r} + \frac{4kn^2t^{r-3}}{\sqrt{s}} + \frac{2n}{t(n-t+1)}\mathbb{E}\left[|A\left(\cdot, q^{t-1}_{(w)}, \ldots, q^{t-1}_{(w)}\right)|_1\right]$$

We have that $\left|A\left(\cdot, q^{t-1}_{(w)}, \ldots, q^{t-1}_{(w)}\right)\right|_1 \leq \max_{y\in Y}\left|A\left(\cdot, q^{t-1}_{(y)}, \ldots, q^{t-1}_{(y)}\right)\right|_1$. Thus, by Lemma 3.5 the last term in the expression above can be bounded by $O\left(\frac{n}{t(n-t+1)}\frac{n^{r-1}}{\sqrt{t}}\sqrt{\frac{n-t}{n}}\frac{\log k}{\epsilon}\right)$. Bounding the sum by integral as in the proof of Lemma 3.4 we get that the sum in the second term of 1 is bounded as $O\left(\frac{n^r}{\sqrt{t_1}}\frac{\log k}{\epsilon} + \epsilon n^r\right)$. This is where we use the fact that $t_1 = \Theta\left(\frac{\log^2 k}{\epsilon^4}\right)$ to conclude that the sum is bounded by $O(\epsilon n^r)$.

For the first term $\mathbb{E}\left[A\left(\hat{x}^{t_1}_{(w)}, \hat{x}^{t_1}_{(w)}, \ldots, \hat{x}^{t_1}_{(w)}\right)\right]$ of (1) we have:

$$\begin{aligned}
\mathbb{E}\left[A\left(\hat{x}^{t_1}_{(w)}, \hat{x}^{t_1}_{(w)}, \ldots, \hat{x}^{t_1}_{(w)}\right)\right] &\leq \mathbb{E}\left[A\left(\frac{n}{t_1}\hat{x}^{t_1}_{(w)}, \frac{n}{t_1}\hat{x}^{t_1}_{(w)}, \ldots, \frac{n}{t_1}\hat{x}^{t_1}_{(w)}\right)\right] + \mathbb{E}\left[\left|A\left(q^{t_1}_{(w)}, q^{t_1}_{(w)}, \ldots, q^{t_1}_{(w)}\right)\right|\right] \\
&\leq \mathbb{E}\left[A\left(\frac{n}{t_1}\hat{x}^{t_1}_{(w)}, \frac{n}{t_1}\hat{x}^{t_1}_{(w)}, \ldots, \frac{n}{t_1}\hat{x}^{t_1}_{(w)}\right)\right] + \mathbb{E}\left[\max_{y\in Y}\left|A\left(q^{t_1}_{(y)}, q^{t_1}_{(y)}, \ldots, q^{t_1}_{(y)}\right)\right|\right] \\
&\leq \mathbb{E}\left[A\left(\frac{n}{t_1}x^{t_1}_{(w)}, \frac{n}{t_1}x^{t_1}_{(w)}, \ldots, \frac{n}{t_1}x^{t_1}_{(w)}\right)\right] + O(\epsilon n^{r-1}) \\
&\leq \mathbb{E}\left[A\left(\frac{n}{t_1}x^{t_1}_{(x^*)}, \frac{n}{t_1}x^{t_1}_{(x^*)}, \ldots, \frac{n}{t_1}x^{t_1}_{(x^*)}\right)\right] + O(\epsilon n^{r-1}) \\
&\leq \mathbb{E}\left[A\left(\hat{x}^{t_1}_{(x^*)}, \hat{x}^{t_1}_{(x^*)}, \ldots, \hat{x}^{t_1}_{(x^*)}\right)\right] + \mathbb{E}\left[\left|A\left(q^{t_1}_{(x^*)}, q^{t_1}_{(x^*)}, \ldots, q^{t_1}_{(x^*)}\right)\right|\right] + O(\epsilon n^{r-1}) \\
&\leq \mathbb{E}\left[A\left(\hat{x}^{t_1}_{(x^*)}, \hat{x}^{t_1}_{(x^*)}, \ldots, \hat{x}^{t_1}_{(x^*)}\right)\right] + \mathbb{E}\left[\max_{y\in Y}\left|A\left(q^{t_1}_{(y)}, q^{t_1}_{(y)}, \ldots, q^{t_1}_{(y)}\right)\right|\right] + O(\epsilon n^{r-1}) \\
&\leq \mathbb{E}\left[A\left(\hat{x}^{t_1}_{(x^*)}, \hat{x}^{t_1}_{(x^*)}, \ldots, \hat{x}^{t_1}_{(x^*)}\right)\right] + O\left(\frac{n^{r-1}}{\sqrt{t_1}}\sqrt{\frac{n-t_1}{n}}\frac{\log k}{\epsilon}\right) + O(\epsilon n^{r-1}) \\
&\leq \mathbb{E}\left[A\left(\hat{x}^{t_0}_{(x^*)}, \hat{x}^{t_0}_{(x^*)}, \ldots, \hat{x}^{t_0}_{(x^*)}\right)\right] + O\left(\epsilon n^r\right) \\
&= OPT + O(\epsilon n^r),
\end{aligned}$$

where the first inequality is by the triangle inequality and the definition of $q_{(1)}^{t_1}$, the second inequality is by a union bound, the third inequality is by Lemma 3.5, using the fact that $O\left(\frac{n^{r-1}}{\sqrt{t_1}}\sqrt{\frac{n-t_1}{n}}\frac{\log k}{\epsilon}\right) = O\left(\frac{n^{r-1}}{\sqrt{t_1}}\frac{\log k}{\epsilon}\right) = O(\epsilon n^{r-1})$, the fourth inequality follows since $x^* \in Y$, the fifth inequality holds by the triangle inequality and the definition of $q_{x^*}^{t_1}$, the sixth inequality is by a union bound, the seventh inequality is by Lemma 3.5, the eighth inequality is by Lemma 3.4 and uses the fact that $t_1 = \Theta(\log^2 k/\epsilon^4)$ and the last equality holds since by definition $A\left(\hat{x}_{(x^*)}^{t_0}, \hat{x}_{(x^*)}^{t_0}, \ldots, \hat{x}_{(x^*)}^{t_0}\right) = OPT$.

Finally, the analysis of the running time follows from computing the overall size of all samples as done for Algorithm 1. In order to implement random sampling from the set of all critical r-tuples we assume that the instance is given the input is given as the r-dimensional matrix A. Then we can check whether a randomly sampled critical r-tuple corresponds to a critical constraint in constant time. ∎

Discussion of the running time and implementation of sampling.

4 Conclusion

In this section we briefly explain how to use our algorithms in a parallel setting, when we have m identical machines available as in modern massive parallel computational models [FMS+10, KSV10, GSZ11, BKS13, ANOY14]. This allows to reduce the total computational time of our algorithms by a factor of m. The loop in the line 3 is very easy to parallelize since different iterations can be done independently on different machines. The greedy pass in the line 5, however, is executed only once and because the decisions made in the previous iterations affect the greedy choices in the subsequent steps we have to modify the algorithm in order to make it parallel. We can do this by partitioning the work into $O(\log_{1+\epsilon} n)$ supersteps. Instead of processing one vertex at a time in the loop in the line 5 in every superstep we increase the size of the current set of processed vertices by a $(1+\epsilon)$ multiplicative factor. Consider one such superstep τ when we increase the set of vertices from $S^\tau - 1$ to S^τ. In this superstep we process all new vertices in $S^\tau \setminus S^{\tau-1}$ independently in random order, dividing the work uniformly between m machines. Crucially, when processing a vertex $r_t \in S^\tau \setminus S^{\tau-1}$ we can no longer take the random sample V_t from the entire set of vertices $r_1, \ldots, r_{t-1}$ processed before it. However, we can take a random sample from $S^{\tau-1}$, which was computed in the previous superstep and differs from $\{r_1, \ldots, r_{t-1}\}$ on at most an ϵ-fraction of points. This suffices for the analysis in Section 3 to still yield a PTAS.

It remains open whether our results can be reproduced in the streaming model, i.e. whether there exist both sublinear time and space algorithms for dense r-CSPs.

5 Acknowledgment

We would like to thank an anonymous reviewer for SODA'15 for multiple insightful comments, including a suggestion to use time-dependent sample size.

References

[AdlVKK03] Noga Alon, Wenceslas Fernandez de la Vega, Ravi Kannan, and Marek Karpinski. Random sampling and approximation of max-csps. *J. Comput. Syst. Sci.*, 67(2):212–243, 2003.

[AFNS09] Noga Alon, Eldar Fischer, Ilan Newman, and Asaf Shapira. A combinatorial characterization of the testable graph properties: It's all about regularity. *SIAM J. Comput.*, 39(1):143–167, 2009.

[AGM12] Kook Jin Ahn, Sudipto Guha, and Andrew McGregor. Graph sketches: sparsification, spanners, and subgraphs. In Michael Benedikt, Markus Krötzsch, and Maurizio Lenzerini, editors, *PODS*, pages 5–14. ACM, 2012.

[AKK99] Sanjeev Arora, David R. Karger, and Marek Karpinski. Polynomial time approximation schemes for dense instances of np-hard problems. *J. Comput. Syst. Sci.*, 58(1):193–210, 1999.

[ANOY14] Alexandr Andoni, Aleksandar Nikolov, Krzysztof Onak, and Grigory Yaroslavtsev. Parallel algorithms for geometric graph problems. In David B. Shmoys, editor, *STOC*, pages 574–583. ACM, 2014.

[BHHS11] Boaz Barak, Moritz Hardt, Thomas Holenstein, and David Steurer. Subsampling mathematical relaxations and average-case complexity. In *SODA*, pages 512–531, 2011.

[BK96] András A. Benczúr and David R. Karger. Approximating s-t minimum cuts in $\tilde{o}(n^2)$ time. In Gary L. Miller, editor, *STOC*, pages 47–55. ACM, 1996.

[BKS13] Paul Beame, Paraschos Koutris, and Dan Suciu. Communication steps for parallel query processing. pages 273–284, 2013.

[BSS14] Joshua D. Batson, Daniel A. Spielman, and Nikhil Srivastava. Twice-ramanujan sparsifiers. *SIAM Review*, 56(2):315–334, 2014.

[CEG95] Ran Canetti, Guy Even, and Oded Goldreich. Lower bounds for sampling algorithms for estimating the average. *Inf. Process. Lett.*, 53(1):17–25, 1995.

[dlV96] Wenceslas Fernandez de la Vega. Max-cut has a randomized approximation scheme in dense graphs. *Random Struct. Algorithms*, 8(3):187–198, 1996.

[dlVKKV05] Wenceslas Fernandez de la Vega, Marek Karpinski, Ravi Kannan, and Santosh Vempala. Tensor decomposition and approximation schemes for constraint satisfaction problems. In Harold N. Gabow and Ronald Fagin, editors, *STOC*, pages 747–754. ACM, 2005.

[dlVKM07] Wenceslas Fernandez de la Vega and Claire Kenyon-Mathieu. Linear programming relaxations of maxcut. In Nikhil Bansal, Kirk Pruhs, and Clifford Stein, editors, *SODA*, pages 53–61. SIAM, 2007.

[FK99] Alan M. Frieze and Ravi Kannan. Quick approximation to matrices and applications. *Combinatorica*, 19(2):175–220, 1999.

[FMS+10] Jon Feldman, S. Muthukrishnan, Anastasios Sidiropoulos, Clifford Stein, and Zoya Svitkina. On distributing symmetric streaming computations. *ACM Transactions on Algorithms*, 6(4), 2010. Previously in SODA'08.

[GG06] Ioannis Giotis and Venkatesan Guruswami. Correlation clustering with a fixed number of clusters. In *SODA*, pages 1167–1176. ACM Press, 2006.

[GGR98] Oded Goldreich, Shafi Goldwasser, and Dana Ron. Property testing and its connection to learning and approximation. *J. ACM*, 45(4):653–750, 1998.

[GKP12] Ashish Goel, Michael Kapralov, and Ian Post. Single pass sparsification in the streaming model with edge deletions. *CoRR*, abs/1203.4900, 2012.

[GSZ11] Michael T. Goodrich, Nodari Sitchinava, and Qin Zhang. Sorting, searching, and simulation in the MapReduce framework. In *ISAAC*, pages 374–383, 2011.

[KL13] Jonathan A. Kelner and Alex Levin. Spectral sparsification in the semi-streaming setting. *Theory Comput. Syst.*, 53(2):243–262, 2013.

[KSV10] Howard J. Karloff, Siddharth Suri, and Sergei Vassilvitskii. A model of computation for mapreduce. In Moses Charikar, editor, *SODA*, pages 938–948. SIAM, 2010.

[KTM+14] M. Kapralov, Y. Tat Lee, C. Musco, C. Musco, and A. Sidford. Single Pass Spectral Sparsification in Dynamic Streams. *ArXiv e-prints*, July 2014.

[MS08] Claire Mathieu and Warren Schudy. Yet another algorithm for dense max cut: go greedy. In Shang-Hua Teng, editor, *SODA*, pages 176–182. SIAM, 2008.

[Sch12] Warren Schudy. *Approximation Schemes for Inferring Rankings and Clusterings from Pairwise Data*. PhD thesis, Brown Univeristy, 2012.

[SS11] Daniel A. Spielman and Nikhil Srivastava. Graph sparsification by effective resistances. *SIAM J. Comput.*, 40(6):1913–1926, 2011.

[ST11] Daniel A. Spielman and Shang-Hua Teng. Spectral sparsification of graphs. *SIAM J. Comput.*, 40(4):981–1025, 2011.

[YZ14] Yuichi Yoshida and Yuan Zhou. Approximation schemes via sherali-adams hierarchy for dense constraint satisfaction problems and assignment problems. In Moni Naor, editor, *ITCS*, pages 423–438. ACM, 2014.

A Lower bound

Theorem A.1. *There exist constants $c_1, c_2 > 0$ such that any algorithm that queries less than $c_1 n/\epsilon^2$ edges has to incur additive error at least $c_2 \epsilon n^2$ for* Max-Cut.

Proof. Consider the following random family of graphs. Let $V = V_0 \cup V_1 \cup V_2$, where $|V_0| = |V_1| = 4n/9$ and $|V_2| = n/9$. The subgraph induced by $V_0 \cup V_1$ is a complete bipartite graph $K_{4n/9,4n/9}$. For every vertex $v \in V_2$ we construct the set of edges adjacent to v by picking a random value $r_v \in \{0,1\}$ uniformly at random and choosing drawing edges from v to vertices in V_{r_v} independently with probability $1/2 + \epsilon$ each and edges to vertices in V_{1-r_v} independently with probability $1/2$ each. With high probability this gives us two random sets of neighbors $S_{r_v} \subseteq V_{r_v}$ of size $2n/9 \pm o(n)$ and $S_{1-r_v} \subseteq V_{1-r_v}$ of size $(1+\epsilon)2n/9 \pm o(n)$. In the presentation below we will often omit the $o(n)$ terms since they don't matter for the analysis.

First, we give a high-probability bound on the cost of the optimum solution for such instances.

Proposition A.2. *With high probability the cost of the optimum solution for the class of instances constructed as described above is equal to $18n^2/81 + 2\epsilon n^2/81 + o(n^2)$.*

Proof. We use sets C_0 and C_1 for the two sides of the cut. First, we show that the optimum solution for this family of instances always places V_0 in C_0, V_1 in C_1 and every vertex $v \in V_2$ in C_{r_v} and thus has the size of the cut equal to $OPT = |V_0||V_1| + |V_2||S_{r_v}| = 4n/9 \cdot 4n/9 + n/9 \cdot (1+\epsilon)2n/9 = 18n^2/81 + 2\epsilon n^2/81$. Indeed, we can choose C_0 so that the majority of vertices from V_0 are placed in C_0. We denote this majority as $V_0^0 = V_0 \cap C_0$ and define analogously $V_j^i = V_j \cap C_i$ for $i, j \in \{0,1\}$. Note that the total number of cut edges in the subgraph induced by $V_0 \cup V_1$ is $|V_0^0| \cdot |V_1^1| + |V_0^1| \cdot |V_1^0|$, which is maximized if $|V_1^1| = |V_1| = 4n/9$. Thus, the number of such edges is at most $|V_0^0| 4n/9$. The number of cut edges adjacent to the vertices in V_2 is at most $|V_2| \cdot (|S_{r_v}| + |S_{1-r_v}|) = |V_2| \cdot (4n/9 + 2\epsilon n/9) = 4n^2/81 + 2\epsilon n^2/81$. Thus, the total number of cut edges is at most $|V_0^0| 4n/9 + 4n^2/81 + 2\epsilon n^2/81$. This implies that $|V_0^0| \geq (OPT - 4n^2/81 - 2\epsilon n^2/81)/(4n/9) = 7n/18 > n/3$. Now assume that V_1^0 is non-empty and consider any vertex in $v \in V_1^0$. The number of edges from v to vertices in C_0 is at least $|V_0^0| > n/3$, while the number of edges from v to vertices in C_1 is at most $|V_0^1| + |V_2| < n/9 + n/9 = 2n/9$. By moving v to C_1 we can improve the solution, which means that V_1^0 is empty. Similarly, V_0^1 is empty and hence $V_0^0 = V_0$ and $V_1^1 = V_1$. Given this the optimum solution places every vertex $v \in V_2$ greedily on the side r_v, which completes the analysis of the cost of the optimum solution. ∎

Using Yao's principle it suffices to consider performance of deterministic algorithms under the hard distribution that we have constructed.

Definition A.1 (Bad vertices)**.** *A vertex $v \in V_2$ is bad if there exists a cut (C_0, C_1) of $V_0 \cup V_1$ such that the degree of v in on of the parts V_j^i the partition deviates from the expectation by more than $(4n/9)^{3/4}$.*

Proposition A.3. *With high probability there are at most $n^{3/4}$ vertices.*

Proof. We denote $m = 4n/9$. Fix a vertex $v \in V_2$. W.l.o.g $r_v = 0$. For a set S let $n_v(S)$ denote the number of neighbors of v in S. We have $\mathbb{E}[n_v(V_0^0 \cup V_1^0)] = \mathbb{E}[n_v(V_0^0)] + \mathbb{E}[n_v(V_1^0)] = |V_0^0|(1+\epsilon)1/2 + |V_1^0|1/2 = 1/2(|V_0^0 \cup V_1^0|) + \epsilon/2|V_0^0|$. By a union bound $\Pr[n_v(V_0^0 \cup V_1^0) - \mathbb{E}[n_v(V_0^0 \cup V_1^0)] > 2m^{3/4}] \leq \Pr[n_v(V_0^0) - \mathbb{E}[n_v(V_0^0)] > m^{3/4}] + \Pr[n_v(V_1^0) - \mathbb{E}[n_v(V_1^0)] > m^{3/4}]$. By Hoeffding bound we have $\Pr[n_v(V_0^0) - \mathbb{E}[n_v(V_0^0)] > m^{3/4}] \leq e^{-\frac{2m^{3/2}}{|V_0^0|}} \leq e^{-2\sqrt{m}}$ and similarly $\Pr[n_v(V_1^0) - \mathbb{E}[n_v(V_1^0)] > m^{3/4}] \leq e^{-2\sqrt{m}}$. Thus $\Pr[n_v(V_0^0 \cup V_1^0) - \mathbb{E}[n_v(V_0^0 \cup V_1^0)] > 2m^{3/4}] \leq 2e^{-2\sqrt{m}}$. The probability that for a fixed cut there are at least $n^{3/4}$ bad vertices is thus $\sum_{t=n^{3/4}}^{|V_2|} \binom{|V_2|}{t} p^t (1-p)^{|V_2|-t} \leq \sum_{t=n^{3/4}}^{|V_2|} \binom{|V_2|}{t} p^t \leq p^{n^{3/4}} \sum_{t=n^{3/4}}^{|V_2|} \binom{|V_2|}{t} \leq p^{n^{3/4}} 2^{|V_2|} \leq (2e^{-2\sqrt{4n/9}})^{n^{3/4}} 2^{n/9}$. Finally, taking a union bound over all $2^{2m} = 2^{8n/9}$ possible cuts induced on $V_0 \cup V_1$ we get that the probability that there exists at least one such cut for which at least $n^{3/4}$ vertices are bad is at most $2^{n+n^{3/4}} \cdot e^{-4/3n^{5/4}} = 2^{-\Omega(n^{1/4})}$, which is exponentially small in n. ∎

The previous lemma essentially allows us to assume that there are no bad vertices, becuase their total contribution to the cut is at most $o(n^2)$, which is negligible. Now we define *biased vertices*, which intuitively are vertices in V_2 which are easy for an algorithm to detect since they have a much larger than expected bias towards one of the sides of the planted solution on the vertices $V_0 \cup V_1$. Most importantly, bad vertices are defined with respect to an arbitrary cut on $V_0 \cup V_1$ since we don't know which of these cuts was selected by the algorithm.

Definition A.2 (Biased vertices)**.** *We say that a vertex $v \in V_2$ has a* bias *towards the side C_0 of the cut, if $n_v(V_0^0 \cup V_1^0) - n_v(V_0^1 \cup V_1^1) > \epsilon n$. Vertices biased towards the side C_1 of the cut are defined analogously.*

Proposition A.4. *There no vertices in V_2, which a bias towards one of the sides of the cut induced by the algorithm on $V_0 \cup V_1$.*

Proof. Suppose that there exists a vertex v with $r_v = 0$ (the case $r_v = 1$ is symmetric) such that $n_v(V_0^0 \cup V_1^0) - n_v(V_0^1 \cup V_1^1) > \epsilon n$. Because we assume that there are no bad vertices this implies that $\mathbb{E}[n_v(V_0^0 \cup V_1^0)] - \mathbb{E}[n_v(V_0^1 \cup V_1^1)] > \epsilon n - 2m^{3/4} > 3\epsilon n/4$, where the last inequality holds for large enough n. Expanding the expectations we have $(1+\epsilon)1/2|V_0^0| + 1/2|V_1^0| - (1+\epsilon)1/2|V_0^1| - 1/2|V_1^1| > 3\epsilon n/4$ or equivalently $|V_0^0| + |V_1^0| - |V_0^1| - |V_1^1| > (3\epsilon n/2 - \epsilon|V_0^0| + \epsilon|V_0^1|) > 3\epsilon n/2 - 4\epsilon n/9 > \epsilon n$. This implies that all vertices with $r_v = 0$ are optimally placed on the side C_1 of the cut. Similarly for vertices with $r_v = 1$ we have that $\mathbb{E}[n_v(V_0^0 \cup V_1^0)] - \mathbb{E}[n_v(V_1^0 \cup V_1^1)] \geq 1/2|V_0^0| + (1+\epsilon)1/2|V_1^0| - 1/2|V_0^1| - (1+\epsilon)1/2|V_1^1| - 2m^{3/4} \geq 1/2(|V_0^0| + |V_1^0| - |V_0^1| - |V_1^1|) - \epsilon/2|V_1^1| - 2m^{3/4} \geq \epsilon n/2 - 2\epsilon n/9 - 2m^{3/4} > 0$ and thus all such vertices are also optimally placed on the C_1 side of the cut.

Consider now the number of cut edges in a solution, which is given as $|V_0^0||V_1^1| + |V_0^1||V_1^0| + |E_2|$, where by E_2 we denote the set of cut edges adjacent to vertices in E_2. With high probability the number of vertices $v \in V_2$ such that $r_v = 0$ is at most $n/18 + n^{3/4}$ and the number of vertices such that $r_v = 1$ is also at most $n/18 + n^{3/4}$. Because there are no bad vertices $|E_2| \leq (n/18 + n^{3/4})\left((1+\epsilon)1/2|V_0^0| + 1/2|V_1^0| + m^{3/4}\right) + (n/18 + n^{3/4})\left(1/2|V_0^0| + (1+\epsilon)1/2|V_1^0| + m^{3/4}\right) = (1+\epsilon/2)n/18(|V_0^0| + |V_1^0|) + o(n^2)$. In the optimum solution vertices in V_2 contribute (roughly) $2n^2/9 + 2\epsilon n^2/9$ edges. Thus, the overall gain from vertices in V_2 compared to the optimum solution is at most $(1+\epsilon/2)n/18(|V_0^0| + |V_1^0|) - 2n^2/9 - 2\epsilon n^2/9 + o(n^2)$ edges. This can be written as $n/9((1/2+\epsilon/4)(|V_0^0| + |V_1^0|) - 2n/9 - 2\epsilon n/9) + o(n^2)$. On the other hand, we have $m^2 - |V_0^0||V_1^1| - |V_0^1||V_1^0| = |V_0^0||V_1^0| + |V_0^1||V_1^1| \geq (|V_0^0| + |V_1^0| - |V_0^1| - |V_1^1|)/2 \cdot |V_0| = 2n/9(|V_0^0| + |V_1^0| - |V_0^1| - |V_1^1|)$. Putting this together, the overall gain compared to the optimum solution is at most:

$$
\begin{aligned}
&n/9(((1/2+\epsilon/4)(|V_0^0| + |V_1^0|) - 2n/9 - 2\epsilon n/9) - 2(|V_0^0| + |V_1^0| - |V_0^1| - |V_1^1|)) + o(n^2) \\
&= n/9(((1/2+\epsilon/4)(4n/9 + (|V_0^0| + |V_1^0| - |V_0^1| - |V_1^1|)/2) - 2n/9 - 2\epsilon n/9) - 2(|V_0^0| + |V_1^0| - |V_0^1| - |V_1^1|)) + o(n^2) \\
&= n/9((\epsilon/8 - 7/4)(|V_0^0| + |V_1^0| - |V_0^1| - |V_1^1|) - \epsilon n/9) + o(n^2) < 0.
\end{aligned}
$$

Using the fact that there are no biased vertices the rest of the proof follows. If there are at least $n/100$ vertices with bias at most $\epsilon n/100$ then since the bias of these vertices is so small each of them loses $\Omega(\epsilon n)$ cut edges compared to the optimum solution which places these vertices on the side $1 - r_v$ and thus achieves $2n/9 + 2\epsilon n/9$ cut edges for each such vertex compared to at most $2n/9 + \epsilon n/9 + \epsilon n/100$ edges for each vertex with a small bias. Otherwise there are at least $n/9 - n/100$ vertices with bias at least $\epsilon n/100$ and at most ϵn (recall that there are no *biased vertices*). Thus, the algorithm has to achieve a non-trivial advantage over probability $1/2$ in guessing the value of r_v for vertices in V_2 if it places these vertices on the correct side. By a Chernoff-type sampling lower bound of [CEG95] this implies that for a constant fraction of vertices in V_2 it has to sample at least $\Omega(1/\epsilon^2)$ edges, which gives the $\Omega(n/\epsilon^2)$ lower bound, completing the proof. ∎

B Omitted proofs

The key difference between our work and [MS08, Sch12] is that we use sampling in the greedy step of our algorithms. However, some of their lemmas aren't affected by this difference. For completeness we present proofs for the lemmas that we used black-box from [Sch12], which contains an extended version of [MS08].

B.1 Proof of Lemma 2.5

First, we prove the following two lemmas:

Lemma B.1. *$\frac{t}{n-t} q^t_{vi}$ is a martingale.*

Lemma B.2. *$\frac{t}{n-t} A_{vk} q^t$ is a martingale with step size bounded by $\frac{4n}{n-t}$.*

Proof of Lemma B.1. We consider two cases:

Case 1: $v \in S^{t-1}$. Then we have $\hat{x}^{t-1}_{vi} = x^{t-1}_{vi} = \hat{x}^t_{vi} = x^t_{vi}$. Let's denote this common value as x. Then we have $\frac{t}{n-t} q^t_{vi} = \frac{t}{n-t}\left(x - \frac{n}{t}x\right) = -x$. Also, $\frac{t-1}{n-t+1} q^{t-1}_{vi} = \frac{t-1}{n-t+1}\left(x - \frac{n}{t-1}x\right) = -x$, as desired.

Case 2: $v \notin S^{t-1}$. We have $\hat{x}^{t-1}_{vi} = \frac{1}{t-1}\sum_{j=1}^{t-1} g^j_{vi}$ and $x^{t-1}_{vi} = 0$. This gives $\frac{t-1}{n-t+1} q^{t-1}_{vi} = \frac{1}{n-t+1}\sum_{j=1}^{t-1} g^j_{vi}$. On the other hand we have $x^t_{vi} = g^t_{vi}$ with probability $\frac{1}{n-t+1}$ and $x^t_{vi} = 0$ with probability $1 - \frac{1}{n-t+1}$. Thus, $\hat{x}^t_{vi} = g^t_{vi}$ with probability $\frac{1}{n-t+1}$ and $\hat{x}^t_{vi} = \frac{1}{t}\sum_{j=1}^{t} g^j_{vi}$ with probability $1 - \frac{1}{n-t+1}$. This gives us:

$$\mathbb{E}\left[\frac{t}{n-t} q^t_{vi}\right] = \frac{t}{n-t}\left(\frac{1}{n-t+1}\left(g^t_{vi} - \frac{n}{t} g^t_{vi}\right) + \left(1 - \frac{1}{n-t+1}\right)\frac{1}{t}\sum_{j=1}^{t} g^j_{vi}\right) = \frac{1}{n-t+1}\sum_{j=1}^{t-1} g^j_{vi} = \frac{t-1}{n-t+1} q^{t-1}_{vi}. \quad ∎$$

Proof of Lemma B.2. The martingale property follows from Lemma B.1. The components q^t_u fall into three cases:

Case 1: $u \in S^{t-1}$. In this case $q^t_u = q^{t-1}_u$ so $A_{vk,ui}(q^t_{ui} - q^{t-1}_{ui}) = 0$.

Case 2: $u = r^t$. In this case $x^t_{ui} = \hat{x}^t_{ui} = g^t_{ui}$, $x^{t-1}_{ui} = 0$ and $\hat{x}^{t-1}_{ui} = \frac{1}{t-1}\sum_{j=1}^{t-1} g^j_{ui}$. This gives us:

$$\frac{t}{n-t} A_{vk,ui} q^t_{ui} - \frac{t-1}{n-t+1} A_{vk,ui} q^{t-1}_{ui} = \frac{t}{n-t} g^t_{ui}\left(1 - \frac{n}{t}\right) - \frac{t-1}{n-t+1}\frac{1}{t-1}\sum_{j=1}^{t-1} g^j_{ui}. \geq -1 - \frac{t-1}{n-t+1} = -\frac{n}{n-t+1}$$

This gives the overall contribution of at most $\frac{2n}{n-t+1}$.

Case 3: $u \notin S^t$. In this case we have $x^t_{ui} = x^{t-1}_{ui} = 0$, $\hat{x}^t_{ui} = \frac{1}{t}\sum_{j=1}^{t} g^j_{ui}$ and $\hat{x}^{t-1}_{ui} = \frac{1}{t-1}\sum_{j=1}^{t-1} g^j_{ui}$. This gives us:

$$\frac{t}{n-t}A_{vk,ui}q^t_{ui} - \frac{t-1}{n-t+1}A_{vk,ui}q^{t-1}_{ui} \le \frac{t}{n-t}\frac{1}{t}\sum_{j=1}^{t} g^j_{ui} - \frac{t-1}{n-t+1}\frac{1}{t-1}\sum_{j=1}^{t-1} g^j_{ui}$$

$$= \frac{g^t_{ui}}{n-t} + \frac{1}{(n-t)(n-t+1)}\sum_{j=1}^{t-1} g^j_{ui} \le \frac{n}{(n-t)(n-t+1)}.$$

The number of vertices $u \notin S^t$ is $n-t$, hence their overall contribution is at most $\frac{2n}{n-t+1}$.

Putting all together, the step size is bounded by $\frac{4n}{n-t+1} \le \frac{4n}{n-t}$. ∎

We will use the following version of the Azuma-Hoeffding inequality:

Theorem B.3 (Azuma-Hoeffding)**.** *Let $X_0, X_1, \ldots, X_t$ be a martingale such that $|X_k - X_{k-1}| \le c_k$ for all k. Then for all $\lambda > 0$ it holds that:*

$$\Pr[|X_t - X_0| \ge \lambda] \le 2e^{-\frac{\lambda^2}{2\sum_{k=1}^{t} c_k^2}}$$

Now we are ready to complete the proof of Lemma 2.5.

Proof of Lemma 2.5. From Azuma-Hoeffding, using a union bound we have:

$$\Pr\left[\left|\frac{t}{n-t}Aq^t - \frac{t_0}{n-t_0}Aq^{t_0}\right|_1 \ge \lambda\right] \le 4ne^{-\frac{\lambda^2}{2\sum_{i=1}^{t}\frac{16n^2}{(n-i)^2}}} \le 4ne^{-\frac{\lambda^2}{32t}},$$

where the bound on the step size of the martingale at time t follows from Lemma B.2. Let $w = \frac{\lambda(n-t)}{t}$. Then $\Pr\left[\left|Aq^t - \frac{t_0(n-t)}{t(n-t_0)}Aq^{t_0}\right|_1 \ge w\right] \le 4ne^{-\frac{w^2 t}{32(n-t)^2}}$. Thus:

$$\mathbb{E}\left[\left|Aq^t - \frac{t_0(n-t)}{t(n-t_0)}Aq^{t_0}\right|_1\right] = \int_0^\infty \Pr\left[\left|Aq^t - \frac{t_0(n-t)}{t(n-t_0)}Aq^{t_0}\right|_1 \ge w\right] dw$$

$$\le 4n\int_0^\infty e^{-\frac{w^2 t}{32(n-t)^2}} = 16n(n-t)\sqrt{\frac{\pi}{t}} = O\left(\frac{n(n-t)}{\sqrt{t}}\right) \quad ∎$$

In order to bound $\mathbb{E}[|Aq^t|_1]$ note that $\mathbb{E}[|Aq^t|_1] \le \mathbb{E}\left[\left|Aq^t - \frac{t_0(n-t)}{t(n-t_0)}Aq^{t_0}\right|_1\right] + \mathbb{E}\left[\left|\frac{t_0(n-t)}{t(n-t_0)}Aq^{t_0}\right|_1\right]$. Bounding the second term we have:

$$\mathbb{E}\left[\left|\frac{t_0(n-t)}{t(n-t_0)}Aq^{t_0}\right|_1\right] = O\left(\frac{n-t}{\epsilon^2 tn}\right)\mathbb{E}\left[\left|Aq^{t_0}\right|_1\right].$$

It suffices to show that $\mathbb{E}\left[\left|Aq^{t_0}\right|_1\right] = O(\epsilon n^2)$. Indeed, consider $|A_{vi}q^{t_0}|_1 = \left|A_{vi}\left(\hat{x}^{t_0} - \frac{n}{t_0}x^{t_0}\right)\right|_1 = \left|A_{vi}\left(x^* - \frac{n}{t_0}x^{t_0}\right)\right|_1$. Recall that x^{t_0} is defined as:

$$x_i^{t_0} = \begin{cases} x_i^* & \text{if } i \in S^{t_0} \\ 0, & \text{otherwise.} \end{cases}$$

Random variable $\frac{n}{t_0}A_{vi}x^{t_0}$ has expectation $A_{vi}x^*$, by the Chernoff bound $\mathbb{E}\left[\left|A_{vi}\left(x^* - \frac{n}{t_0}x^{t_0}\right)\right|_1\right] = O(\epsilon n)$.

B.2 Proof of Lemma 3.5

Proof of Lemma 3.5. To simplify presentation we observe that the following proposition (Lemma 2.20 in [Sch12]) can be used black-box here since it doesn't rely on the specific way the greedy choice is made in the definition of x^t and $\hat{x}^t$ and only uses the fact that the variables at every step are chosen randomly together with the definition of a fictitious assignment.

Explanation of why there is no induction in the proof as per Comment 5.

Proposition B.4 (Adapted from Lemma 2.20 in[Sch12])**.** *For every t and a q-dimensional array A with $|A|_\infty \leq 1$ we have:*

$$\Pr\left[\left|A\left(q^t, q^t, \ldots, q^t\right)\right| \geq \sigma + \lambda\right] \leq e^{-\frac{\lambda^2}{\sigma^2}},$$

where $\sigma = O(n^q\sqrt{\frac{n-t}{nt}})$.

Using Lemma 2.20 from [Sch12] we have for every $y \in Y$:

$$\Pr\left[\left|A\left(\cdot, q^t_{(y)}, \ldots, q^t_{(y)}\right)\right| \geq \lambda\right] \leq e^{-\frac{\lambda^2}{\sigma^2}}.$$

The set Y has size $2^{\frac{\log k}{\epsilon^2}}$, so by taking a union bound we have:

$$\Pr\left[\max_{y\in Y}|A\left(\cdot, q^t_{(y)}, \ldots, q^t_{(y)}\right)|_1 \geq \lambda\right] \leq e^{O\left(\frac{\log k}{\epsilon^2}\right)-\lambda^2/\sigma^2}$$

The bound follows by integration over λ. This completes the proof of the first part of the lemma, the second part is identical. ∎

B.3 Proof of Lemma 3.1

Proof.

Proof added for completeness, includes modified dependence on k.

We use the following analog of Lemma 2.2 for r-CSPs, whose proof can be adapted from [Sch12].

Lemma B.5. *(Analogous to Lemma 2.13 in [Sch12], analog of Lemma 2.2) For every t it holds that:*

$$A(\hat{x}^t, \ldots, \hat{x}^t) - A(\hat{x}^{t-1}, \ldots, \hat{x}^{t-1}) \leq \frac{2^{r+2}n^r}{t^r} + rA(\hat{x}^t - \hat{x}^{t-1}, \hat{x}^{t-1}, \ldots, \hat{x}^{t-1}).$$

By Lemma B.5 it suffices to bound $\mathbb{E}\left[A(\hat{x}^t - \hat{x}^{t-1}, \hat{x}^{t-1}, \ldots, \hat{x}^{t-1})\right]$. We have:

$$A(\hat{x}^t - \hat{x}^{t-1}, \hat{x}^{t-1}, \ldots, \hat{x}^{t-1}) = \frac{n^{r-1}}{t^{r-1}}A(\hat{x}^t - \hat{x}^{t-1}, x^{t-1}, \ldots, x^{t-1}) + A(\hat{x}^t - \hat{x}^{t-1}, q^{t-1}, \ldots, q^{t-1}),$$

using linearity of A in each of its arguments and the definition $q^{t-1} = \hat{x}^{t-1} - \frac{n}{t-1}x^{t-1}$. We bound the first term using Proposition 3.2. If $u = r_t$ then $\hat{x}^t_u - \hat{x}^{t-1}_u = g^t_u - \hat{x}^{t-1}_u$. Hence $\mathbb{E}\left[A(\hat{x}^t_{ui} - \hat{x}^{t-1}_{ui}, x^{t-1} \ldots, x^{t-1})\right] \leq \frac{2kt^{r-1}}{\sqrt{s_t}}$. Because there are k different values of i we have overall $\mathbb{E}\left[A(\hat{x}^t_u - \hat{x}^{t-1}_u, x^{t-1} \ldots, x^{t-1})\right] \leq \frac{2k^2t^{r-1}}{\sqrt{s_t}}$. If $u \notin S^t$ then we have $\hat{x}^t - \hat{x}^{t-1} = \frac{1}{t}(g^t_u - \hat{x}^{t-1}_u)$. For such vertices we have $\mathbb{E}\left[A(\hat{x}^t_u - \hat{x}^{t-1}_u, x^{t-1}, \ldots, x^{t-1})\right] \leq \frac{2k^2t^{r-1}}{\sqrt{s_t}}$. The total number of such vertices is $n - t$, so overall we have:

$$\mathbb{E}\left[A(\hat{x}^t - \hat{x}^{t-1}, x^{t-1}, \ldots, x^{t-1})\right] \leq \frac{2k^2t^{t-1}}{\sqrt{s_t}} + \frac{2(n-t)k^2t^{r-2}}{\sqrt{s_t}} = \frac{2k^2nt^{r-2}}{\sqrt{s_t}}.$$

Thus, the first term is at most $\frac{2k^2n^r}{t\sqrt{s_t}}$.

The second term can be bounded using the same reasoning as in Lemma 2.3. We give a proof for completeness. Conditioning on the choices of S^{t-1} we have:

$$\begin{aligned}
&\mathbb{E}\left[A(\hat{x}^t - \hat{x}_u^{t-1}, q^{t-1}, \ldots, q^{t-1})|S^{t-1}\right] \\
&= \sum_v \mathbb{E}\left[A(\hat{x}_v^t - \hat{x}_v^{t-1}, q^{t-1}, \ldots, q^{t-1})|S^{t-1}\right] \\
&= \sum_v \mathbb{E}[\hat{x}_v^t - \hat{x}_v^{t-1}|S^{t-1}] \cdot A(e_u, q^{t-1}, \ldots, q^{t-1}) \\
&\leq \sum_v |\mathbb{E}[\hat{x}_v^t - \hat{x}_v^{t-1}]|_1 \cdot |A(e_u, q^{t-1}, \ldots, q^{t-1})|_1 \\
&\leq \frac{2n}{t(n-t+1)} \cdot |A(\cdot, q^{t-1}, \ldots, q^{t-1})|_1,
\end{aligned}$$

where the last inequality follows from the analysis given below. For fixed S^{t-1} and u, if $u \in S^{t-1}$ then $\hat{x}_u^t - \hat{x}_u^{t-1} = 0$. Otherwise, as shown in the proof of Lemma 2.2 it holds that $\sum_u |\hat{x}_u^t - \hat{x}_u^{t-1}| \leq \frac{2n}{t}$ and hence:

$$\mathbb{E}\left[|\hat{x}_u^t - \hat{x}_u^{t-1}|_1|S^{t-1}\right] = \mathbb{E}\left[\frac{|\hat{x}^t - \hat{x}^{t-1}|_1}{n-t+1}|S^{t-1}\right] \leq \frac{2n}{t(n-t+1)}.$$

∎