Electronic Colloquium on Computational Complexity, Revision 1 of Report No. 106 (2010)

Optimal Constant-Time Approximation Algorithms and (Unconditional) Inapproximability Results for Every Bounded-Degree CSP

Yuichi Yoshida[*]

School of Informatics, Kyoto University, and
Preferred Infrastructure, Inc.
yyoshida@lab2.kuis.kyoto-u.ac.jp

Abstract

Raghavendra (STOC 2008) gave an elegant and surprising result: if Khot's Unique Games Conjecture (STOC 2002) is true, then for every constraint satisfaction problem (CSP), the best approximation ratio is attained by a certain simple semidefinite programming and a rounding scheme for it.

In this paper, we show that similar results hold for constant-time approximation algorithms in the bounded-degree model. Specifically, we present the followings: (i) For every CSP, we construct an oracle that serves an access, in constant time, to a nearly optimal solution to a basic LP relaxation of the CSP. (ii) Using the oracle, we give a constant-time rounding scheme that achieves an approximation ratio coincident with the integrality gap of the basic LP. (iii) Finally, we give a generic conversion from integrality gaps of basic LPs to hardness results. All of those results are *unconditional*. Therefore, for every bounded-degree CSP, we give the best constant-time approximation algorithm among all.

A CSP instance is called ϵ-far from satisfiability if we must remove at least an ϵ-fraction of constraints to make it satisfiable. A CSP is called testable if there is a constant-time algorithm that distinguishes satisfiable instances from ϵ-far instances with probability at least $2/3$. Using the results above, we also derive, under a technical assumption, an equivalent condition under which a CSP is testable in the bounded-degree model.

Key words: Constant-time approximation, constraint satisfaction problems, linear programmings, rounding schemes, property testing.

[*]Supported by MSRA Fellowship 2010. This work was conducted while the author was visiting Rutgers University.

ISSN 1433-8092

1 Introduction

In a *constraint satisfaction problem* (CSP), the objective is to find an assignment to a set of variables that satisfies the maximum number of a given set of constraints on them. Formally, a CSP Λ is specified by a set of predicates over alphabets $[q] = \{1, \ldots, q\}$. Every instance of Λ consists of a set of variables V, and a set of constraints $\mathcal{P}$ on them. Each constraint consists of a predicate from Λ applied to a subset of variables. The objective is to find an assignment $\beta \in [q]^V$ to the variables that satisfies the maximum number of constraints. A large number of fundamental optimization problems, such as $\mathsf{Max\ Cut}$ and Max k-Sat, are examples of CSPs.

Approximation algorithms for CSPs have been intensively studied. Goemans and Williamson [9] first exploited semidefinite programmings (SDP) to $\mathsf{Max\ Cut}$ and $\mathsf{Max\ 2SAT}$ achieving the approximation ratio ≈ 0.878. After this breakthrough, plethora of approximation algorithms using SDPs have been developed [15, 21]. For inapproximability side, tight hardness results have been successfully obtained for some important optimization problems such as $\mathsf{Max\ 3SAT}$ [14]. However, the approximability of many interesting CSPs such as $\mathsf{Max\ Cut}$ and $\mathsf{Max\ 2SAT}$ remains open. Towards tightening this gap, Khot [16] introduced the Unique Games Conjecture (UGC). Assuming the UGC, tight hardness have been shown for $\mathsf{Max\ Cut}$ [17], $\mathsf{Max\ 2SAT}$ [4], and $\mathsf{Max\ k\text{-}CSP}$ [5, 27]. Finally, Raghavendra [25] succeeded to unify and generalize those approximation and inapproximability results for every CSP. Specifically, Raghavendra showed that, assuming the UGC, for every CSP, a certain SDP combined with a certain rounding scheme attains the best approximation ratio among all polynomial-time approximation algorithms. The ingenious technique in the proof is giving a generic conversion from integrality gaps of SDPs to hardness results via the UGC.

In this paper, we are concerned with constant-time approximation algorithms CSPs. That is, algorithms are supposed to run in time independent of sizes of instances. We use the *bounded-degree model*, which was originally introduced for graphs [11]. In this model, the number of alphabets, the maximum arity (the number of inputs to a predicate), the maximum degree (the number of constraints where a variable appears), and the maximum weight of constraints are bounded by constants. Let $\mathcal{I}$ be a Λ-CSP instance. Since a constant-time algorithm cannot read the whole $\mathcal{I}$, we assume the existence of an oracle $\mathcal{O}_\mathcal{I}$ with which we can get information of $\mathcal{I}$. By specifying a variable v and an index i, $\mathcal{O}_\mathcal{I}$ returns a constraint P where P is the i-th constraint where v appears. The efficiency of an algorithm is measured by the number of accesses to $\mathcal{O}_\mathcal{I}$, which is called *query complexity*.

In this paper, we show an analogous result to Raghavendra's result: for every CSP, a certain linear programming (LP) combined with a certain rounding scheme attains the best approximation ratio among all constant-time approximation algorithms. Furthermore, our results are *unconditional*. To give the statements precisely, we need to define several notions. For a Λ-CSP instance $\mathcal{I}$ with the variable set V and the constraint set $\mathcal{P}$, there is a natural generic LP relaxation which we call $\mathsf{BasicLP}$ (see Section 2). Let $\mathbf{lp}(\mathcal{I})$ denote the objective value of an optimal solution to $\mathsf{BasicLP}$ for $\mathcal{I}$, $\mathbf{opt}(\mathcal{I})$ denote the value of an optimal solution of $\mathcal{I}$, and $\mathbf{val}(\mathcal{I}, \beta)$ denote the value obtained by an assignment $\beta \in [q]^V$. We define $\boldsymbol{w}_\mathcal{I}$ as the sum of weights of constraints in $\mathcal{I}$. Then, we define $\overline{\mathbf{lp}}(\mathcal{I}) = \mathbf{lp}(\mathcal{I})/\boldsymbol{w}_\mathcal{I}, \overline{\mathbf{opt}}(\mathcal{I}) = \mathbf{opt}(\mathcal{I})/\boldsymbol{w}_\mathcal{I}$ and $\overline{\mathbf{val}}(\mathcal{I}, \beta) = \mathbf{val}(\mathcal{I}, \beta)/\boldsymbol{w}_\mathcal{I}$. The *integrality gap curve* $S_\Lambda(c)$ and the *integrality gap* α_Λ of a CSP Λ is defined as

$$S_\Lambda(c) = \inf_{\mathcal{I} \in \Lambda, \overline{\mathbf{lp}}(\mathcal{I}) \geq c} \overline{\mathbf{opt}}(\mathcal{I}), \quad \alpha_\Lambda = \inf_{\mathcal{I} \in \Lambda} \overline{\mathbf{opt}}(\mathcal{I})/\overline{\mathbf{lp}}(\mathcal{I}).$$

The first result of this paper gives a tight approximation algorithm for every CSP.

Theorem 1.1. *In the bounded-degree model, for every CSP Λ and $\epsilon > 0$, there exists an algorithm that, given a Λ-CSP instance $\mathcal{I}$ with n variables and $\overline{\mathbf{lp}}(\mathcal{I}) = c \in (0, 1]$, with probability at least*

$2/3$, outputs a value x such that $S_\Lambda(c-\epsilon)\boldsymbol{w}_\mathcal{I} - \epsilon n \le x \le \mathbf{opt}(\mathcal{I})$. Also, for some fixed assignment β such that $S_\Lambda(c-\epsilon)\boldsymbol{w}_\mathcal{I} - \epsilon n \le \mathbf{val}(\mathcal{I},\beta) \le \mathbf{opt}(\mathcal{I})$, given a variable v in $\mathcal{I}$, it computes β_v in constant time.

The algorithm computes β_v by rounding an LP solution to BasicLP for $\mathcal{I}$. Note that, for an instance $\mathcal{I}$ with $\overline{\mathbf{lp}}(\mathcal{I}) = c$, $S_\Lambda(c)\boldsymbol{w}_\mathcal{I}$ is the best value we can hope for from the definition of $S_\Lambda(c)$. Thus, in this sense, we will give a *optimal rounding scheme* for BasicLP.

We mention that the additive error ϵn cannot be removed. To see this, suppose an instance $\mathcal{I}$ consisting of n variables and only one constraint. Then, we have to see this constraint to approximate $\mathbf{opt}(\mathcal{I})$ if we do not allow the additive error. However, it obviously takes $\Omega(n)$ queries.

For hardness side, we show the following.

Theorem 1.2. *In the bounded-degree model, for every CSP $\Lambda, c \in [0,1]$ and $\epsilon > 0$, there exists a $\delta > 0$ such that any algorithm that, given an instance $\mathcal{I}$ with n variables and $\overline{\mathbf{opt}}(\mathcal{I}) = c \in [0,1]$, with probability at least $2/3$, outputs a value x such that $(S_\Lambda(c)+\epsilon)\boldsymbol{w}_\mathcal{I} - \delta n \le x \le \mathbf{opt}(\mathcal{I})$ requires $\Omega(\sqrt{n})$ queries.*

Note that, using the algorithm in Theorem 1.1, given an instance $\mathcal{I}$, we can distinguish the case $\mathbf{opt}(\mathcal{I}) \ge c\boldsymbol{w}_\mathcal{I}$ from the case $\mathbf{opt}(\mathcal{I}) \le S_\Lambda(c-\epsilon)\boldsymbol{w}_\mathcal{I} - \epsilon n$ (Technically, we need that $S_\Lambda(c)$ is non-decreasing, but this is obvious from the definition). On the contrary, Theorem 1.2 asserts that we cannot distinguish the case $\mathbf{opt}(\mathcal{I}) \ge c\boldsymbol{w}_\mathcal{I}$ from the case $\mathbf{opt}(\mathcal{I}) \le (S_\Lambda(c)+\epsilon)\boldsymbol{w}_\mathcal{I} - \delta n$. Thus, the algorithm given in Theorem 1.1 is not just the best among constant-time approximation algorithm using BasicLP, but the best among all constant-time approximation algorithms.

A value x is called an (α,β)-*approximation* to a value x^* if it satisfies $\alpha x^* - \beta \le x \le x^*$. An algorithm is called an (α,β)-*approximation algorithm* for a CSP Λ if, given a Λ-CSP instance $\mathcal{I}$, it computes an (α,β)-approximation to $\mathbf{opt}(\mathcal{I})$ with probability at least $2/3$ [23, 24]. The following is an immediate corollary achieved by Theorems 1.1 and 1.2.

Corollary 1.3. *In the bounded-degree model, for every CSP Λ and $\epsilon > 0$, there exists a constant-time $(\alpha_\Lambda - \epsilon, \epsilon n)$-approximation algorithm for the CSP Λ. On the other hand, for every CSP Λ and $\epsilon > 0$, there exists a $\delta > 0$ such that any $(\alpha_\Lambda + \epsilon, \delta n)$-approximation algorithm for the CSP Λ requires $\Omega(\sqrt{n})$ queries.*

Theorem 1.2 has much implication to property testing. A Λ-CSP instance $\mathcal{I}$ is called *satisfiable* if there is an assignment to variables that satisfies all the constraints. Also, $\mathcal{I}$ is called *ϵ-far from satisfiability* if we must remove at least ϵtwn constraints to make it satisfiable, where t, w, n is the maximum degree, the maximum weight, and the number of variables, respectively. An algorithm is called a *testing algorithm* for (the satisfiability of) a CSP Λ if, given a Λ-CSP instance, it accepts with probability at least $2/3$ if the instance is satisfiable, and rejects with probability at least $2/3$ if the instance is ϵ-far from satisfiability. Unlike the hardness result given in [25], Theorem 1.2 holds also for $c = 1$, i.e., satisfiable instances. Using this observation, we have the following theorem.

Theorem 1.4. *In the bounded-degree model, the following holds for a CSP Λ. If $S_\Lambda(1) < 1$, then any testing algorithm for the CSP Λ requires $\Omega(\sqrt{n})$ queries. If $S_\Lambda(1) = 1$ and $S_\Lambda(c)$ is continuous at $c = 1$, then there exists a constant-time testing algorithm for the CSP Λ.*

We mention that Theorem 1.4 gives an "if and only if" condition of the testability of CSPs when their integrality gap curves are continuous at the point one while we are not aware of any CSP for which the curve is not continuous at that point.

We give two direct applications of Theorem 1.4. An instance of 2-SAT is a CNF formula where each constraint consists of at most two literals. It is known that $S_{\text{Max 2-SAT}}(1) = 1/2$, and it follows that we need $\Omega(\sqrt{n})$ queries to test 2-SAT. On the contrary, 2-SAT is known to be testable with $\tilde{O}(\sqrt{n})$ queries [10]. This fact implies that the lower bound in Theorem 1.2 is almost tight. An instance of Horn Sat is a CNF formula where each constraint has at most one positive literal, From [31], it is easy to derive that $S_{\textsf{Max Horn SAT}}(1) = 1$ and $S_{\textsf{Max Horn SAT}}(c)$ is continuous at $c = 1$. Thus, Horn SAT is testable in constant time.

Related Work: Subsequent to Raghavendra's work [25], under the UGC, certain SDPs and LPs are shown to be the best approximation algorithms for several classes of problems, such as graph labeling problems (including k-Way Cut, 0-Extension, and Metric Labeling) [22], kernel clustering problems [18], ordering CSPs (including Maximum Acyclic Subgraph) [13], and strict monotone CSPs (including Minimum Vertex Cover) [20].

There have been many studies on constant-time approximation algorithms in the bounded-degree model. For algorithmic side, mainly graph problems have been studied, e.g., Minimum Spanning Tree [7], Minimum Vertex Cover [23, 24, 30], Maximum Matching [23, 30], Maximum Independent Set [1], and Minimum Dominating Set [23, 30]. For inapproximability results of graph problems, Minimum Dominating Set [1] and Maximum Independent Set [1, 29] have been considered. For CSPs, it is known that, for every $\epsilon > 0$, there exists $\delta > 0$ such that any $(1/2 + \epsilon, \delta n)$-approximation algorithm for Max E2LIN2 and $(7/8 + \epsilon, \delta n)$-approximation algorithm for Max E3SAT require linear number of queries [6].

We can compute the optimal value of a CSP instance within an additive error $O(\epsilon n^s)$ by sampling $\mathrm{poly}(1/\epsilon)$ variables and by solving the induced problem, where n is the number of variables and s is the maximum arity [2, 3]. Thus, it is easy to approximate the solution of a dense instance in constant time. Hence, we are concerned with the bounded-degree model in this paper.

Proof Overview: We describe a proof sketch of Theorem 1.1. Let $\mathcal{O}_{\mathcal{I}}$ be the oracle access to a Λ-CSP instance $\mathcal{I}$. First, we construct an oracle access $\mathcal{O}_{\mathbf{lp}}$ to a nearly optimal solution to BasicLP for $\mathcal{I}$, Namely, if we specify a variable in BasicLP, $\mathcal{O}_{\mathbf{lp}}$ outputs its value by accessing $\mathcal{O}_{\mathcal{I}}$ constant number of times. To this end, we use a distributed algorithm for packing/covering LP given in [19]. In the distributed setting, a linear programing is bound to a graph $G = (V, E)$. Each primal variable $\boldsymbol{x}_i$ and each dual variable $\boldsymbol{y}_j$ is associated with a vertex $v_i^p \in V$ and $v_j^d \in V$, respectively. There are edges between primal and dual vertices wherever the respective variables occur in the corresponding inequality. Thus, $(v_i^p, v_j^d) \in E$ if and only if $\boldsymbol{x}_i$ occurs in the j-th inequality of the primal. Let $G_{v,k}$ denote the graph induced by vertices whose distance from v is at most k. Then, a *distributed algorithm in k rounds* works in such a way that each vertex outputs a value of the corresponding variable based on $G_{v,k}$. In [19], it is shown that if the matrix in the LP is "sparse," then there is a distributed algorithm that computes a nearly optimal solution to the LP in k rounds, where k is an integer determined by the sparsity of the LP. Suppose that the degree of the graph is bounded by Δ. Then, given a variable, we can compute the value of it by performing Δ^k queries to $\mathcal{O}_{\mathcal{I}}$ by simulating the process of the distributed algorithm. With this method, we achieve $\mathcal{O}_{\mathbf{lp}}$. Though BasicLP is not a packing/covering LP, after applying several number of transformations, we get a packing LP that has essentially the same behavior under approximation. Technically, we need to show that BasicLP is robust in the sense that even if we violate each constraint by small amount, the optimal value does not significantly increase. We finally mention that, a predicate can return values in $[-1, 1]$ in [25] while it can only return 0 or 1 in this paper. This restriction comes from that we cannot transform BasicLP to a packing LP anymore if we allow negative values.

Next, we exhibit a solution to the original instance by rounding the LP solution given by $\mathcal{O}_{\mathbf{lp}}$. In [26], Raghavendra and Steurer considered a certain SDP relaxation, which we call BasicSDP, and showed an optimal rounding scheme for it. That is, it achieves an approximation ratio coincident with the integrality gap of BasicSDP. Our proof is based on their work. First, from an instance $\mathcal{I}$

and its LP solution, we create another instance $\mathcal{I}'$ by merging variables of $\mathcal{I}$ that are close in the LP solution so that the number of variables in $\mathcal{I}'$ become constant. Though we cannot explicitly construct the whole $\mathcal{I}'$ since the number of constraints is not constant, we can enumerate variables in $\mathcal{I}'$. Then, we perform brute force search on $\mathcal{I}'$. Specifically, we estimate the value obtained by each assignment to variables in $\mathcal{I}'$ by accessing the oracle $\mathcal{O}_{\mathbf{lp}}$. Let β'^* be the assignment for $\mathcal{I}'$ that takes the maximum among them. Note that β'^* can be unfolded to an assignment β^* for $\mathcal{I}$. Then, with high probability, we have $S_\Lambda(c-\epsilon) - \epsilon n \leq \mathbf{val}(\mathcal{I}, \beta^*) \leq \mathbf{opt}(\mathcal{I})$. Since, from a variable v in $\mathcal{I}$, we can get the corresponding variable in $\mathcal{I}'$ in constant time, we can compute β^*_v in constant time. The crucial fact used here is that the LP optimum does not change significantly after merging variables.

Now, we describe a proof sketch of Theorem 1.2. Let $\mathcal{I}$ be a Λ-CSP instance such that $\mathbf{lp}(I) = c\boldsymbol{w}_{\mathcal{I}}$ while $\mathbf{opt}(I)$ is arbitrarily close to $S_\Lambda(c)\boldsymbol{w}_{\mathcal{I}}$. Also, let $(\boldsymbol{x}^*, \boldsymbol{\mu}^*)$ be the optimal LP solution to BasicLP for $\mathcal{I}$. First, we create a distribution of instances $\mathcal{D}^{\mathbf{opt}}$ by blowing up variables of BasicLP. With high probability, an instance $\mathcal{J}$ generated by $\mathcal{D}^{\mathbf{opt}}$ satisfies that $\overline{\mathbf{opt}}(\mathcal{J}) \leq \overline{\mathbf{opt}}(\mathcal{I}) + \epsilon$ where ϵ is an arbitrarily small constant. Next, using the LP solution $(\boldsymbol{x}^*, \boldsymbol{\mu}^*)$, we create another distribution of instances $\mathcal{D}^{\mathbf{lp}}$, which has the property that for all $\mathcal{J}$ generated by $\mathcal{D}^{\mathbf{lp}}$, $\overline{\mathbf{opt}}(\mathcal{J}) \geq \overline{\mathbf{lp}}(\mathcal{I})$. From Yao's minimax principle, by showing that any deterministic algorithm that distinguishes $\mathcal{D}^{\mathbf{opt}}$ from $\mathcal{D}^{\mathbf{lp}}$ with high probability requires $\Omega(\sqrt{n})$ queries, we have the desired result.

Organization: In Section 2, we give notations and basic technical tools used in this paper. In Section 3, we present an oracle access $\mathcal{O}_{\mathbf{lp}}$ to a (nearly) optimal solution to BasicLP. Section 4 is devoted to describe how to round the LP solution optimally and to prove Theorem 1.1. We give proofs of Theorems 1.2 and 1.4 in Section 5 and Appendix E, respectively.

2 Preliminaries

2.1 Definitions

For an integer k, $[k]$ denotes the set $\{1, \ldots, k\}$. The *arity* of a predicate $P : [q]^k \to \{0,1\}$ is the number of inputs to P, i.e., k here. The *degree* of a variable is the number of constraints where the variable appears. For a constraint P in a CSP instance, $V(P)$ denotes the set of variables in P. Let β be a vector or a set indexed by elements of a set V. For a subset $S \subseteq V$, we define $\beta_{|S} = \{\beta_v\}_{v \in S}$.

Definition 2.1. *A bounded-degree constraint satisfaction problem Λ is specified by $\Lambda = ([q], s, t, w, \mathbb{P})$, where $[q]$ is a finite domain, s is the maximum arity of predicates, t is the maximum degree of variables, w is the maximum weight of predicates, and $\mathbb{P} = \{P : [q]^k \to \{0,1\} \mid k \leq s\}$ is a set of predicates.*

Definition 2.2. *An instance $\mathcal{I}$ of a CSP $\Lambda = ([q], s, t, w, \mathbb{P})$ is given by $\mathcal{I} = (V, \mathcal{P}, \boldsymbol{w})$, where*

- *$V = \{v_1, \ldots, v_n\}$ is a set of variables taking values over $[q]$,*
- *$\mathcal{P}$ is a set of constraints, consisting of predicates $P \in \mathbb{P}$ applied to sequences S of variables V of size at most s. More precisely, when a predicate P is applied to a sequence $S = \{i_1, \ldots, i_k\} \subseteq [n]^k$, P takes variables $V_{|S} = \{v_{i_1}, \ldots, v_{i_k}\}$ as the input.*
- *$\boldsymbol{w}$ is a set of weights $\{\boldsymbol{w}_P\}_{P \in \mathcal{P}}$ assigned to each constraint $P \in \mathcal{P}$, where $1 \leq \boldsymbol{w}_P \leq w$.*

The objective is to find an assignment to variables $\beta \in [q]^V$ that maximizes the total weight of satisfied constraints, i.e., $\sum_{P \in \mathcal{P}} \boldsymbol{w}_P P(\beta)$.

Definition 2.3 (Bounded-degree Model). *In the bounded-degree model, an algorithm is given a CSP $\Lambda = ([q], s, t, w, \mathbb{P})$ and the set of variables V beforehand. A Λ-CSP instance $\mathcal{I} = (V, \mathcal{P}, \boldsymbol{w})$ is represented by an oracle $\mathcal{O}_{\mathcal{I}}$ such that $\mathcal{O}_{\mathcal{I}}$, on two numbers $v \in V, i \in [t]$, returns $P \in \mathcal{P}$ where P is the i-th constraint where v appears. If no such constraint exists, it returns a special character $\perp$. The query complexity of an algorithm is the number of accesses to $\mathcal{O}_{\mathcal{I}}$.*

In this paper, when there is no ambiguity, symbols q, s, t and w are used to denote the parameters of a considered CSP. Also, symbols $n, \mathcal{O}_{\mathcal{I}}, \boldsymbol{w}_{\mathcal{I}}$ are used to denote the number of variables, the oracle access, and the total weight of an input instance $\mathcal{I}$, respectively.

We consider an LP relaxation for a CSP Λ as follows, which we call BasicLP.

$$\begin{array}{lll}
\max & \sum\limits_{P \in \mathcal{P}} \boldsymbol{w}_P \sum\limits_{\beta \in [q]^{V(P)}} P(\beta)\boldsymbol{\mu}_{P,\beta} & \\
\text{s.t.} & \sum\limits_{a \in [q]} \boldsymbol{x}_{v,a} = 1 & \forall v \in V \\
& \sum\limits_{\beta \in [q]^{V(P)}, \beta_v = a} \boldsymbol{\mu}_{P,\beta} = \boldsymbol{x}_{v,a} & \forall P \in \mathcal{P}, v \in V(P), a \in [q] \\
& \boldsymbol{x}_{v,a} \geq 0 & \forall v \in V, a \in [q] \\
& \boldsymbol{\mu}_{P,\beta} \geq 0 & \forall P \in \mathcal{P}, \beta \in [q]^{V(P)}.
\end{array}$$

Here, $\boldsymbol{x}_v = \{\boldsymbol{x}_{v,a}\}_{a \in [q]}$ (resp., $\boldsymbol{\mu}_P = \{\boldsymbol{\mu}_{P,\beta}\}_{\beta \in [q]^{V(P)}}$) can be seen as a distribution over assignments to a variable $v \in V$ (resp., a constraint $P \in \mathcal{P}$), and we often identify them as distributions. For an LP solution $(\boldsymbol{x}, \boldsymbol{\mu})$, we define $\mathbf{val}(\mathcal{I}, \boldsymbol{x}, \boldsymbol{\mu})$ as the value of the objective function of BasicLP obtained by $(\boldsymbol{x}, \boldsymbol{\mu})$. We call an LP solution $(\boldsymbol{x}, \boldsymbol{\mu})$ *ϵ-infeasible* if it satisfies constraints of the form $\boldsymbol{x}_{v,a} \geq 0$ and $\boldsymbol{\mu}_{P,\beta} \geq 0$ and violates other constraints by at most ϵ. We call a solution to an LP *(α, β)-approximate* if the objective value obtained by the solution is an (α, β)-approximation to the optimal value of the LP.

2.2 Basic Tools

As a simple application of Hoeffding's inequality, we obtain the following.

Lemma 2.4. *Suppose that we have an oracle access to a function $f : [n] \to [0, w]$. That is, by specifying $x \in [n]$ as a query, we can see the value of $f(x)$. Then, by querying $O(\frac{w^2}{\epsilon^2} \log \frac{1}{\delta})$ times, with probability at least $1 - \delta$, we can compute a $(1, \epsilon n)$-approximation to $\sum_i f(i)$.* □

Let $\mathcal{I}$ be a Λ-CSP instance. Not surprisingly, we cannot compute the optimal solution $(\boldsymbol{x}^*, \boldsymbol{\mu}^*)$ of BasicLP for $\mathcal{I}$ in constant time. Even worse, it is also hard to obtain a feasible solution in constant time. Instead, we will compute a feasible (nearly) optimal solution $(\boldsymbol{x}, \boldsymbol{\mu})$ of an LP obtained by relaxing equality constraints. Though this is an infeasible solution in the original LP, The following lemma states that $\mathbf{val}(\mathcal{I}, \boldsymbol{x}, \boldsymbol{\mu})$ is close to $\mathbf{lp}(\mathcal{I})$. The proof, which needs Fourier analysis, is given in Appendix A.

Lemma 2.5 (Robustness of BasicLP). *Let $\mathcal{I}$ be a Λ-CSP instance. Suppose that $(\boldsymbol{x}, \boldsymbol{\mu})$ is an ϵ-infeasible LP solution for $\mathcal{I}$ of value $c\boldsymbol{w}_{\mathcal{I}}$. Then, it holds that*

$$\overline{\mathbf{lp}}(\mathcal{I}) \geq c - \epsilon \cdot \mathrm{poly}(qs).$$

3 A $(1 - \epsilon, \epsilon n)$-approximation algorithm for BasicLP

In this section, we show the following theorem.

Theorem 3.1. *In the bounded-degree model, given a Λ-CSP instance $\mathcal{I}$, for any $\epsilon > 0$, we can construct an oracle $\mathcal{O}_{\mathbf{lp}}$ that gives an access to an ϵ-feasible $(1-\epsilon, \epsilon n)$-approximate solution to* BasicLP *for $\mathcal{I}$. For each query, the number of queries performed to $\mathcal{O}_{\mathcal{I}}$ is at most* $\exp(\exp(\mathrm{poly}(qstw/\epsilon)))$.

A packing LP is a problem of maximizing $\boldsymbol{b}^T\boldsymbol{z}$ subject to $A^T\boldsymbol{z} \le \boldsymbol{c}$ and $\boldsymbol{z} \ge 0$, where $A \in \mathbb{R}_+^{m\times n}$ is a non-negative matrix and $\boldsymbol{b}, \boldsymbol{c} \in \mathbb{R}_+^n$ are non-negative vectors. There is a constant-round distributed algorithm to compute a nearly optimal solution to the packing LP (see Appendix B for a formal statement). When a variable $\boldsymbol{x}_{v,a}$ or $\boldsymbol{\mu}_{P,\beta}$ is specified as a query, we locally simulate the distributed algorithm and output the value for it. The only issue is that BasicLP is not a packing LP. In this section, we transform BasicLP to a packing LP, and we will show that we can restore a good approximation to BasicLP from an approximation to the resulting packing LP. First, we substitute $\boldsymbol{x}_{v,a}$ by $1-\boldsymbol{x}_{v,a}$ and relax each equality constraint by ϵ. Then, we obtain the following LP.

$$\begin{array}{lll} \max & \sum_{P\in\mathcal{P}} \boldsymbol{w}_P \sum_{\beta\in[q]^{V(P)}} P(\beta)\boldsymbol{\mu}_{P,\beta} & \\ \text{s.t.} & |\sum_{a\in[q]} \boldsymbol{x}_{v,a} - (q-1)| \le \epsilon & \forall v \in V \\ & |\boldsymbol{x}_{v,a} + \sum_{\beta\in[q]^{V(P)},\beta_v=a} \boldsymbol{\mu}_{P,\beta} - 1| \le \epsilon & \forall P\in\mathcal{P}, v\in V(P), a\in[q] \\ & \boldsymbol{x}_{v,a} \ge 0 & \forall v\in V, a\in[q] \\ & \boldsymbol{\mu}_{P,\beta} \ge 0 & \forall P\in\mathcal{P}, \beta\in[q]^{V(P)}. \end{array} \tag{1}$$

Lemma 3.2. *Let $\mathcal{I}$ be a Λ-CSP instance and $(\boldsymbol{x}, \boldsymbol{\mu})$ be an ϵ-infeasible solution to LP (1) of value $c\boldsymbol{w}_{\mathcal{I}}$. Then, $\overline{\mathbf{lp}}(\mathcal{I}) \ge c - \epsilon\cdot\mathrm{poly}(qs)$ holds.*

Proof. Clearly, $(1-\boldsymbol{x}, \boldsymbol{\mu})$ is an 2ϵ-infeasible solution to BasicLP of value $c\boldsymbol{w}_{\mathcal{I}}$. From Lemma 2.5, the lemma holds. □

Next, to make the directions of the inequalities the same, we introduce a complement variable for each variable, i.e., we define $\overline{\boldsymbol{x}}_{v,a} = 1-\boldsymbol{x}_{v,a}$ and $\overline{\boldsymbol{\mu}}_{P,\beta} = 1-\boldsymbol{\mu}_{P,\beta}$. However, such equality constraints cannot be used in a packing LP. Thus, we relax those equality constraints again. That is, we introduce constraints of the form $\boldsymbol{x}_{v,a} + \overline{\boldsymbol{x}}_{v,a} \le 1$ and $\boldsymbol{\mu}_{P,\beta} + \overline{\boldsymbol{\mu}}_{P,\beta} \le 1$. Instead, to discourage them to become much smaller than one, we add additional terms to the objective function. By letting $\boldsymbol{C} = (C, C, \ldots, C)$ where $C = q^{O(s)}\mathrm{poly}(tw)/\epsilon^2$ is a large constant, we get the following LP.

$$\begin{array}{lll} \max & \sum_{P\in\mathcal{P}} \boldsymbol{w}_P \sum_{\beta\in[q]^{V(P)}} P(\beta)\boldsymbol{\mu}_{P,\beta} + \boldsymbol{C}^T(\boldsymbol{x}+\overline{\boldsymbol{x}}) + \boldsymbol{C}^T(\boldsymbol{\mu}+\overline{\boldsymbol{\mu}}), & \\ \text{s.t.} & \sum_{a\in[q]} \boldsymbol{x}_{v,a} \le q-1+\epsilon & \forall v\in V \\ & \sum_{a\in[q]} \overline{\boldsymbol{x}}_{v,a} \le 1+\epsilon & \forall v\in V \\ & \boldsymbol{x}_{v,a} + \sum_{\beta\in[q]^{V(P)},\beta_v=a} \boldsymbol{\mu}_{P,\beta} \le 1+\epsilon & \forall P\in\mathcal{P}, v\in V(P), a\in[q] \\ & \overline{\boldsymbol{x}}_{v,a} + \sum_{x\in[q]^{V(P)},\beta_v=a} \overline{\boldsymbol{\mu}}_{P,\beta} \le q^{|V(P)|-1}+\epsilon & \forall P\in\mathcal{P}, v\in V(P), a\in[q] \\ & \boldsymbol{x}_{v,a}+\overline{\boldsymbol{x}}_{v,a} \le 1, \quad \boldsymbol{x}_{v,a}\ge 0, \quad \overline{\boldsymbol{x}}_{v,a} \ge 0 & \forall v\in V, a\in[q] \\ & \boldsymbol{\mu}_{P,\beta}+\overline{\boldsymbol{\mu}}_{P,\beta} \le 1, \quad \boldsymbol{\mu}_{P,\beta}\ge 0, \quad \overline{\boldsymbol{\mu}}_{P,\beta}\ge 0 & \forall P\in\mathcal{P}, \beta\in[q]^{V(P)}. \end{array} \tag{2}$$

Fortunately, the optimal solutions to LP (2) and LP (1) are essentially the same.

Lemma 3.3 (Theorem 7 of [8], in a special form)**.** *Let $\mathcal{I}$ be a Λ-CSP instance and $(\boldsymbol{x}^*, \overline{\boldsymbol{x}}^*, \boldsymbol{\mu}^*, \overline{\boldsymbol{\mu}}^*)$ be the optimal solution to LP (2) with value $c\boldsymbol{w}_{\mathcal{I}} + CN$ where N is the number of variables in LP (2). Then, $\boldsymbol{x}^* + \overline{\boldsymbol{x}}^* = \mathbf{1}$ and $\boldsymbol{\mu}^* + \overline{\boldsymbol{\mu}}^* = \mathbf{1}$ hold. Also, $(\boldsymbol{x}^*, \boldsymbol{\mu}^*)$ is the optimal solution to LP (1) with value $c\boldsymbol{w}_{\mathcal{I}}$.* □

Now, using the distributed algorithm given by [19], we have the following lemma. The analysis of the query complexity is tedious and the proof is given in Appendix B.

Lemma 3.4. *In the bounded-degree model, given a Λ-CSP instance $\mathcal{I}$, for any $\epsilon > 0$, we can construct an oracle that serves an access to $(\boldsymbol{x}, \overline{\boldsymbol{x}}, \boldsymbol{\mu}, \overline{\boldsymbol{\mu}})$, which is a feasible $(1-\epsilon, 0)$-approximate solution to LP (2). For each query, the number of queries performed to $\mathcal{O}_{\mathcal{I}}$ is at most* $\exp(\mathrm{poly}(qstw/\epsilon))$.

Proof of Theorem 3.1. Let $(\boldsymbol{x}, \overline{\boldsymbol{x}}, \boldsymbol{\mu}, \overline{\boldsymbol{\mu}})$ be a feasible $(1-\epsilon', 0)$-approximate solution obtained by Lemma 3.4, where ϵ' is a constant determined later. For notational simplicity, we write the objective function as $\boldsymbol{w}^T\boldsymbol{\mu} + \boldsymbol{C}^T(\boldsymbol{z} + \overline{\boldsymbol{z}})$ where $\boldsymbol{z} = (\boldsymbol{x}, \boldsymbol{\mu})$. Let $(\boldsymbol{z}^*, \overline{\boldsymbol{z}}^*)$ be the optimal solution to LP (2). From Lemma 3.3, $\boldsymbol{z}^* + \overline{\boldsymbol{z}}^* = \boldsymbol{1}$. Also, let $N \le qn + q^s \cdot tn = (q + tq^s)n$ be the number of variables in LP (2). Then, we have

$$\boldsymbol{w}^T\boldsymbol{\mu} + \boldsymbol{C}^T(\boldsymbol{z} + \overline{\boldsymbol{z}}) \ge (1-\epsilon')(\boldsymbol{w}^T\boldsymbol{\mu}^* + \boldsymbol{C}^T(\boldsymbol{z}^* + \overline{\boldsymbol{z}}^*)) = (1-\epsilon')(\boldsymbol{w}^T\boldsymbol{\mu}^* + CN).$$

Thus,

$$\begin{aligned} \boldsymbol{C}^T(\boldsymbol{z} + \overline{\boldsymbol{z}}) &\ge (1-\epsilon')CN + (1-\epsilon')\boldsymbol{w}^T\mu^* - \boldsymbol{w}^T\mu \ge (1-\epsilon')CN - \boldsymbol{w}_{\mathcal{I}} \ge (1-\epsilon' - w/C)CN, \\ \boldsymbol{w}^T\boldsymbol{\mu} &\ge (1-\epsilon')\boldsymbol{w}^T\boldsymbol{\mu}^* + (1-\epsilon')(CN - \boldsymbol{C}^T(\boldsymbol{z} + \overline{\boldsymbol{z}})) - \epsilon'\boldsymbol{C}^T(\boldsymbol{z} + \overline{\boldsymbol{z}}) \ge (1-\epsilon')\boldsymbol{w}^T\boldsymbol{\mu}^* - \epsilon' CN. \end{aligned}$$

In the former inequality, we used the fact that $\boldsymbol{w}_{\mathcal{I}} \le wN$. In the latter inequality, we used the fact that $CN \ge \boldsymbol{C}^T(\boldsymbol{z} + \overline{\boldsymbol{z}})$.

From the former inequality, we have

$$\boldsymbol{1}^T(\boldsymbol{1} - \boldsymbol{z} - \overline{\boldsymbol{z}}) \le (\epsilon' + w/C)N.$$

Let S be the set of variables $\boldsymbol{z}_i$ ($= \boldsymbol{x}_{v,a}$ or $\boldsymbol{\mu}_{P,\beta}$) such that $(1 - \boldsymbol{z}_i - \overline{\boldsymbol{z}_i}) \ge \epsilon''$ where ϵ'' is a constant determined later. From Markov's inequality, we have $|S| \le (\epsilon' + w/C)/\epsilon'' \cdot N$. Let $S_{\boldsymbol{x}} = S \cap \{\boldsymbol{x}_{v,a}\}_{v \in V, a \in [q]}$ and $S_{\boldsymbol{\mu}} = S \cap \{\boldsymbol{\mu}_{P,\beta}\}_{P \in \mathcal{P}, \beta \in [q]^{V(P)}}$. The variables in $S_{\boldsymbol{x}}$ and $S_{\boldsymbol{\mu}}$ are problematic since constraints in LP (2) involving them are far from being satisfied. Thus, in what follows, we modify these variables and obtain nearly feasible solution to LP (2).

First, we construct variables $\{\boldsymbol{x}'_{v,a}\}_{v \in V, a \in [q]}$ by setting $\boldsymbol{x}'_{v,a} = \boldsymbol{x}_{v,a}$ if none of $\{\boldsymbol{x}_{v,a'}\}_{a' \in [q]}$ is in $S_{\boldsymbol{x}}$ and $\boldsymbol{x}'_{v,a} = 1/q$ if otherwise. Then, we construct variables $\{\boldsymbol{\mu}'_{P,\beta}\}_{P \in V(P), \beta \in [q]^{V(P)}}$ as follows. If none of $\{\boldsymbol{x}_{v,a}\}_{v \in V(P), a \in [q]}$ was modified in the previous step, we set $\boldsymbol{\mu}'_{P,\beta} = \boldsymbol{\mu}_{P,\beta}$. If otherwise, we set the values of $\{\boldsymbol{\mu}'_{P,\beta}\}_{\beta \in [q]^{V(P)}}$ in such a way that the distribution $\boldsymbol{\mu}'_P$ becomes consistent with the product distribution determined by $\{\boldsymbol{x}'_v\}_{v \in P}$. Note that each modification to $\boldsymbol{x}$ in the previous step involves at most $2tq^s$ modifications to $\boldsymbol{\mu}$.

We calculate the decrease of the objective function. The decrease caused by the modification to $\boldsymbol{x}_{v,a}$ is at most $\sum_{P \ni v} \boldsymbol{w}_P \le tw$, and the decrease caused by the modification to $\boldsymbol{\mu}_{P,\beta}$ is at most $\boldsymbol{w}_P \le w$. Thus, the total decrease is at most $tw|S| + 2twq^s|S| \le (\epsilon' + w/C)/\epsilon'' \cdot (1 + 2q^s)twN$.

Note that for each unmodified variable $\boldsymbol{z}_i$ ($= \boldsymbol{x}_{v,a}$ or $\boldsymbol{\mu}_{P,\beta}$), $\boldsymbol{z}'_i + \overline{\boldsymbol{z}}'_i \ge 1 - \epsilon''$ holds. Thus, $(\boldsymbol{x}', \overline{\boldsymbol{x}}', \boldsymbol{\mu}', \overline{\boldsymbol{\mu}}')$ is an ϵ''-infeasible solution with value at least

$$\begin{aligned} \boldsymbol{w}^T\boldsymbol{\mu}' &\ge (1-\epsilon')\boldsymbol{w}^T\boldsymbol{\mu}^* - \epsilon' CN - (\epsilon' + w/C)/\epsilon'' \cdot (1 + 2q^s)twN \\ &\ge (1-\epsilon')\boldsymbol{w}^T\boldsymbol{\mu}^* - (\epsilon' C + (\epsilon' + w/C)/\epsilon'' \cdot (1 + 2q^s)tw)(q + tq^s)n. \end{aligned}$$

Thus, $(\boldsymbol{x}', \boldsymbol{\mu}')$ is an ϵ''-infeasible $(1-\epsilon', (\epsilon' C + (\epsilon' + w/C)/\epsilon'' \cdot (1 + 2q^s)tw)(q + tq^s)n)$-approximate solution. By choosing $\epsilon' = \epsilon^3/(q^{O(s)}\mathrm{poly}(tw))$ and $\epsilon'' = \epsilon$, we have an ϵ-infeasible $(1-\epsilon, \epsilon n)$-approximate solution.

We need to look at q variables $\{\boldsymbol{x}_{v,a}\}_{a \in [q]}$ to decide the value of $\boldsymbol{x}'_{v,a}$, and we need to look at at most qs variables $\{\boldsymbol{x}_{v,a}\}_{v \in V(P), a \in [q]}$ to decide the value of $\boldsymbol{\mu}'_{P,\beta}$. Thus, the number of queries performed to $\mathcal{O}_{\mathcal{I}}$ is at most $\max(q, qs)\exp(\mathrm{poly}(qstw/\epsilon')) = \exp(\exp(\mathrm{poly}(qstw/\epsilon)))$. □

4 Optimal Rounding of BasicLP

In this section, using $\mathcal{O}_{\mathbf{lp}}$, we give an algorithm described in Theorem 1.1. Let $\mathcal{I} = (V, \mathcal{P}, \boldsymbol{w})$ be a Λ-CSP instance. For a mapping $\phi : V \to V'$, we define a new Λ-CSP instance $\mathcal{I}/\phi = (V', \mathcal{P}', \boldsymbol{w}')$ on the variable set V' by identifying variables of $\mathcal{I}$ that get mapped to the same variable in V'. For each constraint $P \in \mathcal{P}$ on the variable set $\{v_1, \ldots, v_k\}$ with weight $\boldsymbol{w}_P$, we have a constraint $P' \in \mathcal{P}'$ on the variable set $\{\phi(v_1), \ldots, \phi(v_k)\}$ with weight $\boldsymbol{w}_P$. For $x \in [0, 1]$, we define $x^\epsilon = (k+1)\epsilon$ where k is the positive integer such that $k\epsilon < x \leq (k+1)\epsilon$. We define $x^\epsilon = 0$ when $x = 0$. In what follows, we assume that $1/\epsilon$ is an integer. If not, we slightly decrease ϵ until $1/\epsilon$ become an integer. Let $(\boldsymbol{x}, \boldsymbol{\mu})$ be an LP solution for $\mathcal{I}$. We identify variables v of $\mathcal{I}$ that have the same values $\{\boldsymbol{x}^\epsilon_{v,a}\}_{a\in[q]}$. Formally, we consider another Λ-CSP instance $\mathcal{I}/\phi_{\boldsymbol{x}}$ where $\phi_{\boldsymbol{x}} : V \to \{0, \ldots, 1/\epsilon\}^q$ is defined as $\phi_{\boldsymbol{x}}(v) = (\boldsymbol{x}^\epsilon_{v,1}, \ldots, \boldsymbol{x}^\epsilon_{v,q})$. We have following two lemmas, the proofs of which are in Appendix C.

Lemma 4.1. *Let $\mathcal{I}$ be a Λ-CSP instance and $(\boldsymbol{x}, \boldsymbol{\mu})$ be an ϵ-infeasible LP solution for $\mathcal{I}$. Then, $(\boldsymbol{x}^\epsilon, \boldsymbol{\mu})$ is a $(q + 1)\epsilon$-infeasible LP solution for $\mathcal{I}$.*

Lemma 4.2. *Let $\mathcal{I}$ be a Λ-CSP instance and $(\boldsymbol{x}, \boldsymbol{\mu})$ be an ϵ-infeasible $(1 - \epsilon, \epsilon)$-approximate LP solution for $\mathcal{I}$, where $\epsilon > 0$ is a small constant. Then, the variable folding $\mathcal{I}/\phi_{\boldsymbol{x}}$ satisfies that*

- $\mathbf{lp}(\mathcal{I}/\phi_{\boldsymbol{x}}) \geq \mathbf{lp}(\mathcal{I}) - \epsilon \cdot \mathrm{poly}(qstw)n$,
- *The variable set of $\mathcal{I}/\phi_{\boldsymbol{x}}$ has a cardinality* $\exp(\mathrm{poly}(q/\epsilon))$.

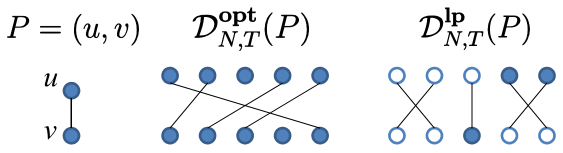
Proof of Theorem 1.1. Let ϵ' be a constant determined later and $(\boldsymbol{x}, \boldsymbol{\mu})$ be an ϵ'-infeasible $(1-\epsilon', \epsilon')$-approximate solution for $\mathcal{I}$. Consider a folded instance $\mathcal{I}' = \mathcal{I}/\phi_{\boldsymbol{x}}$ on the variable set $V' := \{0, \ldots, 1/\epsilon\}^q$. Since there are at most $\exp(\mathrm{poly}(q/\epsilon'))$ variables in V', there are at most $N := \exp(\exp(\mathrm{poly}(q/\epsilon')))$ assignments to V'. For each assignment $\beta' \in [q]^{V'}$, we estimate the value $\mathbf{val}(\mathcal{I}', \beta')$ as follows. First, we note that β' can be unfolded to an assignment $\beta \in [q]^V$ to $\mathcal{I}$ with the same value. Then, for each variable $v \in V$, we associate a value $f_v = \sum_{P \ni v} P(\beta)/|P|$. It is clear that $0 \leq f_v \leq tw$ and $\sum_{v\in V} f_v = \mathbf{val}(\mathcal{I}, \beta) = \mathbf{val}(\mathcal{I}', \beta')$. Also, we can calculate the value f_v by querying $\mathcal{O}_{\mathbf{lp}}$ at most qst times. Thus, using the algorithm given in Lemma 2.4, we get a $(1, \epsilon n/2)$-approximation to $\mathbf{val}(\mathcal{I}', \beta')$ with probability at least $1 - 1/3N$ by querying $\mathcal{O}_{\mathbf{lp}}$ at most $\mathrm{poly}(qstw/\epsilon)O(\log N)$ times.

By the union bound, with probability at least $2/3$, we obtain a $(1, \epsilon n/2)$-approximation to $\mathbf{val}(\mathcal{I}', \beta')$ for every assignment β'. Let $\beta'^* \in [q]^{V'}$ be the assignment that takes the maximum value among those assignments. Then, $\mathbf{val}(\mathcal{I}', \beta'^*)$ is a $(1, \epsilon n/2)$-approximation to $\mathbf{opt}(\mathcal{I}')$. The number of queries performed to $\mathcal{O}_{\mathbf{lp}}$ is at most $\mathrm{poly}(qstw/\epsilon)O(N \log N)$.

Let $\beta^* \in [q]^V$ be the unfolded assignment of β'^*. We can safely assume that $\boldsymbol{w}_{\mathcal{I}'} = \boldsymbol{w}_{\mathcal{I}} \geq \epsilon n$. If not, $\mathbf{val}(\mathcal{I}, \beta^*)$ is indeed a $(1, \epsilon n)$-approximation to $\mathbf{opt}(\mathcal{I})$. When $\boldsymbol{w}_{\mathcal{I}'} = \boldsymbol{w}_{\mathcal{I}} \geq \epsilon n$, it holds that

$$\begin{aligned}
\mathbf{val}(\mathcal{I}, \beta^*) &\geq \mathbf{opt}(\mathcal{I}') - \frac{\epsilon n}{2} \geq S_\Lambda(\overline{\mathbf{lp}}(\mathcal{I}'))\boldsymbol{w}_{\mathcal{I}} - \frac{\epsilon n}{2} \\
&\geq S_\Lambda\left(\overline{\mathbf{lp}}(\mathcal{I}) - \frac{\epsilon' \cdot \mathrm{poly}(qstw)n}{\boldsymbol{w}_{\mathcal{I}}}\right)\boldsymbol{w}_{\mathcal{I}} - \frac{\epsilon n}{2} \quad \text{(using Lemma 4.2)} \\
&\geq S_\Lambda\left(\overline{\mathbf{lp}}(\mathcal{I}) - \frac{\epsilon' \cdot \mathrm{poly}(qstw)}{\epsilon}\right)\boldsymbol{w}_{\mathcal{I}} - \frac{\epsilon n}{2} \quad (\text{using } \boldsymbol{w}_{\mathcal{I}} \geq \epsilon n)
\end{aligned}$$

We are done by setting $\epsilon' = \epsilon^2/\mathrm{poly}(qstw)$. The number of queries performed to $\mathcal{O}_{\mathcal{I}}$ is at most $\mathrm{poly}(qstw/\epsilon)O(N \log N) \cdot \exp(\exp(\mathrm{poly}(qstw/\epsilon'))) = \exp(\exp(\mathrm{poly}(qstw/\epsilon)))$. Once we have fixed β^*, given a variable $v \in V$, we can compute β^*_v by accessing $\mathcal{O}_{\mathbf{lp}}$ q times. The query complexity is at most $\exp(\exp(\mathrm{poly}(qstw/\epsilon)))$. □



$P = (u, v)$
$\mathcal{D}^{\mathbf{opt}}_{N,T}(P)$
$\mathcal{D}^{\mathbf{lp}}_{N,T}(P)$
u
v

Figure 1: Construction of $\mathcal{D}^{\mathbf{opt}}_{N,T}(P)$ and $\mathcal{D}^{\mathbf{lp}}_{N,T}(P)$. Here, the alphabet size $q = 2$, and we choose $N = 5$ and $T = 1$. Also, $\boldsymbol{\mu}^*_{P,00} = 0.4$, $\boldsymbol{\mu}^*_{P,01} = 0.2$, $\boldsymbol{\mu}^*_{P,10} = 0.4$, and $\boldsymbol{\mu}^*_{P,11} = 0$. It follows that $\boldsymbol{x}^*_{u,0} = 0.6$, $\boldsymbol{x}^*_{u,1} = 0.4$, $\boldsymbol{x}^*_{v,0} = 0.8$, and $\boldsymbol{x}^*_{v,1} = 0.2$. White (resp., black) variables in $\mathcal{D}^{\mathbf{lp}}_{N,T}(P)$ indicate that they are assigned to 0 (resp., 1).

5 Lower Bounds

In this section, we prove Theorem 1.2. As we described in the introduction, we utilize Yao's minimax principle. That is, we construct two distributions of instances such that they have much different optimal values and also it is hard to distinguish them in constant time. We fix a Λ-CSP instance $\mathcal{I} = (V, \mathcal{P}, \boldsymbol{w})$ with the optimal LP solution $(\boldsymbol{x}^*, \boldsymbol{\mu}^*)$ throughout this section. To convert the LP integrality gap $\overline{\mathbf{opt}}(\mathcal{I})/\overline{\mathbf{lp}}(\mathcal{I})$ of $\mathcal{I}$ to hardness results, we construct two distributions $\mathcal{D}^{\mathbf{opt}}_{N,T}$ and $\mathcal{D}^{\mathbf{lp}}_{N,T}$ using $\mathcal{I}$ and $(\boldsymbol{x}^*, \boldsymbol{\mu}^*)$. Here, N and T will determine the number of variables and the maximum degree of instances generated by $\mathcal{D}^{\mathbf{opt}}_{N,T}$ and $\mathcal{D}^{\mathbf{lp}}_{N,T}$, respectively. We show that, by taking T as a large constant (independent of N), almost all instances $\mathcal{J}$ in $\mathcal{D}^{\mathbf{opt}}_{N,T}$ satisfy that $\overline{\mathbf{opt}}(\mathcal{J}) \leq \overline{\mathbf{opt}}(\mathcal{I}) + \epsilon$. Also, we show that all instances $\mathcal{J}$ in $\mathcal{D}^{\mathbf{lp}}_{N,T}$ satisfy that $\overline{\mathbf{opt}}(\mathcal{J}) \geq \overline{\mathbf{lp}}(\mathcal{I})$. Finally, we define $\mathcal{D}^{\star}_{N,T}$ as the distribution that chooses $\mathcal{D}^{\mathbf{opt}}_{N,T}$ or $\mathcal{D}^{\mathbf{lp}}_{N,T}$ randomly and outputs an instance generated by the chosen distribution. Then, given an oracle access $\mathcal{O}_{\mathcal{J}}$ to an instance $\mathcal{J}$ generated by $\mathcal{D}^{\star}_{N,T}$, a deterministic algorithm is supposed to guess the original distribution ($\mathcal{D}^{\mathbf{opt}}_{N,T}$ or $\mathcal{D}^{\mathbf{lp}}_{N,T}$) of $\mathcal{J}$ with probability at least 2/3. By showing that such an algorithm requires $\Omega(\sqrt{N})$ queries, we conclude that any randomized algorithm that, given an instance $\mathcal{J}$, distinguishes the case $\overline{\mathbf{opt}}(\mathcal{J}) \geq \overline{\mathbf{lp}}(\mathcal{I})$ from the case $\overline{\mathbf{opt}}(\mathcal{J}) \leq \overline{\mathbf{opt}}(\mathcal{I}) + \epsilon$ requires $\Omega(\sqrt{N})$ queries. By choosing as $\mathcal{I}$ an instance with the worst integrality gap, we have the desired result.

Construction of $\mathcal{D}^{\mathbf{opt}}_{N,T}$: Before stating the construction of $\mathcal{D}^{\mathbf{opt}}_{N,T}$, we introduce a distribution $\mathcal{D}^{\mathbf{opt}}_{N,T}(P)$ for a constraint $P \in \mathcal{P}$ applied to a sequence of variables $\{v_1, \ldots, v_k\}$ (see Fig. 1). An instance $\mathcal{J}_P$ of $\mathcal{D}^{\mathbf{opt}}_{N,T}(P)$ is generated as follows. The variable set of $\mathcal{J}_P$ is $\{v_1, \ldots, v_k\} \times [N]$. We naturally regard that the set of variables $V_i = \{(v_i, j) \mid j \in [N]\}$ corresponds to a variable v_i. Next, we create TN constraints among those variables. To this end, after splitting each variable of $\mathcal{J}_P$ into T copies, we take random perfect k-partite matching in such a way that each matching takes one variable from each V_i. For each such matching $\{u_1, \ldots, u_k\}$ where u_i is of the form (v_i, j_i), we create a constraint $P(u_1, \ldots, u_k)$ of weight $\boldsymbol{w}_P$. Finally, we merge the split variables.

We define the distribution $\mathcal{D}^{\mathbf{opt}}_{N,T}$ using $\mathcal{D}^{\mathbf{opt}}_{N,T}(P)$. An instance $\mathcal{J}$ of $\mathcal{D}^{\mathbf{opt}}_{N,T}$ is generated as follows. For each $P \in \mathcal{P}$, we create an instance $\mathcal{J}_P$ according to the distribution $\mathcal{D}^{\mathbf{opt}}_{N,T}(P)$. Then, $\mathcal{J}$ is a union of $\{\mathcal{J}_P\}_{P \in \mathcal{P}}$ obtained by merging variable sets as follows. Let $P_1, \ldots, P_\ell \in \mathcal{P}$ be the set of constraints containing a variable $v \in V$. We let $V_i (i \in [\ell])$ denote the set of variables in $\mathcal{J}_{P_i}$ corresponding to v. Then, we take random perfect ℓ-partite matching among $V_1, \ldots, V_\ell$, and we merge variables in each matching. We repeat the same process for every $v \in V$. We note that the variable set of $\mathcal{J}$ is $V \times [N]$, and the number of constraints in $\mathcal{J}$ is $|\mathcal{P}|TN$. Now, we decide the indices of constraints, which are used as arguments of the oracle access $\mathcal{O}_{\mathcal{J}}$. We use the following rule. Suppose that P is the i-th constraint where $v \in P$ appears (in the sense of $\mathcal{I}$), then for a variable $(v, j) (j \in [N])$, we randomly assign T indices $\{(T-1)i + 1, \ldots, Ti\}$ to designate T constraints made by $\mathcal{J}_P$. Finally, labels of vertices are randomly permuted.

Construction of $\mathcal{D}^{\mathbf{lp}}_{N,T}$: Before stating the construction of $\mathcal{D}^{\mathbf{lp}}_{N,T}$, again we introduce another distribution $\mathcal{D}^{\mathbf{lp}}_{N,T}(P)$ for a constraint $P \in \mathcal{P}$ applied to a sequence of variables $\{v_1, \ldots, v_k\}$. (see Fig. 1). An instance $\mathcal{J}_P$ is generated as follows. The variable set of $\mathcal{J}_P$ is $\{v_1, \ldots, v_k\} \times [N]$. We naturally regard that the set of variables $V_i = \{(v_i, j) \mid j \in [N]\}$ corresponds to a variable v_i. For each $\beta \in [q]^{V(P)}$, we take a $\boldsymbol{\mu}^*_{P,\beta}$-fraction of variables from each V_i, and let $V_{i,\beta} \subseteq V_i$ denote the set of such variables. Variables in $V_{i,\beta}$ are said to be *assigned to* $\beta_{v_i} \in [q]$. A subtlety here is that $\boldsymbol{\mu}_{P,\beta}N$ may not be an integer. We ignore this issue for simplicity since we can make the error arbitrarily small by choosing N large enough. Next, we create $\boldsymbol{\mu}_{P,\beta}TN$ constraints among $V_{1,\beta}, \ldots, V_{k,\beta}$. To this end, after splitting each variable into T copies, we take random perfect k-partite matching in such a way that each matching takes one variable from each $V_{i,\beta}$. For each matching $\{u_1, \ldots, u_k\}$ where u_i is of the form (v_i, j_i), we create a constraint $P(u_1, \ldots, u_k)$ of weight $\boldsymbol{w}_P$. Finally, we merge the split variables again. We note that, for any variable $v_i \in V(P)$, an $\boldsymbol{x}^*_{v_i,a}$-fraction of variables of V_i is assigned to a.

We define the distribution $\mathcal{D}^{\mathbf{lp}}_{N,T}$ using $\mathcal{D}^{\mathbf{lp}}_{N,T}(P)$. An instance $\mathcal{J}$ of $\mathcal{D}^{\mathbf{lp}}_{N,T}$ is generated as follows. For each $P \in \mathcal{P}$, we create an instance $\mathcal{J}_P$ according to the distribution $\mathcal{D}^{\mathbf{lp}}_{N,T}(P)$. Then, $\mathcal{J}$ is a union of $\{\mathcal{J}_P\}_{P \in \mathcal{P}}$ obtained by merging variable sets as follows. Let $P_1, \ldots, P_\ell \in \mathcal{P}$ be the set of constraints containing a variable $v \in V$. We let $V_{i,a} (i \in [\ell], a \in [q])$ denote the set of variables in $\mathcal{J}_{P_i}$ that correspond to v and are assigned to a. Note that the sizes of $V_{i,a} (i \in [\ell])$ are the same from the construction of $\mathcal{D}^{\mathbf{lp}}_{N,T}(P_i)$. We take random perfect ℓ-partite matching among $V_{1,a}, \ldots, V_{\ell,a}$ and we merge vertices in each matching. We repeat the same process for every $v \in V$ and $a \in [q]$. Note that the variable set of $\mathcal{J}$ is $V \times [N]$ and the number of constraints in $\mathcal{J}$ is $|\mathcal{P}|TN$. To decide the indices of constraints and labels of vertices, we use the same rule as $\mathcal{D}^{\mathbf{opt}}_{N,T}$.

We have the following three lemmas, the proofs of which are in Appendix D.

Lemma 5.1. *For every $\epsilon > 0$, there is a $T > 0$ satisfying the following. Let $\mathcal{J}$ be a Λ-CSP instance generated by $\mathcal{D}^{\mathbf{opt}}_{N,T}$. With probability $1 - o(1)$, $\overline{\mathbf{opt}}(\mathcal{J}) \leq \overline{\mathbf{opt}}(\mathcal{I}) + \epsilon$.*

Lemma 5.2. *Let $\mathcal{J}$ be a Λ-CSP instance generated by $\mathcal{D}^{\mathbf{lp}}_{N,T}$. Then, $\overline{\mathbf{opt}}(\mathcal{J}) \geq \overline{\mathbf{lp}}(\mathcal{I})$ holds.*

Lemma 5.3. *In the bounded-degree model, any deterministic algorithm that, given an oracle access to $\mathcal{O}_{\mathcal{J}}$ generated by $\mathcal{D}^{\star}_{N,T}$, correctly guesses the original distribution of $\mathcal{J}$ with probability at least $3/5$ requires at least $\Omega(\sqrt{N})$ queries.*

Proof of Theorem 1.2. Let us fix $c \in [0, 1]$ and $s = S_\Lambda(c)$. Then, there exists a Λ-CSP instance $\mathcal{I}$ such that $\overline{\mathbf{lp}}(\mathcal{I}) = c$ and $\overline{\mathbf{opt}}(\mathcal{I})$ is arbitrarily close to s. Suppose that there exists a deterministic algorithm $\mathcal{A}$ with query complexity $o(\sqrt{n})$ that, given an instance $\mathcal{J}$ of n variables, with probability at least $2/3$, distinguishes the case $\mathbf{opt}(\mathcal{J}) \geq c\boldsymbol{w}_{\mathcal{J}}$ from the case $\mathbf{opt}(\mathcal{J}) \leq (s + \epsilon)\boldsymbol{w}_{\mathcal{J}} - \epsilon n$. Let T be a constant given by Lemma 5.1 by replacing ϵ with $\epsilon/2$.

Suppose that $\mathcal{J}$ is generated by $\mathcal{D}^{\mathbf{opt}}_{N,T}$. Then, from Lemma 5.1, with probability at least $1 - o(1)$, it holds that $\mathbf{opt}(\mathcal{J}) \leq (s + \epsilon/2)\boldsymbol{w}_{\mathcal{J}} = (s + \epsilon)\boldsymbol{w}_{\mathcal{J}} - \epsilon\boldsymbol{w}_{\mathcal{J}}/2 \leq (s + \epsilon)\boldsymbol{w}_{\mathcal{J}} - \epsilon n$. In the last inequality, we use the fact that $\boldsymbol{w}_{\mathcal{J}}/2 \geq \boldsymbol{w}_{\mathcal{J}}/T \geq n$ when ϵ is small. Suppose that $\mathcal{J}$ is generated by $\mathcal{D}^{\mathbf{lp}}_{N,T}$. Then, from Lemma 5.2, it holds that $\mathbf{opt}(\mathcal{J}) \geq c\boldsymbol{w}_{\mathcal{J}}$.

Thus, in total, the algorithm outputs the correct answer with probability at least $1/2 \cdot (1 - o(1)) \cdot 2/3 + 1/2 \cdot 2/3 = 2/3 - o(1)$. This contradicts Lemma 5.3. □

References

[1] Noga Alon. On constant time approximation of parameters of bounded degree graphs, 2010. manuscript.

[2] Noga Alon, Wenceslas Fernandez de la Vega, Ravi Kannan, and Marek Karpinski. Random sampling and approximation of MAX-CSPs. *Journal of Computer and System Sciences*, 67(2):212–243, 2003.

[3] Gunnar Andersson and Lars Engebretsen. Property testers for dense constraint satisfaction programs on finite domains. *Random Struct. Algorithms*, 21(1):14–32, 2002.

[4] Per Austrin. Balanced MAX 2-SAT might not be the hardest. In *Proc. of STOC 2007*, pages 189–197, 2007.

[5] Per Austrin and Elchanan Mossel. Approximation resistant predicates from pairwise independence. In *Proc. of CCC 2008*, pages 249–258, 2008.

[6] Andrej Bogdanov, Kenji Obata, and Luca Trevisan. A lower bound for testing 3-colorability in bounded-degree graphs. In *Proc. of FOCS 2002*, pages 93–102, 2002.

[7] Bernard Chazelle, Ronitt Rubinfeld, and Luca Trevisan. Approximating the minimum spanning tree weight in sublinear time. In *Proc. of ICALP 2001*, pages 190–200, 2001.

[8] Dimitris A. Fotakis and Paul G. Spirakis. Linear programming and fast parallel approximability, 1997. manuscript.

[9] Michel X. Goemans and David P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *J. ACM*, 42(6):1115–1145, 1995.

[10] Oded Goldreich and Dana Ron. A sublinear bipartiteness tester for bounded degree graphs. *Combinatorica*, 19(3):335–373, 1999.

[11] Oded Goldreich and Dana Ron. Property testing in bounded degree graphs. *Algorithmica*, 32(2):302–343, 2008.

[12] Oded Goldreich and Luca Trevisan. Three theorems regarding testing graph properties. *Random Struct. Algorithms*, 23(1):23–57, 2003.

[13] Venkatesan Guruswami, Rajsekar Manokaran, and Prasad Raghavendra. Beating the random ordering is hard: Inapproximability of maximum acyclic subgraph. In *Proc. of FOCS 2008*, pages 573–582, 2008.

[14] Johan Håstad. Some optimal inapproximability results. *J. ACM*, 48(4):798–859, 2001.

[15] David Karger, Rajeev Motwani, and Madhu Sudan. Approximate graph coloring by semidefinite programming. *J. ACM*, 45(2):246–265, 1998.

[16] Subhash Khot. On the power of unique 2-prover 1-round games. In *Proc. of STOC 2002*, pages 767–775, 2002.

[17] Subhash Khot, Guy Kindler, Elchanan Mossel, and Ryan O'Donnell. Optimal inapproximability results for MAX-CUT and other 2-variable CSPs? In *Proc. of FOCS 2004*, pages 146–154, 2004.

[18] Subhash Khot and Assaf Naor. Sharp kernel clustering algorithms and their associated grothendieck inequalities. *CoRR*, abs/0906.4816, 2009.

[19] Fabian Kuhn, Thomas Moscibroda, and Roger Wattenhofer. The price of being near-sighted. In *Proc. of SODA 2006*, pages 980–989, 2006.

[20] Amit Kumar, Rajsekar Manokaran, Madhur Tulsiani, and Nisheeth K. Vishnoi. On the optimality of a class of LP-based algorithms. *CoRR*, abs/0912.1776, 2009.

[21] Michael Lewin, Dror Livnat, and Uri Zwick. Improved rounding techniques for the MAX 2-SAT and MAX DI-CUT problems. In *Proc. of IPCO 2002*, pages 67–82, 2002.

[22] Rajsekar Manokaran, Joseph (Seffi) Naor, Prasad Raghavendra, and Roy Schwartz. SDP gaps and UGC hardness for multiway cut, 0-extension, and metric labeling. In *Proc. of STOC 2008*, pages 11–20, 2008.

[23] Huy N. Nguyen and Krzysztof Onak. Constant-time approximation algorithms via local improvements. In *Proc. of FOCS 2008*, pages 327–336, 2008.

[24] Michal Parnas and Dana Ron. Approximating the minimum vertex cover in sublinear time and a connection to distributed algorithms. *Theor. Comput. Sci.*, 381(1-3):183–196, 2007.

[25] Prasad Raghavendra. Optimal algorithms and inapproximability results for every CSP? In *Proc. of STOC 08*, pages 245–254, 2008.

[26] Prasad Raghavendra and David Steurer. How to round any CSP. In *Proc. of FOCS 2009*, pages 586–594, 2009.

[27] Alex Samorodnitsky and Luca Trevisan. Gowers uniformity, influence of variables, and PCPs. In *Proc. of STOC 2006*, pages 11–20. ACM, 2006.

[28] Nick Wormald. Models of random regular graphs. In *Surveys in Combinatorics*, pages 239–298. Cambridge University Press, 1999.

[29] Yuichi Yoshida. Lower bounds on query complexity for testing bounded-degree CSPs, 2010. manuscript.

[30] Yuichi Yoshida, Masaki Yamamoto, and Hiro Ito. An improved constant-time approximation algorithm for maximum matchings. In *Proc. of STOC 2009*, pages 225–234, 2009.

[31] Uri Zwick. Finding almost-satisfying assignments. In *Proc. of STOC 1998*, pages 551–560, 1998.

Appendix

A Robustness of BasicLP

In this section, we give a proof of Lemma 2.5. Our strategy is transforming $(\boldsymbol{x}, \boldsymbol{\mu})$ to a feasible solution without decreasing the LP value much. In the first step, we construct $\boldsymbol{x}'$ from $\boldsymbol{x}$ that satisfies $\sum_{a \in [q]} \boldsymbol{x}'_{v,a} = 1$ for every $v \in V$.

Lemma A.1. *Let $(\boldsymbol{x}, \boldsymbol{\mu})$ be an ϵ-infeasible LP solution for a Λ-CSP instance $\mathcal{I}$ where $\epsilon > 0$ is a small constant. Then, $\boldsymbol{x}$ can be transformed to $\boldsymbol{x}'$ so that*

$$\sum_{a \in [q]} \boldsymbol{x}'_{v,a} = 1 \quad \forall v \in V, \tag{3}$$

$$|\boldsymbol{x}'_{v,a} - \boldsymbol{x}_{v,a}| \leq 2\epsilon \quad \forall v \in V, a \in [q]. \tag{4}$$

In particular, $(\boldsymbol{x}', \boldsymbol{\mu})$ is a 3ϵ-infeasible LP solution that satisfies $\sum_{a \in [q]} \boldsymbol{x}'_{v,a} = 1$ for every $v \in V$.

Proof. We define $\boldsymbol{x}'_{v,a} = \boldsymbol{x}_{v,a} / \sum_{a \in [q]} \boldsymbol{x}_{v,a}$. The condition (3) clearly holds. From the ϵ-infeasibility of $\boldsymbol{x}$, $|\sum_{a \in [q]} \boldsymbol{x}_{v,a} - 1| \leq \epsilon$ holds. It follows that $|\boldsymbol{x}'_{v,a} - \boldsymbol{x}_{v,a}| \leq \epsilon/(1-\epsilon) \leq 2\epsilon$ when ϵ is small. □

In the second step, we construct $\boldsymbol{\mu}'$ that satisfies $\sum_{\beta \in [q]^{V(P)}, \beta_v = a} \boldsymbol{\mu}'_{P,\beta} = \boldsymbol{x}'_{v,a}$ for all $P \in \mathcal{P}, v \in V(P)$.

Lemma A.2. *Let $(\boldsymbol{x}, \boldsymbol{\mu})$ be an ϵ-infeasible solution for a Λ-CSP instance $\mathcal{I}$ satisfying $\sum_{a \in [q]} \boldsymbol{x}_{v,a} = 1$ for every $v \in V$. Then, $\boldsymbol{\mu}$ can be transformed to $\boldsymbol{\mu}'$ so that*

$$\Pr_{\beta \sim \boldsymbol{\mu}'_P}[\beta_v = a] = (1-\delta)\boldsymbol{x}_{v,a} + \frac{\delta}{q} \quad \forall P \in \mathcal{P}, v \in V(P), a \in [q],$$

$$||\boldsymbol{\mu}_P - \boldsymbol{\mu}'_P||_1 \leq 2\delta \quad \forall P \in \mathcal{P}.$$

where $\delta = kq^3\epsilon$.

Proof. Let us fix a predicate $P \in \mathcal{P}$ and $S = V(P)$. We may assume $S = \{1, \ldots, k\}$ where $k \leq s$. We can think of $\boldsymbol{\mu}_P$ as a function $f : [q]^k \to \mathbb{R}$ such that $f(\beta)$ is the probability of the assignment β under the distribution $\boldsymbol{\mu}_P$.

Let $\chi_1, \ldots, \chi_q$ be an orthonormal basis of the vector space $\{f : [q] \to \mathbb{R}\}$ such that $\chi_1 \equiv 1$. Here, orthonormal means that $E_{a \in [q]}[\chi_i(a)\chi_j(a)] = \delta_{ij}$ for all $i, j \in [q]$ where δ is Kronecker's delta. By tensoring this basis, we obtain the orthonormal basis $\{\chi_\rho\}_{\rho \in [q]^k}$ of the vector space $\{f : [q]^k \to \mathbb{R}\}$. That is, for $\rho \in [q]^k, \beta \in [q]^k$, we have $\chi_\rho(\beta) = \chi_{\rho_1}(\beta_1) \cdots \chi_{\rho_k}(\beta_k)$. For a function $f : [q]^k \to \mathbb{R}$, we define $\hat{f}(\sigma) = \sum_{\beta \in [q]^k} f(x)\chi_\sigma(\beta)$. Note that $f(\beta) = E_{\sigma \in [q]^k}[\hat{f}(\sigma)\chi_\sigma(\beta)]$. Therefore, if we let f again be the function corresponding to $\boldsymbol{\mu}_P$, we have

$$\Pr_{\beta \sim \boldsymbol{\mu}_P}[\beta_i = a] = \sum_{\beta \in [q]^k, \beta_i = a} \mathop{\mathrm{E}}_{\sigma \in [q]^k}\left[\widehat{f}(\sigma)\chi_\sigma(\beta)\right] = \mathop{\mathrm{E}}_{\sigma \in [q]}\left[\widehat{f}_i(\sigma)\chi_\sigma(a)\right].$$

Here, $\widehat{f}_i(s) = \widehat{f}(\sigma)$ where $\sigma_i = s$ and $\sigma_r = 1$ for all $r \in [k] \setminus \{i\}$. In the second inequality, we used that for every σ with $\sigma_r \neq 1$ for some $r \in [k] \setminus \{i\}$, the sum over the values of χ_σ vanishes.

We let $g_i : [q] \to \mathbb{R}$ be the function $g_i(a) = \boldsymbol{x}_{i,a}$. We define a function $f' : [q]^k \to \mathbb{R}$ as follows.

$$\widehat{f'}(\sigma) = \begin{cases} \widehat{g_i}(s) & \text{if } \sigma_i = s \text{ and } \sigma_r = 1 \text{ for all } r \in [k] \setminus \{i\}, \\ \widehat{f}(\sigma) & \text{otherwise.} \end{cases}$$

This is well-defined since for any $i \in [k]$, it holds that $\widehat{g}_i(1) = \sum_{a\in[q]} g_i(a) = \sum_{a\in[q]} \boldsymbol{x}_{i,a} = 1$. Therefore, the function f' satisfies $\sum_{\beta\in[q]^k} f'(\beta) = \widehat{f}(1) = 1$, Then, we can define a distribution $\boldsymbol{\mu}'_P$ corresponding to f', and we have

$$\Pr_{\beta\sim\boldsymbol{\mu}'_P}[\beta_i = a] = \mathop{\mathrm{E}}_{\sigma\in[q]}\left[\widehat{f}'_i(\sigma)\chi_\sigma(a)\right] = \boldsymbol{x}_{v,a}.$$

Thus, it looks that the $\boldsymbol{\mu}'_P$ is the desired distribution. However, in general, the function f' might take negative values. We will show that these values cannot be too negative and that the function can be made to a proper distribution by smoothing.

Let K be an upper bound on the values of the functions $\chi_1, \ldots, \chi_q$. From the orthonormality of the functions, it follows that $K \le \sqrt{q}$. Let $f_i(a) = \Pr_{\beta\sim\boldsymbol{\mu}_P}[\beta_i = a]$. Since the LP solution $(\boldsymbol{x}, \boldsymbol{\mu})$ is ϵ-infeasible, we have

$$\left|\widehat{g}_i(s) - \widehat{f}_i(s)\right| = \left|\sum_{a\in[q]} g_i(a)\chi_s(a) - \sum_{a\in[q]} f_i(a)\chi_s(a)\right| \le Kq\epsilon.$$

Therefore, $|\widehat{f}'(\sigma) - \widehat{f}(\sigma)| \le Kq\epsilon$ for all $\sigma \in [q]^k$. Recall that $|\widehat{f}'(\sigma) - \widehat{f}(\sigma)| = 0$ for $\sigma \in [q]^k$ if there are $i \ne j$ such that $\sigma_i \ne 1, \sigma_j \ne 1$. Thus,

$$|f'(\beta) - f(\beta)| = \left|\mathop{\mathrm{E}}_{\sigma\in[q]^k}\left[\widehat{f}'(\sigma)\chi_\sigma(\beta) - \widehat{f}(\sigma)\chi_\sigma(\beta)\right]\right| \le \delta/q^k, \tag{5}$$

where $\delta = K^2kq^2\epsilon$. Hence, if we let $h = (1-\delta)f' + \delta U$, where $U : [q]^k \to \mathbb{R}$ is the uniform distribution $U \equiv 1/q^k$, then

$$h(x) = (1-\delta)f'(x) + \delta/q^k \ge (1-\delta)f(x) \ge 0.$$

It follows that h corresponds to another distribution $\boldsymbol{\mu}'_P$ over assignments $[q]^k$. Furthermore, it holds

$$\Pr_{\beta\sim\boldsymbol{\mu}'_P}[\beta_i = a] = (1-\delta)\boldsymbol{x}_{i,a} + \frac{\delta}{q}.$$

Finally, let us estimate the statistical distance between the distributions $\boldsymbol{\mu}_P$ and $\boldsymbol{\mu}'_P$.

$$||f - h||_1 = ||(1-\delta)(f - f') + \delta(f - U)||_1 \le ||f - f'||_1 + \delta \le 2\delta.$$

The first inequality is from the triangle inequality and the second inequality is from (5). □

Proof of Lemma 2.5. Let us consider an ϵ-infeasible LP solution $(\boldsymbol{x}, \boldsymbol{\mu})$ for a Λ-CSP instance $\mathcal{I}$ of value $c\boldsymbol{w}_{\mathcal{I}}$. First, we construct vector $\boldsymbol{x}'$ as in Lemma A.1. These variables together with the original local distributions $\boldsymbol{\mu}$ form an 3ϵ-infeasible LP solution for $\mathcal{I}$. Next, we construct local distributions $\boldsymbol{\mu}'$ as in Lemma A.2. Define new variables

$$\boldsymbol{x}''_{i,a} = (1-\delta)\boldsymbol{x}'_{i,a} + \delta/q.$$

It follows that $(\boldsymbol{x}'', \boldsymbol{\mu}')$ is a feasible LP solution for $\mathcal{I}$. The LP value of this solution is

$$\begin{aligned}\sum_{P\in\mathcal{P}} \boldsymbol{w}_P \mathop{\mathrm{E}}_{\beta\sim\boldsymbol{\mu}'_P}[P(\beta)] &= c\boldsymbol{w}_{\mathcal{I}} - \sum_{P\in\mathcal{P}} \boldsymbol{w}_P \sum_{\beta\in[q]^{V(P)}} P(\beta)\left(\boldsymbol{\mu}_{P,\beta} - \boldsymbol{\mu}'_{P,\beta}\right) \\ &\ge c\boldsymbol{w}_{\mathcal{I}} - \sum_{P\in\mathcal{P}} \boldsymbol{w}_P ||\boldsymbol{\mu}_P - \boldsymbol{\mu}'_P||_1 \\ &\ge c\boldsymbol{w}_{\mathcal{I}} - \epsilon\cdot\mathrm{poly}(kq)\boldsymbol{w}_{\mathcal{I}}.\end{aligned}$$

We used $|P(x)| \le 1$ for the first inequality, and the second inequality follows from Lemma A.2. □

B Proof of Lemma 3.4

In this section, we give a proof of Lemma 3.4. We consider a more restricted form of a packing LP:

$$\begin{aligned} \max \quad & \mathbf{1}^T \boldsymbol{z} \\ \text{s.t.} \quad & A^T \boldsymbol{z} \le \boldsymbol{c} \\ & \boldsymbol{z} \ge 0, \end{aligned} \tag{6}$$

where $A \in \mathbb{R}_+^{m \times n}$ is a non-negative matrix such that $a_{ji} = 0$ or $a_{ji} \ge 1$ for any $j \in [m], i \in [n]$, and $\boldsymbol{c} \in \mathbb{R}_+^n$ is a non-negative vector.

Define

$$c_{\max} = \max_i c_i, \quad \Gamma_p = \max_i \frac{c_{\max}}{c_i} \sum_{j=1}^m a_{ji}, \quad \Gamma_d = \max_j \sum_{i=1}^n a_{ji}.$$

Then, there is a distributed algorithm that solves this packing LP.

Lemma B.1 ([19])**.** *For sufficiently small $\epsilon > 0$, there exists a deterministic distributed algorithm that computes a feasible $(1-\epsilon, 0)$-approximate solution to LP (6) in $O(\log \Gamma_p \log \Gamma_d / \epsilon^4)$ rounds.* □

In order to apply Lemma B.1 to LP (2), we transform it to the form LP (6). Note that, in the objective function, the coefficient of $\boldsymbol{\mu}_{P,\beta}$ is $\boldsymbol{w}_P P(\beta) + C$ and the coefficients of $\boldsymbol{x}_{v,a}, \overline{\boldsymbol{x}}_{v,a}, \overline{\boldsymbol{\mu}}_{P,\beta}$ are C. Thus, by replacing $\boldsymbol{\mu}_{P,\beta}$ with $\boldsymbol{\mu}_{P,\beta}/(\boldsymbol{w}_P P(\beta) + C)$ and replacing $\boldsymbol{x}_{v,a}, \overline{\boldsymbol{x}}_{v,a}, \overline{\boldsymbol{\mu}}_{P,\beta}$ with $\boldsymbol{x}_{v,a}/C, \overline{\boldsymbol{x}}_{v,a}/C, \overline{\boldsymbol{\mu}}_{P,\beta}/C$, respectively, we obtain the following LP.

$$\begin{array}{lll} \max & \mathbf{1}^T(\boldsymbol{x} + \overline{\boldsymbol{x}} + \boldsymbol{\mu} + \overline{\boldsymbol{\mu}}) & \\ \text{s.t.} & \sum_{a \in [q]} \frac{\boldsymbol{x}_{v,a}}{C} \le q - 1 + \epsilon & \forall v \in V \\ & \sum_{a \in [q]} \frac{\overline{\boldsymbol{x}}_{v,a}}{C} \le 1 + \epsilon & \forall v \in V \\ & \frac{\boldsymbol{x}_{v,a}}{C} + \sum_{\beta \in [q]^{V(P)}, \beta_v = a} \frac{\boldsymbol{\mu}_{P,\beta}}{\boldsymbol{w}_P P(\beta) + C} \le 1 + \epsilon & \forall P \in \mathcal{P}, v \in V(P), a \in [q] \\ & \frac{\overline{\boldsymbol{x}}_{v,a}}{C} + \sum_{\beta \in [q]^{V(P)}, \beta_v = a} \frac{\overline{\boldsymbol{\mu}}_{P,\beta}}{C} \le q^{V(P)-1} + \epsilon & \forall P \in \mathcal{P}, v \in V(P), a \in [q] \\ & \frac{\boldsymbol{x}_{v,a}}{C} + \frac{\overline{\boldsymbol{x}}_{v,a}}{C} \le 1, \quad \boldsymbol{x}_{v,a} \ge 0, \quad \overline{\boldsymbol{x}}_{v,a} \ge 0 & \forall v \in V \\ & \frac{\boldsymbol{\mu}_{P,\beta}}{\boldsymbol{w}_P P(\beta) + C} + \frac{\overline{\boldsymbol{\mu}}_{P,\beta}}{C} \le 1, \quad \boldsymbol{\mu}_{P,\beta} \ge 0, \quad \overline{\boldsymbol{\mu}}_{P,\beta} \ge 0 & \forall P \in \mathcal{P}, \beta \in [q]^{V(P)}. \end{array}$$

We multiply each constraint in order to make every coefficient in the LHS at least 1. Then, we have the following LP.

$$\begin{array}{lll} \max & \mathbf{1}^T(\boldsymbol{x} + \overline{\boldsymbol{x}} + \boldsymbol{\mu} + \overline{\boldsymbol{\mu}}) & \\ \text{s.t.} & \sum_{a \in [q]} \boldsymbol{x}_{v,a} \le C(q - 1 + \epsilon) & \forall v \in V \\ & \sum_{a \in [q]} \overline{\boldsymbol{x}}_{v,a} \le C(1 + \epsilon) & \forall v \in V \\ & \frac{w+C}{C} \boldsymbol{x}_{v,a} + \sum_{\beta \in [q]^{V(P)}, \beta_v = a} \frac{w+C}{\boldsymbol{w}_P P(\beta) + 1} \boldsymbol{\mu}_{P,\beta} \le (1 + \epsilon)(w + C) & \forall P \in \mathcal{P}, v \in V(P), a \in [q] \\ & \overline{\boldsymbol{x}}_{v,a} + \sum_{\beta \in [q]^{V(P)}, \beta_v = a} \overline{\boldsymbol{\mu}}_{P,\beta} \le C(q^{V(P)-1} + \epsilon) & \forall P \in \mathcal{P}, v \in V(P), a \in [q] \\ & \boldsymbol{x}_{v,a} + \overline{\boldsymbol{x}}_{v,a} \le C, \quad \boldsymbol{x}_{v,a} \ge 0, \quad \overline{\boldsymbol{x}}_{v,a} \ge 0 & \forall v \in V \\ & \frac{w+C}{\boldsymbol{w}_P P(\beta) + 1} \boldsymbol{\mu}_{P,\beta} + \frac{w+C}{C} \overline{\boldsymbol{\mu}}_{P,\beta} \le w + C, \quad \boldsymbol{\mu}_{P,\beta} \ge 0, \quad \overline{\boldsymbol{\mu}}_{P,\beta} \ge 0 & \forall P \in \mathcal{P}, \beta \in [q]^{V(P)}. \end{array} \tag{7}$$

Proof of Lemma 3.4. Note that LP (7) is of the form LP (6). After a calculation, we have

$$c_{\max} = O(C(w+q^s)), \quad \Gamma_p = O((s+t)(w+C)\cdot C(w+q^s)), \quad \Gamma_d = O((w+C)q^s).$$

We define the *degree* of a variable in an LP as the number of inequalities where the variable appears. Let Δ_p and Δ_d be the maximum degree of primal variables and dual variables, respectively. Here, we treat LP (7) as a dual formulation. We have

$$\Delta_p = O(q^s), \quad \Delta_d = O(s+t).$$

Applying the algorithm given in Lemma B.1 to LP (7), we obtain a distributed algorithm that calculates $(1-\epsilon, 0)$-approximate solution. The number of rounds is $O(\log \Gamma_p \log \Gamma_d/\epsilon^4)$. Note that, given a variable, we can simulate the computation of the distributed algorithm involved by the variable with $(\Delta_p\Delta_d)^r$ queries, where r is the number of rounds. Thus, the query complexity becomes

$$(\Delta_p\Delta_d)^{O(\log\Gamma_p \log\Gamma_d/\epsilon^4)} = \exp(\mathrm{poly}(qstw/\epsilon)).$$

□

C Proofs from Section 4

C.1 Proof of Lemma 4.1

Proof. Since we move each $\boldsymbol{x}_{v,a}$ by at most ϵ, each constraint $\sum_{a\in[q]} \boldsymbol{x}^{\epsilon}_{v,a} = 1$ can be at most $(q+1)\epsilon$-infeasible. Also, each constraint $\sum_{\beta\in[q]^{V(P)},\beta_v=a} \boldsymbol{\mu}_{P,\beta} = \boldsymbol{x}^{\epsilon}_{v,a}$ can be at most 2ϵ-infeasible. □

C.2 Proof of Lemma 4.2

Proof. Since the size of the range of $\phi_{\boldsymbol{x}}$ is $(1/\epsilon)^{O(q)}$, the second claim is obvious.

Suppose that $(\boldsymbol{x},\boldsymbol{\mu})$ has an LP value $c\boldsymbol{w}_{\mathcal{I}}$. From the fact that $(\boldsymbol{x},\boldsymbol{\mu})$ is a $(1-\epsilon,\epsilon)$-approximate solution, we have $c\boldsymbol{w}_{\mathcal{I}} \geq (1-\epsilon)\mathbf{lp}(\mathcal{I}) - \epsilon n$. Also, by Lemma 4.1, $(\boldsymbol{x}^{\epsilon},\boldsymbol{\mu})$ is a $(q+1)\epsilon$-infeasible LP solution. Since only $\boldsymbol{\mu}$ affects the value of the objective function, the LP value of $(\boldsymbol{x}^{\epsilon},\boldsymbol{\mu})$ equals $c\boldsymbol{w}_{\mathcal{I}}$. A key observation is that $(\boldsymbol{x}^{\epsilon},\boldsymbol{\mu})$ is also an LP solution for the folded instance $\mathcal{I}/\phi_{\boldsymbol{x}}$. Thus, we see that $\mathcal{I}/\phi_{\boldsymbol{x}}$ has a $(q+1)\epsilon$-infeasible solution of value at least $c\boldsymbol{w}_{\mathcal{I}}$. From Lemma 2.5, we have

$$\begin{aligned}\mathbf{lp}(\mathcal{I}/\phi_{\boldsymbol{x}}) &\geq (c-(q+1)\epsilon\cdot\mathrm{poly}(qs))\boldsymbol{w}_{\mathcal{I}} \\ &\geq (1-\epsilon)\mathbf{lp}(\mathcal{I}) - \epsilon n - \epsilon\cdot\mathrm{poly}(qs)\boldsymbol{w}_{\mathcal{I}} \\ &\geq \mathbf{lp}(\mathcal{I}) - \epsilon\cdot\mathrm{poly}(qstw)n.\end{aligned}$$

In the last inequality, we use the fact that $\mathbf{lp}(\mathcal{I}) \leq \boldsymbol{w}_{\mathcal{I}} \leq twn$. □

D Proofs from Section 5

D.1 Proof of Lemma 5.1

Let $\mathcal{J}_P$ be an instance generated by $\mathcal{D}^{\mathbf{opt}}_{N,T}(P)$. Let $P_i (1 \leq i \leq 2)$ be a constraint on a variable sequence $\{u^i_1,\ldots,u^i_k\}$ in $\mathcal{J}_P$. Note that the arities of P_i are the same since they both are copies of P. For each $j \in [k]$, we choose $v^1_j \in \{u^1_j, u^2_j\}$ arbitrarily and v^2_j be the remaining one, i.e., $\{u^1_j,u^2_j\}\setminus\{v^1_j\}$. Then, we define a constraint $Q_i (1\leq i\leq 2)$ on the variable sequence $\{v^i_1,\ldots,v^i_k\}$.

We create another instance $\mathcal{J}'_P$ from $\mathcal{J}_P$ by replacing $\{P_1, P_2\}$ by $\{Q_1, Q_2\}$. We call this method *switching*. The following concentration bound is obtained by a simple application of Theorem 2.19 in [28].

Lemma D.1. *If $\mathbf{X}$ is a random variable defined on $\mathcal{D}^{\mathbf{opt}}_{N,T}(P)$ such that $|\mathbf{X}(\mathcal{J}_P) - \mathbf{X}(\mathcal{J}'_P)| \le c$ holds where $\mathcal{J}_P$ and $\mathcal{J}'_P$ are instances of $\mathcal{D}^{\mathbf{opt}}_{N,T}(P)$ that only differ by a switching, then*

$$\Pr_{\mathcal{J}_P \sim \mathcal{D}^{\mathbf{opt}}_{N,T}(P)} [|\mathbf{X}(\mathcal{J}_P) - \mathrm{E}[\mathbf{X}(\mathcal{J}_P)]| \ge t] \le 2\exp\left(-\frac{t^2}{TNc^2}\right)$$

for all $t > 0$. □

Proof of Lemma 5.1. Let $\alpha \in [q]^{V \times [N]}$ be an assignment to $\mathcal{J}$. For $v \in V$ and $a \in [q]$, we define $\boldsymbol{x}_{v,a} = \#\{i \in [N] \mid \alpha_{(v,i)} = a\}/N$. Also, for $P \in \mathcal{P}$ and $\beta \in [q]^{V(P)}$, we define $\boldsymbol{\mu}_{P,\beta} = \prod_{v \in P} \boldsymbol{x}_{v,\beta_v}$. Note that $\boldsymbol{x}_v$ (resp., $\boldsymbol{\mu}_P$) gives a probability distribution over assignments to the variable v (resp., the variable set $V(P)$).

Let $\mathcal{J}_P$ be the sub-instance of $\mathcal{J}$ generated by $\mathcal{D}^{\mathbf{opt}}_{N,T}(P)$ for $P \in \mathcal{P}$. The expectation (over $\mathcal{D}^{\mathbf{opt}}_{N,T}(P)$) of the value gained by a constraint P in $\mathcal{J}_P$ is $\mathrm{E}_{\beta_P \sim \boldsymbol{\mu}_P}[P(\beta_P)]$. Thus, it holds that

$$\begin{aligned}\mathop{\mathrm{E}}_{\mathcal{J} \sim \mathcal{D}^{\mathbf{opt}}_{N,T}} [\mathbf{val}(\mathcal{J}, \alpha)] &= \sum_{P \in \mathcal{P}} \mathop{\mathrm{E}}_{\mathcal{J}_P \sim \mathcal{D}^{\mathbf{opt}}_{N,T}(P)} [\mathbf{val}(\mathcal{J}_P, \alpha_{|V(P)})] = TN \sum_{P \in \mathcal{P}} \boldsymbol{w}_P \mathop{\mathrm{E}}_{\beta_P \sim \boldsymbol{\mu}_P} [P(\beta_P)] \\ &= TN \sum_{P \in \mathcal{P}} \boldsymbol{w}_P \mathop{\mathrm{E}}_{\beta \sim \boldsymbol{\mu}} [P(\beta_{|V(P)})] = TN \mathop{\mathrm{E}}_{\beta \sim \boldsymbol{\mu}} [\sum_{P \in \mathcal{P}} \boldsymbol{w}_P P(\beta_{|V(P)})] = TN \mathop{\mathrm{E}}_{\beta \sim \boldsymbol{\mu}} [\mathbf{val}(\mathcal{I}, \beta)]\end{aligned}$$

Thus, it follows that

$$\mathop{\mathrm{E}}_{\mathcal{J} \sim \mathcal{D}^{\mathbf{opt}}_{N,T}} [\mathbf{val}(\mathcal{J}, \alpha)] \le TN\mathbf{opt}(\mathcal{I}). \tag{8}$$

Note that, for instances $\mathcal{J}_P$ and $\mathcal{J}'_P$ generated by $\mathcal{D}^{\mathbf{opt}}_{N,T}(P)$ such that they differ by a switching, $\mathbf{val}(\mathcal{J}_P, \alpha_{|V(P)})$ and $\mathbf{val}(\mathcal{J}'_P, \alpha_{|V(P)})$ can differ by at most $2\boldsymbol{w}_P \le 2w$. Then, from Lemma D.1,

$$\Pr\left[\left|\mathbf{val}(\mathcal{J}_P, \alpha_{|V(P)}) - \mathrm{E}[\mathbf{val}(\mathcal{J}_P, \alpha_{|V(P)})]\right| \ge t\right] \le 2\exp\left(-\frac{t^2}{4TNw^2}\right).$$

Then,

$$\begin{aligned}\Pr\left[|\mathbf{val}(\mathcal{J}, \alpha) - \mathrm{E}[\mathbf{val}(\mathcal{J}, \alpha)]| \ge t|\mathcal{P}|\right] &\le \Pr\left[\exists P \in \mathcal{P},\ |\mathbf{val}(\mathcal{J}_P, \alpha_{|V(P)}) - \mathrm{E}[\mathbf{val}(\mathcal{J}_P, \alpha_{|V(P)})]| \ge t\right] \\ &\le 2|\mathcal{P}| \exp\left(-\frac{t^2}{4TNw^2}\right).\end{aligned}$$

The last inequality is from the union bound.

We choose $t = \epsilon TN$ so that $t|\mathcal{P}| = \epsilon|\mathcal{P}|TN \le \epsilon \boldsymbol{w}_{\mathcal{J}}$. We have

$$\Pr\Big[|\mathbf{val}(\mathcal{J}, \alpha) - \mathrm{E}[\mathbf{val}(\mathcal{J}, \alpha)]| \ge \epsilon \boldsymbol{w}_{\mathcal{J}}\Big] \le 2|\mathcal{P}| \exp\left(-\frac{\epsilon^2 TN}{4w^2}\right). \tag{9}$$

We combine (8) and (9) with the union bound over all $q^{|V|N}$ assignments. It holds that

$$\begin{aligned}\Pr\Big[\exists \alpha, \mathbf{val}(\mathcal{J}, \alpha) \ge TN\mathbf{opt}(\mathcal{I}) + \epsilon \boldsymbol{w}_{\mathcal{J}}\Big] &= \Pr\Big[\exists \alpha, \overline{\mathbf{val}}(\mathcal{J}, \alpha) \ge \overline{\mathbf{opt}}(\mathcal{I}) + \epsilon\Big] \\ &\le 2|\mathcal{P}| \exp\left(-\frac{\epsilon^2 TN}{4w^2}\right) q^{|V|N}.\end{aligned}$$

by choosing $T = \Theta(w^2 \log q/\epsilon^2)$, we have the desired result. Note that $|V|$ and $|\mathcal{P}|$ can be seen as constants when N is sufficiently large. □

D.2 Proof of Lemma 5.2

Proof. Let $\alpha \in [q]^{V \times [N]}$ be the natural assignment to variables in $\mathcal{J}$. That is, $\alpha(v,i) = a$ when the variable (v,i) is assigned to the value a in the construction of $\mathcal{D}^{\mathbf{lp}}_{N,T}$. Then,

$$\mathbf{opt}(\mathcal{J}) \geq \mathbf{val}(\mathcal{J}, \alpha) = \sum_{P \in \mathcal{P}} \mathbf{val}(\mathcal{J}_P, \alpha_{|V(P)}) = TN \sum_{P \in \mathcal{P}} \boldsymbol{w}_P \mathop{\mathrm{E}}_{\beta_P \sim \boldsymbol{\mu}^*_P}[P(\beta_P)] = TN\mathbf{lp}(\mathcal{I}).$$

□

D.3 Proof of Lemma 5.3

For notational simplicity, we omit subscripts N and T in this section. We define some notions. At each step of an algorithm, a variable v is called *seen* if v is appeared in queries to the oracle or answers by the oracle so far. Also, an index i of a variable v is called *seen* if the i-th constraint of v is already returned by the oracle.

Here, we only show a lower bound for a (randomized) algorithm whose behavior is slightly restricted. That is, when an algorithms asks for a constraint incident to an unseen variable, we assume that the algorithm chooses the variable uniformly at random from the set of unseen variables. We can get rid of this assumption using the technique presented in Section 4 of [12]. Details are deferred to the full version of the paper. In what follows, we regard that the oracle accepts two types of queries. The first one is same as the original, i.e., when we specify a variable v and an index i, the oracle returns the i-th constraint of v. The second one simply returns a random variable from the set of unseen variables without receiving any argument. When an algorithm asks for a constraint incident to an unseen variable, it uses the second type of queries to get a variable first, and then it uses the first type of queries to get a constraint incident to the variable.

Now, we prove Lemma 5.3. Recall that, from Yao's minimax principle, it suffices to consider deterministic algorithms. We basically follow the approach presented in Section 7 of [11]. Let $\mathcal{A}$ be a deterministic algorithm. We introduce a randomized process $\mathcal{P}^{\mathbf{opt}}$ (resp., $\mathcal{P}^{\mathbf{lp}}$), which interacts with $\mathcal{A}$ so that $\mathcal{P}^{\mathbf{opt}}$ (resp., $\mathcal{P}^{\mathbf{lp}}$) answers queries of $\mathcal{A}$ to the oracle while constructing a random instance from $\mathcal{D}^{\mathbf{opt}}$ (resp., $\mathcal{D}^{\mathbf{lp}}$). The final distribution of instances generated by $\mathcal{P}^{\mathbf{opt}}$ (resp., $\mathcal{P}^{\mathbf{lp}}$) coincides with $\mathcal{D}^{\mathbf{opt}}$ (resp., $\mathcal{D}^{\mathbf{lp}}$) no matter how $\mathcal{A}$ makes queries. The interaction between $\mathcal{A}$ and $\mathcal{P}^{\mathbf{opt}}$ (resp.,$\mathcal{P}^{\mathbf{lp}}$) precisely simulates the interaction between $\mathcal{A}$ and $\mathcal{O}_{\mathcal{J}}$ where $\mathcal{J}$ is an instance generated by the distribution $\mathcal{D}^{\mathbf{opt}}$ (resp., $\mathcal{D}^{\mathbf{lp}}$). The process $\mathcal{P}^\star$, which corresponds to the distribution $\mathcal{D}^\star$, is simply a process that chooses $\mathcal{P}^{\mathbf{opt}}$ or $\mathcal{P}^{\mathbf{lp}}$ randomly and behave as the chosen process.

A *transcript* is the part of an instance that $\mathcal{A}$ has seen through the interaction with a randomized process. Note that, the transcript contains the information about labels of vertices and indices of constraints. Let $\mathcal{K}^{\mathbf{opt}}_\tau$ (resp., $\mathcal{K}^{\mathbf{lp}}_\tau$) be the distribution of transcripts after τ-step interaction between $\mathcal{A}$ and $\mathcal{P}^{\mathbf{opt}}$ (resp., $\mathcal{P}^{\mathbf{lp}}$) (Here, $\mathcal{K}$ stands for *knowledge*). The statistical distance between $\mathcal{K}^{\mathbf{opt}}_\tau$ and $\mathcal{K}^{\mathbf{lp}}_\tau$ is defined as follows.

$$d_{\mathrm{TV}}(\mathcal{K}^{\mathbf{opt}}_\tau, \mathcal{K}^{\mathbf{lp}}_\tau) = \frac{1}{2} \sum_K \left| \Pr_{K' \sim \mathcal{K}^{\mathbf{opt}}_\tau}[K' = K] - \Pr_{K' \sim \mathcal{K}^{\mathbf{lp}}_\tau}[K' = K] \right|$$

From the argument given in Section 7 of [11], by showing that $d_{\mathrm{TV}}(\mathcal{K}^{\mathbf{opt}}_\tau, \mathcal{K}^{\mathbf{lp}}_\tau) = o(1)$ when $\tau = o(\sqrt{N})$, we have the desired result.

We can safely assume that $\mathcal{A}$ never asks for the same constraint twice or more. Also, we assume that, if $\mathcal{P}^\star$ returns a constraint containing a variable in the transcript, $\mathcal{A}$ can correctly guess the process ($\mathcal{P}^{\mathbf{opt}}$ or $\mathcal{P}^{\mathbf{lp}}$) with which $\mathcal{A}$ is interacting. In other words, we are assuming that, when $\mathcal{P}^{\mathbf{opt}}$ (resp., $\mathcal{P}^{\mathbf{lp}}$) returns a constraint containing a variable in the transcript, it also returns a certificate stating that the current process is $\mathcal{P}^{\mathbf{opt}}$ (resp., $\mathcal{P}^{\mathbf{lp}}$). This only improves the ability of $\mathcal{A}$ and makes the lower bound smaller.

Now, we define the randomized process $\mathcal{P}^{\mathbf{opt}}$. We omit the definition of $\mathcal{P}^{\mathbf{lp}}$ as it is very similar to the construction of $\mathcal{P}^{\mathbf{opt}}$. The process $\mathcal{P}^{\mathbf{opt}}$ has two stages. The first stage proceeds as long as $\mathcal{A}$ perform queries. In this stage, $\mathcal{P}^{\mathbf{opt}}$ chooses an answer for each query. In the second stage, the process completes the transcript into an instance $\mathcal{J}$.

We identify $[n]$ (resp., $[nN]$) with the set of variables of $\mathcal{I}$ (resp., an instance generated by $\mathcal{P}^{\mathbf{opt}}$). Recall that, in an instance generated by $\mathcal{D}^{\mathbf{opt}}$, the variable set $[nN]$ can be separated into n sets, each of which corresponds to a variable $i \in [n]$. The process $\mathcal{P}^{\mathbf{opt}}$ incrementally constructs this correspondence. A (partial) correspondences is represented by a map $\rho\colon [nN] \to [n] \cup \{\perp\}$. For a variable $i \in [n]$, let $V_i = \{v \in [nN] \mid \rho(v) = i\}$ and $N_i = |V_i|$. Also, for each vertex $v \in [nN]$ and an index $i \in [d]$, let $D_i(v) = \{j \in \{(T-1)i+1, \ldots, Ti\} \mid j\text{-th constraint of } v \text{ is seen}\}$ and $d_i(v) = |D_i(v)|$.

In the first stage, given a query by an algorithm $\mathcal{A}$, $\mathcal{P}^{\mathbf{opt}}$ chooses an answer for it as follows.

- When the query asks for a random unseen variable: we choose a random unseen variable $v \in [nN]$, and set $\rho(v) = i$ with probability $\frac{N-N_i}{\sum_{j\in[n]}(N-N_j)}$. Then, we return v to $\mathcal{A}$.
- When the query asks for the p-th constraint of v: Note that $\rho(v) \neq \perp$ from the assumption that, when $\mathcal{A}$ asks for a constraint incident to an unseen variable, it asks for a random unseen variable beforehand. Let q be such that $(T-1)q+1 \le p \le Tq$, and P be the q-th constraint of $\rho(v)$ in $\mathcal{I}$, which is applied to a sequence of variables $\{i_1, \ldots, i_k\}$ in $\mathcal{I}$ for which $i_\ell = \rho(v)$ for some $\ell \in [k]$. Also, let q_j be such that P is the q_j-th constraint of the variable i_j in $\mathcal{I}$. Note that $q_\ell = q$.

  Then, we choose a set of variables $\{v_j\}_{j\in[k]\setminus\{\ell\}}$ as follows. For each variable u with $\rho(u) = i_j$, we choose u as v_j with probability $\frac{T-d_{q_j}(u)}{\sum_{w\in V_{i_j}}(T-d_{q_j}(w))+(N-N_{i_j})T}$. If otherwise, we choose a random unused variable u as v_j ans set $\rho(u) = i_j$.

  Let P' be a constraint applied to a sequence $\{v_1, \ldots, v_k\}$ of weight $\boldsymbol{w}_P$. Finally, we determine indices for each variable $v_j (j \neq \ell)$. We choose a random index p_j from unused indices in $\{(T-1)q_j+1, \ldots, Tq_j+1\}$, and set P' be as the p_j-th constraint of v_j. Then, we return P' as the answer for the query.

In the second stage of $\mathcal{P}^{\mathbf{opt}}$, the process uniformly selects an instance $\mathcal{J}$ among all those who are consistent with the final transcript.

Lemma D.2. *For every algorithm $\mathcal{A}$, the randomized process $\mathcal{P}^{\mathbf{opt}}$ (resp., $\mathcal{P}^{\mathbf{lp}}$) when interacting with $\mathcal{A}$, uniformly generates an instance $\mathcal{J}$ in $\mathcal{D}^{\mathbf{opt}}$ (resp., $\mathcal{D}^{\mathbf{lp}}$).* □

Proof. The lemma easily follows by induction on the query complexity of $\mathcal{A}$. The base case is clear since if no query is made, then the distribution on instances generated by $\mathcal{P}^{\mathbf{opt}}$ (or, $\mathcal{P}^{\mathbf{lp}}$) is clearly uniform. The induction step follows directly from the definition of the process. In particular, the distribution on instances resulting from the process switching to the second stage after it answers the query is exactly the same as the distribution resulting from the process performing the second stage without answering the query. □

Proof of Lemma 5.3. Let $\mathcal{A}$ be a deterministic algorithm. It is convenient to think that labels of variables are determined on the fly. That is, $\mathcal{P}^\star$ decides labels of variables from $[nN]$ at the time when the variable appears for the first time in the interaction between an algorithm and $\mathcal{P}$. The distribution never change by this modification. Also, we can think that the sequence of labels is determined beforehand, and for each time when a new variable appears, a new label for the variable is taken from the front of the sequence. Let $\mathcal{P}^\star_\ell$ be the process obtained from $\mathcal{P}^\star$ by fixing the sequence to ℓ. It is clear that $\mathcal{P}^\star$ coincides with the process that takes ℓ uniformly at random and acts as $\mathcal{P}^\star_\ell$. Let $\mathcal{P}^{\mathbf{opt}}_\ell$ (resp., $\mathcal{P}^{\mathbf{lp}}_\ell$) be the process obtained from $\mathcal{P}^{\mathbf{opt}}$ (resp., $\mathcal{P}^{\mathbf{lp}}$) by fixing the sequence to ℓ. Then, it suffices to bound the statistical distance between the distribution of transcripts when $\mathcal{A}$ interacts with $\mathcal{P}^{\mathbf{opt}}_\ell$ and the one when $\mathcal{A}$ interacts with $\mathcal{P}^{\mathbf{lp}}_\ell$ for any sequence ℓ.

A deterministic algorithm $\mathcal{A}$ with query complexity τ can be expressed as a decision tree of depth at most τ. Here, each node in the decision tree corresponds to a query to the oracle, and each branch from the node corresponds to the answer by the oracle. Recall that, from the rule of indices, if we fix an index, the process always returns the same predicate (though the set of vertices to which the predicate is applied should differ). Also, since we have fixed the sequence of labels ℓ, at each node in the decision tree, there is just one branch corresponding to the case that $\mathcal{A}$ finds a constraint such that any variable in the constraint (except the queried variable) is not in the transcript. Ignoring branches for which $\mathcal{A}$ outputs an answer, the decision tree has the property that the number of children of each node is at most one. Thus, $\mathcal{A}$ is essentially a non-adaptive algorithm. Without loss of generality, we assume that $\mathcal{A}$ outputs that the current instance is generated by $\mathcal{D}^{\mathbf{opt}}$ after τ steps.

Suppose that the current process is $\mathcal{P}^{\mathbf{opt}}_\ell$ and $\mathcal{A}$ is asking for a constraint incident to some variable in the i-th query. Note that $\mathcal{A}$ has seen at most is variables. Then, from the construction of $\mathcal{P}^{\mathbf{opt}}$, the probability that $\mathcal{P}^{\mathbf{opt}}_\ell$ returns a variable in the transcript is at most $\frac{isT}{(N-is)T} \cdot s = \frac{is^2}{N-is}$. Using the same argument, we can show that, in the i-th query, the probability that $\mathcal{P}^{\mathbf{lp}}_\ell$ returns a variable in the transcript is at most $\frac{is^2}{\mu N - is}$ where μ is the minimum of $\{\boldsymbol{\mu}_{P,\beta}\}_{P \in \mathcal{P}, \beta \in [q]^{V(P)}}$ except 0.

Thus, from the union bound, after τ steps, the probability that $\mathcal{P}^\star_\ell$ returns a variable in the transcript is at most

$$\sum_{i=1}^{\tau} \frac{is^2}{\mu N - is} \leq \frac{\tau^2 s^2}{\mu N - \tau s}.$$

Then, the probability that $\mathcal{A}$ outputs the correct answer is at most $\frac{\tau^2 s^2}{\mu N - \tau s} + \frac{1}{2}$. To make this probability at least 3/5, we have to choose $\tau = \Omega(\sqrt{N})$. Note that μ is a positive constant independent of N. □

E Proof of Theorem 1.4

Proof. We show the first part of the theorem. Let Λ be a CSP such that $S_\Lambda(1) = 1 - \gamma$ for some $\gamma > 0$. Suppose that there exists a testing algorithm for the CSP Λ with $o(\sqrt{n})$ queries. Note that a $\frac{\gamma}{3}$-far instance $\mathcal{I}$ satisfies that $\mathbf{opt}(\mathcal{I}) \leq \boldsymbol{w}_\mathcal{I} - \frac{\gamma t w n}{3} \leq (1 - \frac{\gamma}{3})\boldsymbol{w}_\mathcal{I}$. Thus, using the testing algorithm, given an instance $\mathcal{I}$, with probability at least 2/3, we can distinguish the case $\mathbf{opt}(\mathcal{I}) = \boldsymbol{w}_\mathcal{I}$ from the case $\mathbf{opt}(\mathcal{I}) \leq (1 - \frac{\gamma}{3})\boldsymbol{w}_\mathcal{I}$. However, instantiating Theorem 1.2 with $\epsilon = \gamma/3$, the theorem asserts that any algorithm that, given an instance $\mathcal{I}$, with probability at least 2/3, distinguishes the case $\mathbf{opt}(\mathcal{I}) = \boldsymbol{w}_\mathcal{I}$ from the case $\mathbf{opt}(\mathcal{I}) \leq (S_\Lambda(1) + \frac{\gamma}{3})\boldsymbol{w}_\mathcal{I} = (1 - \frac{2\gamma}{3})\boldsymbol{w}_\mathcal{I}$ requires $\Omega(\sqrt{n})$ queries. This is a contradiction.

We show the second part of the theorem. Let Λ be a CSP such that $S_\Lambda(1) = 1$. Since $S_\Lambda(c)$ is continuous at $c = 1$, for any $\epsilon > 0$, there exists δ such that $S_\Lambda(1 - \delta) > 1 - \epsilon/2$. Consider the algorithm obtained by instantiating Theorem 1.1 replacing ϵ with $\min(\epsilon/2, \delta)$. Suppose that $\mathcal{I}$ is a satisfiable instance. Then, we obtain a value $x \geq S_\Lambda(1-\delta)\boldsymbol{w}_\mathcal{I} - \epsilon n/2 > (1-\epsilon/2)\boldsymbol{w}_\mathcal{I} - \epsilon n/2$. Suppose that $\mathcal{I}$ is an instance ϵ-far from satisfiability. Then, we obtain a value $x \leq \mathbf{opt}(\mathcal{I}) \leq \boldsymbol{w}_\mathcal{I} - \epsilon twn \leq (1-\epsilon/2)\boldsymbol{w}_\mathcal{I} - \epsilon twn/2$. Thus, we can test the satisfiability of the CSP Λ in constant time. □

Acknowledgements

The author is grateful to Hiro Ito and Suguru Tamaki for valuable comments on an earlier draft of this paper.