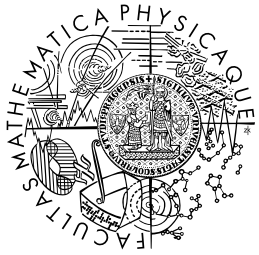

FACULTY
OF MATHEMATICS
AND PHYSICS
Charles University

MASTER THESIS

Filip Bialas

Combinatorial Gap Label Cover

Department of Algebra

Supervisor of the master thesis: doc. Mgr. Libor Barto, Ph.D.

Study programme: Mathematics (N1101)

Study branch: MSTR (1101T039)

Prague 2022

I declare that I carried out this master thesis independently, and only with the cited sources, literature and other professional sources. It has not been used to obtain another or the same degree.

I understand that my work relates to the rights and obligations under the Act No. 121/2000 Sb., the Copyright Act, as amended, in particular the fact that the Charles University has the right to conclude a license agreement on the use of this work as a school work pursuant to Section 60 subsection 1 of the Copyright Act.

In ..... date ....

Author's signature

Firstly, I would like to thank my supervisor doc. Mgr. Libor Barto, Ph.D. for introducing the topic of this thesis to me and spending a lot of his precious time on helping me with the project. Secondly, I would like to thank L. P. for her support and quick responses to questions concerning geometry in more dimensions than I am able to picture well. Though, unfortunately, categorically rejecting questions about four or more dimensional spaces.

Title: Combinatorial Gap Label Cover

Author: Filip Bialas

Department: Department of Algebra

Supervisor: doc. Mgr. Libor Barto, Ph.D., Department of Algebra

Abstract: Theorems about probabilistically checkable proofs (PCP) are famous hard-to-prove results from the theoretical computer science. They provide constructions of PCP systems with interesting surprising properties and serve as a starting point for proofs of NP-hardness of many approximation problems. Recently, a weaker combinatorial version of one of these theorems (Gap Label Cover) was proved using only combinatorial tools. After summarizing the main results in the classical PCP theory, I explore the combinatorial version thoroughly. Original results of this thesis consist of counterexamples to the expected behavior of two concepts from the classical theory in the combinatorial setting — probabilistic version and parallel repetition.

Keywords: PCP Parallel Repetition Combinatorial Gap Label Cover

Contents

Introduction

PCP theorem (standing for probabilistically checkable proofs) is a famous theorem from complexity theory first proved in [1]. In its most famous form, it states that any proof can be rewritten as a proof of polynomial length, such that there exists a randomized algorithm that checks the proof validity with more than $\frac{1}{2}$ accuracy by looking only at a fixed number of letters. The theorem can be reformulated in terms of NP-hardness of distinguishing solvable instances from instances in which only a certain fraction of constraints can be satisfied in specific constraint satisfaction problems.

The PCP theorem has a lot of interesting corollaries, such as proofs of NP-hardness of different approximation algorithms. Other related problems in the field remain unsolved, such as Unique Games conjecture or d-to-1 conjecture [10].

The theorem is useful, but since its original proof is complicated, the theorem is often used just as a black box. Later, a more accessible proof using expander graphs was given in [6]. Although much easier than the original one, even this one is still quite long. In [3] slightly different (more combinatorial) problem called Baby PCP was proposed and proved with combinatorial tools as a corollary of a technical combinatorial lemma, which was the main result of the paper. Baby PCP is a weaker theorem, but it's strong enough to prove some of the corollaries of real PCP. There is a chance that some of the open problems (such as d-to-1 conjecture) could be easier to prove in the Baby PCP setting and would provide a good stepping stone and understanding so that the proof of their full versions would be later more manageable.

In this thesis, I explore the Baby PCP setting mainly concerning the combinatorial version of the famous Parallel Repetition theorem ([11]). Unfortunately, I end up with only negative results, though they are not trivial and still noteworthy.

In the first chapter, I explore the classic theory. After concluding that the two formulations of the PCP theorem are the same, I talk about PCP mainly in its CSP formulation. Next, I discuss stronger versions of the PCP theorem — NP-hardness results with the possible instances made more specific. Those can also be described in the language of specific cooperative two-player games introduced in [5]. Especially the version called the *Label Cover* for so-called projection games is considered. The d-to-1 conjecture and a version of the famous Unique Games conjecture are also stated here as an intuitive way to make the theorem even stronger.

The Parallel Repetition theorem is used to provide an arbitrary large gap while increasing the size of the domain/alphabet in some of the PCP theorems. I don't give a proof of it, but I discuss counterexamples for the most intuitive approach.

In the second chapter, I summarize part of the article [3], repeating proofs of the concepts regarding Baby PCP while leaving the harder technical lemma, which was the main result of the paper, without proof. The lemma is nevertheless discussed thoroughly, and a not-yet-published counterexample for its version, which would prove a combinatorial version of the d-to-1 conjecture, is given.

In the third chapter, I explore a slight variation of the combinatorial definition calling it *probabilistically combinatorial* or *pc*. This definition was considered

informally by authors of [3], though not developed, and all results in this chapter are original. I show that the pc version is more different from the combinatorial one than one would probably think and that the pc version of PCP lies between the real and Baby PCP theorem in strength. The proof of the pc version of PCP is unfortunately given only using the real PCP theorem. The reasons why the technical lemma from [3] can not be used here easily are also discussed.

In the final chapter, I consider a combinatorial version of the Parallel Repetition theorem and give a counterexample. Creating and proving the correctness of the counterexample takes the entire chapter and is the thesis's main original result.

1. PCP and Gap Laber Cover

This chapter summarizes the general context for results in later chapters. There are no original results here. Most of the chapter builds on lecture notes [9].

In the first section, I formulate the PCP theorem, define constraint satisfaction problems (CSPs, [2]) and prove equivalence of the most famous formulation of PCP with the CSP formulation of it. In the second section, the concept of Two-prover One-round games from [5] is developed, and a somewhat stronger PCP result called the Gap Label Cover theorem is shown. In the last section, I discuss the Parallel Repetition theorem, which is used to introduce an arbitrary large gap in the Gap Label Cover theorem.

1.1 PCP

The abbreviation PCP stands for probabilistically checkable proofs. For context, we first talk about standard proof systems, which work as a formalization of proving mathematical statements.

Having a finite alphabet Σ, any statement (a string of finite length with letters from Σ) can be true or false. The proof system has a verifier, which for any proof (a string in the same alphabet) decides if it is valid for the given statement or not; it accepts the proof or rejects it. The conditions that must be satisfied for the proof system to work are completeness (for any true statement, there exists a proof that the verifier accepts) and soundness (the verifier rejects all proofs of false statements).

The definition is interesting only if we add more conditions, such as proofs of short statements being short or verifier being a Turing machine. As an example, in this setting, we say that the language $L \subset \bigcup_{i=1}^{\infty} \Sigma^i$ is in NP iff there is a proof system for it whose proofs have to be polynomial in the statement length, and its verifier has to be a deterministic Turing Machine running in polynomial time. The language is in P if in addition we can find a proof that the verifier accepts deterministically in polynomial time in the length of the statement.

In the case of probabilistically checkable proofs, we allow the verifier to be randomized, and we relax the soundness condition. We are ok if the verifier rejects the proof correctly only with some probability. If we want to be more sure if the proof is valid, we can run the verifier independently more times and decrease the probability of an error to arbitrarily small values.

With this introduction, we can state the PCP theorem.

Theorem 1 (PCP Theorem)**.** *For any language from NP, a PCP system exists, such that it accepts the proofs of false statements with probability at most $\frac{1}{2}$ and has the following properties. Given a statement of length n, the proofs of it are of polynomial length in n, and the verifier is a randomized one, which uses at most $O(\log n)$ bits to decide on which C locations in the proof to look at and a deterministic test to do with these C letters, where C is a universal constant. The verifier doesn't see any other bits of the proof.*

This is an interesting result since C is a universal constant, and it's rather surprising that one needs to see just a few bits of the proof to decide with a large

probability of success if the proof is correct even if it is long. Also, the probability can be decreased to arbitrarily small values by independently running the verifier many times. Running it K times, we see at most CK bits of the proof, and we decrease the probability of accepting incorrectly to at most 2^{-K}.

There is an equivalent formulation of the PCP theorem using CSPs. First we define what we mean by CSPs using definitions from [2].

Definition 1 (CSP)**.** *A* Constraint Satisfaction Problem *(or* CSP*) is a triple* (V, D, S)*, where* V *is a finite set of variables;* D *is a finite domain; and* S *is a finite set of constraints, each of them being* (x, R)*, where* x *is an* m*-tuple of variables from* V *for some natural* m *and* R *any* m*-ary relation on* D*.*

An assignment of variables (mapping $f\colon V \to D$*)* satisfies *a constraint* (x, R)*, if* $f(x) \in R$*. A CSP instance is* solvable *if there exists an assignment satisfying all constraints.*

The most basic question we can ask about a CSP instance is if it is solvable. If it is not solvable we can explore how many constraints can be satisfied at most.

Via the CSP we can define other problem classes by restricting how the constraints can look like. One such class of problems we will use a lot are CSPs where each constraint uses at most some fixed number of variables.

Another example of a problem that can be formulated in this way is the well-known NP-complete 3-SAT. The instances are logical formulas on finite number of variables, which are conjunctions of disjunctions of at most 3 literals (variables or their negations). In the CSP setting we can take V as the set of variables in the formula, $D = \{0, 1\}$ (boolean domain) and S having only constraint of type $y_i \vee y_j \vee y_k$, $y_i \vee y_j$, and y_i, where y_l is one of the variables from V or its negation.

On the other hand in some settings it's worth it to relax some of the conditions in the definition. The one we relax later in the text is that all variables have the same domain. We will also consider different domain sizes for different variables later in this text.

With this introduction we are ready to state the PCP theorem in the CSP language.

Theorem 2 (PCP Theorem in CSP setting)**.** *There exists* $C \in \mathbb{N}$ *and a domain* D*, such that it is NP-hard to distinguish between solvable CSPs over* D *that use only* C *variables in every constraint and the ones where only half of the constraints can be satisfied at once. This decision problem is denoted by* $CSP_{\frac{1}{2},1}$*.*

Claim 3. *The two formulations of the PCP theorem (Theorems 1 and 2) are equivalent.*

Proof. The CSP formulation of PCP follows from Theorem 1 when we consider the verifier as a CSP, with domain D being the alphabet Σ. The length of the statement we are about to prove is denoted by n here. The verifier chooses where to look using $O(\log n)$ bits, which means that it has a polynomial (in n) number of options where to look. For every choice, we get one CSP constraint that uses at most C variables. Combinations of values which the PCP theorem accepts are in the relation. The CSP is solvable when there exists a correct proof for the statement (when the verifier answers "yes" with probability 1); otherwise, at most half of its constraints can be satisfied. Thus if we use the PCP theorem for any NP-complete language (such as 3-SAT), the verifier provides us with a

polynomial reduction from the NP language to $\mathrm{CSP}_{\frac{1}{2},1}$. Therefore, distinguishing between solvable CSPs and those for which at most half of constraints can be satisfied is also an NP-hard problem.

On the other hand, when we know the NP-hardness of $\mathrm{CSP}_{\frac{1}{2},1}$, we can construct a polynomial reduction of any NP language to this one, which gives us precisely the proof system we want. We let the proofs write out the values of the CSP variables, to which we reduced the problem (again taking $\Sigma = D$). Since the reduction was polynomial in statement length, the number of constraints and variables is also polynomial in statement length. The verifier then randomly uniformly chooses one constraint and checks if it holds. The proof is correct iff all constraints are satisfied. If the proof is not valid, at most half of the constraints are satisfied and the verifier accepts with probability at most $\frac{1}{2}$. So both completeness and weak soundness of the PCP system are established.

□

At first look, this formulation doesn't look as impressive as the previous one. However, it is a great starting point for proving the NP-hardness of approximation algorithms (algorithms where the goal is not to find the best solution, but just a solution which is at least c-times as good for some constant $c \in (0, 1)$). From the PCP theorem, it follows that there is a C and D, such that the approximation problem (with $c = \frac{1}{2} + \epsilon$ for arbitrarily small $\epsilon > 0$) of satisfying the most constraints in the CSPs with constraints using only C variables over domain D is NP-hard.

We use the identity as the reduction from the problem in the PCP theorem to the corresponding approximation problem. Whenever the approximation algorithm gives an answer satisfying more than half of the constraints, we know that the instance is solvable. On the other hand, whenever we get an answer less or equal to $\frac{1}{2}$, we know that the problem is not solvable since we assumed that our algorithm is an approximation one with $c = \frac{1}{2} + \epsilon$.

1.2 Gap Label Cover

In this section, we explore another result from the theory (the Gap Label Cover theorem), which is stronger in a sense. We will try to achieve similar results as in the previous chapter with the added constraint that we allow the verifier to see only 2 letters of the proof. The PCP statement that there is a gap (that there is an ϵ, such that the verifier accepts incorrectly with probability less than ϵ) and that only 2 letters are used was used and improved in [11]. In this case, we can't decrease the probability by repeating verifier runs (because we are constrained to look only at two letters). Still, it can be shown that the ϵ can be decreased arbitrarily close to zero, but we need to increase the size of the alphabet to do that. The fact that it is possible is in no way trivial, and we use the Parallel Repetition theorem from [11] without proof to see it.

First, we define the so-called Two-prover One-round Games and show how they correspond to the Gap Label Cover theorem in both the proof system and the CSP setting.

Definition 2 (Two-prover One-round Game)**.** *The* Two-prover One-round Game

G has the following rules. There are two players (provers) playing cooperatively and a verifier.

There is a finite alphabet of answers Σ and finite sets of questions X and Y for each player. There is a probability distribution on $X \times Y$ from which the verifier chooses a question for both players. Players need to answer it with one of the elements of Σ without knowing the question for the other player, or in other words, they have to agree on strategies $f_1 \colon X \to \Sigma$, $f_2 \colon Y \to \Sigma$ beforehand. The only remaining piece is the set of the correct answers to all possible questions (formally a subset of $X \times Y \times \Sigma \times \Sigma$).

The provers know every parameter except the question from the verifier to the other prover. Their goal is to answer the questions correctly with the highest possible probability. The highest probability is called the value *of G or* $\mathrm{val}(G)$.

We can construct a PCP system where one of the Two-prover One-round Games is played for every statement. To have similar properties as in the PCP Theorem 1 we additionally assume that if the statement has length n, then the size of the corresponding Two-prover One-round Game sets X, Y is at most polynomial in n and the probability distribution uses only $O(\log n)$ bits of information.

The proof is the concatenation of prepared answers of the two provers. The verifier asks only for two letters, one from each prover. The proof is valid if all the constraints are satisfied (provers answer correctly every time). The goal of the PCP theorems is to find a verifier which is not foolable much by the provers when the proof is invalid.

On the CSP side, we can imagine the game as a complete bipartite graph on sets X, Y, with one constraint on every edge and a probability distribution over the edges.

We state without proof the following result.

Claim 4. *There exists $\epsilon \in (0, 1)$ and a finite alphabet Σ, such that for any language in NP, there is a PCP system arising from the Two-player One-round Game over Σ with polynomial size. The provers satisfy all constraints when the proof is valid and can score with at most ϵ probability for non-valid proofs.*

To decrease the probability of error to arbitrarily small ϵ, we let the game happen more times (let's say k-times), but so that the players see every question simultaneously. Therefore their answer will be in Σ^k (they provide one answer for every question they get). If the game is called G, the k-times repeated game is denoted by $G^{\otimes k}$.

It is pretty easy to see that if the best probability players could achieve with G is ϵ, they can achieve the probability ϵ^k in $G^{\otimes k}$ by answering independently every of the k questions. But the surprising result is that they can do better – seeing all questions simultaneously lets the players find a better strategy. We will consider examples of this in the next section. Now we will just state without proof the Parallel Repetition theorem from [11].

Theorem 5 (Parallel Repetition Theorem)**.** *For every finite alphabet Σ, there exists a function $\alpha \colon [0, 1) \to [0, 1)$, such that for any two-player one-round game G with value $g < 1$, the value of $G^{\otimes k}$ is at most $\alpha(g)^k$.*

The theorem says that the value decreases exponentially, as one would suspect, but not necessarily as g^k. Now we can state the next theorem from the PCP

theory (a slight reformulation of Claim 4), which follows trivially from Claim 4 and Theorem 5.

Theorem 6. *For every $\epsilon \in (0,1)$, there exists a finite alphabet Σ, such that for any language in NP, there is a PCP system arising from the Two-player One-round Game over Σ with polynomial size. The provers satisfy all constraints when the proof is valid and can score with at most ϵ probability for non-valid proofs.*

We could again use the same arguments as in the previous chapter to rewrite this formulation to an equivalent one with CSPs, but I won't repeat the arguments here. For the rest of the chapter, I will use the CSP formulation and consider stronger results by constraining the problem definition more.

One thing one can consider is letting the verifier constraints (a subset of $\Sigma \times \Sigma$) be just functions on some subset of Σ – to allow at most one correct answer for the second player for any answer of the first player. Or we could constrain it further so that the constraints for all edges with a nonzero probability of choosing are just functions on Σ (then it makes more sense to picture the edges as arrows going from vertices of one player to the vertices of the second player). Or we could consider not a complete bipartite graph but any bipartite graph on the two partites and a uniform distribution on the edges for the verifier.

If we constrain all of this, the NP-hardness result still holds and goes by the name Gap Label Cover theorem.

Definition 3 (Label Cover instance)**.** Label cover instance *I over finite domain D with variable sets X, Y, set A of arrows starting in X and ending in Y with constraints given by $F\colon A \to D^D$ is the instance of a CSP problem, where we search for $f\colon X \cup Y \to D$, such that the most constraints are satisfied (a constraint for an arrow a going from x to y is satisfied if $F(a)(f(x)) = f(y)$). I also say that the arrow a is* decorated *by mapping $F(a)$.*

If f satisfies k constraints, we say that the value *of f is $\frac{k}{|A|}$. The* value *of the instance I is the maximum possible value of any such function. If the value of I is 1, we say that I is* solvable*.*

Theorem 7 (Gap Label Cover)**.** *For any $\epsilon \in (0,1)$, there is a finite domain D, such that it is NP-hard to distinguish between solvable Label Cover instances over D and those with value at most ϵ.*

This is the last and the strongest PCP theorem mentioned here. In the later chapters, we will consider this definition as the starting point.

We finish the section with consideration of two open conjectures that try to say even stronger things. The first one is the d-to-1 conjecture:

Conjecture 8 (d-to-1)**.** *For any $\epsilon \in (0,1)$ and $d > 1$, there exists a finite domain D, such that it is NP-hard to distinguish between solvable Label Cover instances over D where all constraints are d-to-1 functions (for any answer of the second player there are at most d correct answers for the first one) and those with value at most ϵ.*

Constraining even more to $d = 1$, we get to so-called Unique Games where all constraints are bijections. In this case, we can solve the corresponding label cover instance easily by the following polynomial algorithm. We try all possible values from D for one vertex in each component and then run a depth-first search from this vertex. Since all constraints are bijection, we have exactly one option

for how to fill values in all vertices we encounter. The instance is solvable if a value exists for each component's starting vertex, so we don't arrive at any contradiction. Tweaking the problem statement just a little bit (taking almost solvable instances instead of the solvable ones), we get a variant of the famous Unique Games conjecture.

Conjecture 9 (Unique Games)**.** *For any $\epsilon \in \left(0, \frac{1}{2}\right)$, there is a finite domain D, such that it is NP-hard to distinguish between Label Cover instances over D, where all constraints are bijections, of value at least $1-\epsilon$ and the ones with value at most ϵ.*

1.3 Parallel Repetition Theorem

I spend a lot of time on the combinatorial version of the Parallel Repetition theorem in the last chapter. To have more context about the theorem, in this section, I quickly discuss why the naive strategy of playing k games independently isn't the best.

The most famous counterexample comes from [8]. The game has value $\frac{1}{2}$ and the second power has the exactly same value, surprisingly.

Let us consider an alphabet $\Sigma = \{X0, X1, Y0, Y1\}$. Both sets X and Y have two elements: $X = \{x_0, x_1\}$, $Y = \{y_0, y_1\}$. The edges form a complete bipartite graph with a uniform probability distribution on them. The constraints are such that between x_i and y_j only (Xi, Xi) or (Yj, Yj) are correct answers. The players answer correctly only if they agree on the same player and correctly guess the question the player got.

The value of this game is at most $\frac{1}{2}$ since for them to succeed, one of the players has to guess the question of the other player, and that can be done correctly with a probability of just $\frac{1}{2}$.

On the other hand, consider this game repeated twice. If the first player answers (Xi, Yi) upon getting question (x_i, x_j) and the second player answers (Xl, Yl) upon getting question (y_k, y_l), then they win exactly when $i = l$ and that clearly happens with probability $\frac{1}{2}$. Therefore the value of the second power of the game is at least $\frac{1}{2}$.

The twice repeated game has a value smaller or equal to the value of the original game, because we can reduce the twice-played game to the single one by choosing the second set of questions randomly and looking only if the first answers agree. By doing this, we achieve at least the same value in expectancy. Therefore, the original and the twice-played game have both values $\frac{1}{2}$.

This game has some nice properties, like there is just one correct answer for the second player given an answer of the first player. But the constraints still don't satisfy everything from our definition of a Label Cover instance (they are not functions). Another example, for which the constraints are bijections (but on the other hand, the edges aren't chosen uniformly), comes from [12]. I won't prove the results here, just describe the *Odd Cycle game* and state the results.

There is a cycle with n vertices, where n is odd, and an alphabet $\{0, 1\}$. Two players are trying to prove to the verifier that the cycle is 2-colorable. Both vertex sets X, Y has n vertices corresponding to the cycle. The verifier now asks both players with probability $\frac{1}{2}$ for the color of the same vertex (vertices

are chosen uniformly) and checks that the answers are the same. Otherwise, it asks for the color of two adjacent vertices (this is again chosen uniformly from all $2n$ possibilities) and checks that the answers differ. This games' value is $1 - \frac{1}{2n}$, which can be achieved by letting both players have the same coloration of vertices with just one connected pair having the same color (we answer incorrectly only if the verifier checks this pair). Clearly, the value can't be greater in the case when both players answer the same color for every question. On the other hand if they differ with answers for one of the vertices, they also answer incorrectly with probability at least $\frac{1}{2n}$.

The result from [12] states that if the value of the game is $1-\epsilon$, then the value of k-th power of the game is at least $(1-\epsilon^2)^{O(k)}$, which gives a counterexample for small enough ϵ (large enough cycle length).

The counterexample I give for the combinatorial version of the Parallel Repetition theorem in Chapter 4 is conceptually similar to this one. It is also a cycle, though it is defined differently.

2. Combinatorial Version

In this chapter, I explore the Combinatorial Gap Label Cover following [3]. In the first section, I define the combinatorial value of the Label Cover instance and investigate its relationship to the classical value. In the second section, I define the combinatorial version of the Gap Label Cover and show how it follows quickly from the classical version and a conceptually much easier proof from [3] without proving the main lemma. Finally, in the last section, I discuss possible applications to understanding the d-to-1 conjecture. I provide a counterexample for a stronger version of the lemma from which the conjecture would follow easily (that is the only not-yet-published result in this chapter).

2.1 Combinatorial value

Definition 4. *Given a Label Cover instance I (with vertex sets X, Y and domain D) and any $s\colon X \cup Y \to \mathcal{P}(D)$, we say that s is a* combinatorial solution of I *if $f(s(x)) \cap s(y) \neq \emptyset$ for every arrow from $x \in X$ to $y \in Y$ decorated by mapping f.*

The combinatorial value (c-value) of combinatorial solution s *is equal to maximal $|s(z)|$ for $z \in X \cup Y$.*

The combinatorial value (c-value) $\mathrm{cval}(I)$ *is the minimal c-value of s over all combinatorial solutions s of I.*

Remark. The difference between the classical and combinatorial value is that in the classical one, we choose just one element from D for every vertex and try to fulfill as many constraints as possible. In the combinatorial one, we need to satisfy all the constraints, but the fulfillment condition is relaxed. We are allowed to choose more elements for every vertex. For every constraint, it suffices to find just one element from each vertex set so that the constraint is satisfied.

The main benefit of introducing this value is a much easier proof of the Gap Label Cover theorem (though just of a weaker version). On the other hand, many techniques, which help us study the classical value, don't work as well in the combinatorial setting. As we shall see in the following chapters probabilistic version with good properties can be constructed from the classical version, but it is not so nice in the combinatorial setting. Also, the Parallel Repetition theorem doesn't hold for the combinatorial version. Although at least we don't need it in the easy proof of the Combinatorial Gap Label Cover, which is not so in the classical case.

Lemma 10. *For every Label Cover instance I, the following inequality holds*

$$val(I) \geq \frac{1}{cval(I)^2}.$$

Proof. Let us take any combinatorial solution $s : X \cup Y \to \mathcal{P}(Y)$ with vertex sets of size at most d. Let us now sample one element of D using uniform distribution on $s(z)$ for every vertex z independently. For any arrow from x to y the probability of the constraint being satisfied is at least $\frac{1}{|s(x)|}\frac{1}{|s(y)|} \geq \frac{1}{d^2}$. By linearity of expected value, the expected value of the number of satisfied constraints is at least $\frac{m}{d^2}$ where

m is the number of constraints. Clearly, there has to exist a classical solution $c : X \cup Y \to D$, such that its value is at least the expected value of the sampled one. It follows that

$$\mathrm{val}(I) \geq \frac{\frac{m}{d^2}}{m} = \frac{1}{d^2}.$$

Since this holds for any combinatorial solution s, we get $\mathrm{val}(I) \geq \frac{1}{\mathrm{cval}(I)^2}$ as we desired.

□

Remark. The proof could be slightly shorter if we use the probabilistic value defined in the next chapter and Lemma 13. This is one of the reasons why it's nice to have a well-behaving probabilistic value defined in the theory.

There is no reasonable way to formulate any inequality going in the opposite direction where we use just classical and combinatorial values. That is because c-value grows quickly just from very local problems. One can consider a union of two instances, where the smaller one has a large c-value (and small classical value), and the much larger one is solvable. The united instance will then have a classical value close to 1 since that one is computed as a weighted average between the values of the two instances. On the other hand, the c-value will be equal to the c-value of the smaller instance. By taking the solvable instance huge, we can construct instances with arbitrary large c-value and classical value arbitrarily close to 1.

2.2 Combinatorial Gap Label Cover

Theorem 11 (Combinatorial Gap Label Cover)**.** *For every positive integer $d > 1$ there exists a domain D, such that it is NP-hard to distinguish between solvable Label Cover instances over domain D (those with c-value 1) and instances with c-value at least d.*

Proof. [Using Gap Label Cover] To prove this for given d, we take the classical Gap Label Cover theorem with $\epsilon = \frac{1}{d^2}$ (while choosing D corresponding to this ϵ) and show that the Combinatorial Gap Label Cover with constant d uses the same domain D.

Solvable instances in the classical setting are the same as in the combinatorial one. On the other hand, we need to prove that any instance with value at most $\epsilon = \frac{1}{d^2}$ corresponds to an instance with c-value at least d. But that follows from Lemma 10 since

$$\frac{1}{d^2} = \epsilon \geq \mathrm{val}(I) \geq \frac{1}{\mathrm{cval}(I)^2}.$$

So $\mathrm{cval}(I) \geq d$ for every instance I with $\mathrm{val}(I) \leq \epsilon$ as we wanted.

□

Next, I provide a proof using the following technical lemma (which I will not prove) from [3] (or more precisely, its special version with just one layer).

Definition 5. *Given a domain D and set of variables X, the* partial assignment system *of arity k is a collection $(s_K)_{K \in \binom{X}{k}}$, where $s_K \subset D^X$ for all K.*

An assignment $f\colon X \to D$ *is called an* m-solution *of partial assignment system if for all* $M \in \binom{X}{m}$ *there exists* $K_M \supset M$*, such that* $f \in (s_{K_M})|_K$.

Two partial assignment systems $(t_L)_{L\in\binom{X}{l}}$ *and* $(s_K)_{K\in\binom{X}{k}}$ *of arities* $l > k$ *are* consistent *if for all* $K \subset L \subset X, |K| = k, |L| = l$ *it holds that* $s_K \cap (t_L)|_K \neq \emptyset$.

Lemma 12. *For any finite set* D *and numbers* $m, d \in \mathbb{N}$ *there exist* $k < l \in \mathbb{N}$*, such that for any finite set of variables* X *of size at least* l *and two consistent partial assignments systems* $(s_K)_{K\in\binom{X}{k}}$ *and* $(t_L)_{L\in\binom{X}{l}}$ *with set sizes at most* d*, there exists a* m*-solution to the second one. Furthermore the* m*-solution can be computed in polynomial time with respect to* $|X|$.

Remark. The lemma can be used in a situation when we are searching for $f\colon X \to D$ solving some problem, and we have only rough guesses (d options) of what could f look like on sets of sizes k and l. When the guesses are consistent in some sense, we can find global f locally consistent with the guesses.

The downside of this lemma is that we have no control over the numbers k, l. The numbers found in [3] are enormous (not polynomial with respect to $|D|$, m or d). In the next section, I provide a counterexample to the possibility that k, l could be chosen close to each other ($l - k$ being constant). That is sad because otherwise the proof of combinatorial d-to-1 conjecture would follow from this lemma, as we will also see in the next section.

We will now use this lemma to provide a reduction of 3-SAT to the Combinatorial Gap Label Cover with any gap d.

Proof. [Using combinatorial lemma] Let us provide a polynomial reduction of 3-SAT to the Combinatorial Gap Label Cover with gap $d+1$. First we get $k < l$ from the lemma for $D = \{0, 1\}$, d, $m = 3$. We reduce to Label Cover instances first allowing different domain for every vertex, though all domains will have size at most 2^l. We correct this shortcoming at the end of the proof.

Now for any instance S of 3-SAT over finite variable set V, we construct the following Label Cover instance. Vertex sets are $X = \binom{V}{l}$, $Y = \binom{V}{k}$. The domain for any $z \in X \cup Y$ is a subset of $\{0, 1\}^z$ of functions satisfying all constraints of S for which all three variables are in z. Finally, we put one arrow for any $x \in X, y \in Y$, such that $x \supset y$ decorated by the restriction map (which is well defined since if the assignment of variables satisfies all 3-SAT conditions with variables in x, its restriction clearly satisfies all remaining 3-SAT conditions). The size of every domain is at most $\max(2^k, 2^l) = 2^l$. The reduction is polynomial with respect to the length of the 3-SAT instance since we have $O(|V|^l)$ vertices with constant size domains, for which it takes linear time in the number of the 3-SAT constraints to find its domain, and $O(|V|^{lk})$ arrows with constant size functions. If $|V| < l$ or any domain is empty, we already solved our 3-SAT instance, so we don't need to worry about these cases.

If the 3-SAT instance S was solvable, then using its solution as the assignment of variables satisfies all the constraints of the Label Cover instance.

It remains to prove that if S is not solvable, then the c-value of the Label Cover instance is at least $d + 1$. Or in other words, if the c-value of the Label Cover instance is at most d, then S is solvable. Now the lemma is finally useful. We are in a situation where we have two partial assignment systems with set sizes at most d assignments. Since all arrows are restrictions, the Label Cover constraints

say that the partial assignment systems are consistent (since all domains are nonempty). Therefore there exists a 3-solution $f\colon V \to \{0,1\}$.

Now we show that any 3-solution is a full solution of the 3-SAT instance. That follows since every constraint in 3-SAT has at most three variables. Take any such constraint c and set $M, |M| \leq 3$ of its variables. By definition of 3-solution, there exists $L \supset M, |L| = l$, such that one of at most d assignments of L in the Label Cover instance restricted to M is the same as $f|_M$. But all d assignments must satisfy all 3-SAT constraints on variables from L, specially c. Therefore f satisfies c too. Since c was chosen arbitrarily, f satisfies all the constraints, so it's a solution to 3-SAT.

Finally, we choose the domain D as any set of 2^l elements and identify each vertex domain with subset of D in any way. The problem remains that some of the constraints are not functions right now (they don't have any function value for the elements from D which were not identified with any element of the domain of the vertex from X). To sort this out, for any vertex from X with original domain size less than 2^l we choose any of its elements and identify this one with all not-used elements of D. By this we don't change the solvability or c-value in any way, since these elements are for all purposes identical and it doesn't make sense to take any of the copies instead of the original one.

□

2.3 Possible improvements

It would be great to decide the combinatorial version of d-to-1 conjecture as a possible source of ideas to approaching the real one. On the other hand, we can't expect to make much progress on the Unique Games conjecture, since it has no obvious combinatorial version. Every unsolvable Label Cover instance has combinatorial value at least 2. So there is no clear way how to translate the condition that the value of the instance is at least $1 - \epsilon$. If the c-value is smaller than 2, then it's 1 and the decision problem becomes solvable in polynomial time.

The proof of the Theorem 11 using technical Lemma 12 constructs Label Cover instances which are 2^{l-k}-to-1 functions, since the arrows are restrictions of functions from a subset of $\{0,1\}^{\binom{X}{l}}$ to sets of size k. If we could choose $l - k$ to be constant, we would construct a reduction from 3-SAT to the Label Cover with added condition that the mappings are d-to-1 for $d = 2^{l-k}$, which would prove the Combinatorial d-to-1 conjecture.

Unfortunately that's not the case. To provide a counterexample for given $c = l - k$ we consider $m = 2$ (which means the same won't work also for larger m), $D = \{0,1\}$ (again the counter-example would work with any larger set) and $d = c + 1$.

Consider the following construction. For any $l > c$ take variable set $V = \{0, 1, \ldots, l\}$ ($|V| = l + 1$). The t_L sets from the lemma have only one function h, such that for $L = V \setminus \{i\}$ it holds that $h(j) = 0$ for $j < i$ and $h(j) = 1$ for $j > i$.

Sets s_K have at most $d = c+1$ elements. Since $|K| = k = l - c = |V| - (c+1)$, we can write $K = V \setminus \{i_0, \ldots, i_c\}$, where $i_0 < i_1 < \cdots < i_c$. And we define at

most $c+1$ functions $g_\iota(j) = 0$ if $j < i_\iota$ and $g_\iota(j) = 1$ otherwise for $\iota \in \{0, 1, \ldots, c\}$ (g functions are not all different if K misses some consecutive numbers).

This is consistent since whenever $K \subset L$, it means that $V \setminus L \subset V \setminus K$ or with the notation from previous two paragraphs $i \in \{i_0, \ldots, i_c\}$, say $i = i_\alpha$. Clearly $g_\alpha = h|_K$, which means our construction satisfies the assumptions of the lemma.

But there are no 2-solutions. Any 2-solution f has to have $f(0) = 0$ and $f(l) = 1$ since there is no L for which the value at 0 would be equal to 1 or the value at l be equal to 0. Therefore there has to be j, such that $f(j) = 0$ and $f(j + 1) = 1$, but these two elements are not consistent with any L, since the functions in L have the same value at any two consecutive numbers. Therefore no such 2-solution f can exist.

This counterexample is clearly not solvable. We have just one option to choose for the larger sets and this option forces the value of its subsets of size k. Different supersets of size l of one set of size k force conflicting values for it.

It would be nice to produce CSP instances via the process from the combinatorial proof of Theorem 11, which are not solvable by easy polynomial algorithms and the lemma with $l - k$ constant does not help with creating the gap. The first step to this could be to produce counterexample Label Cover instances which would be arc consistent (the provided definition is equivalent to the one found in [4]).

Definition 6 (Arc Consistency)**.** *A CSP instance $(V, (D_v)_{v \in V}, S)$ with different domains D_v for every variable v is* arc consistent *if for every constraint $((x_1, \ldots, x_k), R) \in S$ the following holds. For any $i \in \{1, \ldots, k\}$ and $v \in D_{x_i}$, there exists $(v_1, \ldots, v_k) \in R$ with $v_i = v$.*

The arc consistency says only that we can not figure out anything about the solvability of CSP instance by looking just on individual constraints and not considering any interaction between them.

Whenever a CSP instance is not arc consistent, we can (without changing the solvability of an instance) decrease the size of at least one variable set and we can find this one in polynomial time with respect to number and sizes of constraints and domain sizes. By repeating this process we either arrive to arc consistent instance or we find out that the instance is not solvable by deleting the whole domain for one of the variables which is used in the constraints. The latter case happens with the counterexample described here. Unfortunately, I was not able to figure out a better counterexample.

3. Probabilistic Combinatorial Version

In this chapter, I propose a new definition of what I call *probabilistic combinatorial value* (or pc-value for short). The inspiration comes from a similar relaxation of the definition, which was used to prove PCP in the classical setting in [7].

In the classical setting, the two values are the same for reasons I will give in the first section. Unfortunately, that is not the case for the combinatorial version (the counterexample is given in the second section). However, I was at least able to prove a correspondence for Unique Games. The proof is shown in the third section. In the last section, I consider a probabilistic combinatorial version of the Gap Label Cover theorem and show that it lies between the classical and combinatorial one in terms of strength.

I tried to prove the pc-version of the Gap Label Cover also with the tools used in [3], but I wasn't successful.

3.1 Classical probabilistic value

Definition 7 (Classical probabilistic value of a Label Cover instance)**.** *For a Label Cover instance I (with vertex sets X, Y, domain D and set of arrows A, each arrow $a \in A$ starting at $a_x \in X$ and ending at $a_y \in Y$ decorated by mapping f), its* classical probabilistic value (p-value) $\mathrm{pval}(I)$ *is equal to maximum of*

$$\mathbb{E}_{a \in A}\left[\sum_{v \in D} p_{a_x}(v) p_{a_y}\left(f(v)\right)\right]$$

over all functions $p\colon X \cup Y \to [0,1]^D$, such that $\sum_{d \in D} p_v(d) = 1$ for all $v \in X \cup Y$, where the expected value is took using the uniform distribution over all arrows.

For any function $p\colon X \cup Y \to [0,1]^D$, we say that the p-value of p *is equal to the value of the expression above.*

Remark. This definition is the same as the classical one, except for allowing choosing a probabilistic distribution of values for every vertex instead of just a single value. Then we ask the same question: What's the maximum probability of constraint being satisfied if we sample the constraint randomly from the uniform distribution on the arrows?

Theorem 13. *For any Label Cover instance I it holds that*

$$val(I) = pval(I).$$

Proof. Clearly $\mathrm{pval}(I) \geq \mathrm{val}(I)$ since if we take any classical solution, we can reformulate this solution as taking the chosen element for a given vertex with probability 1 and the remaining ones with probability 0. That gives us a probabilistic solution with the same p-value.

On the other hand, if we take any probabilistic solution p and for every vertex $v \in X \cup Y$ independently, we sample a random element from the distribution p_v,

the expected value of the classical solution we get is equal to the p-value of p. Since the expected value is equal to the p-value of p, there has to be at least one classical solution, which value is at least the p-value of p. Since p was chosen arbitrarily, this gives us the opposite inequality.

□

3.2 Probabilistic combinatorial value

Definition 8 (Probabilistic combinatorial value of a Label Cover instance)**.** *For a Label Cover instance I (with vertex sets X, Y, domain D and set of arrows A, each arrow $a \in A$ starting at $a_x \in X$ and ending at $a_y \in Y$ decorated by mapping f), its* probabilistic combinatorial value (pc-value) $\mathrm{pcval}(I)$ *is equal to the maximum of*

$$\min_{a \in A} \left(\sum_{d \in D} p_{a_x}(d) p_{a_y}(f(d)) \right)$$

over all functions $p\colon X \cup Y \to [0,1]^D$, such that $\sum_{d \in D} p_v(d) = 1$ for all $v \in X \cup Y$, where the expected value is took using the uniform distribution over all arrows.

Any $p\colon X \cup Y \to [0,1]^D$ will be called a pc-solution of I *and we say that the* pc-value of p *is equal to the value of the expression above. Also we say that the* pc-value of arrow a *is equal to the expression inside of the minimum function.*

This definition looks almost identical to the classical probabilistic version, except using minimum over all arrows rather than the expected value. Before commenting more on the motivation for this definition and the relationship between pc and combinatorial version, we prove a simple lemma connecting classical value with the pc one, which will be useful later.

Lemma 14. *For a label cover instance I, the following holds:*

$$pcval(I) \leq val(I).$$

Proof. Let's take any pc-solution p, then the p-value is computed as the expected value instead of the minimum, so by using the same solution we get $\mathrm{pcval}(I) \leq \mathrm{pval}(I)$. Since $\mathrm{pval}(I) = \mathrm{val}(I)$ by Theorem 13, we conclude that $\mathrm{pcval}(I) \leq \mathrm{val}(I)$.

□

In the combinatorial setting, it is a little bit less clear how the pc version relates to the combinatorial one. We can provide the intuition with the following lemma and its proof.

Lemma 15. *For any Label Cover instance I the following holds:*

$$pcval(I) \geq \frac{1}{cval(I)^2}.$$

Proof. Take any combinatorial solution $s\colon X \cup Y \to \mathcal{P}(D)$ of I with c-value c. Then define the probabilistic combinatorial solution by $p_v(d) = \frac{1}{|s(v)|}$ if $d \in s(v)$

and $p_v(d) = 0$ otherwise. Since s is a combinatorial solution, for any arrow $v \to w$ decorated by f, there is at least one d, such that $d \in s(v) \wedge f(d) \in s(w)$. Since $p_v(d) = \frac{1}{|s(v)|} \geq \frac{1}{c}$ and the same holds for $p_w(f(d))$, the probabilistic combinatorial value of this instance is at least $\frac{1}{c^2}$.

□

It would be nice to have inequality in the opposite direction, which doesn't depend on anything other than pc and c-values. The following counterexample shows that no such estimate is possible – we construct an instance with pc-value $\frac{1}{2}$ and c-value arbitrary large (of course, we need to take a sufficiently large domain to do this).

Claim 16. *For any even natural n, there exists a Label Cover instance I with domain set D of size n, such that $pcval(I) \geq \frac{1}{2}$, but $cval(I) \geq \frac{n}{2} + 1$.*

Proof. Take $Y = \binom{D}{\frac{n}{2}}$ and $X = D^Y$ and choose two special elements $0 \neq 1 \in D$. Now for any pair $(x, y) \in X \times Y$ take one arrow from x to y decorated by the following function f.

If $x(y) \neq 0$: $f(d) = 0$ if $d \in y$ and $f(d) = x(y)$ otherwise. And if $x(y) = 0$, $f(d) = 1$ if $d \in Y$ and $f(d) = 0$ otherwise.

With mappings defined like this, we can easily construct a pc-solution p with pc-value $\frac{1}{2}$ by defining $p_x(d) = \frac{1}{n}$ for every $x \in X$ and $d \in D$, and $p_y(0) = 1$ and 0 otherwise for every $y \in Y$. Since all the arrows map exactly one half of elements from D into 0, we get a pc-value of $\frac{n}{2} \cdot \frac{1}{n} = \frac{1}{2}$ for every arrow and therefore also the pc-value of p is such.

On the other hand, if we want to construct a combinatorial solution s with a c-value less than n, we need to leave out at least one element in every vertex of Y. Choosing one such element from every Y gives us a function from Y to D or, in other words, an element of $D^Y = X$, which I will call x_0. For every $y \in Y$, the arrow from x_0 to y forces x to have an element from y in its element set. Therefore it needs to have at least $\frac{n}{2} + 1$ elements in its element set $s(x_0)$, otherwise we would look at any $\frac{n}{2}$-element subset y_0 of $D \setminus s(x_0)$ and the constraint from x_0 to y_0 wouldn't be satisfied.

Therefore $\text{cval}(I) \geq \min(n, \frac{n}{2} + 1) = \frac{n}{2} + 1$, as we wanted to prove.

□

This counterexample uses mappings that compress the domain D into just two-element images every time. In the next section, we prove that if we control the compression (allow only d-to-1 mappings for fixed d), we can also get inequality in the other direction. Particular interest can be given to the case of $d = 1$, which corresponds to the Unique Games.

3.3 Pc-value for Unique Games

Theorem 17. *For any d-to-1 Label Cover instance I, the following holds:*

$$cval(I) \leq \frac{d+1}{pcval(I)}.$$

Proof. Let's take any pc-solution p of I and construct function $s\colon X \cup Y \to \mathcal{P}(D)$ by letting $s(z) = \left\{e; p_z(e) \geq \frac{\text{pcval}(I)}{d+1}\right\}$ for any $z \in X \cup Y$.

It's clear that $|s(z)| \leq \frac{d+1}{\text{pcval}(I)}$ for every z since the values of p_z has to add to one over all elements of D. The only thing reamining is to prove that s is a combinatorial solution to the d-to-1 label cover instance I.

Assume for contradiction that s is not a combinatorial solution. That means that we can choose an arrow from x to y decorated by d-to-1 mapping f, such that there is no $e \in D$ for which it would hold $e \in s(x) \wedge f(e) \in s(y)$ or in other words $p_x(e) \geq \frac{\text{pcval}(I)}{d+1} \wedge p_y(f(e)) \geq \frac{\text{pcval}(I)}{d+1}$. Let us estimate the pc-value of this arrow.

For any numbers $a \in [0,1], b \in [0,c]$ it holds that $ab \leq ac \leq (a+b)c$ with strict inequality in the last estimate if both a, b are nonzero. Since the last expression is symmetric in a, b, we get that $ab \leq (a+b)c$ holds also if we don't know which one of a, b lies in which of the intervals. In our case we know that every summand in the sum used to compute the value of an arrow is of this type with $c = \frac{\text{pcval}(I)}{d+1}$ and that $p_y(e)$ appears in maximally d instances. Therefore we get

$$\begin{aligned}\sum_{e\in D} p_x(e)p_y(f(e)) &< \sum_{e\in D} \frac{\text{pcval}(I)}{d+1}(p_x(e) + p_y(f(e)) \leq \\ &\leq \frac{\text{pcval}(I)}{d+1}\left(\sum_{e\in D} p_x(e) + d\sum_{e\in D} p_y(e)\right) = \text{pcval}(I),\end{aligned}$$

which is a contradiction. (The first inequality is strict since at least one of the summands has to be non-zero. Otherwise, the pc-value of the arrow must be 0. That is also a contradiction with $\text{pcval}(I)$ being non-zero.)

□

Corollary. For any Unique Games Label Cover instance I, the following holds:

$$\text{cval}(I) \leq \frac{2}{\text{pcval}(I)}.$$

3.4 Gap Label Cover Theorem in the pc setting

In this section, we prove that the pc version of the Gap Label Cover theorem lies between the classical and the combinatorial version. First, we prove the pc version by showing that it follows from the classical version. Next, we show how the combinatorial version follows from the pc version.

So far, the only proof of the pc version of the gap label cover theorem known to me uses the classical version. It would be nice to have a more straightforward proof, as is the case with the combinatorial version, but I couldn't find one. At the end of this section, I discuss why the pc version doesn't follow easily from the combinatorial Lemma 12.

Theorem 18. *For any $1 > \epsilon > 0$ there exists domain D, such that it is NP-hard to decide if a given Label Cover instance I with domain D is solvable or has pc-value at most ϵ.*

Proof. The proof is similar to the one of Theorem 11. The reduction between problem instances is just an identity.

Instance I is solvable in the classical sense if it's solvable in the pc one. So we identify solvable instances correctly.

On the other hand, any instance I with a value less than ϵ also has pc-value less than ϵ by Lemma 14. So we respond correctly also to these instances when we use an algorithm for the pc case.

□

In a similar way the Combinatorial Gap Label Cover theorem follows from the pc one. We can show that ϵ version of pc Gap Label Cover implies $\sqrt{\frac{1}{\epsilon}}$ version of Combinatorial Gap Label Cover. By Lemma 15 we know that if we have any instance I with $\text{pcval}(I) \leq \epsilon$, then

$$\text{cval}(I) \geq \sqrt{\frac{1}{\text{pcval}(I)}} \geq \sqrt{\frac{1}{\epsilon}}.$$

So also in this case we can distinguish between pc-values by the algorithm which distinguishes c-values.

Unfortunately, I could not prove the pc version using only combinatorial Lemma 12 or something similar. It is not straightforward since we don't generally have any inequality between c-value and pc-value going in the direction we need (because of Claim 16). We have one we could use well in the case of d-to-1 Label Cover instances (Theorem 17). However, from our discussion in Section 2.3 we know that we can't have a version of the lemma so that the Label Cover instance created from the 3-SAT has d-to-1 property. The validity of the Combinatorial d-to-1 conjecture would imply the pc version of the Gap Label Cover theorem easily.

4. Parallel Repetition Theorem

In this chapter, I first repeat the Parallel Repetition theorem without proof, but this time just for the Label Cover problem in its language. I won't use the language of Two-prover One-round Games in this chapter.

In the rest of the chapter, I provide a counterexample to the combinatorial (and probabilistic combinatorial) version of the theorem. That is the main original result of this thesis.

Definition 9. *Let n be a positive integer and I a Label Cover instance with finite domain D, variable sets X, Y, set of arrows A and a function $F : A \to D^D$. Then I^n (n-th power of I) is a Label Cover instance with finite domain D^n, variable sets X^n, Y^n, set of arrows A^n and a function $F^n : A^n \to (D^n)^{D^n}$) defined by $F^n((a_1, \ldots, a_n))((d_1, \ldots, d_n)) = (F(a_1)(d_1), \ldots, F(a_n)(d_n))$.*

Theorem 19 (Parallel Repetition Theorem)**.** *For every finite set D, there exists a function $\alpha : [0, 1) \to [0, 1)$, such that if an unsolvable Label Cover instance I over domain D has value ϵ, then for every positive integer n the instance I^n has value at most $\alpha(\epsilon)^n$.*

4.1 Combinatorial version

The Parallel Repetition theorem can be stated in the combinatorial setting as follows:

For every finite set D, there exists a function $\beta : \mathbb{N} \setminus \{1\} \to (1, \infty)$, such that if an unsolvable Label Cover instance I over domain D has c-value $N \geq 2$, then for every positive integer n the instance I^n has value at least $\beta(N)^n$.

The probabilistic combinatorial version looks exactly like the standard theorem, except for using the pc-value instead of the normal value.

Since $\mathrm{val}(I) \geq \frac{1}{\mathrm{cval}(I)^2}$ for every Label Cover instance I by Lemma 10, it holds that

$$\mathrm{cval}(I^n) \geq \sqrt{\frac{1}{\mathrm{val}(I^n)}} \geq \sqrt{\frac{1}{\alpha(\mathrm{val}(I))}}^n .$$

Therefore to find a counterexample to the combinatorial version, looking only at one specific instance surely is not enough. That's because for unsolvable I we have $\alpha(\mathrm{val}(I)) < 1$. Looking just at this instance, we can choose $\beta = \sqrt{\frac{1}{\alpha(\mathrm{val}(L))}} > 1$ and satisfy all conditions.

However, the counterexample can be constructed. In this section, I'll describe a family of instances with c-value 2 over domain $\{0, 1\}$, which will force $\beta(2) < 1 + \epsilon$ for arbitrary $\epsilon > 0$. From that, we would have $\beta(2) \leq 1$, which is not allowed.

Definition 10. *Cyclic instance C_k (of length $2k$) is the following Label Cover instance over domain $\{0, 1\}$. Both sets of variables X, Y have k elements. There are $2k - 1$ arrows which are decorated by the identity mapping: between x_i and y_i for all $1 \leq i \leq k$ and between x_{i+1} and y_i for all $1 \leq i \leq k - 1$. Finally, there is one arrow decorated by mapping switching zero and one between x_1 and y_k.*

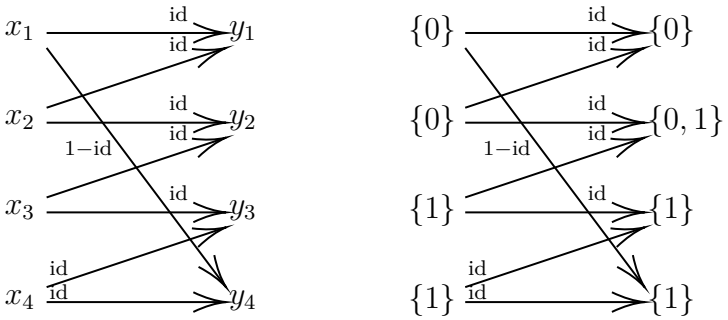

Figure 4.1: Instance C_4 with one possible combinatorial solution with c-value 2.

Remark. C_k is an unsolvable instance since the first $2k-1$ arrows force all variables to have the same value, which makes the last constraint invalid. On the other hand, by choosing the same value for all variables, we satisfy all but one constraint, so $\text{val}(C_k) = \frac{2k-1}{2k}$.

Now it is almost clear that $\text{cval}(C_k) = 2$. This follows from unsolvability and the domain having only 2 elements (both can be chosen for every vertex).

One can also notice that it suffices to choose both values from $\{0,1\}$ only for one variable. Constraints using this variable are clearly satisfied every time. The rest of them forms a path of bijections we can fulfill easily. That means that instances of this family are almost solvable in the sense of having the standard value close to 1 and needing two values for only one variable in the combinatorial setting. Therefore they form a natural candidate for a counterexample to the combinatorial version of the Parallel Repetition theorem. One can hardly find a simpler family with the stated properties.

Also note that $\text{pcval}(C_k) = \frac{1}{2}$. One inequality follows from putting probabilities $\frac{1}{2}$ to both values on all vertices. Then every arrow has value $2 \cdot \left(\frac{1}{2}\right)^2 = \frac{1}{2}$, which means that the pc-value of this instance is at least $\frac{1}{2}$.

To prove the opposite inequality we choose any pc-solution p and color red vertices v, for which $p_v(0) \geq p_v(1)$ and blue the other ones. Either the vertices connected by the only non-identity arrow have the same color or there exist two vertices connected by identity with opposite colors (because the color has to change even number of times in the cycle). In both of these cases the value of the arrow is given by a computation of type $pq + (1-p)(1-q)$, where $p \leq \frac{1}{2}$ and $q \geq \frac{1}{2}$. Denoting $a = \frac{1}{2} - p$, $b = q - \frac{1}{2}$ (both of them being non-negative), we get

$$pq + (1-p)(1-q) = \left(\frac{1}{2} - a\right)\left(\frac{1}{2} + b\right) + \left(\frac{1}{2} + a\right)\left(\frac{1}{2} - b\right) = \frac{1}{2} - 2ab \leq \frac{1}{2}.$$

This shows that the pc-value can't be greater than $\frac{1}{2}$ as we wanted.

Theorem 20. *For every positive integer n there exists positive integer k, such that $\text{cval}(C_k^n) \leq n+1$.*

Remark. This inequality will be enough to disregard the combinatorial version of the Parallel Repetition theorem, but I don't claim to know the value of the instance exactly. We know that for $n = 1$, the c-value is 2, and "minimal" solutions use 2 options only on one vertex, which can be chosen arbitrarily. By

taking a product of two "minimal" solutions having the vertices with two values in sets on different sides, we get a combinatorial solution to C_{2k}^2 with c-value 2. That's smaller than the estimate from the theorem. (Similarly, we get a combinatorial solution with c-value 4 for $n = 4$ by taking the second power of this combinatorial solution.)

Using this theorem, the main result of this chapter follows quickly.

Theorem 21. *The Parallel Repetition theorem doesn't hold in combinatorial or probabilistic combinatorial versions.*

Proof. Assume for contradiction that there exists a function $\beta : \mathbb{N}\setminus\{1\} \to (1, \infty)$ from the formulation of the combinatorial version. Therefore $\mathrm{cval}(I^n) \geq \beta(2)^n$ for every instance I of combinatorial value 2. By Theorem 20 there exists an instance I for which $\mathrm{cval}(I^n) \leq n + 1$ for every n. It follows $\beta(2)^n \leq n + 1$. Since every exponential function with base larger than one grows faster than any linear function for large enough n, it has to be $\beta(2) \leq 1$, which is the desired contradiction.

Probablistic combinatorial version is not much more difficult. From Lemma 15 we get

$$\mathrm{pcval}(C_k^n) \geq \frac{1}{\mathrm{cval}(C_k^n)^2} \geq \frac{1}{(n+1)^2}.$$

From remark below Definition 10 we know that $\mathrm{pcval}(C_k) = \frac{1}{2}$ for every natural k.

Similarly as in the combinatorial case we get to the inequality

$$\frac{1}{(n+1)^2} \leq \mathrm{pcval}(C_k^n) \leq \alpha \left(\frac{1}{2}\right)^n,$$

which can't hold for every n unless $\alpha\left(\frac{1}{2}\right) \geq 1$ since the exponential function with smaller base decreases faster than any rational one for large enough n.

□

In the next section, I spend considerable time proving Theorem 20.

4.2 Cubes in a torus

The first thing needed for the proof is translating the problem into a geometrical setting. After I have the translation, I first forget the discrete nature of the problem and solve a more accessible continuous version. Finally, I argue that by choosing a sufficiently large length of the cycle $2k$, a discretization of the continuous problem gives the construction I need.

How can one think well about the powers of the Cyclic instances? Let's build the following picture. Let us number the vertices from both sides of the Cyclic instance by elements of $\mathbb{Z}_{2k}$ so that an arrow connects vertices that differ by one, and the only non-identity arrow goes between $2k-1$ and 0. Note that the vertices in X are precisely those with even numbers and the vertices in Y are the odd ones.

Let us denote the vertex sets of the C_k^n by X_k^n and Y_k^n. With the mapping from the previous paragraph, we can identify these sets with the following subsets

of $\mathbb{Z}_{2k}^n$. Set X_k^n consists of n-tuples with all coordinates even, and Y_k^n consists of the ones with all coordinates odd. The arrows go between vertices that differ by exactly one in each coordinate. They have identity maps in all coordinates but the ones where the numbers are 0 and $2k-1$.

Let us picture $\mathbb{Z}_{2k}^n$ as an n-dimensional torus by drawing a unit hypercube with opposite hypersides glued together, divided to $(2k)^n$ hypercubes of side length $\frac{1}{2k}$. Now consider only subset $X_k^n \cup Y_k^n$. We are left with only $2k^n$ hypercubes. No two of them share more than one point. The pairs of cubes which share a corner are exactly the ones between which an arrow exists. And the arrow is the non-identity one in a specific coordinate iff the shared corner of the pair lies on the boundary of the large unit hyper-cube in the direction corresponding to the coordinate.

For example, when $n = 1$ we picture just a segment of unit length divided to $2k$ smaller segments of the same length while remembering that the opposite ends are the same. When $n = 2$ we picture squares of just one color on a chessboard (and again, we remember that the chessboard is not just a square but a torus in reality).

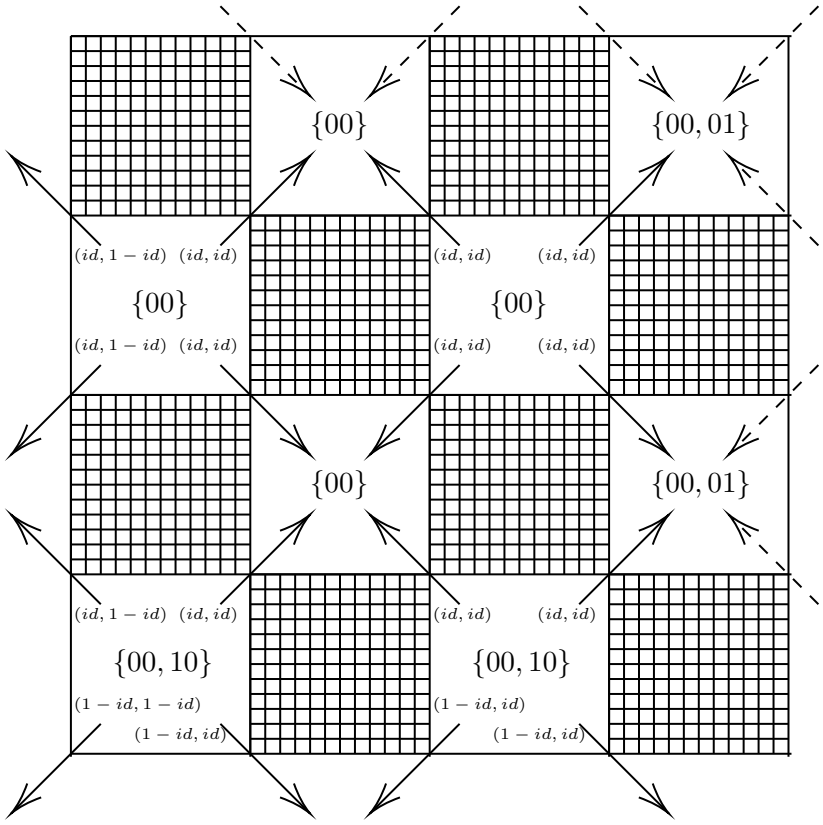

Figure 4.2: Instance C_2^2 pictured as squares in a torus with a combinatorial solution with c-value 2.

Now we need to put a set of maximally $n+1$ n-bit binary numbers into each hypercube, such that every constraint is satisfied. That means that every pair of neighboring hypercubes has a non-empty intersection of the solution sets after switching bits in one of them in coordinates corresponding to the directions they neighbor through the boundary of the unit hypercube.

Now with this picture in mind, let me formulate a continuous problem from

which we later construct the counterexample.

4.2.1 Continuous version

Note that hypercube has a standard numbering of vertices by binary numbers, which we get by using $[0,1]^n$ hypercube and taking the coordinates of vertices. We use this numbering in the following claim.

Claim 22. *The n-dimensional unit hypercube can be cut into 2^n hyperprisms numbered by n-bit binary numbers, such that no point in the interior of the hypercube lies on the boundary of more than $n+1$ prisms, every vertex lies in the hyperprism with the same binary number and the point $(x_1, \ldots, x_{j-1}, 0, x_{j+1}, \ldots, x_n)$ lies on the boundary of prism b iff the point $(x_1, \ldots, x_{j-1}, 1, x_{j+1}, \ldots, x_n)$ lies on the boundary of prism $b \oplus 2^j$ for every $j \in \{1, \ldots, n\}$.*

Remark. Conceptually and a little bit informally, we are trying to achieve the following: Take an n-dimensional torus and canonical generators e_i of its fundamental group $\mathbb{Z}^n$. Next, make for every canonical generator a piece-wise hyperplanar cut (which has no boundary) into n-dimensional torus (cuts can overlap) such that we still have a connected manifold (with a boundary from the cuts). Moreover, every closed path p_i representing e_i has to go through i-th cut an odd number of times and through the others even number of times. Finally, we want to do this such that if we took any small enough ball in the torus, the cuts divide it into maximally $n+1$ different components.

The cuts represent places where we need to use more values (corresponding to neighboring hyperprisms) as part of our solution (in the discrete version, we do this for all hypercubes with a center close enough to the cut).

When completing the discrete case construction, only the proof of no point lying in more than $n+1$ hyperprisms will be used. The fact that the opposite sides of the unit hypercube match is not used there. The proof can be completed without noting this fact. However, I consider it an interesting and important property of the construction, which deserves a proof here.

Proof. Our construction works in the following way. In n dimensions the hyperprisms are given by the system of n inequalities. Hyperprism corresponding to a binary number $\overline{b_1 \ldots b_n}$ is defined by

$$f_i(x_1, \ldots, x_i) - \frac{f_{i-1}(b_1, \ldots, b_{i-1})}{2} \lesseqgtr \frac{1}{2},$$

where the symbol is "$\leq$" if $b_i = 0$ and "$\geq$" otherwise and

$$f_i(a_1, \ldots, a_i) = \sum_{l=1}^{i} \frac{a_l}{2^{i-l}}.$$

I use induction to prove that the construction satisfies the conditions from the statement. We also show that if the i-th coordinate of the point is 0, then the point is only in prisms with i-th bit being 0. And the same for 1. From this added condition the part of the claim that the corners of hypercube lie in the hyperprisms with the same bit number follows immediately.

Note that for a binary number b in $n+1$ dimensions the first n inequalities are the same as for b without the last bit in n dimensions. Essentially, when adding a

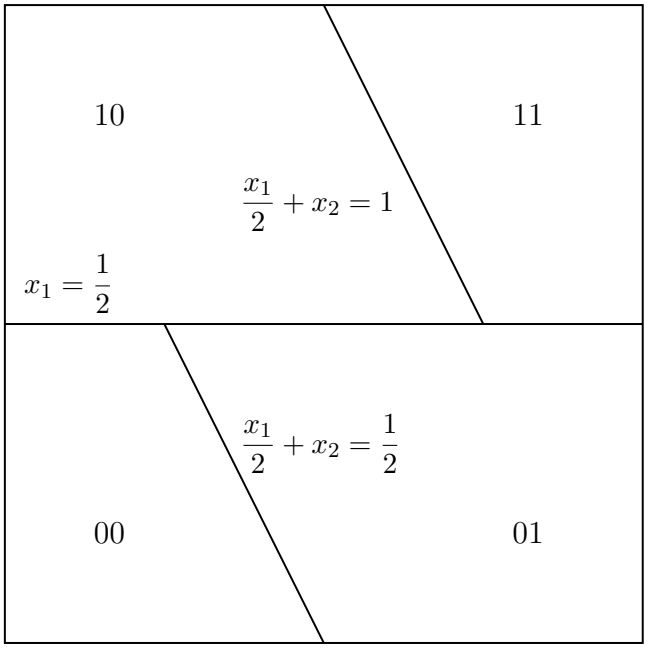

Figure 4.3: Continuous construction for $n = 2$.

dimension, we first multiply all hyperprisms by a unit interval in that dimension and then divide each into two using a hyper-plane. All the hyperplanes used for the division have the same normal. After proving the claim that the opposite sides match we can picture this as using a part of just one hyperplane cutting the hypercube glued into a torus.

For $n = 1$ the construction looks like this:

$$0\colon x_1 \leq \tfrac{1}{2}$$
$$1\colon x_1 \geq \tfrac{1}{2}$$

These two segments clearly satisfy the needed conditions. Point (0) lies only in segment 0, point (1) only in segment 1. Because off that, opposite ends of the unit segment lie in different smaller segments. Finally, no point inside the segment lies on more than two smaller segments.

Consider now points $(x_1, \ldots, x_{j-1}, 0, x_{j+1}, \ldots, x_{n+1})$ and $(x_1, \ldots, x_{j-1}, 1, x_{j+1}, \ldots, x_{n+1})$. We need to show that the former one lies only in hyperprisms with j-th bit equal to 0, the latter only in hyperprisms with j-th bit equal to 1, and that all the other inequalities work on the same side with these two points.

For $j \leq n$ we know that the first first n inequalities behave well from the induction assumption. That means that the points lie in the hyperprisms with correct bits on j-th position. We need to check that the new inequality works the same for both of them, which is just checking that

$$f_{n+1}(x_1, \ldots, x_{j-1}, 0, x_{j+1}, \ldots, x_{n+1}) - \frac{f_n(b_1, \ldots, b_{j-1}, 0, b_{j+1}, \ldots b_{n+1})}{2} \leq \frac{1}{2} \iff$$
$$f_{n+1}(x_1, \ldots, x_{j-1}, 1, x_{j+1}, \ldots, x_{n+1}) - \frac{f_n(b_1, \ldots, b_{j-1}, 1, b_{j+1}, \ldots b_{n+1})}{2} \leq \frac{1}{2}.$$

Since the second inequality is the same as the first one except for adding $\frac{1}{2^{n+1-j}}$ in the first expression and subtracting the same number in the second one, the equivalence holds.

We are left with the case $j = n+1$. In this case we don't expect any change in the direction of the first n inequalities when switching zero and one in $(n+1)$-th coordinate. That is trivially true, since the $(n+1)$-th coordinate is not represented in those.

The only thing that remains is to show that if x_{n+1} is 0, then $b_{n+1} = 0$, and the same with 1. For the "zero" case we compute

$$f_{n+1}(x_1, \ldots, x_n, 0) - \frac{f_n(b_1, \ldots, b_n)}{2} = \sum_{l=1}^{n} \frac{x_l - b_l}{2^{n+1-l}} \leq \sum_{l=1}^{n} \frac{1}{2^{n+1-l}} < \frac{1}{2}.$$

And similarly for the "one" case

$$f_{n+1}(x_1, \ldots, x_n, 1) - \frac{f_n(b_1, \ldots, b_n)}{2} = 1 + \sum_{l=1}^{n} \frac{x_l - b_l}{2^{n+1-l}} \geq 1 + \sum_{l=1}^{n} \frac{-1}{2^{n+1-l}} > 1 - \frac{1}{2} = \frac{1}{2}.$$

Now we know that the cuts match properly on the hypersides of the cube and the last thing to check is that no point inside the hypercube lies in more than $n+1$ hyperprisms.

Let us choose any such point $(y_1, \ldots, y_{n+1})$. From the induction assumption we know that maximally $n+1$ combinations of the first n inequalities hold. Let us denote the set of n-dimensional hyper-prisms from our construction in which point $(y_1, \ldots, y_n)$ lies by C. Since f_n restricted to $\{0,1\}^n$ is an injective function (its value is equal to the binary number we get from reading the arguments backwards divided by 2^{n-1}), the equality of $f_{n+1}(y_1, \ldots, y_{n+1})$ and $\frac{f_n(c_1,\ldots,c_n)}{2} + \frac{1}{2}$ can hold for maximally one prism $\overline{c_1 \ldots c_n}$ from C. If the equality holds, the point $(y_1, \ldots, y_{n+1})$ lies in both $\overline{c_1 \ldots c_n 0}$ and $\overline{c_1 \ldots c_n 1}$ prisms. Otherwise it lies in only one of them. Therefore the selected point in $[0,1]^{n+1}$ lies in maximally $|C| + 1 \leq n+2$ hyperprisms and the proof of the claim is completed.

□

4.2.2 Discrete version

In this subsection I show that for any n we can choose suitable ϵ, such that if we change the inequalities in the construction from the last proof to be of type $X \leq Y + \epsilon$ and $X \geq Y - \epsilon$ instead of just $X \leq Y$ and $X \geq Y$, the claim still holds. After this I conclude that one can choose sufficiently large k, so that if we include n-bit binary value b in the values for hypercube v iff center of v lies in hyperprism b, the construction will provide a combinatorial solution to C_k^n instance. And this solution is of c-value smaller or equal to $n+1$, since center of any hypercube can't lie in more than $n+1$ hyperprisms.

Claim 23. *The construction from the proof of Claim 22 with all inequalities relaxed by sufficiently small $\epsilon > 0$ depending on the dimension n still satisfies that each point lies in at most $n+1$ hyperprisms.*

Proof. For the induction step from n to $n+1$ to hold, we can choose $\epsilon = \frac{1}{3 \cdot 2^n}$. That's because the adjacent values on the right hand side differ by $\frac{1}{2^n}$, so if we choose $\epsilon < \frac{1}{2^{n+1}}$, we still keep the property that there exists maximally one hyperprism $\overline{c_1 \dots c_n}$ from C, such that the chosen point lies in both $\overline{c_1 \dots c_n 0}$ and $\overline{c_1 \dots c_n 1}$. Our chosen value of ϵ decreases in each induction step, which means that for n dimensions we can choose $\epsilon_n = \frac{1}{3 \cdot 2^{n-1}}$ and we get the desired property. □

Claim 24. *For every n we can choose sufficiently large k, such that the construction from Claim 23 gives rise to a combinatorial solution of C_k^n instance of c-value at most $n+1$ in the following way. n-bit binary number b lies in a solution set of vertex v when the center of hypercube associated with v lies in the hyperprism associated with b.*

Proof. The only thing we need to show is that such defined valuation of C_k^n is indeed a solution. We show that for any pair of not-through-the-boundary adjacent hypercubes G, H, there is a hyperprism b, such that centers g, h of G,H lie in b. More generally g lies in b and h lies in $b \oplus c$, where c has ones for the directions in which G and H neighbor through the boundary of unit hypercube ($\oplus$ denotes xor).

We will construct the hyperprism b inductively coordinate after coordinate. When including i-th coordinate (for $i \in \{1, \dots, n\}$), we need to decide on the b_i, so that the i-th inequality is satisfied in conditions for both $g \in b$ and $h \in b \oplus c$. In order to have the existence of this b_i we need to show that both of the following inequalities hold either with taking the same (up or down) options everywhere.

For $c_i = 0$:

$$f_i(g_1, \dots, g_i) - \frac{f_{i-1}(b_1, \dots, b_{i-1})}{2} \lesseqgtr \frac{1}{2} \pm \epsilon_n$$
$$f_i(h_1, \dots, h_i) - \frac{f_{i-1}(b_1 \oplus c_1, \dots, b_{i-1} \oplus c_{i-1})}{2} \lesseqgtr \frac{1}{2} \pm \epsilon_n$$

If this wouldn't be the case, the upper inequality would have to be false in one case and the lower one in the second. (The centers of the hyper-cubes would have to lie outside of the strip on different sides). Assume for contradiction that it's so and wlog assume that both of the following hold:

$$f_i(g_1, \dots, g_i) - \frac{f_{i-1}(b_1, \dots, b_{i-1})}{2} < \frac{1}{2} - \epsilon_n$$
$$f_i(h_1, \dots, h_i) - \frac{f_{i-1}(b_1 \oplus c_1, \dots, b_{i-1} \oplus c_{i-1})}{2} > \frac{1}{2} + \epsilon_n$$

So necessarily (after plugging in for f functions and subtracting):

$$\sum_{l=1}^{i} \frac{(h_l - g_l) - (b_l \oplus c_l - b_l)}{2^{i-l}} > 2\epsilon_n.$$

Since G, H are adjacent, it holds that

- $|h_j - g_j| = \frac{1}{2k}$ if $c_j = 0$,

- $1-(h_j-g_j)=\frac{1}{2k}$ if $c_j=1$ and $b_j=0$,
- $1-(g_j-h_j)=\frac{1}{2k}$ if $c_j=1$ and $b_j=1$.

In either case we get that $|(h_j-g_j)-(b_j\oplus c_j-b_j)|=\frac{1}{2k}$. So in order to get a contradiction it suffices to choose k, such that

$$\frac{1}{2k}\sum_{l=1}^{i}2^{l-i}<\frac{1}{k}\leq 2\epsilon_n=\frac{1}{3\cdot 2^{n-2}}.$$

So it suffices to choose $k=3\cdot 2^{n-2}$ for $n\geq 2$.

For $c_i=1$ we can do the computation faster (though it is still conceptually the same). Wlog $g_i<h_i$ from which it follows $g_i=\frac{1}{4k}$ and $h_i=\frac{4k-1}{4k}$. We will show that $b_i=0$ works for sufficiently large k. For the hypercube G we compute

$$\begin{aligned}f_i(g_1,\ldots,g_i)-\frac{f_{i-1}(b_1,\ldots,b_{i-1})}{2}&=\frac{1}{4k}+\sum_{l=1}^{l-1}\frac{g_j-b_j}{2^{i-l}}\leq\\&\leq\frac{1}{4k}+\sum_{l=1}^{i-1}\frac{1}{2^{i-l}}<\frac{1}{4k}+\frac{1}{2}\leq\frac{1}{2}+\epsilon_n.\end{aligned}$$

This inequality clearly holds $k\geq\frac{1}{4\epsilon_n}=3\cdot 2^{n-3}$.

And similarly for hypercube H

$$\begin{aligned}f_i(h_1,\ldots,h_i)-\frac{f_{i-1}(b_1\oplus c_1,\ldots,b_{i-1}\oplus c_{i-1})}{2}&=1-\frac{1}{4k}+\sum_{l=1}^{i-1}\frac{h_l-(b_l\oplus c_l)}{2^{i-l}}\geq\\&\geq 1-\frac{1}{4k}-\sum_{l=1}^{i-1}\frac{1}{2^{i-l}}>1-\frac{1}{4k}-\frac{1}{2}>\frac{1}{2}-\epsilon_n.\end{aligned}$$

for exactly the same k as in the previous inequality.

The choice $k=3\cdot 2^{n-2}$ for $n\geq 2$ (and $k=2$ for $n=1$) is therefore a valid one, satisfying all conditions of the claim. The claim is therefore proven and the whole counterexample finished.

□

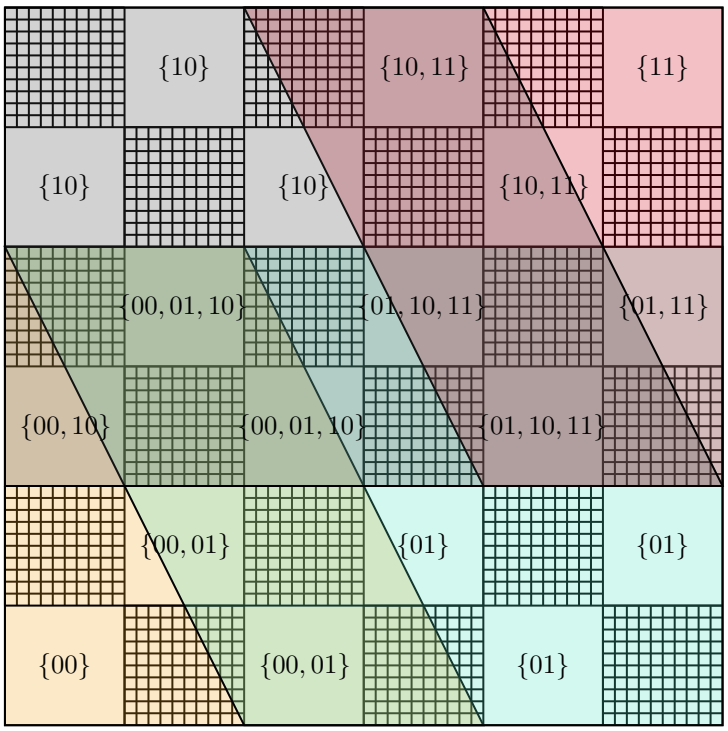

Figure 4.4: The construction for $n = 2$ (with $\epsilon = \frac{1}{6}$ and $k = 3$).

Conclusion

I explored the theory surrounding the combinatorial version of the Gap Label Cover theorem. The two original results are understanding the differences between a probabilistic combinatorial version and the combinatorial one (Chapter 3) and providing a counterexample to the combinatorial version of the Parallel Repetition Theorem (Chapter 4). Unfortunately, both results were in the direction of the combinatorial version not behaving as nicely as the classical one. On the other hand, more interesting results may arise from the theory of the combinatorial version. Negative results in this thesis do not make other advancements necessary harder. Chapter 3 only says we can't make a good probabilistic picture as in the classical case. After Chapter 4, we are left without the Parallel Repetition theorem, which is needed for the proof of the Gap Label Cover theorem in the classical case, but not in the combinatorial one.

Further research can be done on the technical Lemma 12. It would be interesting to see if the assumptions can be made weaker and if any stronger versions of combinatorial PCP theorems would follow. A counterexample for the constant difference between sizes of partial solutions was given in the second chapter. Still, this counterexample can be solved using a polynomial algorithm (it's not even arc consistent). It would be interesting to know if there is a better, harder-to-solve counterexample. If that wasn't the case, the d-to-1 conjecture could still follow from an improved version of the lemma.

There are also more specific things in the thesis which could be improved. In Chapter 4, I used the Cyclic instance C_k (20) to provide a counterexample to the Parallel Repetition theorem. However, I couldn't compute the combinatorial value of n-th power exactly for a large enough k (only an estimate in one direction was provided). It would be interesting to compute this value more precisely. At this moment, I don't have a better lower bound than 2 for any n.

Bibliography

[1] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and the hardness of approximation problems. *J. ACM*, 45(3):501–555, may 1998. ISSN 0004-5411. doi: 10.1145/278298.278306. URL https://doi.org/10.1145/278298.278306.

[2] L. Barto. Constraint satisfaction problem and universal algebra. *ACM SIGLOG News*, 1(2):14–24, oct 2014. doi: 10.1145/2677161.2677165. URL https://doi.org/10.1145/2677161.2677165.

[3] L. Barto and M. Kozik. *Combinatorial Gap Theorem and Reductions between Promise CSPs*, pages 1204–1220. 2022. doi: 10.1137/1.9781611977073.50. URL https://epubs.siam.org/doi/abs/10.1137/1.9781611977073.50.

[4] L. Barto, A. Krokhin, and R. Willard. Polymorphisms, and How to Use Them. In A. Krokhin and S. Zivny, editors, *The Constraint Satisfaction Problem: Complexity and Approximability*, volume 7 of *Dagstuhl Follow-Ups*, pages 1–44. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 2017. ISBN 978-3-95977-003-3. doi: 10.4230/DFU.Vol7.15301.1. URL http://drops.dagstuhl.de/opus/volltexte/2017/6959.

[5] M. Ben-Or, S. Goldwasser, J. Kilian, and A. Wigderson. Multi-prover interactive proofs: How to remove intractability assumptions. In *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, STOC '88, page 113–131, New York, NY, USA, 1988. Association for Computing Machinery. ISBN 0897912640. doi: 10.1145/62212.62223. URL https://doi.org/10.1145/62212.62223.

[6] I. Dinur. The PCP theorem by gap amplification. *J. ACM*, 54(3):12–es, jun 2007. ISSN 0004-5411. doi: 10.1145/1236457.1236459. URL https://doi.org/10.1145/1236457.1236459.

[7] I. Dinur and D. Steurer. Analytical approach to parallel repetition. In *Proceedings of the Forty-Sixth Annual ACM Symposium on Theory of Computing*, STOC '14, page 624–633, New York, NY, USA, 2014. Association for Computing Machinery. ISBN 9781450327107. doi: 10.1145/2591796.2591884. URL https://doi.org/10.1145/2591796.2591884.

[8] U. Feige. On the success probability of the two provers in one-round proof systems. In *Proceedings of the Sixth Annual Structure in Complexity Theory Conference, Chicago, Illinois, USA, June 30 - July 3, 1991*, pages 116–123. IEEE Computer Society, 1991. doi: 10.1109/SCT.1991.160251. URL https://doi.org/10.1109/SCT.1991.160251.

[9] V. Guruswami and R. O'Donnell. Cse 533: The PCP theorem and hardness of approximation, 2005. URL https://courses.cs.washington.edu/courses/cse533/05au/.

[10] S. Khot. On the power of unique 2-prover 1-round games. In *Proceedings of the Thiry-Fourth Annual ACM Symposium on Theory of Computing*, STOC

'02, page 767–775, New York, NY, USA, 2002. Association for Computing Machinery. ISBN 1581134959. doi: 10.1145/509907.510017. URL https://doi.org/10.1145/509907.510017.

[11] R. Raz. A parallel repetition theorem. *SIAM Journal on Computing*, 27(3): 763–803, 1998. doi: 10.1137/S0097539795280895. URL https://doi.org/10.1137/S0097539795280895.

[12] R. Raz. A counterexample to strong parallel repetition. *SIAM Journal on Computing*, 40(3):771–777, 2011. doi: 10.1137/090747270. URL https://doi.org/10.1137/090747270.