Finding Optimal Solutions With Neighborly Help

Elisabet Burjons
Department of Computer Science, ETH Zürich, Universitätstrasse 6, CH-8092 Zürich, Switzerland
eburjons@inf.ethz.ch

Fabian Frei
Department of Computer Science, ETH Zürich, Universitätstrasse 6, CH-8092 Zürich, Switzerland
fabian.frei@inf.ethz.ch

Edith Hemaspaandra
Department of Computer Science, Rochester Institute of Technology, Rochester, NY 14623, USA
eh@cs.rit.edu

Dennis Komm
Department of Computer Science, ETH Zürich, Universitätstrasse 6, CH-8092 Zürich, Switzerland
dennis.komm@inf.ethz.ch

David Wehner
Department of Computer Science, ETH Zürich, Universitätstrasse 6, CH-8092 Zürich, Switzerland
david.wehner@inf.ethz.ch

Abstract

Can we efficiently compute optimal solutions to instances of a hard problem from optimal solutions to neighboring (i.e., locally modified) instances? For example, can we efficiently compute an optimal coloring for a graph from optimal colorings for all one-edge-deleted subgraphs? Studying such questions not only gives detailed insight into the structure of the problem itself, but also into the complexity of related problems; most notably graph theory's core notion of critical graphs (e.g., graphs whose chromatic number decreases under deletion of an arbitrary edge) and the complexity-theoretic notion of minimality problems (also called criticality problems, e.g., recognizing graphs that become 3-colorable when an arbitrary edge is deleted).

We focus on two prototypical graph problems, Colorability and Vertex Cover. For example, we show that it is NP-hard to compute an optimal coloring for a graph from optimal colorings for *all* its one-vertex-deleted subgraphs, and that this remains true even when optimal solutions for *all* one-edge-deleted subgraphs are given. In contrast, computing an optimal coloring from all (or even just two) one-edge-added supergraphs is in P. We observe that Vertex Cover exhibits a remarkably different behavior, demonstrating the power of our model to delineate problems from each other more precisely on a structural level.

Moreover, we provide a number of new complexity results for minimality and criticality problems. For example, we prove that MINIMAL-3-UNCOLORABILITY is complete for DP (differences of NP sets), which was previously known only for the more amenable case of deleting vertices rather than edges. For Vertex Cover, we show that recognizing β-vertex-critical graphs is complete for Θ_2^{p} (parallel access to NP), obtaining the first completeness result for a criticality problem for this class.

2012 ACM Subject Classification Theory of computation → Problems, reductions and completeness; Theory of computation → Complexity classes

Keywords and phrases Critical Graphs, Computational Complexity, Structural Self-Reducibility, Minimality Problems, Colorability, Vertex Cover, Satisfiability, Reoptimization, Advice

Digital Object Identifier 10.4230/LIPIcs.MFCS.2019.78

Related Version A full version of this paper, which includes the appendices, is published on arXiv [3] and available at `https://arxiv.org/abs/1906.10078`.

Funding *Edith Hemaspaandra*: Research done in part while on sabbatical at ETH Zürich.

Acknowledgements We thank the anonymous referees and Hans-Joachim Böckenhauer, Rodrigo R. Gumucio Escobar, Lane Hemaspaandra, Juraj Hromkovič, Rastislav Královič, Richard Královič, Xavier Muñoz, Martin Raszyk, Peter Rossmanith, Walter Unger, and Koichi Wada for helpful comments and discussions.

44th International Symposium on Mathematical Foundations of Computer Science (MFCS 2019).
Editors: Peter Rossmanith, Pinar Heggernes, and Joost-Pieter Katoen; Article No. 78; pp. 78:1–78:14
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Introduction and Related Work

In Subsection 1.1, we introduce and motivate our new model, which we then compare and contrast to related notions in Subsection 1.2. Finally, we present in Subsection 1.3 an overview of our most interesting results and place them into the context of the wider literature.

1.1 Our Model

In view of the almost complete absence of progress in the question of P versus NP, it is natural to wonder just how far and in what way these sets may differ. For example, how much additional information enables us to design an algorithm that solves an otherwise NP-hard problem in polynomial time? We are specifically interested in the case where this additional information takes the form of optimal solutions to neighboring (i.e., locally modified) instances. This models situations such as that of a newcomer who may ask experienced peers for advice on how to solve a difficult problem, for instance finding an optimal work route. Similar circumstances arise when new servers join a computer network. Formally, we consider the following oracle model: An algorithm may, on any given input, repeatedly select an arbitrary instance neighboring the given one and query the oracle for an optimal solution to it. Occasionally, it will be interesting to limit the number of queries that we grant the algorithm. In general, we do not impose such a restriction, however.

What precisely constitutes a local modification and thus a neighbor depends on the specific problem, of course. We examine the prototypical graph problems Colorability and Vertex Cover, considering the following four local modifications, which are arguably the most natural choices: deleting an edge, adding an edge, deleting a vertex (including adjacent edges), and adding a vertex (including an arbitrary, possibly empty, set of edges from the added vertex to the existing ones). For example, we ask whether there is a polynomial-time algorithm that computes a minimum vertex cover for an input graph G if it has access to minimum vertex covers for all one-edge-deleted subgraphs of G. We will show that questions of this sort are closely connected to and yet clearly distinct from research in other areas, in particular the study of critical graphs, minimality problems, self-reducibility, and reoptimization.

1.2 Related Concepts

Criticality. The notion of criticality was introduced into the field of graph theory by Dirac [7] in 1952 in the context of Colorability with respect to vertex deletion. Thirty years later, Wessel [19] generalized the concept to arbitrary graph properties and modification operations. Nevertheless, Colorability has remained a central focus of the extensive research on critical graphs. Indeed, a graph G is called *critical* without any further specification if it is χ-critical under edge deletion, that is, if its chromatic number $\chi(G)$ (the number of colors used in an optimal coloring of G) changes when an arbitrary edge is deleted. Besides Colorability, one other problem has received a comparable amount of attention and thorough analysis in three different manifestations: Independent Set, Vertex Cover, and Clique. The corresponding notions are α-criticality, β-criticality, and ω-criticality, where α is the *independence number* (size of a maximum independent set), β is the *vertex cover number* (size of a minimum vertex cover), and ω is the *clique number* (size of a maximum clique). Note that these graph numbers are all monotone – either nondecreasing or nonincreasing – with respect to each of the local modifications examined in this paper.

Minimality. Another strongly related notion is that of minimality problems. An instance is called *minimal* with respect to a property if only the instance itself but none of its neighbors has this property; that is, it inevitably loses the property under the considered local modification. The corresponding minimality problem is to decide whether an instance is minimal in the described sense. For example, a graph G is minimally 3-uncolorable (with respect to edge deletion) if it is not 3-colorable, yet all its one-edge-deleted neighbors are. The minimality problem MINIMAL-3-UNCOLORABILITY is the set of all minimally 3-uncolorable graphs. Note that a graph is critical exactly if it is minimally k-uncolorable for some k.

While minimality problems tend to be in DP (i.e., differences of two NP sets, the second level of the Boolean hierarchy), DP-hardness is so difficult to prove for them that only a few have been shown to be DP-complete so far; see for instance Papadimitriou and Wolfe [14]. Note that the notion of minimality is not restricted to graph problems. Indeed, minimally unsatisfiable formulas figure prominently in many of our proofs.

Auto-Reducibility. Our model provides a refinement of the notion of functional auto-reducibility; see Faliszewski and Ogihara [8]. An algorithm solves a function problem $R \subseteq \Sigma_1^* \times \Sigma_2^*$ if on input $x \in \Sigma_1^*$ it outputs some $y \in \Sigma_2^*$ with $(x, y) \in R$. The problem R is *auto-reducible* if there is a polynomial-time algorithm with unrestricted access to an oracle that provides solutions to all instances except x itself. The task of finding an optimal solution to a given instance is a special kind of function problem. Defining all instances to be neighbors (local modifications) of each other lets the two concepts coincide.

Self-Reducibility. Self-reducibility is auto-reducibility with the additional restriction that the algorithm may query the oracle only on instances that are smaller in a certain way. There are a multitude of definitions of self-reducibility that differ in what exactly is considered to be "smaller," the two seminal ones stemming from Schnorr [15] and from Meyer and Paterson [13]. For Schnorr, an instance is smaller than another one if its encoding input string is strictly shorter. While his definition does allow for functional problems (i.e., more than mere decision problems, in particular the problem of finding an optimal solution), it is too restrictive for self-reducibility to encompass our model since not all neighboring graphs have shorter strings under natural encodings.

Meyer and Paterson are less rigid and allow instead any partial order having short downward chains to determine which instances are considered smaller than the given one.[1] The partial orders induced by deleting vertices, by deleting edges, and by adding edges all have short downward chains. The definition by Meyer and Paterson is thus sufficient for our model to become part of functional self-reducibility for all local modifications considered in this paper but one, namely, the case of adding a vertex, which is too generous a modification to display any particularly interesting behavior.

As an example, consider the graph decision problem COLORABILITY $= \{(G, k) \mid \chi(G) \leq k\}$, which is self-reducible by the following observation. Any graph G with at least two vertices that is not a clique is k-colorable exactly if at least one of the polynomially many graphs that result from merging two non-adjacent vertices in G is k-colorable. This works for the optimization variant of the problem as well. Any optimal coloring of G assigns at

[1] Formally, a partial order is said to *have short downward chains* if the following condition is satisfied: There is a polynomial p such that every chain decreasing with respect to the considered partial order and starting with some string x is shorter than $p(|x|)$ and such that all strings preceding x in that order are bounded in length by $p(|x|)$.

least two vertices the same color, except in the trivial case of G being a clique. An optimal coloring for the graph that has two such vertices merged then yields an optimal coloring for G. This contrasts well with the findings for Colorability's behavior under our new model discussed below.

Reoptimization. Reoptimization examines optimization problems under a model that is tightly connected to ours. The notion of reoptimization was coined by Schäffter [16] and first applied by Archetti et al. [1]. The reoptimization model sets the following task for an optimization problem:

> *Given an instance, an optimal solution to it, and a local modification of this instance, compute an optimal solution to the modified instance.*

The proximity to our model becomes clearer after a change of perspective. We reformulate the reoptimization task by reversing the roles of the given and the modified instance.

> *Given an instance, a local modification of it, and an optimal solution to the modified instance, compute an optimal solution to the original instance.*

Note that this perspective switch flips the definition of local modification; for example, edge deletion turns into edge addition. Aside from this, the task now reads almost identical to that demanded in our model. The sole but crucial difference is that in reoptimization, the neighboring instance and the optimal solution to it are given as part of the input, whereas in our model, the algorithm may select any number of neighboring instances and query the oracle for optimal solutions to them. Even if we limit the number of queries to just one, our model is still more generous since the algorithm is choosing (instead of being given) the neighboring instance to which the oracle will supply an optimal solution. Thus, hardness in our model always implies hardness for reoptimization, but not vice versa. In fact, all problems examined under the reoptimization model so far remain NP-hard. Only for some of them could an improvement of the approximation ratio be achieved after extensive studies, the first discovered examples being TSP under edge-weight changes [4] and addition or deletion of vertices [2]. This stands in stark contrast to the results for our model, as outlined in the next section.

1.3 Results

We shed a new light on two of the most prominent and well-examined graph problems, Colorability and Vertex Cover. Our results come in two different types.

The first type concerns the hardness of the two problems in our model for the most common local modifications; Table 1 summarizes the main results of this type. In addition, Corollaries 2 and 14 show that Satisfiability and Vertex Cover remain NP-hard for any number of queries if the local modification is the deletion of a clause or a triangle, respectively. The results for the vertex-addition columns are trivial since we can just query an optimal solution for the graph with an added isolated vertex; see Theorem 16 in Appendix A [3]. The hardness results for the one-query case all follow from the same simple Theorem 17, variations of which appear in the study of self-reducibility and many other fields; see Appendix B [3]. The findings of Theorems 10, 12, and 19 in Appendix F [3] clearly delineate our model from that in reoptimization, where the NP-hard problems examined in the literature remain NP-hard despite the significant amount of advice in form of the provided optimal solution; see Böckenhauer et al. [5].

Table 1 An overview of our results regarding the hardness of Colorability and Vertex Cover in our model for the most common definitions of a local modification. The v stands for a vertex and the e stands for an edge. The question mark indicates an interesting open problem. The results in the vertex-addition columns are trivial; see Theorem 16 in Appendix A [3]. The NP-hardness results for the 1-query case all follow from rather simple Turing reductions; see Theorem 17 in Appendix B [3].

No. of Queries	Colorability				Vertex Cover			
	Add v	Delete v	Add e	Delete e	Add v	Delete v	Add e	Delete e
1	P	NP-hard	NP-hard	NP-hard	P	NP-hard	NP-hard	NP-hard
2 or more	P [Thm. 16]	NP-hard [Thm. 4]	P [Thm. 10]	NP-hard [Thm. 6]	P [Thm. 16]	P [Thm. 12]	P [Thm. 19]	?

The results of the second type locate criticality problems in relation to the complexity classes DP and Θ_2^{p}. The class Θ_2^{p} was introduced by Wagner [17] and represents the languages that can be decided in polynomial time by an algorithm that has access to an NP oracle under the restriction that all queries are submitted at the same time. The definitions of the classes immediately yield the inclusions $\mathrm{NP} \cup \mathrm{coNP} \subseteq \mathrm{DP} \subseteq \Theta_2^{\mathrm{p}}$.

Papadimitriou and Wolfe [14] have shown that MINIMAL-UNSAT (the set of unsatisfiable formulas that become satisfiable when an arbitrary clause is deleted) is DP-complete. Cai and Meyer [6] built upon this to prove DP-completeness of VERTEXMINIMAL-k-UNCOLORABILITY (the set of graphs that are not k-colorable but become k-colorable when an arbitrary vertex is deleted), for all $k \geq 3$. With Theorems 7 and 8, we were able to extend this result to classes that are analogously defined for the much smaller local modification of edge deletion, which is considered the default setting; namely, we prove DP-completeness of MINIMAL-k-UNCOLORABILITY, for all $k \geq 3$.

In Theorem 9, we show that recognizing criticality and vertex-criticality are in Θ_2^{p} and DP-hard. As Joret [12] points out, a construction by Papadimitriou and Wolfe [14] proves the DP-hardness of recognizing β-critical graphs. This problem also lies in Θ_2^{p}, but no finer classification has been achieved so far. In Theorem 15, we show that this problem is in fact Θ_2^{p}-hard, yielding the first known Θ_2^{p}-completeness result for a criticality problem.

2 Preprocessing 3-SAT

Our main technique for proving the nontrivial hardness results in our model is the following: We build in polynomial-time computable solutions for each locally modified problem instance. That way, the solutions to the locally modified problem instances do not give away any information about the instance to be solved. A similar approach is taken in some proofs of DP-completeness for minimality problems. Indeed, we can occasionally combine the proof of DP-hardness with that of the NP-hardness of computing an optimal solution from optimal solutions to locally modified instances. Denote by 3-CNF the set of nonempty CNF-formulas with exactly three distinct literals per clause.[2] We begin by showing in Theorem 1 that there is a reduction from 3-SAT (the set of satisfiable 3-CNF-formulas) to 3-SAT that builds in polynomial-time computable solutions for all one-clause-deleted subformulas of the resulting 3-CNF-formula. At first glance, this very surprising result may seem dangerously close to

[2] This set is often denoted E3-CNF in the literature.

proving P = NP; Corollary 2 will make explicit where the hardness remains. We will then use the reduction of Theorem 1 as a preprocessing step in reductions from 3-Sat to other problems.

▶ **Theorem 1.** *There is a polynomial-time many-one reduction f from 3-Sat to 3-Sat and a polynomial-time computable function s such that, for every 3-CNF-formula Φ and for every clause C in $f(\Phi)$, $s(f(\Phi) - C)$ is a satisfying assignment for $f(\Phi) - C$.*

Proof. Papadimitriou and Wolfe [14, Lemma 1] give a reduction from 3-UnSat to Minimal-UnSat (the set of CNF-formulas that are unsatisfiable but that become satisfiable with the removal of an arbitrary clause). In Appendix C [3], we show how to enhance this reduction such that it has all properties of our theorem. First, we carefully prove that there is a function s that together with the original reduction satisfies all properties of our theorem, except that we may output a formula that is not in 3-CNF. In order to rectify this, we show that the standard reduction from Sat to Sat that decreases the number of literals per clause to at most three maintains all the required properties. The same is then shown for the standard reduction that transforms CNF-formulas with at most three literals per clause into 3-CNF-formulas that have exactly three distinct literals per clause. Combining these three reductions, we can therefore satisfy all requirements of our theorem. ◀

▶ **Corollary 2.** *Computing a satisfying assignment for a 3-CNF-formula whose one-clause-deleted subformulas all have a satisfying assignment from these assignments is NP-hard.*

Proof. Given a 3-CNF-formula Φ, compute $f(\Phi)$, where f is the reduction from Theorem 1. Now compute $s(f(\Phi) - C)$ for every clause C in $f(\Phi)$ and compute a satisfying assignment for $f(\Phi)$ from these solutions. Use this assignment to determine whether Φ is satisfiable. ◀

3 Colorability

As mentioned in the previous section, the constructions of some DP-completeness results for minimality problems can be adapted to obtain NP-hardness for computing optimal solutions from optimal solutions to locally modified instances. There are remarkably few complexity results for minimality problems; fortunately, however, VertexMinimal-3-UnColorability (the graphs that are not 3-colorable but that are 3-colorable after deleting any vertex)[3] is DP-complete by reduction from Minimal-3-UnSat [6]. We will show how to extract from said reduction a proof of the fact that computing an optimal coloring for a graph from optimal colorings for its one-vertex-deleted subgraphs is NP-hard (Theorem 4). However, the standard notion of criticality is χ-criticality under edge deletion, and the construction by Cai and Meyer [6] does unfortunately not yield the analogous result for deleting edges. This was to be expected, since working with edge deletion is much harder. Surprisingly, however, a targeted modification of the constructed graph allows us to establish, through a far more elaborate case distinction, that computing an optimal coloring for a graph from optimal colorings for its one-edge-deleted subgraphs is NP-hard (Theorem 6) as well as that the related minimality problem Minimal-3-UnColorability is DP-complete (Theorem 7).

[3] It should be noted that VertexMinimal-3-UnColorability is denoted by Minimal-3-UnColorability by Cai and Meyer [6] despite the fact that minimality problems usually refer to the case of edge deletion.

▶ **Lemma 3.** *There is a polynomial-time many-one reduction g from* 3-Sat *to* 3-Colorability *and a polynomial-time computable function* opt *such that, for every* 3-CNF*-formula* Φ *and for every vertex v in $g(\Phi)$,* $\mathrm{opt}(g(\Phi) - v)$ *is an optimal coloring for $g(\Phi) - v$.*

Proof. Given a 3-CNF-formula Φ, let $g(\Phi) = h(f(\Phi))$, where f is the reduction from Theorem 1 and h is the reduction from Minimal-3-UnSat to VertexMinimal-3-UnColorability by Cai and Meyer [6]. Since h also reduces 3-Sat to 3-Colorability [6, Lemma 2.2], so does g. A careful inspection of the reduction g reveals that there is a polynomial-time computable function opt such that, for every vertex v in $g(\Phi)$, $\mathrm{opt}(g(\Phi) - v)$ is a 3-coloring of $g(\Phi) - v$. We can also verify that $g(\Phi) - v$ does not have a 2-coloring, hence $\mathrm{opt}(g(\Phi) - v)$ is an optimal coloring. We do not dive into the details as this lemma immediately follows from the proof of the analogous result for edge deletion (Lemma 5), as explained in Appendix D [3]. ◀

▶ **Theorem 4.** *Computing an optimal coloring for a graph from optimal colorings for its one-vertex-deleted subgraphs is* NP*-hard.*

Proof. Given a 3-CNF-formula Φ, compute $g(\Phi)$, where g is the reduction from Lemma 3, compute $\mathrm{opt}(g(\Phi) - v)$ for every vertex v in $g(\Phi)$, and from these optimal solutions compute one for $g(\Phi)$. This determines whether $g(\Phi)$ is 3-colorable and thus whether Φ is satisfiable. ◀

▶ **Lemma 5.** *There is a polynomial-time many-one reduction g from* 3-Sat *to* 3-Colorability *and a polynomial-time computable function* opt *such that, for every* 3-CNF*-formula* Φ *and for every edge e in $g(\Phi)$,* $\mathrm{opt}(g(\Phi) - e)$ *is an optimal coloring of $g(\Phi) - e$.*

Proof. Given a 3-CNF-formula Φ, let $g(\Phi) = h(f(\Phi)) - e$, where f is the reduction from Theorem 1, h is the reduction to VertexMinimal-3-UnColorability by Cai and Meyer [6], and e is the edge $\{v_{\mathrm{c}}, v_{\mathrm{s}}\}$, with v_{c} being the unique vertex adjacent to all variable-setting vertices and v_{s} being the only remaining neighbor vertex of v_{c}. We prove in detail that g has all the desired properties in Appendix D [3]. See Figure 1 in Appendix D [3] for an example of the construction. ◀

▶ **Theorem 6.** *Computing an optimal coloring for a graph from optimal colorings for its one-edge-deleted subgraphs is* NP*-hard.*

Proof. The same argument as for Theorem 4 can be applied here. ◀

▶ **Theorem 7.** Minimal-3-UnColorability *is* DP*-complete.*

Proof. Membership in DP is immediate, since given a graph $G = (V, E)$, determining whether $G - e$ is 3-colorable for every $e \in E$ is in NP and so is determining whether G is 3-colorable. As for DP-hardness, the argument from the proof of Lemma 5 shows that mapping Φ to $h(\Phi) - \{v_{\mathrm{c}}, v_{\mathrm{s}}\}$, where h is the reduction from Minimal-3-UnSat to VertexMinimal-3-UnColorability by Cai and Meyer [6], gives a reduction from Minimal-3-UnSat to Minimal-3-UnColorability (and to VertexMinimal-3-UnColorability as well). Recall that Minimal-3-UnSat is DP-hard [14]. ◀

Cai and Meyer [6] show DP-completeness for VertexMinimal-k-UnColorability, for all $k \geq 3$. We now prove that the analogous result for deletion of edges holds as well.

▶ **Theorem 8.** Minimal-k-UnColorability *is* DP*-complete, for every $k \geq 3$.*

Proof. Membership in DP is again immediate. To show hardness for $k \geq 4$, we reduce MINIMAL-3-UNCOLORABILITY to MINIMAL-k-UNCOLORABILITY. We use the construction for deleting vertices [6, Theorem 3.1] and map graph G to $G + K_{k-3}$.[4] Note that $\chi(K_{k-3}) = k-3$ and $\chi(H+H') = \chi(H)+\chi(H')$ for any two graphs H and H'. First suppose $G+K_{k-3}$ is in MINIMAL-k-UNCOLORABILITY. Then $G + K_{k-3}$ is not k-colorable, and so G is not 3-colorable. Let e be an edge in G. Then $(G-e)+K_{k-3} = (G+K_{k-3})-e$ is k-colorable, and thus $G-e$ is 3-colorable. It follows that G is in MINIMAL-3-UNCOLORABILITY.

Now suppose G is in MINIMAL-3-UNCOLORABILITY. Then $G + K_{k-3}$ is not k-colorable. Let e be an edge in $G + K_{k-3}$. If e is an edge in G, then $G - e$ is 3-colorable and so $(G + K_{k-3}) - e = (G - e) + K_{k-3}$ is k-colorable. If e is an edge in K_{k-3}, then $K_{k-3} - e$ is $(k-4)$-colorable and G is 4-colorable (let $\hat{e}$ be any edge in G, take a 3-coloring of $G - \hat{e}$, and change the color of one of the vertices incident to $\hat{e}$ to the remaining color), so $(G + K_{k-3}) - e = G + (K_{k-3} - e)$ is k-colorable. Finally, if $e = \{v, w\}$ for a vertex v in G and a vertex w in K_{k-3}, let $\hat{e}$ be an edge in G incident to v, take a 3-coloring of $G - \hat{e}$, take a disjoint $(k-3)$-coloring of K_{k-3}, and change the color of v to the color of w. As a result, for all edges e in $G + K_{k-3}$, $(G + K_{k-3}) - e$ is k-colorable. It follows that $G + K_{k-3}$ is in MINIMAL-k-UNCOLORABILITY. ◀

The construction above does not prove the analogues of Lemmas 3 and 5: Note that G is 3-colorable if and only if $(G + K_{k-3}) - v$ and $(G + K_{k-3}) - e$ are both $(k-1)$-colorable for every vertex v in K_{k-3} and for every edge e in K_{k-3}, and so we can certainly determine whether a graph is 3-colorable from the optimal solutions to the one-vertex-deleted subgraphs and one-edge-deleted subgraphs of $G + K_{k-3}$ in polynomial time. Turning to criticality and vertex-criticality, we can bound their complexity as follows.

▶ **Theorem 9.** *The two problems of determining whether a graph is critical and whether it is vertex-critical are both in Θ_2^{p} and* DP*-hard.*

Proof. For the Θ_2^{p}-membership of the two problems, we observe that the relevant chromatic numbers of a graph $G = (V, E)$ and its neighbors can be computed by querying the NP oracle COLORABILITY $= \{(G, k) \mid \chi(G) \leq k\}$ for every (G, k), $(G - e, k)$, and $(G - v, k)$ for every $e \in E$, $v \in V$, and $k \leq \|V(G)\|$ in parallel.

For the DP-hardness of the two problems, we prove that $h(\Phi) - \{v_{\mathrm{c}}, v_{\mathrm{s}}\}$ is a reduction from MINIMAL-3-UNSAT to both of them. We have already seen that it reduces MINIMAL-3-UNSAT to MINIMAL-3-UNCOLORABILITY. Hence, for every $\Phi \in$ MINIMAL-3-UNSAT, the graph $h(\Phi) - \{v_{\mathrm{c}}, v_{\mathrm{s}}\}$ is in MINIMAL-3-UNCOLORABILITY $\subseteq$ VERTEXMINIMAL-3-UNCOLORABILITY and thus both critical and vertex-critical. For the converse it suffices to note that, for every $\Phi \in$ CNF with clauses of size at most 3, $h(\Phi) - \{v_{\mathrm{c}}, v_{\mathrm{s}}\}$ is 4-colorable and thus in MINIMAL-3-UNCOLORABILITY (in VERTEXMINIMAL-3-UNCOLORABILITY, respectively) if and only if it is critical (vertex-critical, respectively). ◀

The exact complexity of these problems remains open, however. In particular, it is unknown whether they are Θ_2^{p}-hard. This contrasts with the case of Vertex Cover, for which we prove in Theorem 15 that recognizing β-vertex-criticality is indeed Θ_2^{p}-complete.

Before that, however, we return to our model and consider Colorability under the local modification of adding an edge. If we allow only one query, the problem stays NP-hard via a simple Turing reduction: Iteratively adding edges to the given instance eventually leads to

---

[4] For two graphs G_1 and G_2, the *graph join* $G_1 + G_2$ is the disjoint union $G_1 \cup G_2$ plus a *join edge* added from every vertex of G_1 to every vertex of G_2; see, e.g., Harary's textbook on graph theory [10, p. 21].

a clique as a trivial instance, see Theorem 17 in Appendix B [3]. Note that the restriction to one query is crucial for this reduction to work; without it, the branching may lead to an exponential blowup in the number of instances that need to be considered. The following theorem shows that this breakdown of the hardness proof is inevitable unless P = NP since the problem becomes in fact polynomial-time solvable if just one more oracle call is granted.

▶ **Theorem 10.** *There is a polynomial-time algorithm that computes an optimal coloring for a graph from optimal colorings of all its one-edge-added supergraphs; in fact, two optimal colorings, one for each of two specific one-edge-added supergraphs, suffice.*

For the proof of this theorem, we naturally extend the notion of universal vertices as follows.

▶ **Definition 11.** *An edge $\{u, v\} \in E$ of a graph $G = (V, E)$ is called* universal *if, for every vertex $x \in V - \{u, v\}$, we have $\{x, u\} \in E$ or $\{x, v\} \in E$. A graph is called* universal-edged *if all its edges are universal.*

Additionally, we denote, for any given graph $G = (V, E)$ and any vertex $x \in V$, the *open neighborhood* of x in G by $N(x) := \{y \mid \{x, y\} \in E\}$ and the *closed neighborhood* of x in G by $N[x] := N(x) \cup \{x\}$. We are now ready to give the proof of Theorem 10.

Proof of Theorem 10. We show that COLORER (Algorithm 1), which uses the oracle of our model and SUBCOL (Algorithm 2) as subroutines, has the desired properties.

Algorithm 1 COLORER.

Input: An undirected graph $G = (V, E)$.
Output: An optimal coloring for G.
Description: Optimizes universal-edged graphs with two queries to ORACLE, which provides optimal solutions to one-edge-added supergraphs; other graphs are optimized via SUBCOL.

```
1:  for every edge {u, v} ∈ E do
2:      for every vertex x ∈ V − {u, v} do
3:          if {u, x} ∉ E ∧ {v, x} ∉ E then
4:              f_1 ← ORACLE(G ∪ {u, x})
5:              f_2 ← ORACLE(G ∪ {v, x})
6:              if f_1 uses fewer colors on G than f_2 then
7:                  return f_1
8:              else
9:                  return f_2
10: k ← 1
11: while SUBCOL(G, k) = NO do
12:     k ← k + 1
13: return SUBCOL(G, k)
```

We begin by proving that COLORER is correct. Assume first that the input graph $G = (V, E)$ is not universal-edged. Then COLORER can find an edge $\{u, v\} \in E$ with a non-neighboring vertex $x \in V$ and query the oracle on $G \cup \{u, x\}$ and $G \cup \{v, x\}$ for optimal colorings f_1 and f_2. We argue that at least one of them is also optimal for G. Let f be any optimal coloring of G. Since u and v are connected by an edge, we have $f(u) \neq f(v)$ and hence $f(x) \neq f(u)$ or $f(x) \neq f(v)$; see Figure 2in Appendix E [3]. Thus, f is also an optimal coloring of $G \cup \{x, u\}$ or $G \cup \{x, v\}$, and so we have $\chi(G) = \chi(G \cup \{x, u\})$ or $\chi(G) = \chi(G \cup \{x, v\})$. Therefore, f_1 or f_2 is an optimal coloring for G as well and returned on line 7 or 9, respectively.

Algorithm 2 SUBCOL.

Input: An undirected, universal-edged graph $G = (V, E)$ and a positive integer k.
Output: A k-coloring f for G if there is one; NO if there is none.
Description: Works by recursion on k, with $k = 1$ and $k = 2$ serving as the base cases.

1: **if** G has no edge **then**
2:     **return** the constant 1-coloring with $f(x) = 1$ for all $x \in V$.
3: **else if** $k = 1$ **then**
4:     **return** NO.
5: **if** G has bipartition $\{A, B\}$ **then**
6:     **return** the 2-coloring $f(x) = \begin{cases} 1 & \text{for } x \in A \text{ and} \\ 2 & \text{for } x \in B. \end{cases}$
7: **else if** k=2 **then**
8:     **return** NO.
9: Choose an arbitrary edge $\{\ell, r\} \in E$.
10: $L \leftarrow N(\ell) - N[r]$; $R \leftarrow N(r) - N[\ell]$; $M \leftarrow N(\ell) \cap N(r)$
11: $g \leftarrow \text{SUBCOL}(G[M], k-2)$
12: **if** $g =$ NO **then**
13:     **return** NO
14: **return** the k-coloring $f(x) = \begin{cases} g(x) & \text{for } x \in M, \\ k-1 & \text{for } x \in L \cup \{r\}, \text{ and} \\ k & \text{for } x \in R \cup \{\ell\}. \end{cases}$

The while loop can be entered only if the graph G is universal-edged. This allows us to compute an optimal solution to G with no queries at all by using SUBCOL (Algorithm 2). We will show that SUBCOL is a polynomial-time algorithm that computes, for any universal-edged graph G and any positive integer k, a k-coloring of G if there is one, and outputs NO otherwise. The while loop of COLORER thus searches the smallest integer k such that G has a k-coloring, that is, $k = \chi(G)$. Hence, an optimal coloring of G is returned on line 13. Due to $k = \chi(G) \leq \|V\|$, COLORER has polynomial time complexity.

It remains to prove the correctness and polynomial time complexity of SUBCOL. This can be done by bounding its recursion depth and verifying the correctness for each of the six return statements; this is hardest for the last two. The proof relies on the properties of the partition $M \cup L \cup R \cup \{\ell\} \cup \{r\}$ as illustrated in Figure 3 in Appendix E [3]; see Appendix E [3] for all details. ◀

In this section, we have proven that MINIMAL-k-UNCOLORABILITY is complete for DP for every $k \geq 3$ and demonstrated that Colorability remains NP-hard in our model for deletion of vertices or edges, whereas it becomes polynomial-time solvable when the local modification is the addition of an edge. In the next section, we turn our attention to Vertex Cover.

4 Vertex Cover

This section will show that the behavior of Vertex Cover in our model is distinctly different from the one that we demonstrated for Colorability in the previous section. In particular, Theorem 12 proves that computing an optimal vertex cover from optimal solutions of one-vertex-deleted subgraphs can be done in polynomial time, which is impossible for optimal colorings according to Theorem 4 unless P = NP.

First, we note that the NP-hardness proof for our most restricted case with only one query still works (i.e., Theorem 17 in Appendix B [3] is applicable): Deleting vertices, adding edges, or deleting edges repeatedly will always lead to the null graph, an edgeless graph, or a clique through polynomially many instances. As we have seen for Colorability in the previous section, hardness proofs of this type may fail due to exponential branching as soon as multiple queries are allowed. We can show that, unless P = NP, this is necessarily the case for edge addition and vertex deletion since two granted queries suffice to obtain a polynomial-time algorithm.

▶ **Theorem 12.** *There is a polynomial-time algorithm that computes an optimal vertex cover for a graph from two optimal vertex covers for some one-vertex-deleted subgraphs.*

Proof. Observe what can happen when a vertex v is removed from a graph G with an optimal vertex cover of size k. If v is part of any optimal vertex cover of G, then the size of an optimal vertex cover for $G-v$ is $k-1$. Given any graph G, pick any two adjacent vertices v_1 and v_2. Since there is an edge between them, one of them is always part of an optimal vertex cover, thus either $G-v_1$ or $G-v_2$ or both will have an optimal vertex cover of size $k-1$. Two queries to the oracle return optimal vertex covers for $G-v_1$ and $G-v_2$. The algorithm chooses the smaller of these two covers (or any, if they are the same size) and adds the corresponding v_i. The resulting vertex cover has size k and is thus optimal for G. ◀

Theorem 19 in Appendix F [3] proves that the analogous result for adding an edge holds as well.

At this point, we would like to prove either an analogue to Theorem 6, showing that computing an optimal vertex cover is NP-hard even if we get access to a solution for every one-edge-deleted subgraph, or an analogue to Theorem 19 in Appendix F [3], showing that the problem is in P if we have access to a solution for more than one one-edge-deleted subgraph. We were unable to prove either, however. The latter is easy to do for many restricted graph classes (e.g., graphs with bridges), yet we suspect that the problem is NP-hard in general. We will detail a few reasons for the apparent difficulty of proving this statement after the following theorem and corollary, which look at deleting a triangle as the local modification.

▶ **Theorem 13.** *There is a reduction g from* 3-SAT *to* VERTEXCOVER *such that, for every* 3-CNF*-formula Φ and for every triangle T in $g(\Phi)$, there is a polynomial-time computable optimal vertex cover of $g(\Phi)-T$.*

The proof of Theorem 13 relies on the standard reduction from 3-SAT to VERTEXCOVER; see [9], where clauses correspond to triangles; see Appendix G [3] for the details. Applying the same argument as in the proof of Theorem 4 yields the following corollary.

▶ **Corollary 14.** *Computing an optimal vertex cover for a graph from optimal vertex covers for the one-triangle-deleted subgraphs is* NP*-hard.*

What can we say about optimal vertex covers for one-edge-deleted graphs? Papadimitriou and Wolfe show [14, Theorem 4] that there is a reduction g from MINIMAL-3-UNSAT to MINIMAL-k-NOVERTEXCOVER (called CRITICAL-VERTEXCOVER in [14]; asking, given a graph G and an integer k, whether G does not have a vertex cover of size k but all one-edge-deleted subgraphs do). The reduction builds in a polynomial-time computable vertex cover of size k for every one-edge-deleted subgraph. And so g is a reduction from 3-SAT to VERTEXCOVER such that there exists a polynomial-time computable function opt such that for every 3-CNF-formula Φ and $g(\Phi) = (G, k)$, it holds, for every edge e in G, that opt$(G - e)$ is a vertex cover of size k. Unfortunately, it may happen that an optimal vertex cover of $G - e$ has size $k - 1$; namely, if e is an edge connecting two triangles, an edge between two variable-setting vertices, or any edge of the clause triangles. The function opt does thus not give us an optimal vertex cover, thwarting the proof attempt. This shows that we cannot always get results for our model from the constructions for criticality problems.

The following would be one approach to design a polynomial-time algorithm that computes an optimal vertex cover from optimal vertex covers for all one-edge-deleted subgraphs: It is clear that deleting an edge does not increase the size of an optimal vertex cover and decreases it by at most one. If, for any two neighbor graphs, the provided vertex covers differ in size, then we can take the smaller one, restore the deleted edge, and add any one of the two incident vertices to the vertex cover; this gives us the desired optimal vertex cover. If the optimal vertex cover size decreases for all deletions of a single edge, we can do the same with any of them. Thus, it is sufficient to design a polynomial-time algorithm that solves the problem on graphs whose one-edge-deleted subgraphs all have optimal vertex covers of the same size as an optimal vertex cover of the original graph. One might suspect that only very few and simple graphs can be of this kind. However, we obtain infinitely many such graphs by the removal of any edge from different cliques, as already mentioned in the introduction. In fact, there is a far larger class of graphs with this property and no apparent communality to be exploited for the efficient construction of an optimal vertex cover.

We now turn to our complexity results of β-(vertex-)criticality. The reduction from Minimal-3-UnSat to Minimal-k-NoVertexCover by Papadimitriou and Wolfe [14] establishes the DP-hardness of deciding whether a graph is β-critical. However, it seems unlikely that β-criticality is in DP. The obvious upper bound is Θ_2^{p}, since a polynomial number of queries to a VertexCover oracle, namely (G, k) and $(G - e, k)$ for all edges e in G and all $k \leq \|V(G)\|$, in parallel allows us to determine $\beta(G)$ and $\beta(G - e)$ for all edges e in polynomial time, and thus allows us to determine whether G is β-critical. While we have not succeeded in proving a matching lower bound, or even any lower bound beyond DP-hardness, we do get this lower bound for β-vertex-criticality, thereby obtaining the first Θ_2^{p}-completeness result for a criticality problem.

▶ **Theorem 15.** *Determining whether a graph is β-vertex-critical is Θ_2^{p}-complete.*

Proof. Membership follows with the same argument as above, this time querying the oracle VertexCover in parallel for all (G, k) and $(G - v, k)$ for all vertices v in G and all $k \leq \|V(G)\|$. To show that this problem is Θ_2^{p}-hard, we use a similar reduction as the one by Hemaspaandra et al. [11, Lemma 4.12] to prove that it is Θ_2^{p}-hard to determine whether a given vertex is a member of a minimum vertex cover. We reduce from the Θ_2^{p}-complete problem $\mathrm{VC}_= = \{(G, H) \mid \beta(G) = \beta(H)\}$ [18]. Let $n = \max(\|V(G)\|, \|V(H)\|)$, let G' consist of $n + 1 - \|V(G)\|$ isolated vertices, let H' consist of $n + 1 - \|V(H)\|$ isolated vertices, and let $F = (G \cup G') + (H \cup H')$. Note that $\beta(F) = (n + 1) + \min(\beta(G), \beta(H))$. If $\beta(G) = \beta(H)$, then $\beta(F) = (n + 1) + \beta(G) = (n + 1) + \beta(H)$ and for every vertex v in F, $\beta(F - v) = n + \beta(G)$. Thus, F is critical. If $\beta(G) \neq \beta(H)$, assume without loss of generality that $\beta(G) < \beta(H)$. Then $\beta(F) = n + 1 + \beta(G)$. Let v be a vertex in G'. Then $\beta(F - v) = \min(n + 1 + \beta(G), n + \beta(H)) = n + 1 + \beta(G)$, and therefore F is not critical. ◀

5 Conclusion and Future Research

We defined a natural model that provides new insights into the structural properties of NP-hard problems. Specifically, we revealed interesting differences in the behavior of Colorability and Vertex Cover under different types of local modifications. While Colorability remains

NP-hard when the local modification is the deletion of either a vertex or an edge, there is an algorithm that finds an optimal coloring by querying the oracle on at most two edge-added supergraphs. Vertex Cover, in contrast, becomes easy in our model for both deleting vertices and adding edges, as soon as two queries are granted. The question of what happens for the local modification of deleting an edge remains as an intriguing open problem that defies any simple approach, as briefly outlined above. Moreover, examples of problems where one can prove a jump from membership in P to NP-hardness at a given number of queries greater than 2 might be especially instructive.

With its close connections to many distinct research areas, most notably the study of self-reducibility and critical graphs, our model can serve as a tool for new discoveries. In particular, we were able to exploit the tight relations to criticality in the proof that recognizing β-vertex-critical graphs is Θ_2^{P}-hard, yielding the first completeness result for Θ_2^{P} in the field.

References

1 Claudia Archetti, Luca Bertazzi, and Maria Grazia Speranza. Reoptimizing the Traveling Salesman Problem. *Networks*, 42(3):154–159, 2003.

2 Giorgio Ausiello, Bruno Escoffier, Jérôme Monnot, and Vangelis Paschos. Reoptimization of Minimum and Maximum Traveling Salesman's Tours. In *Proceedings of the 10th Scandinavian Workshop on Algorithm Theory (SWAT 2006)*, volume 4059 of *Lecture Notes in Computer Science*, pages 196–207. Springer-Verlag, 2006.

3 Elisabet Burjons, Fabian Frei, Edith Hemaspaandra, Dennis Komm, and David Wehner. Finding Optimal Solutions With Neighborly Help. Technical Report arXiv:1906.10078 [cs.CC], arXiv.org, June 2019. URL: `https://arxiv.org/abs/1906.10078`.

4 Hans-Joachim Böckenhauer, Luca Forlizzi, Juraj Hromkovič, Joachim Kneis, Joachim Kupke, Guido Proietti, and Peter Widmayer. Reusing Optimal TSP Solutions for Locally Modified Input Instances. In *Proceedings of the 4th IFIP International Conference on Theoretical Computer Science (IFIP TCS 2006)*, pages 251–270. Springer-Verlag, 2006.

5 Hans-Joachim Böckenhauer, Juraj Hromkovič, and Dennis Komm. Reoptimization of Hard Optimization Problems. In Teofilo F. Gonzalez, editor, *AAM Handbook of Approximation Algorithms and Metaheuristics*, volume 1, chapter 25, pages 427–454. CRC Press 2018, 2nd edition, 2018.

6 Jin-Yi Cai and Gabriele E. Meyer. Graph Minimal Uncolorability is $\mathrm{D^P}$-Complete. *SIAM Journal on Computing*, 16(2):259–277, 1987.

7 Gabriel A. Dirac. Some Theorems on Abstract Graphs. *Proceedings of the London Mathematical Society*, s3-2(1):69–81, 1952.

8 Piotr Faliszewski and Mitsunori Ogihara. On the Autoreducibility of Functions. *Theory of Computing Systems*, 46(2):222–245, 2010.

9 Michael Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, 1979.

10 Frank Harary. *Graph Theory*. Addison-Wesley, 1991.

11 Edith Hemaspaandra, Holger Spakowski, and Jörg Vogel. The complexity of Kemeny elections. *Theoretical Computer Science*, 349(3):382–391, 2005.

12 Gwenaël Joret. *Entropy and Stability in Graphs*. PhD thesis, Université Libre de Bruxelles, Faculté des Sciences, 2008.

13 Albert R. Meyer and Mike Paterson. With What Frequency Are Apparently Intractable Problems Difficult? Technical Report MIT/LCS/TM-126, Laboratory for Computer Science, MIT, Cambridge, MA, 1979.

14 Christos H. Papadimitriou and David Wolfe. The Complexity of Facets Resolved. *Journal of Computer and System Sciences*, 37(1):2–13, 1988.

15 Claus-Peter Schnorr. Optimal Algorithms for Self-Reducible Problems. In *Proceedings of the 3rd International Colloquium on Automata, Languages, and Programming*, pages 322–337. Edinburgh University Press, 1976.

16 Markus W. Schäffter. Scheduling with forbidden sets. *Discrete Applied Mathematics*, 72(1–2):155–166, 1997.

17 Karl Wagner. Bounded Query Classes. *SIAM Journal on Computing*, 19(5):833–846, 1990.

18 Klaus W. Wagner. More Complicated Questions About Maxima and Minima, and some Closures of NP. *Theoretical Computer Science*, 51(1–2):53–80, 1987.

19 Walter Wessel. Criticity with respect to properties and operations in graph theory. In László Lovász András Hajnal and Vera T. Sós, editors, *Finite and Infinite Sets. (6th Hungarian Combinatorial Colloquium, Eger, 1981)*, volume 2 of *Colloquia Mathematica Societatis Janos Bolyai*, pages 829–837. North-Holland, 1984.