Electronic Colloquium on Computational Complexity, Report No. 54 (2007)

Testing Properties of Constraint-Graphs

Shirley Halevy* Oded Lachish† Ilan Newman‡ Dekel Tsur§

Abstract

We study a model of graph related formulae that we call the *Constraint-Graph model*. A constraint-graph is a labeled multi-graph (a graph where loops and parallel edges are allowed), where each edge e is labeled by a distinct Boolean variable and every vertex is associate with a Boolean function over the variables that label its adjacent edges. A Boolean assignment to the variables satisfies the constraint graph if it satisfies every vertex function. We associate with a constraint-graph G the property of all assignments satisfying G, denoted $SAT(G)$.

We show that the above model is quite general. That is, for every property of strings $\mathcal{P}$ there exists a property of constraint-graphs $\mathcal{P}_G$ such that $\mathcal{P}$ is testable using q queries iff $\mathcal{P}_G$ is thus testable. In addition, we present a large family of constraint-graphs for which $SAT(G)$ is testable with constant number of queries. As an implication of this, we infer the testability of some edge coloring problems (e.g. the property of two coloring of the edges in which every node is adjacent to at least one vertex of each color). Another implication is that every property of Boolean strings that can be represented by a Read-twice CNF formula is testable. We note that this is the best possible in terms of the number of occurrences of every variable in a formula.

1 Introduction

Property Testing considers the following relaxation of standard decision problems: given a property $\mathcal{P}$ of some combinatorial structures, one wants to decide whether a given instance (structure) S has the property $\mathcal{P}$ or it is ϵ-far from having the property. By ϵ-far we mean that at least an ϵ-fraction of the representation of S should be modified in order to make S satisfy $\mathcal{P}$. The goal in property testing is to design randomized algorithms which read a very small portion of the input and distinguish between the above two cases. Such algorithms are called *property tests*. Blum, Luby and Rubinfeld [2] were the first to formulate a question of this type. The general notion of property testing was first formulated by Rubinfeld and Sudan [9], who studied various algebraic properties such as linearity of functions. For detailed surveys on the subject see [4, 6, 8]. We use the formal framework of *Combinatorial Property Testing* as defined by Goldreich Goldwasser and Ron [6].

In this paper we focus on properties of constraint-graphs. A constraint-graph is a labeled multi-graph (a graph where loops and parallel edges are allowed), where each edge e is labeled by a distinct Boolean variable and every vertex is associate with a Boolean function over the variables that label its adjacent edges [1]. A Boolean assignment to the variables satisfies the constraint graph if it satisfies every vertex function. We associate with a constraint-graph G the property of all assignments satisfying G, denoted $SAT(G)$.

*The Caesarea Rothschild Institute for Computer Science, University of Haifa, Haifa 31905, Israel. Email: shirleyh@cs.technion.ac.il

†Department of Computer Science, University of Haifa, Haifa 31905, Israel. Email: loded@cs.haifa.ac.il

‡Department of Computer Science, University of Haifa, Haifa 31905, Israel. Email: ilan@cs.haifa.ac.il, Research supported in part by an Israel Science Foundation grant number 55/03.

§Department of Computer Science , Ben-Gurion University of the Negev, Be'er Sheva 84105, Israel. Email: dekelts@cs.bgu.ac.il

[1]One should not confuse this definition of 'constraint-graphs' with the definition used by Dinur in [3]. There, each vertex is associated with a *distinct* Boolean variable and the edges are labeled by constraints.

ISSN 1433-8092

Every Boolean function has a representation as a Constraint-Graph. Our aim is to use the Constraint-Graph model in order to identify wide classes of testable properties. The approach we use towards this goal is to examine classes of constraint-graphs defined by certain conditions on the underlying graph and on the functions that label the vertices.

We present a wide family of constraint-graphs, denoted by $\mathcal{LD}_3$, such that $SAT(G)$ is testable for every $G \in \mathcal{LD}_3$. A constraint graph is in $\mathcal{LD}_3$ if for every vertex v with degree at least 3, the hamming distance between any two assignments not satisfying f_v is at least 3. We show that this result is best possible in these terms; it can be shown that if the bound on the hamming distance is decreased from 3 to 2 then the set of constraint-graphs that is obtained is almost as general as the set of all constraint-graphs.

This result has a number of immediate consequences asserting the testability of certain natural properties. For example, the edge-coloring property in which every vertex is adjacent to at least two different colored edges is testable (for every graph).

Another more general implication is that for every read-twice CNF formula[2] θ, the property of all assignments satisfying θ is testable.

A natural question is whether this can be extended to read-3-times CNF formulae. The answer to this question is negative. Ben-Sasson et al. showed in [1] that there exists an infinite family of 3-CNF formulae such that the property $SAT(\theta)$ (the property of all satisfying assignments of θ) has linear query complexity. This result implies, in turn, that there exists a corresponding non-testable infinite family of read-3-Times 3-CNF formulae (requiring, in fact, linear query complexity). It should be noted that even if we restrict ourselves to read-3-times 2-CNF formulae the answer remains negative due to a construction by Fischer et al. [5]. Thus the testability result for read-twice CNF formula is the best possible in this respect.

The constraint-graph model can be extended in many natural ways. One may allow variables and functions over domains that are not necessarily Boolean. It also has other interesting interpretations for the Boolean case; One may interpret the value assigned to an edge as an orientation of the edge. That is 1 implies that the edge is oriented in a certain predetermined direction and 0 in the opposite direction. Thus, a property of assignments to a constraint-graphs can be viewed as a property of orientations of the edges. One, such natural property is the property of being Eulerian. Properties of orientations are by themselves an interesting line of research. For example, the query complexity of natural orientation properties such as strongly connected, acyclic etc. is still open.

Organization. In Section 2 we formally define the model of constraint-graphs and the set $\mathcal{LD}_3$. In Section 3 we present the results, the proofs to some, and a number of applications of the main result. Section 4 contains the main tools required in order to prove our main result and a proof of a weaker version of that result. Finally in Section 5 we prove the main result.

2 Preliminaries

In this work we consider properties that are subsets of $\{0,1\}^n$. Following [6], an ϵ-test for a property $\mathcal{P}$ is a randomized algorithm that given a distance parameter ϵ and oracle access to individual locations in the input string, can distinguish with high probability (say 2/3) between strings satisfying $\mathcal{P}$ and those that are ϵ-far from satisfying it. The query complexity $q(\epsilon, n)$ is defined as the maximum number of queries that the algorithm makes on any length n input. An $(\epsilon, q(\epsilon, n))$-test is an ϵ-test that on input of length n uses at most $q(\epsilon, n)$ queries.

A Property $\mathcal{P}$ is said to be *testable* if it has a $q(\epsilon, n)$-test (for any $\epsilon > 0$) for which $q(\epsilon, n)$ can be bounded from above by a function that depends only on ϵ (and independent of n). We say that a test for a property has *one-sided error* if it always accepts every input in the property. If the test may err in both directions then it is said to have *two-sided error*. A test is *adaptive* if its queries depend on the answers to

[2] in a read-twice formula every variable appears at most twice

previous queries, and *non-adaptive* otherwise (that is, if the queries that the test makes are only based on the internal random coins of the test).

Remark: In this paper all upper bounds on the query complexity are proven for the one-sided error *non-adaptive* case and all lower bounds are shown for two-sided error *adaptive* tests.

We now formally present the notion of constraint-graphs.

Definition 2.1. [Constraint-Graph] *A constraint-graph is a labeled multi-graph $G = (V(G), E(G), \mathcal{F})$ in which*

- *$(V(G), E(G))$ is an undirected multi-graph (may contain parallel edges and loops), and is called the underlying graph.*
- *Each edge $e \in E(G)$ is associated with a distinct Boolean variable x_e. We sometimes identify between an edge $e \in E(G)$ and the Boolean variable associated with it.*
- *Each vertex $v \in V(G)$ is labeled by a Boolean function f_v over the variables associated with the edges adjacent to v. We set $\mathcal{F} = \{f_v\}_{v \in V(G)}$.*

2.1 Assignments and Constraint-graphs

From here on by a graph we mean a constraint-graph (unless stated otherwise). For a vertex v, E_v denotes the set of edges adjacent to v, and $deg_G(v)$ denotes its degree. We denote by $B_G(v, r)$ the induced subgraph in G that contains all vertices that are at distance at most r from v.

An assignment to $E(G)$ is a mapping $\sigma : E(G) \longrightarrow \{0, 1\}$. We denote by $\sigma(e)$ the value assigned to e. We say that an assignment σ satisfies v if $f_v(\sigma) = 1$. We say that σ satisfies G if it satisfies every vertex in G. We denote by $\mathbf{UNSAT_G}(\sigma)$ the set of all vertices $v \in V(G)$ that are not satisfied by σ. Let $\mathbf{\Delta_G}(\sigma)$ be the minimum degree of a vertex in $UNSAT_G(\sigma)$.

We associate with the graph G the property $\mathbf{SAT(G)}$ that contains all the satisfying assignments of G. For a formula θ we associate with θ the property $SAT(\theta)$ that contains all assignments satisfying θ. Thus, for a constraint graph G, $SAT(G) = SAT(\phi_G)$, where $\phi_G \equiv \wedge_{v \in V(G)} f_v$ over the variables $E(G)$.

Given two assignments σ_1 and σ_2 for $E(G)$, the distance between σ_1 and σ_2, denoted $dist(\sigma_1, \sigma_2)$, is the hamming distance between σ_1 and σ_2. The distance between an assignment σ and a property $SAT(G)$, denoted $dist(\sigma, SAT(G))$, is the minimum distance between σ and an assignment in $SAT(G)$. We say that an assignment σ is ϵ-far from $SAT(G)$ if its distance from $SAT(G)$ is at least $\epsilon \cdot |E(G)|$ and ϵ-close otherwise.

2.2 The Hierarchy $\mathcal{LD}_i$

Given a vertex v and two assignments σ_1, σ_2 to $E(G)$, the *local distance* at v, denoted by $ldist_{G,v}(\sigma_1, \sigma_2)$, is the number of edges $e \in E_v$ for which $\sigma_1(e) \neq \sigma_2(e)$, where loops are counted twice.

Definition 2.2. *Let $\mathcal{LD}_i$ be the set of graphs G such that for every two assignments σ_1, σ_2 and every vertex v with $deg_G(v) \geq 3$, if σ_1, σ_2 do not satisfy f_v then either $\sigma_1(E_v) = \sigma_2(E_v)$ or $ldist_v(\sigma_1, \sigma_2) \geq i$.*

Remark: We omit G from the subscript of the various symbols when the graph is clear from the context.

3 Results and Applications

We first observe that the Constraint-graph model is a general model in the following sense.

Proposition 3.1. *For every Boolean function θ there exists a constraint-graph G such that $\theta \equiv \phi_G$.*

Proof. Let G be a star with n leaves in which the center is labeled by θ, the n edges are the labeled by the distinct n variables of θ, and the leaves are associated with the constant 1-function. □

In [1] the authors show that there exists family of 3-CNF formulae that are highly non-testable.

Theorem 3.2. **[a]** *There exist $\epsilon > 0$ and a 3-CNF formula ϕ on n variables (for every n large enough) for which every two-sided error ϵ-test makes $\Omega(n)$ queries.*

Corollary 3.3. *There exist $\epsilon > 0$ and a constraint graph G with n edges (for every large enough n) for which every two-sided error ϵ-test for $SAT(G)$ makes $\Omega(n)$ queries.*

The following claim enables us to prove an even stronger result.

Claim 3.4. *For every k-CNF formula θ on n variables, where $k \geq 2$, there is a read-3-times k-CNF formula η such that $SAT(\theta)$ has an $(\epsilon, q(\epsilon))$-test iff $SAT(\eta)$ has an $(\epsilon, q(\epsilon))$-test.*

Proof. The proof uses the standard way of constructing an equivalent read-3-times formula of a given formula.

Let θ be a k-CNF formula on variables $X = \{x_1, \ldots, x_n\}$. Let r be the maximum number of appearances of any single variable in θ. We create the new k-CNF formula η as follows: For each variable x_i we introduce $r = r(i)$ variables $x_i^1, \ldots, x_i^r$ which will be required to have the same value. This can be done by adding 2-size clauses and so that every x_i^j appears in at most two of these clauses. Then, we replace each appearance of a variable x_i in θ by a distinct copy. Hence the resulting formula η containing the original clauses with the replaced variables and the additional 2-size clauses, is a k-CNF, read-3-times formula.

It is straight forward to see that θ is equivalent to η (w.r.t satisfiability). More over, it can be shown that a test for η can be turned into a test for θ of essentially the same complexity and vice versa (one should be careful as the relative distance is not always maintained, but this can be taken care of). We differ further details to the final version. □

Corollary 3.5. *There exists $\epsilon > 0$ and a* **read 3-times** *3-CNF formula ϕ on n variables for which every 2-sided error ϵ-test makes $\Omega(n)$ queries.*

Corollary 3.6. *There exists $\epsilon > 0$ and a 3-bounded-degree graph G with n edges (for every sufficiently large n) for which every 2-sided error ϵ-test for $SAT(G)$ requires $\Omega(n)$ queries.*

Proof. Let θ be a read-3-times 3-CNF formula as asserted by Corollary 3.5. We construct a constraint-graph G, whose maximal vertex degree is 3, and if $(\epsilon/3, q(3/\epsilon))$-testable then $SAT(\theta)$ is $(\epsilon, O(q(1/\epsilon)))$-testable. This suffices as for some fixed ϵ, $SAT(\theta)$ is not $(\epsilon, o(n))$-testable.

Assume first that each variable appears exactly 3 times in θ. Let $\mathcal{C}$ be the clause set of θ and X be its variable set. We set $G = (X \cup \mathcal{C}, E, \mathcal{F})$ to be the following constraint graph. E contains an edge between $x \in X$ and $C \in \mathcal{C}$ if and only if x appears in C (as either x or $\bar{x}$). For each $x \in X$, f_x is the Boolean function expressing that $a \equiv b$ for every $a, b \in E_x$. For each $C \in \mathcal{C}$, f_C is defined to be the clause we get from C by replacing each appearance of a variable x in C by the variable e, where e is the edge connecting x to C (e appears in negated from if x appears negated in C and not negated otherwise).

We define the following mapping γ from assignments α for G to assignments σ for θ. We set $\gamma(\alpha)(e) = \alpha(x_i)$ if e is adjacent to x_i. It is easy to see that if $\alpha \in SAT(G)$, then $\gamma(\alpha) \in SAT(\theta)$. Also, if α is ϵ-far from $SAT(G)$ then $\gamma(\alpha)$ is $\epsilon/3$-far from $SAT(\theta)$. Finally, given an (ϵ, q)-test for $SAT(G)$ there is an (ϵ, q)-test for $SAT(\theta)$ by simulating every x-query by a query to an edge e that is adjacent to x.

If some variables appear less than 3 times in θ then one just needs to add parallel edges in G, so that the degree of every vertex $x \in X$ is exactly 3. □

Thus, even if we restrict ourselves to the set of graphs of maximal vertex degree 3 we can not hope to prove any general testability result. However, if we restrict the functions labeling the vertices then we can show a positive result.

Theorem 3.7. *For every constraint-graph $G \in \mathcal{LD}_3$ there exists a one sided error, non-adaptive $(\epsilon, 2^{\tilde{O}(1/\epsilon)})$-test for $SAT(G)$.*

The proof of Theorem 3.7 appears in Section 5. It requires several 'cleaning' steps of which the final resulting graph is the 'hard-core' of the problem. For example, it is easy to see that we may restrict ourselves to connected graphs. This motivates the following definition, which captures this 'hard-core'.

Definition 3.8. [Hard-Constraint-Graph] *A graph $G = (V, E, \mathcal{F})$ is hard if it is in $\mathcal{LD}_3$ and it satisfies the following conditions.*

- *G is connected and contains at least one vertex of degree at least 3.*
- *$f_v \equiv 1$ for every vertex of degree 1.*
- *For every vertex v of degree 2, and every two assignments σ_1, σ_2 that do not satisfy f_v, either $ldist_v(\sigma_1, \sigma_2) = 2$ or $\sigma_1(E_v) = \sigma_2(E_v)$.*

The proof in Section 5 is based on the following theorem that is weaker than Theorem 3.7 in two senses. First, it implies the existence an algorithm that behaves like a test only if it gets as input an assignment σ for which $\Delta_G(\sigma) \geq 3$ (recall, that $\Delta_G(\sigma)$ is the minimum degree of an unsatisfied vertex). Second, the algorithm is correct only for hard-constraint-graphs.

Theorem 3.9. *For every hard-constraint-graph G and $\epsilon > 0$ there exists a $2^{\tilde{O}(1/\epsilon)}$ query complexity algorithm that on input σ: accepts with probability 1 if $\sigma \in SAT(G)$, rejects with probability 2/3 if σ is ϵ-far from $SAT(G)$ and $\Delta_G(\sigma) \geq 3$.*

The proof of this theorem captures the main essence of the testing problem, while enabling us to avoid a number technicalities. The following lemma which relates $dist(\sigma, SAT(G))$ to $\Delta_G(\sigma)$ is a crucial part of the proof of Theorem 3.9 and is proved separately in Section 4.4.

Lemma 3.10. *Let G be a hard-graph. Then for every assignment σ, $dist_G(\sigma, SAT(G)) \leq 6/\Delta_G(\sigma)$.*

Lemma 3.10 implies that for a hard-graph G and assignment σ, if $\Delta_G(\sigma) > 6/\epsilon$ then σ is ϵ-close to $SAT(G)$. Consequently, if the degree of every vertex is greater than $6/\epsilon$ then testing $SAT(G)$ is trivial.

One might hope that the result of Theorem 3.7 can be extended to $\mathcal{LD}_2$. However, Eldar Fischer observed that this is not the case, since the set $\mathcal{LD}_2$ is almost as general as the set of all constraint-graphs.

Theorem 3.11 (Fischer)**.** *For every Boolean formula θ and $\epsilon > 0$ there exists a graph $G \in \mathcal{LD}_2$ such that $SAT(G)$ is (ϵ, q)-testable if and only if $SAT(\theta)$ is $(\epsilon, O(q))$-testable.*

Proof. Let θ be a Boolean formula over a set variables $X = \{x_1, \ldots, x_n\}$. Let G be the constraint-graph on two vertices $\{v, t\}$, that has $n + 1$ parallel edges between v and t, where one is identified with the variable y and each of the rest n edges is labeled with a distinct variable in X. Let $f_v = y \oplus (\bigoplus_{i=1}^n x_i)$ and $f_t = \theta(x_1, \ldots, x_n) \vee (y = \bigoplus_{i=1}^n x_i)$. Obviously the resulting graph G is in $\mathcal{LD}_2$.

Let γ be the a mapping from assignments α to $X \cup \{y\}$ to assignments σ to X, such that $\sigma(x_i) = \alpha(x_i)$ for every $x_i \in X$. Observe that f_t is satisfied if and only if $t \neq (\bigoplus_{i=1}^n x_i)$. This implies that an assignment α satisfies both f_t and f_v if and only if $\gamma(\alpha) \in \mathcal{P}_\theta$. Hence, $dist(\alpha, \mathcal{P}_G) = dist(\gamma(\alpha), \mathcal{P}_\theta) + i$, where $i \in \{0, 1\}$. Given a test for one of the properties it is straightforward to build a test for the other property. □

We present here several immediate applications of Theorem 3.7

Theorem 3.12. *For every read-twice CNF formula θ, $SAT(\theta)$ is $(\epsilon, 2^{\tilde{O}(1/\epsilon)})$-testable.*

Proof. Let θ be a read-twice CNF formula with clause-set $\mathcal{C}$ and variable set X. Set G to be the following constraint graph. For each $C \in \mathcal{C}$ there is vertex v_C labeled with $f_{v_C} = C$. For each variable x that appears in two different clauses C, D (in any polarity) there is an edge labeled by x between v_C and v_D. For a variable x that appears in only one clause C there is a edge labeled by x between v_C and a vertex u_C of degree 1 that is labeled with $f_{u_C} = 1$.

By definition θ is identical to ϕ_G. Hence their query complexity is the same. In addition, every vertex in G that is associated with a non-constant function has at most one non-satisfying assignment, hence $G \in \mathcal{LD}_3$. □

This result is the best possible in terms of the maximum number of times a variable appears in a formula, even for 2-CNF. In order to show this we need the following theorem which is a direct result of [5].

Theorem 3.13. *[5] There exists $\epsilon > 0$ and a 2-CNF formula ϕ on n variables (for every sufficiently large n) such that $SAT(\phi)$ is not testable.*

By taking these hard-to-test formulae that are asserted by Theorem 3.13, and applying Claim 3.4 we get the following corollary.

Corollary 3.14. *There exists $\epsilon > 0$ and a* **read-3-times** *2-CNF formula ϕ on n variables (for every sufficiently large n) for which $SAT(\phi)$ is not testable.*

An edge-coloring of a multi-graph G by a set of colors C is called 'no-where-monochromatic' if there exist no vertex such that all its adjacent edges have the same color.

Theorem 3.15. *For every multi-graph G the property of all 'no-where-monochromatic' colorings has an $(\epsilon, 2^{\tilde{O}(1/\epsilon)})$-test.*

Proof. Let G be a multi-graph and assume that $C = \{0, 1\}$. We view an edge-coloring of G as a Boolean assignment of its edges. We associate with a vertex v the Boolean function that is satisfied by all assignments to E_v that contain both a '1' and a '0'. Then, by definition, $SAT(G)$ contains all the no-where-monochromatic coloring of G. It can also be verified that $G \in \mathcal{LD}_3$. Hence, Theorem 3.7 implies the claim.

In order to show the same for coloring any number (not necessarily constant) of colors one needs to show that Theorem 3.7 holds even if we extend $\mathcal{LD}_3$ to non-Boolean variables. We omit further details in this draft. □

4 Proof of Theorem 3.9

This section is organized as follows. In Subsection 4.1 we study the relation between heavy-graphs and their assignments. In Subsection 4.2 we study the relation between $SAT(G)$ and the satisfiability of graphs that are obtain from G via edge-contraction. This, in turn, plays a crucial role in the proof of Theorem 3.9 and Lemma 3.10. In Subsection 4.3 we prove Theorem 3.9. Finally, in Subsection 4.4 we prove Lemma 3.10.

4.1 Subgraphs and Assignments

Let G be a hard-graph. Given two assignments σ_1, σ_2 to $E(G)$ and a (not necessarily induced) subgraph T of G we use the notation $\sigma_1 \overset{T}{\sim} \sigma_2$ to denote that $\sigma_1(e) = \sigma_2(e)$ for every $e \notin E(T)$. We say an assignment σ **agrees** with a subgraph T of G if there exists an assignment σ^* such that $\sigma^* \overset{T}{\sim} \sigma$ and $UNSAT_G(\sigma^*) \cap V(T) = \emptyset$. That is, a subgraph T of G agrees with an assignment σ to $E(G)$, if we can get an assignment that satisfies all the vertices in $V(T)$ by changing σ only on $E(T)$.

The notion of a subgraph agreeing with an assignment will turn useful. For example, for every hard-graph G, the distance of an assignment η from $SAT(G)$, is the minimum number of edges in a subgraph T that agrees with η and contains every vertex in $UNSAT_G(\eta)$. In our proof we are especially interested in the structures of subgraphs that agree with every assignment and the structure of subgraphs that don't agree with some assignments.

It will be shown in Claim 4.3 below that if a connected subgraph T contains a cycle or a vertex v for which $f_v \equiv 1$, then it agrees with every assignment. Claim 4.3 implies the following corollary, due to the fact that every hard-graph either contains a cycle or contains a vertex of degree 1.

Corollary 4.1. *Every hard-graph is satisfiable.*

In the case of a connected subgraph that agrees with some assignments we can even learn more on their structure as in Claim 4.4. In order to prove these results we first show that for every connected subgraph T of a hard-graph G, and every assignment η to $E(G)$, we can select any vertex $v \in V(T)$ and by 'correcting' the values of η on $E(T)$ we can get an assignment that satisfies every vertex in $V(T)$ except possibly v. The ability to "select" the vertex that is not satisfied is useful in the proofs of the results in this subsection.

Claim 4.2. *Let G be a hard-graph, T a connected subgraph of G and v a vertex in $V(T)$. Then, for every assignment η to $E(G)$ there exists an assignment η^* such that $\eta \overset{T}{\sim} \eta^*$ and $UNSAT_G(\eta^*) \cap V(T) \subseteq \{v\}$.*

Proof. If $|UNSAT_G(\eta) \cap V(T) \setminus \{v\}| = 0$ then η is the required assignment. We proceed by induction on $\ell = |UNSAT_G(\eta) \cap V(T) \setminus \{v\}|$. Assume that the claim holds for every η' and T' when $|(UNSAT_G(\eta') \cap V(T')) \setminus \{v\}| < \ell$.

Let $u \in V(T) \setminus \{v\}$ be such that $u \in UNSAT_G(\eta)$ and $dist_T(u, v)$ is minimum. Let R be a shortest path in T that connects u and v (such a path exists since T is connected). Denote the vertices of R by $x_1, \ldots, x_k$, where $x_1 = u$ and $x_k = v$. Let $e_1, \ldots, e_{k-1}$ be the edges of R, where for every $1 \leq i < k$, e_i is adjacent to x_i and x_{i+1}. By definition, $x_i \notin UNSAT_G(\eta)$ for every $2 \leq i \leq k-1$.

Let η^* be such that $\eta^* \overset{T}{\sim} \eta$ and $\eta^*(e) = \neg\eta(e)$ for every $e \in E(R)$. Since $ldist_{x_1}(\eta^*, \eta) = 1$ and since $G \in \mathcal{LD}_3$, $x_1 \notin UNSAT_G(\eta')$. If we also have $x_i \notin UNSAT_G(\eta^*)$ for every $2 \leq i < k$, then $|(UNSAT(\eta^*) \cap V(T)) \setminus \{v\}| \leq \ell - 1$ and hence by the induction hypothesis we are done.

Thus, we assume that there exists $i \in \{2, \ldots, k-1\}$ such that $x_i \in UNSAT_G(\eta^*)$. Let j be the minimum integer such that $x_j \in UNSAT_G(\eta^*)$. Let η^{**} be such that $\eta^{**} \overset{T}{\sim} \eta$, and $\eta^{**}(e_i) = \eta^*(e_i)$ for every $1 \leq i < j$ and $\eta^{**}(e_i) = \eta(e_i)$ for every $j \leq i < k$. Since $ldist_{x_j}(\eta^{**}, \eta^*) = 1$, the fact that $G \in \mathcal{LD}_3$ implies that $x_j \notin UNSAT_G(\eta^{**})$. Thus every vertex on R, except possibly v, is satisfied by η^{**}. Since $|(UNSAT(\eta^{**}) \cap V(T)) \setminus \{v\}| \leq \ell - 1$, the induction hypothesis completes the proof. □

Claim 4.3. *Let G be a hard-graph, T a connected subgraph of G. If T contains a cycle, or contains a vertex v for which $f_v \equiv 1$, then T agrees with every assignment to $E(G)$.*

Proof. Let η be an unsatisfying assignment to $E(G)$. Assume first that there exists $v \in V(T)$ with $f_v \equiv 1$. According to Claim 4.2 there exists an assignment ξ such that $\xi \overset{T}{\sim} \eta$ and $UNSAT_G(\eta) \cap V(T) \subseteq \{v\}$. Since $f_v \equiv 1$, T agrees with η.

Assume that T contains a simple cycle. Since G is a hard-graph, there exists $v_0 \in V(T)$ such that $deg_G(v_0) \geq 3$ and v_0 is in a cycle in T. By Claim 4.2 there exists an assignment η^* such that $\eta^* \overset{T}{\sim} \eta$ and $UNSAT_T(\eta^*) \subseteq \{v_0\}$. We may assume that $UNSAT_T(\eta^*) = \{v_0\}$ as otherwise we are done.

Let $R = (v_0, v_1, \ldots, v_{k-1}, v_0)$, be a simple cycle (possibly a loop) in T. Denote the edges of R by $e_0, \ldots, e_{k-1}$, where for every $i \leq k-1$, e_i is adjacent to x_i and $x_{(i+1) \bmod k}$. Let η^{**} be the assignment we get from η^* by changing the value assigned to e_0. Obviously, η^{**} satisfies v_0. If it satisfies v_1 we are finished. Otherwise we look at the assignment we get from η^{**} by changing the value assigned to e_2, check if v_2 is satisfied and so on until we stop at some stage i for which v_i is satisfied and we are done. Otherwise, we end by changing the value of e_{k-1} arriving back at v_0. The assignment η' at this point satisfies v_0 too as the local distance at v_0 from η' to η^* is exactly 2. □

For an assignment η that does not satisfy a certain vertex $v \in V(G)$, we are interested in "locally" fixing η is a subgraph T so that v and all vertices in T are satisfied. The following claim states that such a correction can always be made along a simple path starting at v + possibly an edge between the last vertex in the path and some other vertex in the path.

Claim 4.4. *Let G be a hard-graph, σ an assignment to $E(G)$ and T a subgraph that agrees with σ. Let $v \in UNSAT_G(\sigma) \cap V(T)$, then there exists a path in T that starts in v and agrees with σ. Moreover this path, possibly excluding the last vertex is simple.*

The proof of this claim uses similar idea as before and appears in Appendix A.

4.2 Subgraph Contraction

Given a graph G and an edge $e \in E(G)$ adjacent to vertices x, y, we set $G/\{e\}$ to be the graph we get by contracting e. By contracting e (edge contraction) we mean that we remove e from $E(G)$ and merge the two vertices x, y into one vertex v_e. That is, all the edges that where adjacent to x or y are now adjacent to the merged vertex. Note that this operation may result in parallel edges and self loops that remain in the resulting graph. The vertex v_e is labeled with the function $f_{v_e} = (f_x \wedge f_y)\mid_{e=1} \ \vee \ (f_x \wedge f_y)\mid_{e=0}$. That is, an assignment ν to E_{v_e} satisfies f_{v_e} if and only if ν extended by $x_e = 1$ satisfies f_x and f_y or ν extended by $x_e = 0$ satisfies f_x and f_y.

Given a subgraph T of G, we set G/T to be the graph we get by applying edge contraction to every edge in $E(T)$ according to some arbitrary order. It is easy to see that G/T does not depend on the order in which the edges were contracted. Note that contracting T collapses each connected component of T into a unique vertex in G/T. For a connected component M of T we denote this vertex by v_M.

The following corollary is immediate from the definition of contraction and Claim 4.3.

Corollary 4.5. *Let G be a hard-graph and T a subgraph of G. For every connected component M of T, if M contains a cycle then $f_{v_M} \equiv 1$ in G/T.*

We show next that hard-graphs are closed under contraction.

Claim 4.6. *Let G be a hard-graph and T a subgraph of G then G/T is a hard-graph.*

The proof is based on the fact that for $e \in T$, $G/T = (G/\{e\})/(T/\{e\})$, hence it is enough to prove the claim for a single edge. This is done by case study and appears in Appendix B.

The following mappings between assignments of G and assignments of G/T will prove to be useful.

Definition 4.7. *Let T be a subgraph of G.*

- *Given an assignment η for G we define $\eta^{G \to G/T}$ to be an assignment for G/T that is the restriction of η on $E(G/T) = E(G) \setminus E(T)$ (formally there is some renaming of the edges in G/T).*
- *Given an assignment ξ for G/T we define $\xi^{G/T \to G}$ to be an assignment for G, such that $\xi^{G/T \to G}(e) = \xi(e^G/T)$ for every $e \in E(G) \setminus E(T)$, where $e^{G/T}$ is the edge in $E(G/T)$ corresponding to e. The assignments to edges in $E(T)$ are such that $|UNSAT_G(\xi^{G/T \to G})|$ is the smallest possible.*

The following corollary directly follows from the definition of subgraph contraction and Definition 4.7.

Corollary 4.8. *Let T be a subgraph of a hard graph G. Then, for every assignment η to $E(G)$, $\eta \in SAT(G)$ if and only if $\eta^{G \to G/T} \in SAT(G/T)$ and for every assignment ξ to $E(G/T)$, $\xi \in SAT(G/T)$ if and only if $\xi^{G/T \to G} \in SAT(G)$.*

Proof Technique Based on Subgraph Contraction
Our technique for using subgraph contraction is based on following proposition.

Proposition 4.9. *Let T be a subgraph of a hard graph G and σ an assignment to $E(G)$. Then,*

$$dist(\sigma, SAT(G)) \leq dist(\sigma^{G \to G/T}, SAT(G/T)) + |E(T)|.$$

Proof. By definition there exists an assignment $\nu \in SAT(G/T)$ such that $dist(\sigma^{G \to G/T}, \nu) = dist(\sigma^{G \to G/T}, SAT(G/T))$. Set $\sigma^* = \nu^{G/T \to G}$. According to Corollary 4.8, $\nu \in SAT(G/T)$ implies that $\sigma^* \in SAT(G)$. Consequently, $dist(\sigma, SAT(G)) \leq dist(\sigma, \sigma^*)$. By definition

$$dist(\sigma, \sigma^*) = \sum_{e \in E(G) \setminus E(T)} |\sigma(e) - \sigma^*(e)| + \sum_{e \in E(T)} |\sigma(e) - \sigma^*(e)|. \tag{1}$$

Observe that,

$$dist(\sigma^{G\to G/T}, SAT(G/T)) = \sum_{e\in E(G/T)\setminus E(T)} |\sigma^{G\to G/T}(e) - \nu(e)| = \sum_{e\in E(G)\setminus E(T)} |\sigma(e) - \sigma^*(e)|.$$

Hence, by plugging the above instead of the first term of Equation (1) and noting that the second term is bounded by $|E(T)|$ completes the proof. □

According to Proposition 4.9 if we want to prove an upper bound on $dist(\sigma, SAT(G))$ it is sufficient to find a subgraph T of G such that we can show the required upper bound on $|E(T)| + dist(\sigma^{G\to G/T}, SAT(G/T))$. This is the technique we use in order to prove Lemma 3.10.

In order to prove Theorem 3.9 we use a variation of the previous technique. That is, we are given an assignment σ, that is ϵ-far from $SAT(G)$ (I.e. $dist(\sigma, SAT(G)) \geq \epsilon \cdot |E(G)|$). Our goal is to show a lower bound the number of edges in a subgraph T of G, whose choice depends on σ. Proposition 4.9 implies that we only need to show an appropriate upper bound on $dist(\sigma^{G\to G/T}, SAT(G/T))$.

4.3 Proof of Theorem 3.9

Let G be a hard-graph and σ be an assignment to $E(G)$. We show in this subsection that there exists an algorithm that given ϵ and oracle access to σ accepts with probability 1 if $\sigma \in SAT(G)$ and rejects with probability 2/3 if σ is ϵ-far from $SAT(G)$ and $\Delta_G(\sigma) \geq 3$. The algorithm will be shown to have query complexity $2^{\tilde{O}(1/\epsilon)}$. The existence of such an algorithm implies Theorem 3.9. The goal of the algorithm is to find a vertex $v \in UNSAT_G(\sigma)$ of degree at most $12/\epsilon$. Only then will it reject. Hence if $\sigma \in SAT(G)$ then the algorithm accepts with probability 1. In order to easily refer to such vertices we use the following definition.

Definition 4.10. [**ϵ-relevant/ ϵ-heavy vertices**]
Let G be a graph and $\epsilon > 0$. We say that $v \in V$ is ϵ-relevant *if $3 \leq deg_G(v) \leq 12/\epsilon$ and* ϵ-heavy *if $deg_G(v) > 12/\epsilon$.*

Assume that σ is ϵ-far from $SAT(G)$ and $\Delta_G(\sigma) \geq 3$. It is not necessarily true that G contains many ϵ-relevant vertices that are not satisfied by σ. The way that the algorithm is directed to find such a vertex is via the association of a subgraph, denoted as ϵ-Balloon$_G(v)$, with every ϵ-relevant v. This subgraph will have the following properties.

1. If σ is ϵ-far from $SAT(G)$, then the number of edges e such that $e \in E(\epsilon\text{-Balloon}_G(v))$ for some ϵ-relevant $v \in UNSAT_G(\sigma)$ is at least $\epsilon \cdot |E(G)|/4$.

2. For every edge in $e \in E(G)$ the number of ϵ-relevant vertices v such that ϵ-Balloon$_G(v)$ contains e is $2^{\tilde{O}(1/\epsilon)}$.

Property 1 implies that if σ is ϵ-far from $SAT(G)$ and one selects an edge e uniformly at random, then with probability that depends only on ϵ there exists an ϵ-relevant vertex $v \in UNSAT_G(\sigma)$ whose ϵ-Balloon$_G(v)$ contains e. We refer to such an event as 'good'. If a good event occurs for an edge e then one only needs to check whether σ satisfies f_v, for every ϵ-relevant vertex v for which $e \in E(\epsilon\text{-Balloon}_G(v))$. Property 2 asserts that there are only $2^{\tilde{O}(1/\epsilon)}$ such vertices. Checking whether f_v is satisfied for an ϵ-relevant vertex v requires to query the value of the edges in E_v and there are at most $12/\epsilon$ such edges. Consequently, the total number of queries will depend only on ϵ.

We present the exact definition of ϵ-Balloon$_G(v)$ after introducing the algorithm and formally proving its correctness, which depends only on Property 1 and Property 2 above. We now formally introduce the test.

Algorithm 4.1.
Input: ϵ, σ.

1. *Repeat the following $3/\epsilon$ times, independently.*
   - *Select an edge $e \in E(G)$ uniformly.*
   - *For every ϵ-relevant vertex z, such that $e \in E(\epsilon\text{-}Balloon_G(z))$, query all values assigned to edges in E_z and if f_z is not satisfied then reject.*
2. *If there was no reject then accept.*

Claim 4.11. *Algorithm 4.1 has query complexity $2^{\tilde{O}(1/\epsilon)}$. It accepts $\sigma \in SAT(G)$ with probability 1 and rejects σ that is ϵ-far from $SAT(G)$ with probability at least $2/3$.*

Proof. Algorithm 4.1 rejects an input σ only if it finds a vertex that is not satisfied, hence it cannot reject a satisfying assignment of G.

Assume that σ is ϵ-far from $SAT(G)$. According to Property 1 above, with probability at least $\epsilon/4$ an edge selected in step 1 of the algorithm is in the ϵ-Balloon of some ϵ-relevant vertex $v \in UNSAT_G(\sigma)$. Since step 1 is repeated $10/\epsilon$ independently the probability that in none of the iterations such an edge is selected is $(1-\epsilon/4)^{10/\epsilon} < 1/3$. Consequently, Algorithm 4.1 rejects σ with probability strictly greater than $2/3$.

Finally, the query complexity of the algorithm follows immediately from its definition, Property 2 above and the definition of an ϵ-relevant vertex. □

We next define ϵ-Balloon$_G(v)$ formally. The definition is somewhat technical, it is tailored to facilitate the proofs of Lemma 4.14 and Claim 4.13, which assert Properties 1 and 2 above.

Definition 4.12. [**ϵ-Balloon$_G(v)$**]
Let $v \in V(G)$ be ϵ-relevant. ϵ-Balloon$_G(v)$ is the subgraph with the minimum number of vertices among the following (we choose the first if both alternatives have the same number of vertices).

- *$B_G(v,r)$, where r is the minimum such that $B_G(v,r)$ contains a cycle or a vertex of degree 1 or there are at least $12/\epsilon$ edges not in $B_G(v,r)$ that are adjacent to vertices in $B_G(v,r)$.*
- *The subgraph we get by removing all but one ϵ-heavy vertex from $B_G(v,r)$, where r is the minimum such that $B_G(v,r)$ contains at least one ϵ-heavy vertex.*

Claim 4.13. *For every $e \in E(G)$ there exist $2^{\tilde{O}(1/\epsilon)}$, ϵ-relevant vertices v such that $e \in E(\epsilon\text{-}Balloon_G(v))$.*

Proof. A vertex $v \in V(G)$ is said to be near a vertex $u \in V(G)$, if there exists a path between v and u, that contains at most $1 + (12/\epsilon)$ ϵ-relevant vertices and does not contain any ϵ-heavy vertex. Note, that according to this notation if v is near u, then both vertices are not ϵ-heavy.

We will first show that the statement of the claim is true assuming that for every ϵ-relevant vertex v if $e \in E(\epsilon\text{-Balloon}_G(v))$, then v is near one of the vertices adjacent to e. Afterwards we shall show that this assumption is indeed true.

Obviously, the number of ϵ-relevant vertices that are near a given vertex u is maximized when u is the root of a balanced tree of depth $12/\epsilon$, such that each of it internal vertices has degree $12/\epsilon$. Consequently, there are at most $(12/\epsilon)^{12/\epsilon}$ vertices near u. Hence, there are at most $2 \cdot (12/\epsilon)^{12/\epsilon} = 2^{\tilde{O}(1/\epsilon)}$ vertices near a vertex adjacent to e. According to the assumption this is an upper bound on the number of vertices v such that $e \in E(\epsilon\text{-Balloon}_G(v))$. It remains to show that the assumption is indeed true.

Let v be an ϵ-relevant vertex such that $e \in E(\epsilon\text{-Balloon}_G(v))$ and x, y be the vertices adjacent to e. By definition $V(\epsilon\text{-Balloon}_G(v))$ contains at most one ϵ-heavy vertex. Therefore at least one of the vertices x, y is not ϵ-heavy. With out loss of generality assume that x is not ϵ-heavy.

Let B be a shortest path between x and v. Obviously, B is contained in ϵ-Balloon$_G(v)$. We first show that B does not contain an ϵ-heavy vertex. Indeed, by definition, if ϵ-Balloon$_G(v)$ contains an ϵ-heavy vertex z, then $dist_G(z,v) = max\{dist_G(s,v) \mid s \in V(\epsilon\text{-Balloon}_G(v))\}$. As B is a shortest path, $dist_G(v,s) < dist_G(v,x)$ for every $s \in B$. Hence, none of the vertices in $V(B)$ are ϵ-heavy.

Next we show that B contains at most $1 + (12/\epsilon)$ ϵ-relevant vertices. Indeed, assume towards a contradiction that $V(B)$ contains at least $12/\epsilon+2$, ϵ-relevant vertices. Since $u \notin V(B_G(v, dist_G(x,v)-1))$, by definition, $B_G(v, dist_G(x,v)-1)$ is strictly contained in ϵ-Balloon$_G(v)$. In addition, $B_G(v, dist_G(x,v)-1)$ is a tree with no vertices of degree 1 in G as otherwise, ϵ-Balloon$_G(v)$ is a subgraph of $B_G(v, dist_G(x,v)-1)$. Observe that $B_G(v, dist_G(x,v)-1)$ contains at least $12/\epsilon+2$, ϵ-relevant vertices. Therefore, there are more than $12/\epsilon$ edges not in $B_G(v, dist_G(x,v)-1)$, that are adjacent to vertices in $B_G(v, dist_G(x,v)-1)$. This leads to a contradiction, because it implies that $B_G(v, dist_G(x,v)-1)$ contains ϵ-Balloon$_G(v)$. □

In the following, a union of subgraphs contains the union of the vertex sets and the union of the edge sets.

Lemma 4.14. *Let G be a hard-graph, σ an assignment that is ϵ-far from $SAT(G)$ and*

$$T = \bigcup_{v \in UNSAT_G(\sigma)} \epsilon\text{-}Balloon(v).$$

Then, $|E(T)| \geq \epsilon \cdot |E(G)|/2$.

Proof. By definition $dist_G(\sigma, SAT(G)) \geq \epsilon \cdot |E(G)|$ and hence according to Proposition 4.9 in order to prove the statement of the lemma we only need to show that

$$dist_H(\sigma^{G \to G/T}, SAT(G/T)) \leq \epsilon \cdot |E(G)|/2. \tag{2}$$

According to Lemma 3.10 there exists $\nu \in SAT(G/T)$, such that $dist_{G/T}(\nu, \sigma^{G \to G/T}) \leq \frac{6 \cdot |E(G/T)|}{\Delta_{G/T}(\sigma)}$. By definition $dist_{G/T}(\sigma^{G \to G/T}, SAT(G/T)) \leq dist_{G/T}(\nu, \sigma^{G \to G/T})$ and hence

$$dist_{G/T}(\sigma^{G \to G/T}, SAT(G/T)) \leq \frac{6 \cdot |E(G/T)|}{\Delta_{G/T}(\sigma)}. \tag{3}$$

If $T = G$ then 2 is trivially true and hence we assume that $T \neq G$.

Since $|E(G)| \geq |E(G/T)|$ we can replace $|E(G/T)|$ in Equation (3) by $|E(G)|$. To infer Equation (2) it is enough to show that $\Delta_{G/T}(\sigma) \geq 12/\epsilon$.

Let $T_1, \ldots, T_k$ be the connected components of T. Recall that we chose T such that $UNSAT_G(\sigma) \subseteq V(T)$ and hence every vertex in $UNSAT_{G/T}(\sigma^{G \to G/T})$ is v_{T_i} for some i. We next show that for every $v_{T_i} \in UNSAT_{G/T}(\sigma^{G \to G/T})$, $deg_{G/T}(v_{T_i}) \geq 12/\epsilon$. This is sufficient since it implies that $\Delta_{G/T}(\sigma) \geq 12/\epsilon$.

Let T_j be such that $v_{T_j} \in UNSAT_{G/T}(\sigma^{G \to G/T})$. According to Corollary 4.5 if $v_{T_j} \in UNSAT_{G/T}(\sigma^{G \to G/T})$, then T_j is a tree that does not contain a vertex z for which $deg_G(z) = 1$. Also, by the definition of T, T_j contains ϵ-Balloon$_G(v)$ for some $v \in UNSAT_G(\sigma)$. Since T_j is a tree that does not contain a vertex of degree 1 we infer that ϵ-Balloon$_G(v)$ is also a tree that does not contain a vertex of degree 1. Thus, by the definition of ϵ-Balloon$_G(v)$ at least one of the following is true; ϵ-Balloon$_G(v)$ contains an ϵ-heavy vertex or there are at least $12/\epsilon$ edges connecting the vertices of ϵ-Balloon$_G(v)$ to the vertices of the rest of the graph. Both possibilities imply that there are at least $12/\epsilon$ edges connecting the vertices of T_j to vertices in the rest of the graph. Consequently, $deg_{G/T}(v_{T_j}) \geq 12/\epsilon$. □

4.4 Proof of Lemma 3.10

The intuition behind the proof is the following: For a vertex $v \in UNSAT_G(\sigma)$ we define a subgraph, Fix-Sub$_{G,\sigma}(v)$, that has a small number of edges and agree with σ. Thus σ could be corrected to satisfy v using a small number of edges. We would like to do the same, simultaneously, for every unsatisfied vertex. Indeed Claim 4.17 asserts a bound on the number of edges in the union, T, of these subgraphs. However, we need to take care of the consistency of the simultaneous corrections. Instead, we contract T and use induction on G/T and Proposition 4.9. We now proceed formally.

We prove the lemma by induction on $|UNSAT_G(\sigma)|$. We may assume that $\Delta_G(\sigma) \geq 7$ as otherwise the lemma immediately follows.

The definition of Fix-Sub$_{G,\sigma}(v)$ requires that each edge $e \in E(G)$ is treated as a continuous line of length 1. The distance between two points on edges is simply the minimum distance between them in the graph. This enables us to define the notion of volume at a distance r around a vertex.

Definition 4.15. $[Vol_G(v,r)]$
For $v \in V(G)$ and positive Real r, we set $Vol_G(v,r)$ to be the set of all points x on the edges of G for which $dist_G(v,x) \leq r$. We set $|(Vol_G(v,r))|$ to be the total length of the lines in $Vol_G(v,r)$.

We say that an edge e is in $Vol_G(v,r)$ if and only if every point in e is in $Vol_G(v,r)$. We say $Vol_G(v,r)$ contains a subgraph R if and only if each of its edges is in $Vol_G(v,r)$. We are now ready to define Fix-Sub$_{G,\sigma}(v)$.

Definition 4.16. **[Fix-Rad$_{G,\sigma}(v)$,Fix-Sub$_{G,\sigma}(v)$]**
Let G be a hard-graph. For every $v \in V(G)$ and assignment σ to $E(G)$ we set Fix-Rad$_{G,\sigma}(v)$ to be the minimum real r for which $Vol_{G,\sigma}(v,r)$ contains a subgraph R that agrees with σ. Let Fix-Sub$_{G,\sigma}(v)$ be a subgraph with a minimum number of edges that agrees with σ and is contained in $Vol_{G,\sigma}(v, \text{Fix-Rad}_{G,\sigma}(v))$.

Remark: The value of Fix-Rad$_{G,\sigma}(v)$ is not necessarily an integer; for example, it may be a product of 1/2 when Fix-Sub$_{G,\sigma}(v)$ is an odd cycle. Note also that according to Corollary 4.1 every hard-graph is satisfiable and hence Fix-Rad$_{G,\sigma}(v)$ and Fix-Sub$_{G,\sigma}(v)$ are well defined for every $v \in UNSAT_G(\sigma)$.

Set $T = \bigcup_{v \in UNSAT_G(\sigma)}$ Fix-Sub$_{G,\sigma}(v)$. By Proposition 4.9,

$$dist_{G/T}(\sigma, SAT(G)) \leq dist_{G/T}(\sigma^{G \to G/T}, SAT(G/T)) + |E(T)|$$

Using Claim 4.17, below, implies that

$$dist_{G/T}(\sigma, SAT(G)) \leq dist_{G/T}(\sigma^{G \to G/T}, SAT(G/T)) + \frac{2 \cdot |E(G)|}{\Delta_G(\sigma)} \tag{4}$$

Since $|UNSAT_G(\sigma)| > 0$, Proposition 4.18 below asserts that $|UNSAT_{G/T}(\sigma^{G \to G/T})| < |UNSAT_G(\sigma)|$. Thus, the induction hypothesis applied to G/T, asserts that,

$$dist_{G/T}(\sigma^{G \to G/T}, SAT(G/T)) \leq dist_{G/T}(\nu, \sigma^{G \to G/T}) \leq \frac{6 \cdot |E(G/T)|}{\Delta_{G/T}(\sigma^{G \to G/T})}. \tag{5}$$

Proposition 4.18 asserts that $\Delta_{G/T}(\sigma^{G \to G/T}) \geq 2 \cdot \Delta_G(\sigma) - 2$. Plugging this in Equation (5) and substituting $|E(G/T)|$ with $|E(G)|$ gives,

$$dist_{G/T}(\sigma^{G \to G/T}, SAT(G/T)) \leq \frac{6 \cdot |E(G)|}{2 \cdot \Delta_G(\sigma) - 2}. \tag{6}$$

Finally, as $\Delta_G(\sigma) \geq 7$, Equation (6) and Equation (4) imply the claim. □

We next state Claim 4.17 and Proposition 4.18. Their proof is in Appendix C.

Claim 4.17. *Let σ be an assignment to $E(G)$ and $T = \cup_{v \in UNSAT_G(\sigma)}$Fix-Sub$_{G,\sigma}(v)$, then*

$$|E(T)| \leq \frac{2 \cdot |E(G)|}{\Delta_G(\sigma)}$$

Proposition 4.18. *Let G be a hard-graph, σ an assignment to $E(G)$ and T the minimum subgraph of G such that Fix-Sub$_{G,\sigma}(v)$ is a subgraph of T for every $v \in UNSAT_G(\sigma)$. Then,*

$$\Delta_{G/T}(\sigma^{G \to G/T}) \geq 2 \cdot \Delta_G(\sigma) - 2.$$

Moreover, if $|UNSAT_G(\sigma)| > 0$, then $|UNSAT_{G/T}(\sigma^{G \to G/T})| < |UNSAT_G(\sigma)|$.

5 Proof of Main Result

In this section we show that $SAT(G)$ has query complexity $2^{\tilde{O}(1/\epsilon)}$ for every $G \in \mathcal{LD}_3$ (not necessarily hard). We start by showing that for G that is a simple path or a simple cycle, $SAT(G)$ has query complexity $poly(1/\epsilon)$.

5.1 Simple Cases

5.1.1 Path Test

Let G be a simple path. We show that there exists a width 2 read once oblivious leveled branching program BP_G (we formally define what an Oblivious Leveled Branching Program is further on) such that the language accepted by BP_G is $SAT(G)$. In [7] the authors show that for every property accepted by a bounded width oblivious leveled branching program there exists a test whose query complexity is $(1/\epsilon)^{O(1)}$, where the O notation hides a dependency on the width of the oblivious leveled branching program. Thus, we can use their test for BP_G.

> An *Oblivious Leveled Branching Program*, is a directed graph B, in which the nodes a partitioned into levels $L_0, \ldots, L_m$. There are two special nodes; a *start* node belonging to L_0 and an *accept* node belonging to L_m. Edges are going only from a level to nodes in a consecutive level. Each node has at most two out-going edges one of which is labeled by '0' and the other is labeled by '1'. In addition all edges in between two consecutive levels are associated with a distinct member of $X = \{x_1, \ldots, x_n\}$. An assignment σ to X defines a path starting at the *start*-node: At each level, if the out-going edges are associated with x_i, then the edge with the label identical to $\sigma(x_i)$ is chosen. An Oblivious Leveled Branching Program defines a property P in the following way: $\sigma \in P$ if and only if the path defined by σ reaches *accept*.

The width of an oblivious leveled branching program is the maximum number of vertices in a level. An Oblivious Leveled Branching Program is Read-Once if every variable in X is associated with at most one level in $L_0, \ldots, L_m$.

The following is the main result of [7].

Theorem 5.1. *Let $P \subseteq \{0,1\}^n$ be the language accepted by a read-once branching program of width w. Then testing P requires at most $\left(\frac{2^w}{\epsilon}\right)^{O(w)}$ queries.*

If G is a simple path then obviously $\phi_G = \wedge_{v \in V(G)} f_v$ can be represented as an oblivious leveled branching program, since each f_v depends only on 2 variables and each variable is shared by 2 adjacent vertices. We omit further details here.

5.1.2 Cycle test

Let G be a simple cycle. We call a test for such a graph a cycle test. Let $\{v_0, \ldots, v_n\}$ be the set of vertices $V(G)$, let $\{e_0, \ldots, e_n\}$ be the set of edges $E(G)$, where for every $i \in [n]$, e_i is adjacent to v_i and $v_{(i+1) \mod n}$.

The following algorithm is a test for $SAT(G)$. On input $\epsilon > 0$ and $\sigma \in \{0,1\}^n$ query the value $\sigma(e_n)$. Then it set G' to be the graph we get by removing the edge e_n from G and setting the function that labels v_0,v_n to be $f_{v_0}(\sigma(e_n), e_0)$ and $f_{v_n}(e_{n-1}, \sigma(e_n))$ respectively. Finally, execute the path test for G' with error parameter ϵ and oracle access to $\sigma_{\{0,\ldots,n-1\}}$.

Obviously the query complexity is $(1/\epsilon)^{O(1)}$ and an input in $SAT(G)$ is accepted with probability 1. It is easy to see that if the assignment σ is ϵ-far from G then $\sigma_{\{0,\ldots,n-1\}}$ is ϵ-far from G' as defined in the test.

5.1.3 Consistency Test

In the general case we have to deal with unsatisfied vertices of degree 2. The path test does not suffice as for σ we not only need to reject if σ is 'far' from satisfying the path but also if σ is 'far' from satisfying the path when the first and last values in the path are fixed.

Hence we use an algorithm we call a consistency test, which is a variation of the path test.

Let G be a hard-graph that is a simple path. Let $\{v_1, \ldots, v_n\}$ be the set of vertices $V(G)$, let $\{e_1, \ldots, e_{n-1}\}$ be the set of edges $E(G)$, where for every $i \in [n-1]$, e_i is an edge between v_i and v_{i+1}.

Given a distance parameter $\epsilon > 0$, the requirements from a consistency are the following:

- It accepts with probability 1 if σ satisfies G.
- It rejects with probability 1 if there is no $\alpha \in SAT(G)$ such that $\alpha(e_1) = \sigma(e_1)$ and $\alpha(e_{n-1}) = \sigma(e_{n-1})$.
- If there exists $\alpha \in SAT(G)$ such that $\alpha(e_1) = \sigma(e_1)$ and $\alpha(e_{n-1}) = \sigma(e_{n-1})$, it rejects with probability 2/3 if the minimum distance from σ to such α is at least $\epsilon \cdot |E(G)|$.

The following algorithm is a consistency test for $SAT(G)$. On input $\epsilon > 0$ and assignment σ to $|E(G)|$ query the value $\sigma(e_0), \sigma(e_n)$. Then set G' to be the graph we get from G setting $f_{v_0} = (e_0 \equiv \sigma(e_0))$ and $f_{v_n} = (e_n \equiv \sigma(e_n))$. Execute the path test for G' with error parameter ϵ. Obviously the query complexity is $(1/\epsilon)^{O(1)}$ and the algorithm behaves as required.

5.1.4 Test for Hard-Graphs

Recall that so far we have shown a test for hard-graphs that is correct for satisfying assignments or ϵ-far assignments σ for which $\Delta_G(\sigma) \geq 3$. We now want to generalize it to any ϵ-far assignment. Let G be a hard-graph or a simple path or a simple cycle and let σ be an assignment $E(G)$.

In order to introduce the required test we need the following definition.

Definition 5.2. *For every edge $e \in E(G)$ define R_e to be the maximum path in G that contains e and all its internal vertices have degree 2 in G.*

In the description of Algorithm 5.1 we say that we execute a consistency test for R_e, where e is some edge in G. What we mean is that we run the consistency test on R_e as if it was a simple path, such that for the first vertex in the path x, $f_x \equiv 1$ and for the last vertex in the path y we have $f_y \equiv 1$. All other vertices retain their original labeling.

Algorithm 5.1.

Input: ϵ, σ.

1. (a) *If G is a simple path, execute the path test with distance parameter $\epsilon/2$. If the path test rejects, then reject, otherwise accept.*
   (b) *If G is a simple cycle, execute the cycle test with distance parameter $\epsilon/2$. If the path test rejects, then reject, otherwise accept.*
2. *Other graphs:*
   (a) *Execute Algorithm 4.1 with distance parameter $\epsilon/2$. Reject if the executions of Algorithm 4.1 returned a rejection.*
   (b) *For each edge e queried by Algorithm 4.1 in the previous step execute the consistency test for R_e. Reject if one of the executions of the consistency test with distance parameter $\epsilon/4$ returned a rejection.*
   (c) *Select $64/\epsilon$ edges from $E(G)$ uniformly and independently. For each edge e selected execute the consistency test for R_e with distance parameter $\epsilon/4$. Reject if in one of the executions of the consistency test returns a rejection.*

Claim 5.3. *Algorithm 5.1 is an* $(\epsilon, 2^{\tilde{O}(1/\epsilon)})$*-test for* $SAT(G)$*, where* G *is a hard-graph or a simple path or a simple cycle.*

Obviously the query complexity of Algorithm 5.1 is as required. It is also obvious that if G is a simple path or a simple cycle then the algorithm behaves as required.

Algorithm 5.1 always accepts an input in $SAT(G)$. If σ is far from not being such that $\Delta_G(\sigma) \geq 3$, then it is easy to show that with probability at least 2/3 one of the executions of the consistency test in Step 2c will reject. Now if the input satisfies $\Delta_G(\sigma) \geq 3$, then we know that the execution of Algorithm 4.1 in Step 2a will result in a rejection with probability at least 2/3. However, there is a third possibility, it could be the case that an assignment σ does not satisfy a very small number of ϵ-relevant vertices and is also not far from satisfying all functions of vertices of degree 2.

Proof of Claim 5.3: Let G be a hard-graph and σ be ϵ-far from $SAT(G)$. We prove that Algorithm 5.1 rejects σ with probability at least 2/3.

Assume that σ is $\epsilon/2$-far from being such that $\Delta_g(\sigma) \geq 3$. Then there are at least $\epsilon \cdot |E(G)|/2$ edges in the union of paths R_e such that for every assignment η to $E(G)$ such that $\Delta_G(\eta) \geq 3$ we have $dist(\eta_{E(R_e)}, \sigma_{E(R_e)}) \geq \epsilon \cdot |E(R(e))|/4$. Hence, the probability that in step 2c an edge e is selected for such an R_e is at least $1 - (1 - 12/\epsilon)^{64/\epsilon} > 9/10$. Let e be such an edge that was selected in Step 2c. Since the consistency test is executed 3 times independently on R_e with distance parameter $\epsilon/4$ the probability that one of these execution rejects is a least $1 - (1 - 2/3)^3 > 9/10$. Thus the probability that Algorithm 5.1 rejects σ is at least 2/3.

Assume that σ is $\epsilon/2$-close to being such that $\Delta_g(\sigma) \geq 3$. Let η be an assignment with minimum distance to σ, where $\Delta_g(\eta) \geq 3$. By the triangle inequality η is $\epsilon/2$-far from $SAT(G)$. Set S to be the set of all edges adjacent to ϵ-relevant vertices such that $\sigma(e) \neq \eta(e)$.

Let us look at the execution of Algorithm 4.1 in Step 2a. We may treat this execution as if it is given oracle access to η and not σ, since if Algorithm 4.1 queries an edge e in S we will show that the execution of the consistency test on R_e in Step 2b will reject with probability 1. Consequently, as Algorithm 4.1 is executed in Step 2a with distance parameter $\epsilon/2$ it rejects with probability 2/3.

Let e be an edge from S that was selected in Step 2c. Observe that it must be the case that $|E(R_e)| > 1$. Let e' be the other edge in R_e that is connected to a vertex of degree different than 2. It can not be that there exists $\xi \in SAT(G)$ such that $\xi(e) = \sigma(e)$ and $\xi(e') = \sigma(e')$ because then there exists an assignment η' as follows. Set η' to be such that $\eta'(e) = \sigma(e)$ and $\eta'(e') = \sigma(e')$ and $\eta'(e'') = \eta(e'')$ for every $e'' \notin \{e, e'\}$. Then $\Delta_G(\eta') \geq 3$ and the distance of η' from σ is strictly less than the distance of η from σ. Therefore, there is no assignment κ that satisfies R_e such that $\kappa(e) = \sigma(e)$ and $\kappa(e') = \sigma(e')$. Consequently, when the Algorithm 5.1 executes a consistency test on R_e it rejects with probability 1. □

5.2 The General Case

So far we have shown that $SAT(G)$ has query complexity $2^{\tilde{O}(1/\epsilon)}$ if G is hard or a simple path or a simple cycle. This does not cover a number of other cases. For example if G contains a vertex v such that $f_v \equiv 0$ and therefore $SAT(G) = \emptyset$ and is trivially testable. Another case is that G is not connected. This case is also testable (assuming that we have a test for every connected graph) as follows: On input ϵ and assignment σ to $E(G)$ select a connected component T with probability $|E(M)|/|E(G)|$ and ϵ-test T. It is easy to see that after suitable amplification this becomes an ϵ-test for G.

It remains to deal with the following last case: G is connected, is not a simple path or a simple cycle, and contains a vertex v of degree 1 such f_v is not equivalent to a constant, or a vertex v of degree 2 such that f_v has two unsatisfying assignments with '*ldist*' 1 between them.

5.2.1 Last Case

An *unwanted* vertex, is a vertex v that has degree 1 and f_v is not equivalent to a constant or has degree 2 and f_v has two unsatisfying assignments with '*ldist*' 1 between them.

We show that the statement of Theorem 3.7 holds for $G \in \mathcal{LD}_3$ by using two reductions. We start by showing that there exists a graph $G^* \in \mathcal{LD}_3$ such that G^* has the same set of edge variables as G,

$SAT(G^*) = SAT(G)$ and there is no unwanted vertex of degree 2 in G^*. The graph G^* is not necessarily connected and may contain vertices v of degree 1 for which f_v is not equivalent to a constant.

We finish by showing that there exists a graph $G^{**} \in \mathcal{LD}_3$ such that G^{**} has the same set of edge variables as G^* , $SAT(G^{**}) = SAT(G^*)$, and G^{**} consists of connected components such that each one of them is either a simple cycle or a simple path or does not contain any unwanted vertex.

We set G^* to be the graph we get from G by applying the following operation to every unwanted vertex v of degree 2 in the graph.

We refer to the edges adjacent to v by e_1 and e_2. We remove v from the set of vertices and insert vertices t_v^1, t_v^2 of degree 1 instead, where t_v^1 is adjacent to e_1 and t_v^2 is adjacent to e_2. Set $f_{t_v^1}, f_{t_v^2}$ as follows:

- If v has only one satisfying assignment ν. Without loss of generality assume that $f_v = e_1 \wedge e_2$, then set $f_{t_v^1} = t_v^1$ and $f_{t_v^2} = t_v^2$.

- If v has two unsatisfying assignment ν_1, ν_2, then $f_{t_v^1}, f_{t_v^2}$ are defined as follows. Assume with loss of generality that $\nu_1(e_1) = \nu_2(e_1)$. Set $f_{t_v^1} = (t_v^1 \equiv \neg\nu(e_1))$ and $f_{t_v^2} \equiv 1$.

Obviously the variables of G are the same as the variables of G^*, and there are no unwanted vertices of degree 2 in G^*. By definition an assignment satisfying f_v, where v is an unwanted vertex of degree 2 in G also satisfies $f_{t_v^1}$,$f_{t_v^2}$ in G^*. Since all other vertices where untouched we get that $SAT(G^*) \equiv SAT(G^*)$ and $G^* \in \mathcal{LD}_3$.

We set G^{**} to be the graph we get from G^* by applying the following operation as long there is an unwanted vertex v of degree 1 in a connected component that is not a simple path.

Let S be the minimum path from v to a vertex u of degree at least 3 in the graph. Let e be the edge in S that is connected to u. Since there are no unwanted vertices of degree 2 in the graph, there exists $a \in \{0, 1\}$ such that for every assignment σ that satisfies the graph $\sigma(e) = a$.

We disconnect e from u and set e to be adjacent to a new vertex t, where $f_t \equiv 1$. We change the name of the vertex u to u' and set $f_{u'} = f_u \mid_{e=a}$.

Note that the size of the connected components that are not a simple path decreases in each iteration, and hence the process eventually ends. Obviously the variables of $\phi_{G^{**}}$ are the same as the variables of ϕ_{G^*}, $SAT(G^{**}) = SAT(G^*)$. It is also easy to see that $G^{**} \in \mathcal{LD}_3$.

References

[1] E. Ben-Sasson, P. Harsha, and S. Raskhodnikova, *Some 3-CNF properties are hard to test, STOC* 2003, pp. 345–354.

[2] M. Blum, M. Luby, and R. Rubinfeld, *Self testing/correcting with applications to numerical problems, Journal of Computer and System Science* 47:549–595, 1993.

[3] I. Dinur, *The PCP Theorem by Gap Amplification, STOC* 2006, pp. 241–250, 2006.

[4] E. Fischer, *The art of uninformed decisions: A primer to property testing, The Computational Complexity Column of The bulletin of the European Association for Theoretical Computer Science*, 75:97–126, 2001.

[5] E. Fischer, E. Lehman, I. Newman, S. Raskhodnikova, R. Rubinfeld, and A. Samorodnitsky, *Monotonicity testing over general poset domains, STOC* 2002, pp. 474–483.

[6] O. Goldreich, S. Goldwasser, and D. Ron, *Property testing and its connection to learning and approximation, Journal of the ACM*, 45(4):653–750, 1998.

[7] I. Newman, *Testing Membership in Languages that Have Small Width Branching Programs, SIAM Journal on Computing* 31(5):1557–1570, 2002.

[8] D. Ron, *Property testing (a tutorial), Handbook of Randomized Computing* (S.Rajasekaran, P. M. Pardalos, J. H. Reif and J. D. P. Rolin eds), Kluwer Press (2001).

[9] R. Rubinfeld and M. Sudan, *Robust characterization of polynomials with applications to program testing, SIAM Journal of Computing*, 25(2):252–271, 1996.

Appendix

A Proof of Claim 4.4

We prove the claim for the case that T is connected. This is, sufficient since otherwise we can take the connected component of T that contains v.

Assume that T contains a cycle. Then by Claim 4.3, the minimal connected subgraph that contains both the cycle and v is as required.

We conclude the proof by considering the case when T is a tree. Let S be a minimum subgraph of T that agrees with σ, and contains v and let $\sigma^* \overset{S}{\sim} \sigma$ be an assignment that satisfies f_x for every $x \in S$. Let E' be the set of edges on which σ^* differs from σ. We may assume that $E' = E(S)$ as otherwise it would contradict the minimality of S. We may also assume that there exists $u \in V(S)$, where $deg_S(u) \geq 3$, and that S is connected (otherwise we are done, as S is a simple path or cycle). Let e_1, e_2 be arbitrary edges adjacent to u, not on the path from v (if $u = v$ then e_1, e_2 are arbitrary edges adjacent to v). Let σ_1 be identical to σ^* on every edge but e_1 on which it assigns the same value as σ, and let σ_2 be identical to σ^* every where but e_1 and e_2.

Note that for every $x \neq v$ in the connected component that contains v in $S \setminus \{e_1\}$, both σ_1 and σ^* assign the same values to all edges adjacent to x. Similarly, for every $x \neq v$ in the connected component of $S \setminus \{e_1, e_2\}$ that contains v, σ_2 and σ^* assign the same values to all edges adjacent to x. Thus σ_1,σ_2 satisfy every vertex in the appropriate component containing v, except possibly u. However $ldist_u(\sigma_1, \sigma_2) = 1$ and hence at least one of σ_1,σ_2 must also satisfy u. On the other hand, both σ_1 and σ_2 differ from σ in at most $|S| - 1$ edges in contradiction to the minimality of S. $\square$

B Proof of Claim 4.6

By the definition of contraction G/T is connected. Each vertex in $V(G/T) \cap V(G)$ is labeled by the same function in G/T as in G and hence to conclude the claim we only need to show that for every connected subgraph M of T the function f_{v_M} in G/T is as required by the definition of a hard-graph.

We show that f_{v_T} is as required, by induction on $|E(T)|$. For $|E(T)| = 0$ $G/T = G$ and there is nothing to show. Hence, it is enough to prove the claim for a single edge e since for any subgraph T and $e \in T$, and then by induction we will be done.

If e is a loop then by Claim 4.3 and the definition of subgraph contraction, $f_{v_e} \equiv 1$ in G/T and we are done. The same is also true if $deg_{G/\{e\}}(v_e) = 0$ (by Corollary 4.1).

According to the definition of hard-graphs it remains to show that:

1. If $deg_{G/\{e\}}(v_e) = 1$, then $f_{v_e} \equiv 1$.
2. If $deg_{G/\{e\}}(v_e) = 2$, then $ldist_{G/\{e\},v_e}(\eta_1, \eta_2) = 2$ for every two assignments η_1, η_2 that do not satisfy f_{v_e}.
3. If $deg_{G/\{e\}}(v_e) \geq 3$, then $ldist_{G/\{e\},v_e}(\eta_1, \eta_2) \geq 3$ for every two assignments η_1, η_2 that do not satisfy f_v.

By simple case study it can be shown that this indeed the case. We do not provide further details in this draft. □

C Proof of Claim 4.17 and Proposition 4.18

In order to prove Claim 4.17 and Proposition 4.18 we need the following claim.

Claim C.1. *Let G be a hard-graph and σ an assignment to $E(G)$. Then, for every $v \in UNSAT_G(\sigma)$ we have*

$$|E(\textit{Fix-Sub}_{G,\sigma}(v))| \leq \frac{4}{\Delta_G(\sigma)} \cdot \left| Vol_G\left(v, \frac{\textit{Fix-Rad}_{G,\sigma}(v)}{2}\right)\right|.$$

Proof. Let G, σ be as in the claim and $v \in UNSAT_G(\sigma)$. We first show that

$$\left| Vol_G\left(v, \frac{\text{Fix-Rad}_{G,\sigma}(v)}{2}\right)\right| \geq \Delta_G(\sigma) \cdot \frac{\text{Fix-Rad}_{G,\sigma}(v)}{2} \tag{7}$$

Indeed, it is enough to show that for every $\delta \geq 0$, $Vol_G(v, \text{Fix-Rad}_{G,\sigma}(v) - \delta)$ is a tree rooted in v, and does not contain a vertex x for which $deg_G(x) = 1$. This is sufficient since G is a finite graph and therefore $Vol_G(v, \text{Fix-Rad}_{G,\sigma}(v)) \geq \deg_G(v) \cdot \text{Fix-Rad}_{G,\sigma}(v)$. Consequently, $Vol_G(v, \text{Fix-Rad}_{G,\sigma}(v)/2) \geq \Delta_G(v) \cdot \text{Fix-Rad}_{G,\sigma}(v)/2$.

Let $\delta \geq 0$ be the maximum real such that $Vol_G(v, \text{Fix-Rad}_{G,\sigma}(v) - \delta)$ contains a subgraph that contains a cycle or a vertex of degree 1. We show next that $\delta = 0$. Let K be a subgraph contained in $Vol_G(v, \text{Fix-Rad}_{G,\sigma}(v) - \delta)$ that either contains a cycle or contains a vertex degree 1 or both. If K contains a cycle or contains a vertex u for which $deg_G(u) = 1$, then by Corollary 4.5 there exists an assignment σ^* to $E(G)$ that agrees with K. Thus, by the definition of $\text{Fix-Rad}_{G,\sigma}(v)$ we get that $\text{Fix-Rad}_{G,\sigma}(v) = \text{Fix-Rad}_{G,\sigma}(v) - \delta$. Consequently, $\delta = 0$ which implies Equation (7).

We conclude the claim by showing that

$$2 \cdot \text{Fix-Rad}_{G,\sigma}(v) \geq |E(\text{Fix-Sub}_{G,\sigma}(v))| \tag{8}$$

We do so by showing that in each of the following cases there exists a subgraph S in $Vol_G(v, \text{Fix-Rad}_{G,\sigma}(v))$ that agrees with σ and satisfies $2 \cdot |E(S)| \leq 2\text{Fix-Rad}_{G,\sigma}(v)$.

1. $Vol_G(v, \text{Fix-Rad}_{G,\sigma}(v))$ is a tree.

2. $Vol_G(v, \text{Fix-Rad}_{G,\sigma}(v))$ contains a vertex of degree 1.

3. $Vol_G(v, \text{Fix-Rad}_{G,\sigma}(v))$ contains a cycle.

This is sufficient, since the definition of $\text{Fix-Sub}_{G,\sigma}(v)$ implies that $|E(\text{Fix-Sub}_{G,\sigma}(v))| \leq |E(S)|$ and hence Equation (8) holds.
(1) By Claim 4.4 and the fact that $\text{Fix-Sub}_{G,\sigma}(v)$ is a subgraph that agrees with σ we infer that $Vol_G(v, \text{Fix-Rad}_{G,\sigma}(v))$ contains a subgraph S that agrees with σ and is a simple path. Since $Vol_G(v, \text{Fix-RAD}_{G,\sigma}(v))$ is a tree of diameter at most $2 \cdot \text{Fix-Rad}_{G,\sigma}(v)$ we get that $|E(S)| \leq 2 \cdot \text{Fix-Rad}_{G,\sigma}(v)$.
(2) Let u be a vertex of degree 1 in $Vol_G(v, \text{Fix-Rad}_{G,\sigma}(v))$. Let S be a shortest path between v and u. By definition $|E(S)| \leq \text{Fix-Rad}_{G,\sigma}(v)$ and S is contained in $Vol_G(v, \text{Fix-Rad}_{G,\sigma}(v))$. According to Corollary 4.5, the fact that S contains a vertex u for which $deg_w(u) = 1$, implies that S agrees with σ.
(3) Recall $Vol_G(v, \text{Fix-Rad}_{G,\sigma}(v) - \delta)$ is a tree for every $\delta > 0$. Hence, there exists a point in $Vol_G(v, \text{Fix-Rad}_{G,\sigma}(v))$ and S_1,S_2, that are different shortest 'paths' in $Vol_G(v, \text{Fix-Rad}_{G,\sigma}(v))$ from the point to v (note that it might be the case that this point is not a vertex). Set S to be the minimum subgraph that contains both S_1 and S_2. S contains a cycle and hence according to Corollary 4.5 it agrees with σ. The maximum distance between v and a point in $Vol_G(v, \text{Fix-Rad}_{G,\sigma}(v))$ and hence the length of S_1,S_2 is at most $\text{Fix-Rad}_{G,\sigma}(v)$. Thus, S is a required. □

C.1 Proof of Claim 4.17

Let G,σ,T be as in the claim. By definition

$$|E(T)| \leq \sum_{v \in UNSAT_G(\sigma)} |E(\text{Fix-Sub}_{G,\sigma}(v))|. \tag{9}$$

According to Claim C.1, for every $v \in UNSAT_G(\sigma)$

$$|E(\text{Fix-Sub}_{G,\sigma}(v))| \leq \frac{4}{\Delta_G(\sigma)} \cdot \left| Vol_G\left(v, \frac{\text{Fix-Rad}_{G,\sigma}(v)}{2}\right)\right|. \tag{10}$$

By plugging Equation 10 into Equation 9 we get that

$$|E(T)| \leq \frac{4}{\Delta_G(\sigma)} \cdot \sum_{v \in UNSAT_G(\sigma)} \left| Vol_G\left(v, \frac{\text{Fix-Rad}_{G,\sigma}(v)}{2}\right)\right|. \tag{11}$$

Thus, in order to prove the statement of the claim it is enough to show that the following Equation holds

$$\sum_{v \in UNSAT_G(\sigma)} \left| Vol_G\left(v, \frac{\text{Fix-Sub}_{G,\sigma}(v)}{2}\right)\right| \leq |E(G)|. \tag{12}$$

We do this by showing that for every two different vertices $u, v \in UNSAT_G(\sigma)$,

$$dist_G(u, v) \geq \max\{\text{Fix-Rad}_{G,\sigma}(u), \text{Fix-Rad}_{G,\sigma}(v)\}. \tag{13}$$

This is sufficient since it implies that $|Vol_G(v, \text{Fix-Rad}_{G,\sigma}(v)/2) \cap Vol_G(u, \text{Fix-Rad}_{G,\sigma}(u)/2))| = 0$ and since G is finite we infer that Equation 12 holds.

Assume for the sake of contradiction that $u, v \in UNSAT_G(\sigma)$ are two different vertices such that $dist_G(u, v) < \max\{\text{Fix-Rad}_{G,\sigma}(u), \text{Fix-Rad}_{G,\sigma}(v)\}$. With out loss of generality assume that $\text{Fix-Rad}_{G,\sigma}(v) = \max\{\text{Fix-Rad}_{G,\sigma}(u), \text{Fix-Rad}_{G,\sigma}(v)\}$ and hence for some $\delta > 0$ $dist_G(u, v) = \text{Fix-Rad}_{G,\sigma}(v) - \delta$.

Let R be a shortest path between u and v, x be the vertex in R adjacent to v and $e' \in E(R)$ be the edge between v and x. Let R' be the graph we get from R by removing the edge e' and the vertex v. According to

Claim 4.2 there exists an assignment σ^* such that $\sigma^* \overset{R'}{\sim} \sigma$ and $UNSAT_G(\sigma^*) \cap V(R') \subseteq \{x\}$. Observe that by definition of R', $|E(R')| \leq \text{Fix-Rad}_{G,\sigma}(v) - \delta$ and hence R' is contained in $Vol_G(v, \text{Fix-Rad}_{G,\sigma}(v)) - \delta)$. Consequently, $UNSAT_G(\sigma^*) \cap V(R') = \{x\}$, since otherwise thus contradicts the definition of $\text{Fix-Rad}_{G,\sigma}(v)$.

Set σ' to be an assignment to $E(G)$ such that $\sigma'(e') = \neg\sigma^*(e')$ and $\sigma'(e) = \sigma^*(e)$ for every $e \in E(G) \setminus \{e'\}$. Since $x, u \in UNSAT_G(\sigma^*)$ and $G \in \mathcal{LD}_3$ and $ldist_{G,x}(\sigma', \sigma^*) = 1$ and $ldist_{G,u}(\sigma', \sigma^*) = 1$ then $x, u \notin UNSAT_G(\sigma')$ and hence $UNSAT_G(\sigma') \cap V(R) = \emptyset$. That is, R agrees with σ in contradiction to the choice of $\text{Fix-Rad}_{G,\sigma}(v)$. □

C.2 Proof of Proposition 4.18

Let G,σ,T be as in the claim. As $UNSAT_G(\sigma) \subseteq V(T)$, each vertex in $UNSAT_{G/T}(\sigma^{G \to G/T})$ is of the sort v_M, where M is a connected component of T. We show now that $|V(M) \cap UNSAT_G(\sigma)| \geq 2$ for every such M. Afterwards we show how this implies the proposition.

Assume for the sake of contradiction that there exists such M for which $|V(M) \cap UNSAT_G(\sigma)| < 2$. By definition, there exists $v \in UNSAT_G(\sigma)$ such that M contains $\text{Fix-Sub}_{G,\sigma}(v)$. Therefore, $|V(M) \cap UNSAT_G(\sigma)| = 1$. Let $u \in UNSAT_G(\sigma) \cap V(M)$. According to choice of T as a minimum subgraph we get that $M = \text{Fix-Sub}_{G,\sigma}(u)$. Consequently, according to the definition of Fix-Sub and subgraph contraction we get that $v_M \notin UNSAT_{G/T}(\sigma^{G \to G/T})$ in contradiction to the choice of M.

According to Corollary 4.5 if a connected component M is such that $v_M \in UNSAT_{G/T}(\sigma^{G \to G/T})$, then M is a tree. This together with the fact that for every connected component M of T, $|V(M) \cap UNSAT_G(\sigma)| \geq 2$, implies that every vertex in $UNSAT_{G/T}(\sigma^{G \to G/T})$ has degree at least $2 \cdot \Delta_G(\sigma) - 2$. This proves the first part of the proposition.

We have shown that for every vertex in $UNSAT_{G/T}(\sigma^{G \to G/T})$ there exists at least two distinct vertices in $UNSAT_G(\sigma)$. This implies the second part of the proposition. □

ECCC ISSN 1433-8092

http://eccc.hpi-web.de/