Approximating Dense Max 2-CSPs

Pasin Manurangsi[*1] **and Dana Moshkovitz**[†2]

1 **Dropbox, Inc.**
San Francisco, CA 94107, USA
pasin@dropbox.com

2 **Massachusetts Institute of Technology**
Cambridge, MA 02139, USA
dmoshkov@mit.edu

Abstract

In this paper, we present a polynomial-time algorithm that approximates sufficiently high-value Max 2-CSPs on sufficiently dense graphs to within N^{ε} approximation ratio for *any* constant $\varepsilon > 0$. Using this algorithm, we also achieve similar results for free games, projection games on sufficiently dense random graphs, and the Densest k-Subgraph problem with sufficiently dense optimal solution. Note, however, that algorithms with similar guarantees to the last algorithm were in fact discovered prior to our work by Feige et al. and Suzuki and Tokuyama.

In addition, our idea for the above algorithms yields another by-product: a quasi-polynomial time approximation scheme (QPTAS) for satisfiable dense Max 2-CSPs with better running time than the known algorithms.

Keywords and phrases Max 2-CSP, Dense Graphs, Densest k-Subgraph, QPTAS, Free Games, Projection Games

1 Introduction

Maximum constraint satisfaction problem (Max CSP) is a problem of great interest in approximation algorithms since it encapsulates many natural optimization problems; for instance, Max k-SAT, Max-Cut, Max-DiCut, Max k-Lin, projection games, and unique games are all families of Max CSPs. In Max CSP, the input is a set of variables, an alphabet set, and a collection of constraints. Each constraint's domain consists of all the possible assignments to a subset of variables. The goal is to find an assignment to all the variables that satisfies as many constraints as possible.

In this paper, our main focus is on the case where each constraint depends on exactly $k = 2$ variables and the alphabet size is large. This case is intensively researched in hardness of approximation and multi-prover games. For Max 2-CSP with large alphabet size, the best known polynomial-time approximation algorithm, due to Charikar et al. [10], achieves an approximation ratio of $O((nq)^{1/3})$ where n is the number of variables and q is the alphabet size. On the other hand, it is known that, there is no polynomial-time $2^{\log^{1-\delta}(nq)}$-approximation algorithm for Max 2-CSP unless NP $\not\subseteq$ DTIME($n^{polylog(n)}$) [16]. Moreover, it is believed that, for some constant $c > 0$, no polynomial-time $O((nq)^c)$-approximation algorithm exists

* Part of this work was completed while the author was at Massachusetts Institute of Technology.
† This material is based upon work supported by the National Science Foundation under Grant Number 1218547.

for projection games, a family of Max 2-CSP we shall introduce later, unless P = NP [15]. This is also known as the Projection Games Conjecture (PGC). As a result, if the PGC holds, one must study special cases in order to go beyond polynomial approximation ratio for Max 2-CSP.

One such special case that has been particularly fruitful is dense Max 2-CSP where density is measured according to number of constraints, i.e., an instance is δ-dense if there are δn^2 constraints. Note that, for convenience, we always assume that there is at most one constraint on a pair of variables. In other words, we form a simple graph by letting vertices represent the variables and edges represent the constraints. This is the interpretation that we will use throughout the paper. According to this view, δn is the average degree of the graph.

In 1995, Arora, Karger and Karpinski [3] invented a polynomial-time approximation scheme (PTAS) for dense Max 2-CSP when the density δ and alphabet size q are constants. More specifically, for any constant $\varepsilon > 0$, the algorithm achieves an approximation ratio $1 + \varepsilon$ and runs in time $O(n^{1/\varepsilon^2})$. Unfortunately, the running time becomes quasi-polynomial time when q is not constant.

Another line of development of such PTASs centers around subsampling technique (e.g. [1, 2, 4]). In summary, these algorithms function by randomly sampling the variables according to some distribution and performing an exhaustive search on the induced instance. Since the sampled set of variables is not too large, the running time is not exponential. However, none of these algorithm achieves polynomial running time for large alphabets. In particular, all of them are stuck at quasi-polynomial running time.

Since none of these algorithms runs in polynomial time for large alphabet, a natural and intriguing question is how good a polynomial-time approximation algorithm can be for dense Max 2-CSPs. In this paper, we partially answer this question by providing a polynomial-time approximation algorithm for dense high-value Max 2-CSPs that achieves $(nq)^{\varepsilon}$ approximation ratio for any constant $\varepsilon > 0$. Moreover, our technique also helps us come up with a quasi-polynomial time approximation scheme for satisfiable Max 2-CSPs with running time asymptotically better than that those from [1, 2, 3, 4].

The central idea of our technique is a trade-off between two different approaches: greedy assignment algorithm and "choice reduction" algorithm. In summary, either a simple greedy algorithm produces an assignment that satisfies many constraints or, by assigning an assignment to just one variable, we can reduce the number of optimal assignment candidates of other variables significantly. The latter is what we call the choice reduction algorithm. By applying this argument repeatedly, either one of the greedy assignments gives a high value assignment, or we are left with only few candidates for each variables. In the latter case, we can then just pick a greedy assignment at the end.

We would like to note that the choice reduction idea is not our original idea as it has already been successfully used in a few other algorithms. In fact, the idea has even been explored before for Max 2-CSP by Charikar et al [10]. However, what sets our algorithm apart is that, whereas in [10] the choice reduction was used once, we observe that the choice reduction algorithm can be applied multiple times for dense instances.

Not only that our technique is useful for Max 2-CSP, we are able to obtain approximation algorithms for other problems in dense settings as well. The first such problem is free games, which can be defined simply as Max 2-CSP on balanced complete bipartite graphs. While

free games have been studied extensively in the context of parallel repetition [5, 17] and as basis for complexity and hardness results [1, 9], the algorithm aspect of it has not been researched as much. In fact, apart from the aforementioned algorithms for dense Max 2-CSP that also works for free games, we are aware of only two approximation algorithms, by Aaronson et al. [1] and by Brandao and Harrow [8], specifically developed for free games. Similar to the subsampling lemmas, these two algorithms are PTASs when q is constant but, when q is large, the running times become quasi-polynomial. Interestingly, our result for dense Max 2-CSP directly yields a polynomial-time algorithm that can approximate free games within $(nq)^{\varepsilon}$ factor for any constant $\varepsilon > 0$, which may be the first non-trivial approximation algorithm for free games with such running time.

Secondly, our idea is also applicable for projection games. The projection games problem (also known as Label Cover) is Max 2-CSP on a bipartite graph where, for each assignment to a left vertex of an edge, there is exactly one satisfiable assignment to the other endpoint of the edge. Label Cover is of great significance in the field of hardness of approximation since almost all NP-hardness of approximation results known today are reduced from the NP-hardness of approximation of projection games (e.g. [6, 12]).

The current best polynomial-time approximation algorithm for satisfiable projection games is the authors' with $O((nq)^{1/4})$ ratio [14]. Moreover, as mentioned earlier, if the PGC is true, then, in polynomial time, approximating Label Cover beyond some polynomial ratio is unlikely. In this paper, we exceed this bound on random balanced bipartite graphs with sufficiently high density by proving that, in polynomial time, one can approximate satisfiable projection games on such graphs to within $(nq)^{\varepsilon}$ factor for any constant $\varepsilon > 0$.

Finally, we show a similar result for Densest k-Subgraph, the problem of finding a size-k subgraph of a given graph that contains as many edges as possible. Finding best polynomial-time approximation algorithm for Densest k-Subgraph(DkS) is a big open question in the field of approximation algorithms. Currently, the best known algorithm for DkS achieves an approximation ratio of $O(n^{1/4+\varepsilon})$ for any constant $\varepsilon > 0$ [7]. On the other hand, however, we only know that there is no PTAS for DkS unless P=NP [13].

Even though Densest k-Subgraph on general graphs remains open, the problem is better understood in some dense settings. More specifically, Arora et al. [3] provided a PTAS for the problem when the given graph is dense and $k = \Omega(N)$ where N is the number of vertices of the given graph. Later, Feige et al. [11] and Suzuki and Tokuyama [18] showed that, if we only know that the optimal solution is sufficiently dense, we can still approximate the solution to within any polynomial ratio in polynomial time. Using our approximation algorithm for dense Max 2-CSP, we are able to construct a polynomial-time algorithm for Densest k-Subgraph with similar conditions and guarantees as that from the algorithms from [11] and [18].

The theorems we prove in this paper are stated in Section 3 after appropriate preliminaries in the next section.

2 Preliminaries and Notation

In this section, we formally define the problems we focus on and the notation we use throughout the paper. First, to avoid confusion, let us state the definition of approximation ratio

for the purpose of this paper.

▶ **Definition 1.** An approximation algorithm for a maximization problem is said to have an approximation ratio α if the output of the algorithm is at least $1/\alpha$ times the optimal solution.

Note here that the approximation ratio as defined above is always at least one.

Next, before we define our problems, we review the standard notation of density of a graph.

▶ **Definition 2.** A simple undirected graph $G = (V, E)$ is defined to be of density $|E|/|V|^2$.

Moreover, for a graph G and a vertex u, we use $\Gamma^G(u)$ to denote the set of neighbors of u in G. We also define $\Gamma_2^G(u)$ to denote the set of neighbors of neighbors of u in G, i.e., $\Gamma_2^G(u) = \Gamma^G(\Gamma^G(u))$. When it is unambiguous, we will leave out G and simply write $\Gamma(u)$ or $\Gamma_2(u)$.

Now, we will define the problems starting with Max 2-CSP.

▶ **Definition 3.** An instance $(q, V, E, \{C_e\}_{e\in E})$ of Max 2-CSP consists of

- a simple undirected graph (V, E), and
- for each edge $e = (u, v) \in E$, a constraint (or constraint) $C_e : [q]^2 \to \{0, 1\}$ where $[q]$ denotes $\{1, 2, \ldots, q\}$.

The goal is to find an assignment (solution) $\varphi : V \to [q]$ that maximizes the number of constraints C_e's that are satisfied, i.e. $C_{(u,v)}(\varphi(u), \varphi(v)) = 1$. In other words, find an assignment $\varphi : \{x_1, \ldots, x_n\} \to [q]$ that maximizes $\sum_{(u,v)\in E} C_{(u,v)}(\varphi(u), \varphi(v))$. The value of an assignment is defined as the fraction of edges satisfied by it and the value of an instance is defined as the value of the optimal assignment.

A Max 2-CSP instance $(q, V, E, \{C_e\}_{e\in E})$ is called δ-dense if the graph (V, E) is δ-dense. Throughout the paper, we use n to denote the number of vertices (variables) $|V|$ and N to denote nq, which can be viewed as the size of the problem.

Free games and projection games are specific classes of Max 2-CSP, which can be defined as follows. Note that n, N, density and value are defined in a similar fashion for free games and projection games as well.

▶ **Definition 4.** A free game $(q, A, B, \{C_{a,b}\}_{(a,b)\in A\times B})$ consists of

- Two sets A, B of equal size, and
- for $(a, b) \in A \times B$, a constraint $C_{a,b} : [q]^2 \to \{0, 1\}$.

The goal is to find an assignment $\varphi : A \cup B \to [q]$ that maximizes the number of edges $(a, b) \in A \times B$ that are satisfied, i.e., $C_{a,b}(\varphi(a), \varphi(b)) = 1$.

▶ **Definition 5.** A projection game $(q, A, B, E, \{\pi_e\}_{e\in E})$ consists of

- a simple bipartite graph (A, B, E), and
- for each edge $e = (a, b) \in E$, a "projection" $\pi_e : [q] \to [q]$.

The goal is to find an assignment to the vertices $\varphi : A \cup B \to [q]$ that maximizes the number of edges $e = (a, b)$ that are satisfied, i.e., $\pi_e(\varphi_A(a)) = \varphi_B(b)$.

Both free games and projection games can be viewed as special cases of Max 2-CSP. More specifically, free games are simply Max 2-CSPs on complete balanced bipartite graphs.

For projection games, one can view π_e as a constraint $C_e : [q]^2 \to \{0, 1\}$ where $C_e(\sigma_u, \sigma_v) = 1$ if and only if $\pi_e(\sigma_u) = \sigma_v$. In other words, projection game is Max 2-CSP on bipartite graph where an assignment to the endpoint in A of an edge determines the assignment to the endpoint in B.

For convenience, we will define the notation of "optimal assignment" for Max 2-CSP intuitively as follows.

▶ **Definition 6.** For a Max 2-CSP instance $(q, V, E, \{C_e\}_{e\in E})$, for each vertex $u \in V$, let σ_u^{OPT} be the assignment to u in an assignment to vertices that satisfies maximum number of edges, i.e., $\varphi(u) = \sigma_u^{OPT}$ is the assignment that maximizes $\sum_{(u,v)\in E} C_{(u,v)}(\varphi(u), \varphi(v))$. In short, we will sometimes refer to this as "the optimal assignment".

Note that since projection games and free games are families of Max 2-CSP, the above definition also carries over when we discuss them.

Lastly, we define Densest k-Subgraph.

▶ **Definition 7.** In the Densest k-Subgraph problem, the input is a simple graph $G = (V, E)$ of $N = |V|$ vertices. The goal is to find a subgraph of size k that contain maximum number of edges.

3 Summary of Results

We are finally ready to describe our results and how they relate to the previous results. We will start with the main theorem on approximating high-value dense Max 2-CSP.

▶ **Theorem 8** (Main Theorem). *For every constant $\gamma > 0$, there exists a polynomial-time algorithm that, given a δ-dense* Max 2-CSP *instance of value λ, produces an assignment of value $\Omega((\delta\lambda)^{O(1/\gamma)}N^{-\gamma})$ for the instance.*

Note that, when $\delta, \lambda = N^{-o(1)}$, the algorithm can achieve $(nq)^{\varepsilon}$ approximation ratio for any constant $\varepsilon > 0$.

Since every free game is 1/2-dense, Theorem 8 immediately implies the following corollary.

▶ **Corollary 9.** *For every constant $\gamma > 0$, there exists a polynomial-time algorithm that, given a free game of value λ, produces an assignment of value $\Omega(\lambda^{O(1/\gamma)}N^{-\gamma})$ for the instance.*

Again, note that when $\lambda = N^{-o(1)}$, the algorithm can achieve $(nq)^{\varepsilon}$ approximation ratio for any constant $\varepsilon > 0$.

The next result is a similar algorithm for projection games on sufficiently dense random graphs as stated below.

▶ **Theorem 10.** *For every constant $\gamma > 0$, there exists a polynomial-time algorithm that, given a satisfiable projection game on a random bipartite graph $(A, B, E) \sim \mathcal{G}(n/2, n/2, p)$ for any $p \geq 10\sqrt{\log n/n}$, produces an assignment of value $\Omega(N^{-\gamma})$ for the instance with probability $1 - o(1)$.*

Note that $\mathcal{G}(n/2, n/2, p)$ is defined in Erdős-Rényi fashion, i.e., the graph contains $n/2$ vertices on each side and, each pair of left and right vertices is included as an edge with probability p independently.

In addition, it is worth noting here that the required density for projection games is much lower than that of Max 2-CSP; our Max 2-CSP algorithm requires the degree to be $\Omega(n/N^{-o(1)})$ whereas the projection games algorithm requires only $\tilde{\Omega}(\sqrt{n})$.

As stated earlier, we are unaware of any non-trivial polynomial-time algorithm for dense Max 2-CSP, free games, or projection games on dense random graphs prior to our algorithm.

Next, we state our analogous result for Densest k-Subgraph.

▶ **Corollary 11.** *For every constant $\gamma > 0$, there exists a polynomial-time algorithm that, given a graph $G = (V, E)$ on N vertices such that its densest subgraph with k vertices is δ-dense, produces a subgraph of k vertices that is $\Omega(\delta^{O(1/\gamma)}N^{-\gamma})$-dense with high probability.*

Note that the density condition is on the optimal solution, not the given graph G. The condition and the algorithm are exactly the same as that of [11] and [18]. However, the techniques are substantially different. While [11] deals combinatorially directly with the given graph G and [18] employs subsampling technique, we simply use our algorithm from Theorem 8 together with a simple reduction from Densest k-Subgraph to Max 2-CSP due to Charikar et al. [10].

Lastly, we also give a quasi-polynomial time approximation scheme for satisfiable dense Max 2-CSP as described formally below.

▶ **Corollary 12** (QPTAS for Dense Max 2-CSP)**.** *For any $1 \geq \varepsilon > 0$, there exists an $(1 + \varepsilon)$-approximation algorithm for satisfiable δ-dense* Max 2-CSP *that runs in time $N^{O(\varepsilon^{-1}\delta^{-1}\log N)}$.*

Comparing to the known algorithms, our QPTAS runs faster than QPTASs from [2, 3, 4], each of which takes at least $N^{O(\varepsilon^{-2}\delta^{-1}\log N)}$ time. However, while our algorithm works only for satisfiable instances, the mentioned algorithms work for unsatisfiable instances as well but with an additive error of ε in value instead of the usual multiplicative guarantee of $(1 + \varepsilon)$.

4 Proof of The Main Theorem

In this section, we prove the main theorem. In order to do so, we will first show that we do not have to worry about the density δ at all, i.e., it is enough for us to prove the following lemma.

▶ **Lemma 13.** *For every $\gamma > 0$, there exists a polynomial-time algorithm that, given a free game $(q, A, B, \{C_{(a,b)}\}_{(a,b)\in A\times B})$ of value λ', produces an assignment of value $\lambda'^{O(1/\gamma)}q^{-\gamma}$*

for the instance.

The proof of the main theorem based on the lemma above is shown below.

Proof of Theorem 8 based on Lemma 13. The proof is based on putting in "dummy edges" where the constraints are always false regardless of the assignment to make the game more dense. More specifically, given a Max 2-CSP instance $(q, V, E, \{C_e\}_{e\in E})$ of value λ and density δ, we construct a free game $(q', A, B, \{C'_{(a,b)}\}_{(a,b)\in A\times B})$ as follows:

- Let A, B be copies of V and let $q' = q$.
- For each $a \in A$ and $b \in B$, let $C'_{(a,b)} = C_{(a,b)}$ if $(a, b) \in E$. Otherwise, let $C'_{(a,b)} := 0$.

Observe that, if we assign the optimal assignment of the original instance to the free game, we can see that the value of the free game is at least $\delta\lambda$. Thus, from Lemma 13, for any constant γ, we can find an assignment $\varphi : A \cup B \to [q']$ of value at least $(\delta\lambda)^{O(1/\gamma)}q^{-\gamma}$ for the free game.

From φ, we create an assignment $\varphi' : V \to [q]$ as follows: enumerate through the vertices $v \in V$, pick either the assignment to the copy of v in A or the copy of v in B that satisfies more edges considering only the assignment so far. It is not hard to see that this assignments satisfies at least half as many edges with respect to the original instance as that of φ with respect to the free game instance. As a result, we can conclude that φ' we got has value at least $\Omega((\delta\lambda)^{O(1/\gamma)}N^{-\gamma})$ regarding the instance $(q, V, E, \{C_e\}_{e\in E})$, which completes our proof for the main theorem. ◀

Now, we finally give the proof for Lemma 13. As mentioned in the introduction, the main idea of the proof is a trade-off between the greedy algorithm and the choice reduction algorithm. In other words, either the greedy assignment has high value, or we can reduce the number of candidates of the optimal assignment for many variables significantly by assigning only one variable. This argument needs to be applied multiple times to arrive at the result; the more variables we iterate on, the better guarantee we get on the output assignment value.

For the purpose of analysis, we will define our algorithm recursively and use induction to show that the output assignment meets the desired criteria.

Proof of Lemma 13. First, let us define notation that we will use throughout the proof. For a free game $(q, A, B, \{C_{(a,b)}\}_{(a,b)\in A\times B})$, define E^{OPT} to be the set of edges satisfied by $\{\sigma_u^{OPT}\}_{u\in V}$. In other words, $E^{OPT} = \{(u, v) \in E \mid C_{(u,v)}(\sigma_u^{OPT}, \sigma_v^{OPT}) = 1\}$. We also define $\Gamma^{OPT}(u)$ to be the neighborhood of u with respect to (V, E^{OPT}) and let d_u^{OPT} be the degree of u in (V, E^{OPT}), i.e., $d_u^{OPT} = |\Gamma^{OPT}(u)|$. In addition, let $n' = n/2$ be the size of A and B.

We will prove the lemma by induction. Let $P(i)$ represent the following statement: there exists an $O\left((nq)^{2i}\right)$-time algorithm Approx-FreeGame$_i(q, A, B, \{C_{(a,b)}\}_{(a,b)\in A\times B}, \{S_b\}_{b\in B})$ that takes in a free game instance $(q, A, B, \{C_{(a,b)}\}_{(a,b)\in A\times B})$ of value λ' and a reduced alphabets set S_b for every $b \in B$, and produces an assignment that satisfies at least

$$n'\left(\sum_{b\in B}\left(\frac{d_b^{OPT}}{n'}\right)^{\frac{i+1}{2}}\left(\frac{1}{|S_b|}\right)^{\frac{1}{i}}1_{\sigma_b^{OPT}\in S_b}\right)$$

edges. Note here that $1_{\sigma_b^{OPT} \in S_b}$ denotes an indicator variable for whether $\sigma_b^{OPT} \in S_b$. Moreover, for convenience, we use the expression $(1/|S_b|)^{\frac{1}{i}} 1_{\sigma_b^{OPT} \in S_b}$ to be represent zero when $S_b = \emptyset$.

Before we proceed to the induction, let us note why $P(i)$ implies the lemma. By setting $i = \lceil 1/\gamma \rceil$ and $S_b = [q]$ for every $b \in B$, since $\sigma_b^{OPT} \in S_b$ for every $b \in B$, the number of edges satisfied by the output assignment of the algorithm in $P(i)$ is at least

$$\begin{aligned}
n' \sum_{b \in B} \left(\frac{d_b^{OPT}}{n'} \right)^{\frac{i+1}{2}} \left(\frac{1}{q} \right)^{\frac{1}{i}} &= n' \frac{1}{q^{1/i}} \left(\sum_{b \in B} \left(\frac{d_b^{OPT}}{n'} \right)^{\frac{i+1}{2}} \right) \\
\text{(From Hölder's inequality)} &\geq \frac{(n')^2}{q^{1/i}} \left(\frac{1}{n'} \sum_{b \in B} \frac{d_b^{OPT}}{n'} \right)^{\frac{i+1}{2}} \\
&= \frac{(n')^2}{q^{1/i}} \left(\frac{|E^{OPT}|}{(n')^2} \right)^{\frac{i+1}{2}} \\
\text{(Since } |E^{OPT}|/(n')^2 \text{ is the value of the instance)} &= \frac{(n')^2}{q^{1/i}} (\lambda')^{\frac{i+1}{2}} \\
\text{(From our choice of } i) &\geq (n')^2 \frac{\lambda'^{O(1/\gamma)}}{q^{-\gamma}},
\end{aligned}$$

which is the statement of the lemma.

Now, we finally show that $P(i)$ is true for every $i \in \mathbb{N}$ by induction.

Base Case. The algorithm Approx-FreeGame$_1(q, A, B, \{C_{(a,b)}\}_{(a,b) \in A \times B}, \{S_b\}_{b \in B})$ is a greedy algorithm that works as follows:

1. For each $a \in A$, assign it to $\sigma_a^* \in S_a$ that maximizes $\sum_{b \in B} \frac{1}{|S_b|} \left(\sum_{\sigma_b \in S_b} C_{(a,b)}(\sigma_a, \sigma_b) \right)$.
2. For each $b \in B$, assign it to $\sigma_b^* \in S_b$ that maximizes the number of edges satisfied, i.e., maximizes $\sum_{a \in A} C_{(a,b)}(\sigma_a^*, \sigma_b)$.

It is obvious that the algorithm runs in $O(n^2 q^2)$ time as desired.

Next, we need to show that the algorithm gives an assignment that satisfies at least

$$n' \left(\sum_{b \in B} \left(\frac{d_b^{OPT}}{n'} \right) \left(\frac{1}{|S_b|} \right) 1_{\sigma_b^{OPT} \in S_b} \right) = \sum_{b \in B} \frac{d_b^{OPT}}{|S_b|} 1_{\sigma_b^{OPT} \in S_b}$$

edges.

To prove this, observe that, from our choice of σ_b^*, the number of satisfied edges by the output assignment can be bounded as follows.

$$\begin{aligned}
\sum_{b \in B} \sum_{a \in A} C_{(a,b)}(\sigma_a^*, \sigma_b^*) &\geq \sum_{b \in B} \frac{1}{|S_b|} \sum_{\sigma_b \in S_b} \left(\sum_{a \in A} C_{(a,b)}(\sigma_a^*, \sigma_b) \right) \\
&= \sum_{a \in A} \sum_{b \in B} \frac{1}{|S_b|} \left(\sum_{\sigma_b \in S_b} C_{(a,b)}(\sigma_a^*, \sigma_b) \right) \\
\text{(From our choice of } \sigma_a^*) &\geq \sum_{a \in A} \sum_{b \in B} \frac{1}{|S_b|} \left(\sum_{\sigma_b \in S_b} C_{(a,b)}(\sigma_a^{OPT}, \sigma_b) \right)
\end{aligned}$$

$$
\begin{aligned}
&\geq \sum_{a\in A}\sum_{b\in B}\frac{1}{|S_b|}C_{(a,b)}(\sigma_a^{OPT},\sigma_b^{OPT})1_{\sigma_b^{OPT}\in S_b}\\
&= \sum_{b\in B}\sum_{a\in A}\frac{1}{|S_b|}C_{(a,b)}(\sigma_a^{OPT},\sigma_b^{OPT})1_{\sigma_b^{OPT}\in S_b}\\
\text{(From definition of } d_b) &= \sum_{b\in B}\frac{1}{|S_b|}d_b 1_{\sigma_b^{OPT}\in S_b}\\
&= \sum_{b\in B}\frac{d_b}{|S_b|}1_{\sigma_b^{OPT}\in S_b}.
\end{aligned}
$$

Thus, we can conclude that $P(1)$ is true.

Inductive Step. Let j be any positive integer. Suppose that $P(j)$ holds.

We will now describe Approx-FreeGame$_{j+1}$ based on Approx-FreeGame$_j$ as follows.

1. For each $a \in A$ and $\sigma_a \in S_a$, do the following:
   a. For each $b \in B$, compute $S_b^{a,\sigma_a} = \{\sigma_b \in S_b \mid C_{(a,b)}(\sigma_a,\sigma_b) = 1\}$.
   b. Call Approx-FreeGame$_j(q, A, B, \{C_{(a,b)}\}_{(a,b)\in A\times B}, \{S_b^{a,\sigma_a}\}_{b\in B})$. Let the output assignment be φ^{a,σ_a}.
2. Execute the following greedy algorithm:
   a. For each $a \in A$, assign it to $\sigma_a^* \in S_a$ that maximizes $\sum_{b\in B}\frac{1}{|S_b|}\left(\sum_{\sigma_b\in S_b}C_{(a,b)}(\sigma_a,\sigma_b)\right)$.
   b. For each $b \in B$, assign it to $\sigma_b^* \in S_b$ that maximizes the number of edges satisfied, i.e., maximizes $\sum_{a\in A}C_{(a,b)}(\sigma_a^*,\sigma_b)$.
3. Output an assignment among the greedy assignment and φ^{a,σ_a} for every a, σ_a that satisfies maximum number of edges.

Since every step except the Approx-FreeGame$_j(q, A, B, \{C_{(a,b)}\}_{(a,b)\in A\times B}, \{S_b\}_{b\in B})$ calls takes $O((nq)^2)$ time and we call Approx-FreeGame$_j$ only at most $(nq)^2$ times, we can conclude that the running time of Approx-FreeGame$_{j+1}$ is $O((nq)^{2j+2})$ as desired.

Define R to be $n'\left(\sum_{b\in B}\left(\frac{d_b^{OPT}}{n'}\right)^{\frac{j+2}{2}}\left(\frac{1}{|S_b|}\right)^{\frac{1}{j+1}}1_{\sigma_b^{OPT}\in S_b}\right)$, our target number of edges we want to satisfy. The only thing left to show is that the assignment output from the algorithm indeed satisfies at least R edges. We will consider two cases.

First, if there exists $a \in A$ and $\sigma_a \in S_b$ such that the output assignment from Approx-FreeGame$_j(q, A, B, \{C_{(a,b)}\}_{(a,b)\in A\times B}, \{S_b^{a,\sigma_a}\}_{b\in B})$ satisfies at least R edges, then it is obvious that the output assignment of Approx-FreeGame$_{j+1}$ indeed satisfies at least R edges as well.

In the second case, for every $a \in A$ and $\sigma_a \in S_a$, the output assignment from Approx-FreeGame$_j(q, A, B, \{C_{(a,b)}\}_{(a,b)\in A\times B}, \{S_b^{a,\sigma_a}\}_{b\in B})$ satisfies less than R edges. For each $a \in A$, since the output assignment from Approx-FreeGame$_j(q, A, B, \{C_{(a,b)}\}_{(a,b)\in A\times B}, \{S_b^{a,\sigma_a^{OPT}}\}_{b\in B})$ satisfies less than R edges, we arrive at the following inequality:

$$
R > n'\left(\sum_{b\in B}\left(\frac{d_b^{OPT}}{n'}\right)^{\frac{j+1}{2}}\left(\frac{1}{|S_b^{a,\sigma_a^{OPT}}|}\right)^{\frac{1}{j}}1_{\sigma_b^{OPT}\in S_b^{a,\sigma_a^{OPT}}}\right)
$$

$$\geq n' \left(\sum_{b \in \Gamma^{OPT}(a)} \left(\frac{d_b^{OPT}}{n'} \right)^{\frac{j+1}{2}} \left(\frac{1}{|S_b^{a,\sigma_a^{OPT}}|} \right)^{\frac{1}{j}} 1_{\sigma_b^{OPT} \in S_b^{a,\sigma_a^{OPT}}} \right).$$

Now, observe that, for every $b \in \Gamma^{OPT}(a)$, we have $1_{\sigma_b^{OPT} \in S_b^{a,\sigma_a^{OPT}}} = 1_{\sigma_b^{OPT} \in S_b}$. This is because, from our definition of Γ^{OPT}, $C_{(a,b)}(\sigma_a^{OPT}, \sigma_b^{OPT}) = 1$ for every $b \in \Gamma^{OPT}(a)$, which means that, if σ_b^{OPT} is in S_b, then it remains in $S_b^{a,\sigma_a^{OPT}}$. Thus, the above inequality can be written as follows:

$$R > n' \left(\sum_{b \in \Gamma^{OPT}(a)} \left(\frac{d_b^{OPT}}{n'} \right)^{\frac{j+1}{2}} \left(\frac{1}{|S_b^{a,\sigma_a^{OPT}}|} \right)^{\frac{1}{j}} 1_{\sigma_b^{OPT} \in S_b} \right). \tag{1}$$

We will use inequality 1 later in the proof. For now, we will turn our attention to the number of edges satisfied by the greedy algorithm, which, from our choice of σ_b^*, can be bounded as follows:

$$\begin{aligned}
\sum_{b \in B} \sum_{a \in A} C_{(a,b)}(\sigma_a^*, \sigma_b^*) &\geq \sum_{b \in B} \frac{1}{|S_b|} \sum_{\sigma_b \in S_b} \left(\sum_{a \in A} C_{(a,b)}(\sigma_a^*, \sigma_b) \right) \\
&= \sum_{a \in A} \sum_{b \in B} \frac{1}{|S_b|} \left(\sum_{\sigma_b \in S_b} C_{(a,b)}(\sigma_a^*, \sigma_b) \right) \\
(\text{From our choice of } \sigma_a^*) &\geq \sum_{a \in A} \sum_{b \in B} \frac{1}{|S_b|} \left(\sum_{\sigma_b \in S_b} C_{(a,b)}(\sigma_a^{OPT}, \sigma_b) \right) \\
(\text{Since } C_{(a,b)}(\sigma_a^{OPT}, \sigma_b) = 1 \text{ for every } \sigma_b \in S_b^{a,\sigma_a^{OPT}}) &\geq \sum_{a \in A} \sum_{b \in B} \frac{1}{|S_b|} |S_b^{a,\sigma_a^{OPT}}| \\
&= \sum_{a \in A} \sum_{b \in B} \frac{|S_b^{a,\sigma_a^{OPT}}|}{|S_b|} \\
&\geq \sum_{a \in A} \sum_{b \in \Gamma^{OPT}(a)} \frac{|S_b^{a,\sigma_a^{OPT}}|}{|S_b|}.
\end{aligned}$$

Moreover, from inequality 1, we can derive the following inequalities:

$$\begin{aligned}
& R^j \left(\sum_{a \in A} \sum_{b \in \Gamma^{OPT}(a)} \frac{|S_b^{a,\sigma_a^{OPT}}|}{|S_b|} \right) \\
&= \sum_{a \in A} R^j \left(\sum_{b \in \Gamma^{OPT}(a)} \frac{|S_b^{a,\sigma_a^{OPT}}|}{|S_b|} \right) \\
(\text{Inequality 1}) &\geq (n')^j \sum_{a \in A} \left(\sum_{b \in \Gamma^{OPT}(a)} \left(\frac{d_b^{OPT}}{n'} \right)^{\frac{j+1}{2}} \left(\frac{1}{|S_b^{a,\sigma_a^{OPT}}|} \right)^{\frac{1}{j}} 1_{\sigma_b^{OPT} \in S_b} \right)^j \left(\sum_{b \in \Gamma^{OPT}(a)} \frac{|S_b^{a,\sigma_a^{OPT}}|}{|S_b|} \right) \\
(\text{Hölder's inequality}) &\geq (n')^j \sum_{a \in A} \left(\sum_{b \in \Gamma^{OPT}(a)} \left(\frac{d_b^{OPT}}{n'} \right)^{\frac{j}{2}} \left(\frac{1}{|S_b|} \right)^{\frac{1}{j+1}} 1_{\sigma_b^{OPT} \in S_b} \right)^{j+1}
\end{aligned}$$

By applying Hölder's inequality once again, the last term above is at least

$$
\begin{aligned}
&(n')^j n' \left(\frac{1}{n'} \sum_{a \in A} \sum_{b \in \Gamma^{OPT}(a)} \left(\frac{d_b^{OPT}}{n'} \right)^{\frac{j}{2}} \left(\frac{1}{|S_b|} \right)^{\frac{1}{j+1}} 1_{\sigma_b^{OPT} \in S_b} \right)^{j+1} \\
&= \left(\sum_{b \in B} \sum_{a \in \Gamma^{OPT}(b)} \left(\frac{d_b^{OPT}}{n'} \right)^{\frac{j}{2}} \left(\frac{1}{|S_b|} \right)^{\frac{1}{j+1}} 1_{\sigma_b^{OPT} \in S_b} \right)^{j+1} \\
(\text{Since } d_b^{OPT} = |\Gamma^{OPT}(b)|) &= \left(\sum_{b \in B} d_b^{OPT} \left(\frac{d_b^{OPT}}{n'} \right)^{\frac{j}{2}} \left(\frac{1}{|S_b|} \right)^{\frac{1}{j+1}} 1_{\sigma_b^{OPT} \in S_b} \right)^{j+1} \\
&= \left(n' \sum_{b \in B} \left(\frac{d_b^{OPT}}{n'} \right)^{\frac{j+2}{2}} \left(\frac{1}{|S_b|} \right)^{\frac{1}{j+1}} 1_{\sigma_b^{OPT} \in S_b} \right)^{j+1} \\
&= R^{j+1}.
\end{aligned}
$$

Hence, we can conclude that

$$
\sum_{a \in A} \sum_{b \in \Gamma^{OPT}(a)} \frac{|S_b^{a, \sigma_a^{OPT}}|}{|S_b|} \geq R.
$$

In other words, our greedy algorithm satisfies at least R edges, which means that $P(j+1)$ is also true for this second case.

As a result, $P(i)$ is true for every positive integer i, which completes the proof for Lemma 13. ◀

5 Approximation Algorithm for Projection Games

In this section, we will present our approximation algorithm for projection games. The main idea of this algorithm is a reduction from projection games on dense random graphs to free games, which we use together with the approximation algorithm for free games from Corollary 9 above to prove Theorem 10. The reduction's properties can be stated formally as follows.

▶ **Lemma 14.** *There is a polynomial-time reduction from a satisfiable projection game $(q, A, B, E, \{\pi_e\}_{e \in E})$ where (A, B, E) is sampled from a distribution $\mathcal{G}(n/2, n/2, p)$ where $p \geq 10\sqrt{\log n / n}$ to a satisfiable free game instance $(q', A', B', \{C_{(a,b)}\}_{(a,b) \in A' \times B'})$ such that, with probability $1 - o(1)$,*

1. *$|A'|, |B'| \leq |A|$ and $q' \leq q$, and*
2. *For any $1 \geq \varepsilon \geq 0$, given an assignment $\varphi' : A' \cup B' \to [q]$ to the free game instance of value ε, one can construct an assignment $\varphi : A \cup B \to [q]$ for the projection game of value $\Omega(\varepsilon)$ in polynomial time.*

Before we describe the reduction, we give a straightforward proof for Theorem 10 based on the above lemma.

Proof of Theorem 10 based on Lemma 14. The proof is simple. First, we use the reduction from Lemma 14 to transform a projection game on dense graph to a free game. Since the approximation ratio deteriorates by only constant factor with probability $1 - o(1)$ in the reduction, we can use the approximation algorithm from Corollary 9 with $\lambda = 1$, which gives us an assignment of value at least $\Omega(1/N^{\gamma})$. ◀

To prove the reduction lemma, we use the following two properties of random graphs. We do not prove the lemmas as they follow from a standard Chernoff bound.

▶ **Lemma 15.** *When $p \geq 10\sqrt{\log n/n}$, with probability $1 - o(1)$, every vertex in $G \sim \mathcal{G}(n/2, n/2, p)$ has degree between $np/10$ and $10np$.*

▶ **Lemma 16.** *In $G \sim \mathcal{G}(n/2, n/2, p)$ with $p \geq 10\sqrt{\log n/n}$, with probability $1 - o(1)$, every pair of vertices a, a' on the left has at least $np^2/10$ common neighbors.*

Now, we are ready to prove the reduction lemma. Roughly speaking, the idea of the proof is to "square" the projection game, i.e., use A as the vertices of the new game and, for each pair of vertices in A, add a constriant between them based on their constraints with their common neighbors in the projection game. This can be formalized as follows.

Proof of Lemma 14. The reduction proceeds as follows.

1. Partition A into A_1, A_2 of equal sizes. Then, set $A' \leftarrow A_1, B' \leftarrow A_2$ and $q' \leftarrow q$.
2. For each $a_1 \in A_1, a_2 \in A_2, \sigma_{a_1}, \sigma_{a_2} \in [q]$, let $C_{(a_1,a_2)}(\sigma_{a_1}, \sigma_{a_2})$ to be one if and only if these two assignments agree on every $b \in \Gamma(a_1) \cap \Gamma(a_2)$. In other words, $C_{(a_1,a_2)}(\sigma_{a_1}, \sigma_{a_2}) = 1$ if and only if $\pi_{(a_1,b)}(\sigma_{a_1}) = \pi_{(a_2,b)}(\sigma_{a_2})$ for every $b \in \Gamma(a_1) \cap \Gamma(a_2)$.

It is obvious that the reduction runs in polynomial time, the first condition holds, and the new game is satisfiable. Thus, we only need to prove that, with probability $1 - o(1)$, the second condition is indeed true.

To show this, we present a simple algorithm that, given an assignment $\varphi' : A' \cup B' \to [q]$ of the free game instance of value ε, output an assignment $\varphi : A \cup B \to [q]$ of the projection game of value $\Omega(\varepsilon)$. The algorithm works greedily as follows.

1. For each $a \in A$, let $\varphi(a) \leftarrow \varphi(a)$.
2. For each $b \in B$, pick $\varphi(b) = \sigma_b^*$ to be the assignment to b that satisfies maximum number of edges, i.e., maximize $|\{a \in \Gamma(b) \mid \pi_{(a,b)}(\varphi_A(a)) = \sigma_b\}|$.

Trivially, the algorithm runs in polynomial time. Thus, we only need to prove that, with probability $1 - o(1)$, the produced assignment is of value at least $\Omega(\varepsilon)$. To prove this, we will use the properties from Lemma 15 and Lemma 16, which holds with probability $1 - o(1)$.

The number of satisfied edges can be rearranged as follows.

$$\sum_{b \in B} \sum_{a \in \Gamma(b)} 1_{\pi_{(a,b)}(\varphi(a)) = \varphi(b)}$$
$$= \sum_{b \in B} \sum_{a \in \Gamma(b)} 1_{\pi_{(a,b)}(\varphi'(a)) = \sigma_b^*}$$

$$
\begin{aligned}
&= \sum_{b\in B}\left[\frac{1}{d_b}\left(\sum_{a\in\Gamma(b)} 1_{\pi_{(a,b)}(\varphi'(a))=\sigma_b^*}\right) d_b\right] \\
&= \sum_{b\in B}\left[\frac{1}{d_b}\left(\sum_{a\in\Gamma(b)} 1_{\pi_{(a,b)}(\varphi'(a))=\sigma_b^*}\right)\left(\sum_{\sigma_b\in[q]}\sum_{a\in\Gamma(b)} 1_{\pi_{(a,b)}(\varphi'(a))=\sigma_b}\right)\right] \\
&= \sum_{b\in B}\left[\frac{1}{d_b}\sum_{\sigma_b\in[q]}\left(\sum_{a\in\Gamma(b)} 1_{\pi_{(a,b)}(\varphi'(a))=\sigma_b^*}\right)\left(\sum_{a\in\Gamma(b)} 1_{\pi_{(a,b)}(\varphi'(a))=\sigma_b}\right)\right] \\
(\text{From the choice of } \sigma_b^*) &\geq \sum_{b\in B}\left[\frac{1}{d_b}\sum_{\sigma_b\in[q]}\left(\sum_{a\in\Gamma(b)} 1_{\pi_{(a,b)}(\varphi'(a))=\sigma_b}\right)^2\right] \\
&= \sum_{b\in B}\left(\frac{1}{d_b}\sum_{\sigma_b\in[q]}\sum_{a,a'\in\Gamma(b)} 1_{\pi_{(a,b)}(\varphi'(a))=\sigma_b} 1_{\pi_{(a',b)}(\varphi'(a'))=\sigma_b}\right) \\
&= \sum_{b\in B}\left(\frac{1}{d_b}\sum_{a,a'\in\Gamma(b)}\sum_{\sigma_b\in[q]} 1_{\pi_{(a,b)}(\varphi'(a))=\sigma_b} 1_{\pi_{(a',b)}(\varphi'(a'))=\sigma_b}\right)
\end{aligned}
$$

Observe that $\sum_{\sigma_b\in[q]} 1_{\pi_{(a,b)}(\varphi'(a))=\sigma_b} 1_{\pi_{(a',b)}(\varphi'(a'))=\sigma_b} = 1_{\pi_{(a,b)}(\varphi'(a))=\pi_{(a',b)}(\varphi'(a'))}$. Thus, the number of satisfied edges is at least

$$
\sum_{b\in B}\left(\frac{1}{d_b}\sum_{a,a'\in\Gamma(b)} 1_{\pi_{(a,b)}(\varphi'(a))=\pi_{(a',b)}(\varphi'(a'))}\right).
$$

Moreover, from Lemma 15, $d_b \leq 10np$ for every $b \in B$ with probability $1-o(1)$. This implies that, with probability $1-o(1)$, the output assignment satisfied at least

$$
\frac{1}{10np}\sum_{b\in B}\sum_{a,a'\in\Gamma(b)} 1_{\pi_{(a,b)}(\varphi'(a))=\pi_{(a',b)}(\varphi'(a'))}
$$

edges.

We can further reorganize this quantity as follows.

$$
\begin{aligned}
\frac{1}{10np}\sum_{b\in B}\sum_{a,a'\in\Gamma(b)} 1_{\pi_{(a,b)}(\varphi'(a))=\pi_{(a',b)}(\varphi'(a'))} &\geq \frac{1}{10np}\sum_{b\in B}\sum_{\substack{(a,a')\in A'\times B' \\ \text{s.t. } a,a'\in\Gamma(b)}} 1_{\pi_{(a,b)}(\varphi'(a))=\pi_{(a',b)}(\varphi'(a'))} \\
&= \frac{1}{10np}\sum_{(a,a')\in A'\times B'}\sum_{b\in\Gamma(a)\cap\Gamma(a')} 1_{\pi_{(a,b)}(\varphi'(a))=\pi_{(a',b)}(\varphi'(a'))}.
\end{aligned}
$$

Now, observe that, from its definition, if $C_{(a,a')}(\varphi'(a),\varphi'(a'))$ is one, then $1_{\pi_{(a,b)}(\varphi'(a))=\pi_{(a',b)}(\varphi'(a'))}$ is also one for every $b \in \Gamma(a)\cap\Gamma(a')$. Thus, we have

$$
\begin{aligned}
&\frac{1}{10np}\sum_{(a,a')\in A'\times B'}\sum_{b\in\Gamma(a)\cap\Gamma(a')} 1_{\pi_{(a,b)}(\varphi'(a))=\pi_{(a',b)}(\varphi'(a'))} \\
&\geq \frac{1}{10np}\sum_{(a,a')\in A'\times B'}\sum_{b\in\Gamma(a)\cap\Gamma(a')} C_{(a,a')}(\varphi'(a),\varphi'(a'))
\end{aligned}
$$

$$= \frac{1}{10np} \sum_{(a,a') \in A' \times B'} |\Gamma(a) \cap \Gamma(a')| C_{(a,a')}(\varphi'(a), \varphi'(a')).$$

From Lemma 16, with probability $1 - o(1)$, $|\Gamma(a) \cap \Gamma(a')| \geq np^2/10$ for every $(a, a') \in A' \times B'$. Hence, we can conclude that the above expression is, with probability $1 - o(1)$, at least

$$\frac{1}{10np} \sum_{(a,a') \in A' \times B'} \frac{np^2}{10} C_{(a,a')}(\varphi'(a), \varphi'(a')) = \frac{p}{100} \sum_{(a,a') \in A' \times B'} C_{(a,a')}(\varphi'(a), \varphi'(a')).$$

Next, note that $\sum_{(a,a') \in A' \times B'} C_{(a,a')}(\varphi'(a), \varphi'(a'))$ is the number of edges satisfied by φ' in the free game, which is at least $\varepsilon |A'||B'| = \varepsilon n^2/16$. Thus, we have

$$\frac{p}{100} \sum_{(a,a') \in A' \times B'} C_{(a,a')}(\varphi'(a), \varphi'(a')) \geq \frac{\varepsilon n^2 p}{1600}.$$

Finally, again from Lemma 15, the total number of edges is at most $5n^2p$ with probability $1 - o(1)$. As a result, with probability $1 - o(1)$, the algorithm outputs an assignment that satisfies at least $\frac{\varepsilon}{8000} = \Omega(\varepsilon)$ fraction of edges of the projection game instance as desired. ◀

6 Approximation Algorithm for Densest k-Subgraph

The main goal of this section is to prove Corollary 11. As stated previously, we simply use our algorithm from Theorem 8 together with a reduction from Max 2-CSP to DkS from [10]. First, let us start by stating the reduction from Theorem 8, which we rephrase as follows.

▶ **Lemma 17** ([10]). *There exists a randomized polynomial-time algorithm that, given a graph G of N vertices and an integer $k \leq N$, produce an instance $(q, V, E, \{C_e\}_{e \in E})$ of* Max 2-CSP *such that*

- $q \leq N, n = k$, *and*
- *any solution to the instance can be translated in polynomial time to a subgraph of G of k vertices such that the number of edges in the subgraph equals to the number of edges satisfied by the* Max 2-CSP *solution, and*
- *with constant probability, the number of edges satisfied by the optimal solution to the instance is at least $1/100$ times the number of edges in the densest k-subgraph of G.*

We will not show the proof of Lemma 17 here; please refer to Theorem 6 from [10] for the proof. Instead, we will now show how to use the reduction to arrive at the proof of Corollary 11.

Proof of Corollary 11. First, we note that, to prove Corollary 11, it is enough to find a randomized polynomial-time algorithm with similar approximation guarantee to that in Corollary 11 except that the probability of success is a constant (instead of high probability as stated in Corollary 11). This is because we can then repeatedly run this algorithm $\Theta(\log n)$ times and produce the desired result.

The algorithm proceeds as follows:

1. Use the reduction from Lemma 17 on the input graph G and k to produce $(q, V, E, \{C_e\}_{e\in E})$.
2. Run the algorithm from Theorem 8 on $(q, V, E, \{C_e\}_{e\in E})$.
3. Transform the assignment from previous step according to Lemma 17 and output the result.

From the property of the reduction, we know that, with constant probability, the optimal assignment to $(q, V, E, \{C_e\}_{e\in E})$ satisfies $\Omega(\delta k^2)$ edges. If this is the case, we can conclude that the density of (V, E) is $\Omega(\delta)$ and, similarly, that the value of the instance is $\Omega(\delta)$. As a result, the output assignment from step 2 has value at least $\Omega(\delta^{O(1/\gamma)}N^{-\gamma})$. Since the reduction from Lemma 17 preserves the optimum, our algorithm produces a subgraph of density at least $\Omega(\delta^{O(1/\gamma)}N^{-\gamma})$ as well, which concludes our proof for this corollary. ◀

7 QPTAS for Dense Max 2-CSPs

At first glance, it seems that the QPTAS would follow easily for our main theorem. This, however, is not the case as the algorithm in the main theorem always loses at least a constant factor. Instead, we need to give an algorithm that is similar to that of the main theorem but have a stronger guarantee in approximation ratio, which can be stated as follows.

▶ **Lemma 18.** *For every positive integer $i > 0$, there exists an $O\left((nq)^{O(i)}\right)$-time algorithm that, for any satisfiable* Max 2-CSP *instance on the complete graph, produces an assignment of value at least $1/q^{1/i}$.*

Lemma 18 can be viewed as a special case of the main theorem when the graph is complete. However, it should be noted that Lemma 18 is more exact in the sense that the guaranteed lower bound of the value of the output assignment is not asymptotic. The proof of this lemma is also similar to that of Lemma 13 except that we need slightly more complicated algorithm and computation to deal with the fact that the underlying graph is not bipartite.

Proof of Lemma 18. We will prove the lemma by induction. Note that through out the proof, we will not worry about the randomness that the algorithm employs; it is not hard to see that the random assignment algorithms described below can be derandomized via greedy approach so that the approximation guarantees are as good as the expected guarantees of the randomized ones and that we still end up with the same asymptotic running time.

Let $P(i)$ represent the following statement: there exists an $O\left((nq)^{3i}\right)$-time algorithm Approx-CompleteGame$_i(q, V, E, \{C_e\}_{e\in E}, \{S_u\}_{u\in V})$ that takes in a satisfiable Max 2-CSP instance $(q, V, E, \{C_e\}_{e\in V})$ where (V, E) is a complete graph and a reduced alphabets set S_u for every $u \in U$ such that, if $\sigma_u^{OPT} \in S_u$ for every $u \in V$, then the algorithm outputs an assignment of value at least $\left(\prod_{u\in V} \frac{1}{|S_u|}\right)^{\frac{1}{ni}}$.

Observe that $P(i)$ implies the lemma by simply setting $S_u = [q]$ for every $u \in V$.

Base Case. The algorithm Approx-CompleteGame$_1(q, V, E, \{C_e\}_{e\in E}, \{S_u\}_{u\in V})$ is a simple random assignment algorithm. However, before we randomly pick the assignment, we need to first discard the alphabets that we know for sure are not optimal. More specifically, Approx-CompleteGame$_1(q, V, E, \{C_e\}_{e\in E}, \{S_u\}_{u\in V})$ works as follows.

1. While there exists $u, v \in U$ and $\sigma_u \in S_u$ such that $C_{(u,v)}(\sigma_u, \sigma_v) = 0$ for every $\sigma_v \in S_v$, remove σ_u from S_u.
2. For each $u \in V$, pick $\varphi(u)$ independently and uniformly at random from S_u. Output φ.

It is obvious that the algorithm runs in $O(n^3q^3)$ time as desired.

Now, we will show that, if $\sigma_u^{OPT} \in S_u$ for every $u \in V$, then the algorithm gives an assignment that is of value at least $\left(\prod_{u \in V} \frac{1}{|S_u|}\right)^{\frac{1}{n}}$ in expectation.

First, observe that σ_u^{OPT} remains in S_u after step 1 for every $u \in V$. This is because $C_{(u,v)}(\sigma_u^{OPT}, \sigma_v^{OPT}) = 1$ for every $v \neq u$.

Next, Consider the expected number of satisfied edges by the output assignment, which can be rearranged as follows:

$$\begin{aligned}\mathbb{E}\left[\sum_{(u,v)\in E} C_{(u,v)}(\varphi(u), \varphi(v))\right] &= \sum_{(u,v)\in E} \mathbb{E}\left[C_{(u,v)}(\varphi(u), \varphi(v))\right] \\ &= \sum_{(u,v)\in E} \frac{1}{|S_u||S_v|} \sum_{\sigma_u \in S_u} \sum_{\sigma_v \in S_v} C_{(u,v)}(\sigma_u, \sigma_v).\end{aligned}$$

From the condition of the loop in step 1, we know that after the loop ends, for each $\sigma_u \in S_u$, there must be at least one $\sigma_v \in S_v$ such that $C_{(u,v)}(\sigma_u, \sigma_v) = 1$. In other words,

$$\sum_{\sigma_u \in S_u} \sum_{\sigma_v \in S_v} C_{(u,v)}(\sigma_u, \sigma_v) \geq \sum_{\sigma_u \in S_u} 1 = |S_u|.$$

Similarly, we can also conclude that

$$\sum_{\sigma_u \in S_u} \sum_{\sigma_v \in S_v} C_{(u,v)}(\sigma_u, \sigma_v) \geq |S_v|.$$

Thus, we have

$$\sum_{\sigma_u \in S_u} \sum_{\sigma_v \in S_v} C_{(u,v)}(\sigma_u, \sigma_v) \geq \max\{|S_u|, |S_v|\}$$

for every $u \neq v$.

Hence, we can bound the expected number of satisfied edges as follows:

$$\begin{aligned}\sum_{(u,v)\in E} \frac{1}{|S_u||S_v|} \sum_{\sigma_u \in S_u} \sum_{\sigma_v \in S_v} C_{(u,v)}(\sigma_u, \sigma_v) &\geq \sum_{(u,v)\in E} \frac{1}{|S_u||S_v|} \max\{|S_u|, |S_v|\} \\ &= \sum_{(u,v)\in E} \frac{1}{\min\{|S_u|, |S_v|\}} \\ &\geq \sum_{(u,v)\in E} \frac{1}{\sqrt{|S_u||S_v|}} \\ \text{(A.M. - G.M. inequality)} &\geq |E| \left(\prod_{(u,v)\in E} \frac{1}{\sqrt{|S_u||S_v|}}\right)^{\frac{1}{|E|}}\end{aligned}$$

$$= |E| \left(\prod_{(u,v)\in E} \frac{1}{\sqrt{|S_u||S_v|}} \right)^{\frac{2}{n(n-1)}}$$

$$\text{(Each } u \in V \text{ appears in exactly } n-1 \text{ edges)} = |E| \left(\left(\prod_{u\in V} \frac{1}{|S_u|} \right)^{(n-1)/2} \right)^{\frac{2}{n(n-1)}}$$

$$= |E| \left(\prod_{u\in V} \frac{1}{|S_u|} \right)^{1/n},$$

which implies that $P(1)$ is true as desired.

Inductive Step. Let j be any positive integer. Suppose that $P(j)$ holds.

We will now describe Approx-CompleteGame$_{j+1}$ based on Approx-CompleteGame$_j$ as follows.

1. Define R to be $\left(\prod_{u\in V} \frac{1}{|S_u|}\right)^{\frac{1}{n(j+1)}}$, our target value we want to achieve.
2. Run the following steps 2(a)i to 2(a)iv until no S_u is modified by neither step 2(a)iv nor step 2(a)ii.
   a. For each $u \in V$ and $\sigma_u \in S_u$, do the following:
      i. For each $v \in V$, compute $S_v^{u,\sigma_u} = \{\sigma_v \in S_v \mid C_{(u,v)}(\sigma_u, \sigma_v) = 1\}$. This is the set of reduced assignment of v if we assign σ_u to u. Note that when $v = u$, let $S_u = \{\sigma_u\}$.
      ii. If $S_v^{u,\sigma_u} = \emptyset$ for some $v \in V$, then remove σ_u from S_u and continue to the next u, σ_u pair.
      iii. Compute $R^{u,\sigma_u} = \left(\prod_{v\in V} \frac{1}{|S_v^{u,\sigma_u}|}\right)^{\frac{1}{nj}}$. If $R' < R$, continue to the next u, σ_u pair.
      iv. Execute Approx-CompleteGame$_j(q, V, E, \{C_e\}_{e\in E}, \{S_v^{u,\sigma_u}\}_{v\in V})$. If the output assignment is of value less than R^{u,σ_u}, then remove σ_u from S_u. Otherwise, return the output assignment as the output to Approx-CompleteGame$_{j+1}$.
3. If the loop in the previous step ends without outputting any assignment, just output a random assignment (i.e. pick $\varphi(u)$ independently and uniformly at random from S_u).

Observe first that the loop can run at most nq times as the total number of elements of S_v's for all $v \in V$ is at most nq. This means that we call Approx-CompleteGame$_j$ at most nq times. Since every step except the Approx-CompleteGame$_j$ calls takes $O((nq)^3)$ time and we call Approx-CompleteGame$_j$ only at most n^2q^2 times, we can conclude that the running time of Approx-CompleteGame$_{j+1}$ is $O((nq)^{3j+3})$ as desired.

The only thing left to show is that the assignment output from the algorithm indeed is of expected value at least R. To do so, we will consider two cases.

First, if step 3 is never reached, the algorithm must terminate at step 2(a)iv. From the return condition in step 2(a)iv, we know that the output assignment is of value at least $R^{u,\sigma_u} \geq R$ as desired.

In the second case where step 3 is reached, we first observe that when we remove s_u from S_u in step 2(a)iv, the instance is still satisfiable. The reason is that, if $\sigma_u = \sigma_u^{OPT}$ is the optimal as-

signment for u, then σ_v^{OPT} remains in S_v^{u,σ_u} for every $v \in V$. Hence, from our inductive hypothesis, the output assignment from Approx-CompleteGame$_j(q, V, E, \{C_e\}_{e\in E}, \{S_v^{u,\sigma_u}\}_{v\in V})$ must be of value at least R^{u,σ_u}. As a result, we never remove s_u^{OPT} from S_u, and, thus, the instance remains satisfiable throughout the algorithm.

Moreover, notice that, if $R^{u,\sigma_u} \geq R$ for any u, σ_u, we either remove σ_u from S_u or output the desired assignment. This means that, when step 3 is reached, $R^{u,\sigma_u} < R$ for every $u \in V$ and $\sigma_u \in S_u$.

Now, let us consider the expected number of edges satisfied by the random assignment. Since our graph (V, E) is a complete, it can be written as follows.

$$
\begin{aligned}
\mathbb{E}\left[\sum_{(u,v)\in E} C_{(u,v)}(\varphi(u),\varphi(v))\right] &= \mathbb{E}\left[\frac{1}{2}\sum_{u\in V}\sum_{\substack{v\in V\\ v\neq u}} C_{(u,v)}(\varphi(u),\varphi(v))\right] \\
&= \frac{1}{2}\sum_{u\in V}\sum_{\substack{v\in V\\ v\neq u}} \mathbb{E}\left[C_{(u,v)}(\varphi(u),\varphi(v))\right] \\
&= \frac{1}{2}\sum_{u\in V}\sum_{\substack{v\in V\\ v\neq u}} \frac{1}{|S_u||S_v|}\left(\sum_{\sigma_u\in S_u}\sum_{\sigma_v\in S_v} C_{(u,v)}(\sigma_u,\sigma_v)\right) \\
\text{(From definition of } S_v^{u,\sigma_u}) &= \frac{1}{2}\sum_{u\in V}\sum_{\substack{v\in V\\ v\neq u}} \frac{1}{|S_u||S_v|}\left(\sum_{\sigma_u\in S_u} |S_v^{u,\sigma_u}|\right) \\
&= \frac{1}{2}\sum_{u\in V}\frac{1}{|S_u|}\sum_{\sigma_u\in S_u}\left(\sum_{\substack{v\in V\\ v\neq u}} \frac{|S_v^{u,\sigma_u}|}{|S_v|}\right) \\
\text{(A.M.-G.M. inequality)} &\geq \frac{1}{2}\sum_{u\in V}\frac{1}{|S_u|}\sum_{\sigma_u\in S_u}(n-1)\sqrt[n-1]{\prod_{\substack{v\in V\\ v\neq u}}\frac{|S_v^{u,\sigma_u}|}{|S_v|}} \\
&= \frac{(n-1)}{2}\sum_{u\in V}\frac{1}{|S_u|}\sum_{\sigma_u\in S_u}\sqrt[n-1]{\frac{\prod_{\substack{v\in n\\ v\neq u}}|S_v^{u,\sigma_u}|}{\prod_{\substack{v\in V\\ v\neq u}}|S_v|}} \\
\text{(From our definition of } R^{u,\sigma_u}, R) &= \frac{(n-1)}{2}\sum_{u\in V}\frac{1}{|S_u|}\sum_{\sigma_u\in S_u}\sqrt[n-1]{\frac{(R^{u,\sigma_u})^{-nj}}{\frac{R^{-n(j+1)}}{|S_u|}}} \\
\text{(Since } R^{u,\sigma_u} < R) &> \frac{(n-1)}{2}\sum_{u\in V}\frac{1}{|S_u|}\sum_{\sigma_u\in S_u}\sqrt[n-1]{|S_u|R^n} \\
&= \frac{(n-1)}{2}\sum_{u\in V}\sqrt[n-1]{|S_u|R^n} \\
&= \frac{(n-1)}{2}R^{n/(n-1)}\left(\sum_{u\in V}\sqrt[n-1]{|S_u|}\right) \\
\text{(A.M.-G.M. inequality)} &\geq \frac{(n-1)}{2}R^{n/(n-1)}\left(n\sqrt[n(n-1)]{\prod_{u\in V}|S_u|}\right)
\end{aligned}
$$

$$
\begin{aligned}
\text{(From our definition of } R) &= \frac{(n-1)}{2} R^{n/(n-1)} \left(n \sqrt[n(n-1)]{\prod_{u \in V} R^{-n(j+1)}} \right) \\
&= \frac{n(n-1)}{2} R^{(n-1-j)/(n-1)} \\
\text{(Since } R \leq 1 \text{ and } j \geq 0) &\geq \frac{n(n-1)}{2} R.
\end{aligned}
$$

Since $\frac{n(n-1)}{2}$ is the number of edges in (V, E), we can conclude that the random assignment is indeed of expected value at least R.

Thus, we can conclude that $P(j+1)$ is true. As a result, $P(i)$ is true for every positive integer i, which completes the proof for Lemma 18. ◀

Next, we will prove Corollary 12 by reducing it to Max 2-CSP on complete graph, and, then plug in Lemma 18 with appropriate i to get the result.

First, observe that, since $\log(1+\varepsilon') = \Omega(\varepsilon')$ for every $1 \geq \varepsilon' > 0$, by plugging in $i = C \log q/\varepsilon'$ for large enough constant C into Lemma 18, we immediately arrive the following corollary.

▶ **Corollary 19.** *For any $1 \geq \varepsilon' > 0$, there exists an $(1+\varepsilon')$-approximation algorithm for satisfiable* Max 2-CSP *on the complete graph that runs in time $N^{O(\varepsilon'^{-1} \log N)}$.*

Now, we will proceed to show the reduction and, thus, prove Corollary 12.

Proof of Corollary 12. First of all, notice that, since $\frac{1}{1+\varepsilon} = 1 - \Theta(\varepsilon)$. It is enough for us to show that there exists a $N^{O(\varepsilon^{-1}\delta^{-1} \log N)}$-time algorithm for satisfiable δ-dense Max 2-CSP that produces an assignment of value at least $1-\varepsilon$.

On input $(q, V, E, \{C_e\}_{e \in E})$, the algorithm works as follows:

1. Construct a Max 2-CSP instance $(q, V, E', \{C'_e\}_{e \in E'})$ where (V, E') is a complete graph and C'_e is defined as C_e if $e \in E$. Otherwise, $C_e := 1$. In other words, we put in dummy constraints that are always true just to make the graph complete.
2. Run the algorithm from Corollary 19 on $(q, V, E', \{C'_e\}_{e \in E'})$ with $\varepsilon' = \varepsilon\delta$ and output the assignment got from the algorithm.

To see that the algorithm indeed produces an assignment with value $1-\varepsilon$ for the input instance, first observe that, since $(q, V, E, \{C_e\}_{e \in E})$ is satisfiable, $(q, V, E', \{C'_e\}_{e \in E'})$ is trivially satisfiable. Thus, from Corollary 19, the output assignment has value at least $1/(1+\delta\varepsilon) \geq 1 - \delta\varepsilon$ with respect to $(q, V, E', \{C'_e\}_{e \in E'})$. In other words, the assignment does not satisfy at most $\delta\varepsilon n^2$ edges. Thus, with respect to the input instance, it satisfies at least $\delta n^2 - \delta\varepsilon n^2 = (1-\varepsilon)\delta n^2$ edges. In other words, it is of value at least $1-\varepsilon$ as desired.

Lastly, note that the running time of this algorithm is determined by that of the algorithm from Corollary 19, which runs in $N^{O(\varepsilon'^{-1} \log N)} = N^{O(\varepsilon^{-1}\delta^{-1} \log N)}$ time as desired. ◀

8 Conclusions and Open Questions

Finally, we conclude by listing the open questions and interesting directions related to the techniques and problems presented here. We also provide our thoughts regarding each question.

- *Can our algorithm be extended to work for* Max k-CSP *for* $k \geq 3$*?* Other algorithms for approximating Max 2-CSP such as those from [2, 3, 4] are applicable for Max k-CSP for any value of k as well. So it is possible that our technique can be employed for Max k-CSP too.
- *Can one also come up with an algorithm that approximates* Max 2-CSP *to within* N^{ε} *factor for any* $\varepsilon > 0$ *for low-value dense* Max 2-CSP*?* Our algorithm needs the value λ to be $N^{-o(1)}$ in order to give such a ratio so it is interesting whether we can remove or relax this condition. However, we do not think that one can remove the condition completely because, with similar technique to the proof of Corollary 12, we can arrive at a reduction from any Max 2-CSP to dense Max 2-CSP where the approximation ratio is preserved but the value decreases. This means that, if we can remove the condition on λ, then we are also able to refute the PGC. This argument nonetheless does not rule out relaxing the condition for λ without removing it completely.
- *Can our QPTAS be extended to unsatisfiable instances?* One of the main disadvantages of our QPTAS is that it requires the instance to be satisfiable. This renders our QPTAS useless against many problems such as Max 2-SAT and Max-Cut because the satisfiable instances of those problems are trivial. If we can extend our QPTAS to work on unsatisfiable instances as well, then we may be able to produce interesting results for those problems. Note, however, that, with similar argument to the preceding question, QPTAS for low-value instances likely does not exists. Instead, the case of unsatisfiable instances where [2, 3, 4] are successful is when they look for an additive error guarantee instead of a multiplicative one. Currently, it is unclear whether our technique can achieve such results.
- *Can one arrive at a similar or even better algorithm using SDP hierarchies?* SDP hierarchies have been very useful in finding approximation algorithms for combinatorial optimization problems. A natural question to ask is whether one can apply SDP hierarchies to get similar results to ours. For example, can the $O(i)$-level of the Lasserre hierarchy produce an approximation algorithm with ratio $O(q^{1/i})$ for dense Max 2-CSP? If so, then this may also be an interesting direction to pursue an algorithm with guarantee additive error discussed previously.

References

1 S. Aaronson, R. Impagliazzo, and D. Moshkovitz. AM with multiple Merlins. In *Computational Complexity (CCC), 2014 IEEE 29th Conference on*, pages 44–55, June 2014.

2 N. Alon, W. F. de la Vega, R. Kannan, and M. Karpinski. Random sampling and approximation of max-CSPs. *J. Comput. Syst. Sci.*, 67(2):212–243, September 2003.

3 S. Arora, D. Karger, and M. Karpinski. Polynomial time approximation schemes for dense instances of NP-hard problems. In *Proceedings of the Twenty-seventh Annual ACM Sym-*

posium on Theory of Computing, STOC '95, pages 284–293, New York, NY, USA, 1995. ACM.

4 B. Barak, M. Hardt, T. Holenstein, and D. Steurer. Subsampling mathematical relaxations and average-case complexity. In *Proceedings of the Twenty-second Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '11, pages 512–531. SIAM, 2011.

5 B. Barak, A. Rao, R. Raz, R. Rosen, and R. Shaltiel. Strong parallel repetition theorem for free projection games. In *Proceedings of the 12th International Workshop and 13th International Workshop on Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, APPROX '09 / RANDOM '09, pages 352–365, Berlin, Heidelberg, 2009. Springer-Verlag.

6 M. Bellare, O. Goldreich, and M. Sudan. Free bits, PCPs, and nonapproximability—towards tight results. *SIAM Journal on Computing*, 27(3):804–915, 1998.

7 A. Bhaskara, M. Charikar, E. Chlamtac, U. Feige, and A. Vijayaraghavan. Detecting high log-densities: An $O(n^{1/4})$ approximation for densest k-subgraph. In *Proceedings of the Forty-second ACM Symposium on Theory of Computing*, STOC '10, pages 201–210, New York, NY, USA, 2010. ACM.

8 F. G.S.L. Brandao and A. W. Harrow. Quantum de finetti theorems under local measurements with applications. In *Proceedings of the Forty-fifth Annual ACM Symposium on Theory of Computing*, STOC '13, pages 861–870, New York, NY, USA, 2013. ACM.

9 M. Braverman, Y. K. Ko, and O. Weinstein. Approximating the best nash equilibrium in $n^{o(\log n)}$-time breaks the exponential time hypothesis. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '15, pages 970–982. SIAM, 2015.

10 M. Charikar, M. Hajiaghayi, and H. Karloff. Improved approximation algorithms for label cover problems. In *ESA*, pages 23–34, 2009.

11 U. Feige, D. Peleg, and G. Kortsarz. The dense k -subgraph problem. *Algorithmica*, 29(3):410–421, 2001.

12 J. Håstad. Some optimal inapproximability results. *Journal of the ACM*, 48(4):798–859, 2001.

13 S. Khot. Ruling out PTAS for graph min-bisection, densest subgraph and bipartite clique. In *Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science*, FOCS '04, pages 136–145, Washington, DC, USA, 2004. IEEE Computer Society.

14 P. Manurangsi and D. Moshkovitz. Improved approximation algorithms for projection games. In *Algorithms – ESA 2013*, volume 8125 of *Lecture Notes in Computer Science*, pages 683–694. Springer Berlin Heidelberg, 2013.

15 D. Moshkovitz. The projection games conjecture and the NP-hardness of $\ln n$-approximating set-cover. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques - 15th International Workshop, APPROX 2012*, volume 7408, pages 276–287, 2012.

16 R. Raz. A parallel repetition theorem. In *SIAM Journal on Computing*, volume 27, pages 763–803, 1998.

17 R. Shaltiel. Derandomized parallel repetition theorems for free games. *Comput. Complex.*, 22(3):565–594, September 2013.

18 A. Suzuki and T. Tokuyama. Dense subgraph problems with output-density conditions. In Xiaotie Deng and Ding-Zhu Du, editors, *Algorithms and Computation*, volume 3827 of *Lecture Notes in Computer Science*, pages 266–276. Springer Berlin Heidelberg, 2005.