Available online at www.sciencedirect.com

SCIENCE DIRECT®

Discrete Applied Mathematics 153 (2005) 3–24

DISCRETE APPLIED MATHEMATICS

www.elsevier.com/locate/dam

On the complexity of unfrozen problems

Adam Beacham, Joseph Culberson

Department of Computing Science, University of Alberta, Edmonton, Al, Canada T6G 2E8

Received 28 November 2003; received in revised form 27 June 2004; accepted 6 May 2005
Available online 12 September 2005

Abstract

We consider questions such as what is the complexity of recognizing instances of (monotonic) NP-complete problems in which no variable is fixed (or *frozen*) by the set of solutions. Since this *unfrozenness* is also a monotonic property in NP, this leads to an inductive sequence of properties for each monotone NP-complete property. In some cases the sequence remains NP-complete, while in others it at some point enters P. Determining the boundaries is particularly challenging. We also consider the related questions of recognizing maximal properties. This study was motivated by results from statistical mechanics being applied to phase transitions of NP-complete problems, which show a correlation of hard instances with a sudden increase in frozen variables.

Keywords: Frozen complexity; NP-completeness; Graph coloring; Satisfiability; Hamiltonian cycle; Subgraph isomorphism

1. Introduction

Empirical studies such as [5,2,20,8] have correlated a high frequency of very hard instances of NP-complete problems with phase transitions, or thresholds [3], in random classes. However, NP-complete problems without such hard regions do exist [13,21].

The distinction between hard and easy thresholds appears to be related to the order of the phase transition [2,8,17,18] in the generation process [4,3]. The *backbone* of an instance is the set of variables that are fixed by the set of solutions. See Section 2 for detailed definitions. Roughly, a phase transition is "second-order" if the backbone is continuous in the limit, and "first-order" otherwise. Hamiltonian cycle and more recently 1-in-*k* SAT

E-mail address: joe@cs.ualberta.ca (J. Culberson).

0166-218X/$ - see front matter © 2005 Elsevier B.V. All rights reserved.
doi:10.1016/j.dam.2005.05.003

[1], are NP-complete and are known to have second order phase transitions. 2-SAT which is polynomial, also has a second order transition and thus there is no direct connection between the order of the phase transition and computational complexity [1]. This is not too surprising since the theory of P versus NP does not consider probability distributions.

But the correlation of hard instances with first order thresholds and not with second order remains. (The results of Achlioptas et al. [1] suggest that there may be an algorithm that finds satisfying assignments for 1-in-*k* SAT at the threshold with high probability.) The first indication is that a large, but not too large, backbone may somehow be related to the difficulty of solving an instance.

These considerations prompted us to ask questions such as what is the complexity of recognizing instances which are *unfrozen*, that is with no backbone, for monotonic properties in NP. By considering the complexity of solving backbone free instances we hope to gain insight into the role of frozen structures in the difficulty of the instance.

Since unfrozenness is also a monotonic property in NP, this leads to an inductive sequence of properties for each monotone NP-complete property. In some cases the sequence remains NP-complete, while in others it at some point enters P. The cases do not correlate in an obvious way with the existence or not of hard instances at the threshold. Determining the boundaries is particularly challenging.

Test beds of hidden solution instances are typically generated by selecting a solution which forbids certain parts being generated and then adding (or deleting as appropriate) parts until some desired constraint density is achieved [2,9]. These methods also sometimes produce the easy–hard–easy phenomenon associated with thresholds [2,7]. Very underconstrained instances have many solutions and one is easy to find. As the number of edges increases there are fewer solutions, and finding one, even the hidden one, becomes harder. Then as the edges continue to increase the problem typically becomes easier again because the unique hidden solution becomes obvious in some manner.

Our second set of questions relates to whether it is possible to have an easy–hard–hard hidden solution phenomenon, where the hidden solution does not become obvious as edges are added. We consider the extreme. An instance is maximal with respect to some property if the addition of any edge (or clause, etc.) means the instance no longer has the property. Maximal properties can be seen as the extension to the generation process in which frozen parts are skipped [2,4,8] taken to the limit. In this paper we note that usually maximal properties are in P, but exhibit two cases where the instances remain hard in the complexity theoretic sense.

Maximal properties and unfrozen properties are also related to certain results in extremal graph theory [3] but typically such results are more concerned with the properties (e.g. number of edges) of the maximum over all maximal graphs.

2. Notation

Let Ω designate the set of all instances of a combinatorial class. A *property* is a subset $\mathbf{X} \subseteq \Omega$ and not-$\mathbf{X} = \Omega \backslash \mathbf{X}$. For graphs we use the notation $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ with complement $\mathbf{G}^c$. We say a property is *monotone down with respect to edges*, or simply *monotone*, if $\mathbf{G} \in \mathbf{X} \Longrightarrow \forall \mathbf{E}' \subseteq \mathbf{E}, \mathbf{G}' = (\mathbf{V}, \mathbf{E}') \in \mathbf{X}$. We consider only those properties $\mathbf{X} \in$ NP that

are monotone. We use the notation $\mathbf{G}+e$ ($\mathbf{G}-e$) to mean the graph with edge set $\mathbf{E}\bigcup\{e\}$ ($\mathbf{E}\backslash\{e\}$). Keep in mind that $(\mathbf{G}+e)-e \neq (\mathbf{G}-e)+e$.

We use the notation $\mathbf{G}[\mathbf{V}']$ for the induced subgraph of $\mathbf{G}$ on $\mathbf{V}' \subseteq \mathbf{V}$, and $\mathbf{G}_1 \times \mathbf{G}_2$ to represent the *join*, that is the union of $\mathbf{G}_1$ and $\mathbf{G}_2$ plus all edges between them. For hypergraphs and satisfiability the only additional requirements are that the arity of the edges or clauses is bounded by a constant, so that the number of edges/clauses in a complete structure is polynomial in the number of vertices/variables. The notation for these cases is a straightforward extension.

For properties such as Hamiltonian Cycle (HC) monotonicity reverses direction. To satisfy our conditions we would need to define the property with respect to the complementary structure, e.g. does $\mathbf{G}^c$ have an HC. However, in most cases the direction is not important and so we typically present arguments using the more familiar model in which the property ceases to hold under deletion of edges. An exception occurs in Theorem 7.4 where the combination of properties requires them to be consistent in order to preserve monotonicity.

We consider two sets of problems. The standard problems: independent set (IS), clique, vertex cover (VC), Hamiltonian Path/Cycle (HP/HC), coloring (COL), k-SAT and not-all-equal-k-SAT (NAE-k-SAT). These are defined in [14] and will be defined here as required. We consider only decision versions of these, e.g. k-COL, k-IS, etc.

The other set consists of a special form of subgraph isomorphism which requires a little care to make it monotonic and symmetric.

Definition (*k-complementary subgraph (k-CSG)*). $\mathbf{G}$ has this property iff there is a subset $\mathbf{V}' \subseteq \mathbf{V}$ such that $|\mathbf{V}'| = k$ and $\mathbf{G}[\mathbf{V}'] \subseteq \mathbf{G}^c[\mathbf{V}\backslash\mathbf{V}']$. Note the subgraph isomorphism is not induced.

For $1 < k \leqslant \lfloor n/2 \rfloor$ this is monotonic, since as edges are added, the complement of any k-subset of $\mathbf{V}$ has decreasing edges, while any k vertex induced subgraph has increasing edges. We show in Theorem 7.2 it is NP-complete.

3. Unfrozen problems

We define a frozen set for any monotonic property $\mathbf{X}$ on graph $\mathbf{G}$ by

$$\mathbf{F}(\mathbf{X}, \mathbf{G}) = \{e : \mathbf{G}+e \in \text{not-}\mathbf{X}\}, \tag{1}$$

where the edge e can be replaced by the appropriate structures, such as hyper-edges or clauses, for other problems. Notice that if $\mathbf{G} \in$ not-$\mathbf{X}$, then every e is in $\mathbf{F}$.

We define a property *unfrozen with respect to* $\mathbf{X}$, denoted by $\mathbf{U}(\mathbf{X})$, by $\mathbf{G} \in \mathbf{U}(\mathbf{X}) \iff (\mathbf{F}(\mathbf{X}, \mathbf{G}) = \emptyset)$. In Lemma 4.1 we show that if $\mathbf{X} \in$ NP then $\mathbf{U}(\mathbf{X})$ is also monotonic and in NP. This leads to the inductive definition

$$\mathbf{U}^0(\mathbf{X}) = \mathbf{X}, \tag{2}$$

$$\mathbf{U}^i(\mathbf{X}) = \mathbf{U}(\mathbf{U}^{i-1}(\mathbf{X})), \quad i > 0. \tag{3}$$

Equivalently, $\mathbf{G} \in \mathbf{U}^i(\mathbf{X})$ iff $\forall e_1, \ldots, e_i$, $\mathbf{G}+e_1+\cdots+e_i \in \mathbf{X}$.

Given a monotone property $\mathbf{X}$, we define the *maximally constrained* property $\max_c(\mathbf{X})$ by $\mathbf{G} \in \mathbf{X}$ and $\mathbf{G}$ is maximal with respect to $\mathbf{X}$; that is, for all $\mathbf{E}' \supset \mathbf{E}(\mathbf{G})$, $\mathbf{G}' \notin \mathbf{X}$. Equivalently, we can define maximally constrained by $\mathbf{G} \in \max_c(\mathbf{X}) \iff \mathbf{F}(\mathbf{X}, \mathbf{G}) = \mathbf{E}^c$ and $\mathbf{G} \in \mathbf{X}$.

Similar definitions apply to bounded arity SAT and hypergraph properties.

For SAT and Constraint problems, a different notion of frozenness is sometimes used [17,18]. In this case the *backbone* is defined to be the set of variables forced to take on the same value under all satisfying assignments.[1] We define $\widehat{\mathbf{U}}^i(\mathbf{X})$ such that for any instance in this set, for any choice of i variables, and any setting of those variables, there is an extension of that setting that satisfies the instance.

There is frequently a close relation between $\widehat{\mathbf{U}}$ and $\mathbf{U}$. For 3-SAT each frozen clause corresponds to a set of three frozen variables, and vice versa. Only when there are exactly one or two frozen variables is there a distinction between the two notions of unfrozen. Also note that a frozen variable corresponds to a frozen unit clause. On the other hand, for any colorable k-COL instance, or satisfiable instance of NAE-k-SAT, the variables are never frozen. We discuss $\widehat{\mathbf{U}}$ only in the case of k-SAT.

In a graph process, we can arrive at the threshold by either starting with an empty graph and adding edges, or starting with a complete graph and deleting edges. Similar to the above, we can define the *critical set* for a property $\mathbf{X}$ and graph $\mathbf{G}$ as

$$\mathbf{C}(\mathbf{X}, \mathbf{G}) = \{e : (\mathbf{G} - e) \in \mathbf{X}\} \tag{4}$$

We define *uncritical* with respect to $\mathbf{X}$, denoted by $\mathbf{D}(\mathbf{X})$ by $\mathbf{G} \in \mathbf{D}(\mathbf{X}) \iff (\mathbf{C}(\mathbf{X}, \mathbf{G}) = \emptyset)$. Note that $\mathbf{G} \in \mathbf{D}(\mathbf{X})$ implies that $\mathbf{G} \in$ not-$\mathbf{X}$. Again, it is easy to see that $\mathbf{D}$ is monotonic up and in CO-NP, where a certificate for not-$\mathbf{D}(\mathbf{X})$ consists of an edge to be deleted and a certificate for $\mathbf{X}$. This leads to the definition

$$\mathbf{D}^0(\mathbf{X}) = \text{not-}\mathbf{X}, \tag{5}$$

$$\mathbf{D}^i(\mathbf{X}) = \mathbf{D}(\text{not-}\mathbf{D}^{i-1}(\mathbf{X})), \quad i > 0. \tag{6}$$

Equivalently, $\mathbf{G} \in \mathbf{D}^i(\mathbf{X})$ iff $\forall e_1, \ldots, e_i$, $\mathbf{G} - e_1 - \cdots - e_i \in$ not-$\mathbf{X}$. These problems are all monotonic up and in CO-NP.

We define a *minimally constrained* property for monotonic up properties not-$\mathbf{X}$ $\min_c(\mathbf{X})$ as $\mathbf{G} \in$ not-$\mathbf{X}$ and $\mathbf{G}$ is minimal with respect to not-$\mathbf{X}$; that is, for all $\mathbf{E}' \subset \mathbf{E}(\mathbf{G})$, $\mathbf{G}' \in \mathbf{X}$. Equivalently, we can define minimally constrained by $\mathbf{G} \in \min_c(\text{not-}\mathbf{X}) \iff \mathbf{C}(\mathbf{X}, \mathbf{G}) = \mathbf{E}$. For graph coloring for example this definition is just the usual definition of an edge critical graph.

Of course, since $\mathbf{U}^i$ and not-$\mathbf{D}^j$ are monotonic properties, combinations such as the following are also well defined:

$$\mathbf{G} \in \mathbf{U}(\text{not-}\mathbf{D}(\mathbf{X})) \iff \forall e_1, \exists e_2, (\mathbf{G} + e_1) - e_2 \in \mathbf{X},$$
$$\mathbf{G} \in \mathbf{D}(\mathbf{U}(\mathbf{X})) \iff \forall e_1, \exists e_2, (\mathbf{G} - e_1) + e_2 \in \text{not-}\mathbf{X}.$$

We can use further alternations and superscript $\mathbf{U}$ and $\mathbf{D}$ as desired, as long as the total set of edges is bounded in size by some constant.

[1] In fact, in those papers a backbone is defined under all assignments causing a minimum violation of clauses.

We call the properties generated by **U** and **D** for property **X** the *frozen hierarchy* of **X**. When **X** ∈ NP this hierarchy is in the union of NP and CO-NP.

We give these latter definitions for completeness; our results in this paper are on **U** and $\max_c$.

4. Generic complexity of unfrozen problems

Let us define some additional problems related to these frozenness conditions. The problem **BB**(**G**, *e*, **X**) is to answer whether or not $e \in \mathbf{F}(\mathbf{G}, \mathbf{X})$. The problem **BB**(**G**, **X**) requires an algorithm to return some *e* that is in **F**(**G**, **X**) or Ø if there is none.

The following indicates the complexity relationships that are determined solely by **X** being a monotonic property in NP.

Lemma 4.1. *If* **X** *is a monotonic property in* NP*, then*:

1. $\mathbf{U}^i(\mathbf{X}) \in$ NP *and is monotone, for fixed i.*
2. $\mathbf{BB}(\mathbf{G}, e, \mathbf{X}) \equiv_T^P \mathbf{X}$.
3. $\mathbf{U}(\mathbf{X}) \propto_T^P \mathbf{BB}(\mathbf{G}, \mathbf{X}) \propto_T^P \mathbf{BB}(\mathbf{G}, e, \mathbf{X})$.
4. $\max_c \propto_T^P \mathbf{X}$.

Proof. For 1, monotonicity is trivial. To see that it is in NP, note that we need at most a polynomial number of certificates of polynomial size. For 2 we note that using an oracle for **X** we can test an *e* by simply adding it to **G**. On the other hand, since every $e \in \mathbf{E}(\mathbf{G})$ is in **F**(**G**, **X**) when **G** ∈ not-**X**, we can test for insolubility of **G** using **BB**(**G**, *e*, **X**).[2] For 3 and 4 the proofs are straightforward. □

It is not so easy to determine the complexity of **U** or of $\max_c$. In fact, for IS and HC we will show that $\mathbf{U}^i$ is NP-complete for any fixed *i*, while for fixed *k*, $\mathbf{U}^i$(*k*-COL) and $\mathbf{U}^i$(*k*-SAT) are in P for *i* sufficiently large.

We find that for every property **X** in the standard set listed in Section 2, $\max_c(\mathbf{X}) \in$ P. (Contrast this with $\min_c$ where many critical problems are DP-complete, see [19].) The obvious conjecture, however, does not seem to hold, since for *k*-CSG and for certain composite properties, we show reductions from complete properties that indicate $\max_c$ is not in P.

Determining the boundaries between P and NP or conditions when $\max_c$ is not in P are challenging open problems.

5. The complexity of some unfrozen properties

For some problems it is easily seen that the unfrozen versions are NP-complete.

[2] It is worth noting that even if we redefine **F** to be empty when **G** ∈ not-**X**, the reduction can be carried out by starting with an empty graph and adding the edges in **G** in some order, testing each in turn to see if frozen. Thus, this relation holds even if we use e.g. the spine definition of [4].

The independent set problem is given a graph $\mathbf{G}$ and integer k, is there a subset $\mathbf{V}' \subseteq \mathbf{V}$, $|\mathbf{V}'| = k$ such that $\mathbf{E}[\mathbf{G}[\mathbf{V}']] = \emptyset$.

Theorem 5.1. $\mathbf{U}^i(\mathbf{IS}) \in$ NP-complete *for all* $i \geqslant 0$.

Proof. Given a graph $\mathbf{G}$, construct $\mathbf{G}'$ by making $i+1$ copies of $\mathbf{G}$ and inserting all possible edges between the copies. Then $\mathbf{G}' \in \mathbf{U}^i(k\text{-IS})$ iff $\mathbf{G}$ contains a k-IS.[3] □

An instance of the k-SAT problem is a pair $S = (V, C)$, where V is a set of boolean *variables* $V = \{x_1, x_2, \ldots, x_n\}$ and C is a set of *clauses* $C = \{c_1, c_2, \ldots, c_m\}$ with k literals per clause. A literal is either a variable or its negation. The query is whether or not there is a *satisfying assignment* $f : V \to \{T, F\}$ of the variables, that is one such that every clause has at least one true literal. An instance $Q = (V, C)$ of NAE-k-SAT also consists of a set of boolean variables and clauses. It differs in that a *satisfying assignment* is a truth assignment of the variables such that every clause contains at least one true and at least one false literal [14].

For k-SAT, we have the following results.

Theorem 5.2. $\mathbf{U}(k\text{-SAT})$ *and* $\widehat{\mathbf{U}}(k\text{-SAT})$ *are* NP-complete, *for fixed* $k \geqslant 3$.

Proof. We reduce from NAE-k-SAT to $\widehat{\mathbf{U}}(k\text{-SAT})$. Construct a k-SAT instance S over the same variables as follows. For every clause $(l_1, l_2, \ldots, l_k)$ in Q, we insert the clauses $(l_1, l_2, \ldots, l_k)$ and $(\overline{l_1}, \overline{l_2}, \ldots, \overline{l_k})$ into S. A variable assignment satisfies Q if and only if it also satisfies S.

On the other hand, if S is satisfiable, then any satisfying assignment can be mapped to another one by inverting the value of every variable. Thus, S is satisfiable if and only if $S \in \widehat{\mathbf{U}}(k\text{-SAT})$. As $\widehat{\mathbf{U}}(k\text{-SAT}) \subseteq \mathbf{U}(k\text{-SAT})$, we also have $S \in \mathbf{U}(k\text{-SAT})$. □

Theorem 5.3. $\mathbf{U}^i(k\text{-SAT})$ *is in* P for $i \geqslant 2^k - 1$, *and* $\widehat{\mathbf{U}}^i(k\text{-SAT})$ *is in* P *for* $i \geqslant k$.

Proof. In either case the sets are trivial, consisting only of instances without clauses. Let F be an instance of k-SAT that contains at least one clause $c = (l_1, l_2, \ldots, l_k)$. If we now add the other $2^k - 1$ clauses on the same variables as those in c, the instance becomes unsatisfiable, and so is frozen for this i. For $\widehat{\mathbf{U}}^i(k\text{-SAT})$, simply freeze the literals of a clause to all false. □

For other problems the analysis is less simple.

A graph $\mathbf{G}$ has a Hamiltonian path (**HP**) if there is a permutation π of $\mathbf{V}$ such that $(v_{\pi_i}, v_{\pi_{i+1}}) \in \mathbf{E}$, $1 \leqslant i < n$. If in addition $(v_{\pi_n}, v_{\pi_1}) \in \mathbf{E}$ then the graph has a Hamiltonian cycle (**HC**). We remind the reader that the direction of monotonicity is reversed for these problems; the graph with no edges has no HP, while the complete graph does. Accordingly, in this section we will also reverse the direction of the definitions for frozen edges; that is, an edge is frozen if its deletion would eliminate all **HP**'s (**HC**'s). It is easy to see that this

[3] Thanks to N. Burch for suggesting this construction.

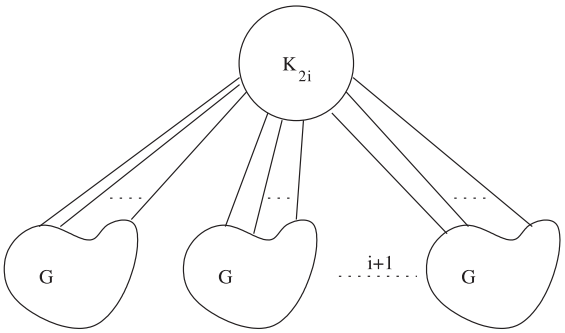

Fig. 1. Construction used in the reduction $\mathbf{HP} \propto_M^P \mathbf{U}^i(\mathbf{HP})$.

is equivalent to defining the problem in terms of the complement $\mathbf{G}^c$ with monotonicity defined in the usual way.

Theorem 5.4. $\mathbf{U}^i(\mathbf{HP}) \in$ NP-complete, $\forall i \geqslant 0$.

Proof. Let $\mathbf{G}$ be any graph, and construct a new graph $\mathbf{G}' = ((i+1)\mathbf{G}) \times K_{2i}$, where there are no edges between the $i+1$ copies of $\mathbf{G}$ (see Fig. 1). We show $\mathbf{G} \in \mathbf{HP} \iff \mathbf{G}' \in \mathbf{U}^i(\mathbf{HP})$ (i.e., $\iff \mathbf{G}'$ contains a Hamiltonian path even after first deleting i arbitrary edges).

Suppose $\mathbf{G} \notin \mathbf{HP}$. Then covering the vertices in each copy of $\mathbf{G}$ requires at least two disjoint paths. Thus, it takes $\geqslant 2(i+1)$ disjoint paths to cover the $i+1$ copies of $\mathbf{G}$. To create a Hamiltonian path in $\mathbf{G}'$, we would need to connect these disjoint paths together by visiting a vertex in the clique in between each subpath. But this would require at least $2i+1$ vertices in the clique. Hence, $\mathbf{G} \notin \mathbf{HP} \Longrightarrow \mathbf{G}' \notin \mathbf{HP} \Longrightarrow \mathbf{G}' \notin \mathbf{U}^i(\mathbf{HP})$.

Now, let $\mathbf{G} \in \mathbf{HP}$. We must show that there is a Hamiltonian path in $\mathbf{G}'$ even if an arbitrary set of i edges is first removed. There are three classes of edges that can be removed: edges inside the copies of $\mathbf{G}$, edges in the clique, and edges with one endpoint in the clique and one endpoint in a copy of $\mathbf{G}$. Call these removed edge sets E_1, E_2, and E_3, respectively, with $|E_1| = h$, $|E_2| = k$, $|E_3| = j$ and $i = h + j + k$.

Since $\mathbf{G} \in \mathbf{HP}$, in the absence of edge deletions we can cover the $i+1$ copies of $\mathbf{G}$ with $i+1$ disjoint paths. By removing E_1 from $\mathbf{G}'$, in the worst case we will need to use $i+h+1$ disjoint paths $\mathscr{P} = \{P_1, P_2, \ldots, P_{i+h+1}\}$. We will construct a path in $\mathbf{G}'$ that traverses each of these paths in order, visits a vertex in the $2i$-clique between successive paths, and then finishes by tracing a path in the remaining vertices in the clique. That is, the Hamiltonian path will look like $P_1 - x_1 - P_2 - x_2 - \cdots - P_{i+h} - x_{i+h} - P_{i+h+1} - x_{i+h+1} - x_{i+h+2} - \cdots - x_{2i}$ where the x's are vertices in the clique K_{2i}.

Let V_1 be the set of vertices in the clique K_{2i}. Let $V_2 \subseteq V_1$ be a set of k vertices such that each edge in E_2 has an endpoint in V_2 (i.e., V_2 covers E_2). Also, order the paths in $\mathscr{P}$ so that for $t > j$, neither endpoint of P_t is adjacent to an edge in E_3.

After removing E_3 from $\mathbf{G}'$, there are still at least $2i - j \geqslant j$ vertices in V_1 adjacent to every vertex in the paths. Let V_3 be any j of these vertices. For $1 \leqslant t \leqslant j$, connect paths P_t and P_{t+1} together with a vertex in V_3. For $j+1 \leqslant t \leqslant j + |V_2 \backslash V_3|$, connect P_t and P_{t+1}

together with a vertex from $V_2 \setminus V_3$; note that $j + |V_2 \setminus V_3| \leqslant j + |V_2| = j + k \leqslant i + h$, so we will be able to visit all the vertices in V_2 by the end of this step. At this point, every unvisited vertex in the clique, $V_4 = V_1 \setminus (V_2 \cup V_3)$, is adjacent to every vertex in all the paths, so for $j + |V_2 \setminus V_3| + 1 \leqslant t \leqslant i + h$ we can connect P_t and P_{t+1} together with vertices in V_4. Finally, if necessary we connect the free endpoint of P_{i+h+1} with any unvisited vertex in V_4. We can now complete the path by visiting the remaining vertices in V_1 because $K_{2i}[V_4]$ is a clique. □

Remark. Note that by adding an extra vertex to the clique in the proof, we can modify the proof to obtain a reduction $\mathbf{HP} \propto_M^P \mathbf{U}^i(\mathbf{HC})$.

Remark. Note that any Hamiltonian path in the graph $\mathbf{G}'$ constructed in the proof of Theorem 5.4 must contain a Hamiltonian path of at least one of the copies of $\mathbf{G}$ as a subpath. Hence, simply finding an **HP** in a graph $\mathbf{G} \in \mathbf{U}^i(\mathbf{HP})$ is as hard as finding a Hamiltonian path in an arbitrary graph.

6. Unfrozen coloring

Our results for coloring are sufficiently complex that we award them a section of their own. In the first subsection we show how to determine whether or not a graph is $\mathbf{U}^i(k\text{-COL})$ for $i \geqslant \binom{k}{2}$ in linear time, that is in $O(m+n)$. In the second subsection we show that $\mathbf{U}(k\text{-COL})$ is NP-complete. It follows that unless P = NP, for each $k \geqslant 3$ there exists an i_k, $1 < i_k \leqslant \binom{k}{2}$ such that $\mathbf{U}^{i_k}(k\text{-COL}) \in \mathrm{P}$ but $\mathbf{U}^{i_k-1}(k\text{-COL}) \notin \mathrm{P}$. In the third subsection we discuss a conjecture that would close the gap between P and NP-complete for $k \geqslant 3$.

6.1. Polynomial time algorithms for $i \geqslant \binom{k}{2}$

Trivially, if it is possible to form a $(k+1)$-clique by adding i edges then $\mathbf{G}$ is not in $\mathbf{U}^i(k\text{-COL})$. Thus, if $i \geqslant \binom{k+1}{2}$ then $\mathbf{U}^i(k\text{-COL})$ contains no graph on more than k vertices and:

Lemma 6.1. *For $k \geqslant 3$, $0 \leqslant i < \binom{k+1}{2}$*

$$\mathbf{U}^i(k\text{-COL}) \subseteq \left\{ G \,\middle|\, \forall H \subseteq \mathbf{V}, n_h = k+1 \Rightarrow m_h < \binom{k+1}{2} - i \right\}.$$

Throughout this section, given a subset $H \subseteq \mathbf{V}$, we will use $n_h = |H|$ and $m_h = |\mathbf{E}(\mathbf{G}[H])|$. It immediately follows that:

Lemma 6.2. *If $\mathbf{G} \in \mathbf{U}^{\binom{k}{2}}(k\text{-COL})$, then all connected components of $\mathbf{G}$ contain $\leqslant k$ vertices.*

The following lemma will be useful in both the proof of Theorem 6.1 and in the subsequent development of a fast algorithm to detect unfrozen instances.

Lemma 6.3. *If $H \subseteq \mathbf{V}$, $n_h \geqslant k+1$ and $m_h \geqslant n_h - 1$ then there exists $H' \subseteq H$ such that $n_{h'} = k+1$ and $m_{h'} \geqslant k$.*

Proof. If $n_h = k+1$ then $H' = H$ and we are done.

Otherwise, choose a vertex v of minimum degree δ. If $\delta \leqslant 1$ then delete v to obtain H' and observe that $m_{h'} \geqslant m_h - 1 \geqslant n_{h'} - 1$. If $\delta \geqslant 2$ then $m_h \geqslant \delta n_h/2 \geqslant n_h$ so we can delete an edge incident on v and still have $m_h \geqslant n_h - 1$.

The proof is completed by induction on repeated deletions. □

Theorem 6.1. *For $k \geqslant 3$ we have*

$$\mathbf{U}^{\binom{k}{2}}(k\text{-COL}) = \{G \mid \forall H \subseteq \mathbf{V}, n_h = k+1 \Rightarrow m_h \leqslant k-1\}.$$

Proof. Lemma 6.1provides the forward inclusion.

Suppose the theorem is false and let $H \subseteq \mathbf{V}$ be minimum such that adding $\binom{k}{2}$ edges to the induced subgraph $\mathbf{G}[H]$ will produce a non-k-colorable graph $\mathbf{G}'$. The minimum degree of $\mathbf{G}'$ must be $\delta \geqslant k$, since otherwise we could color the remaining vertices (since H is minimum) and then color v with a color not found on any neighbor of v.

Thus, the number of edges in $\mathbf{G}'$ is at least $\delta n_h/2$ and so $m_h \geqslant \delta n_h/2 - \binom{k}{2}$. From Lemma 6.3 we see that under the conditions of the theorem $m_h \leqslant n_h - 2$ given that $n_h \geqslant k+1$.

Define α by $n_h = k+1+\alpha$. Then

$$\begin{aligned} n_h - 2 &\geqslant m_h \geqslant \frac{kn_h}{2} - \binom{k}{2}, \\ k + \alpha - 1 &\geqslant \frac{k(k+1+\alpha)}{2} - \binom{k}{2}, \\ \alpha - 1 &\geqslant \frac{\alpha k}{2}. \end{aligned}$$

From this we obtain that $\alpha < 0$ for $k \geqslant 3$, which contradicts the supposition since $n_h \geqslant k+1$ is required to make $\mathbf{G}'$ uncolorable. □

Theorem 6.1 tells us that $\mathbf{U}^{\binom{k}{2}}(k\text{-COL})$ may be solved in $O(n^{k+1})$ time by checking for the existence of a set of $k+1$ vertices that induce k or more edges in $\mathbf{G}$. However, we can do much better.

Our goal is to determine the maximum $U \subseteq \mathbf{V}$ such that $m_u \geqslant n_u - 1$. We begin with the following easily proved observation:

Observation: If $U \subseteq \mathbf{V}$ is such that $m_u \geqslant n_u - 1$ then:

1. If $H \not\subseteq U$ is a component in $\mathbf{G}$ and $U \cap H \neq \emptyset$ then $U' = U \cup H$ has $m_{u'} \geqslant n_{u'} - 1$. Thus, a maximal U satisfying $m_u \geqslant n_u - 1$ will not contain partial components.
2. If H is a component and $m_h \geqslant n_h$ then $U' = U \cup H$ has $m_{u'} \geqslant n_{u'} - 1$. Thus, a maximal U will contain all non-tree components.

3. If U contains a tree component T and there is a larger tree component T' such that $T' \cap U = \emptyset$ then $U' = U \cup T' \setminus T$ has $m_{u'} \geqslant n_{u'} - 1$.
4. If $m_u \geqslant n_u$ and there is a tree component T not in U then $U' = U \cup T$ has $m_{u'} \geqslant n_{u'} - 1$.

Based on these it is easily verified that the following computes the (unique) largest set U such that $m_u \geqslant n_u - 1$. First, compute the union of all components H in which $m_h \geqslant n_h$. Call this subset U. It is straightforward to see that $m_u \geqslant n_u$ even if $U = \emptyset$. The remaining components are trees. We merge the trees in order of decreasing size into U one at a time until $m_u = n_u - 1$ or all components are in U.

Then by Lemma 6.3 and Theorem 6.1 if $n_u \geqslant k + 1$ the graph is frozen, otherwise it is not. Note that we do not actually require the union of the components, only the numbers. Thus, we only need to compute and sort the components and then keep track of n_u and m_u. Using a standard depth first search algorithm we can identify and analyze the components in $O(n + m)$ time. Since the number of vertices in a component is less than or equal n, we can enumerate the components of each order i in $O(n)$ time and thus:

Theorem 6.2. $\mathbf{U}^{\binom{k}{2}}(k\text{-COL})$ *can be solved in* $O(m + n)$ *time.*

Interestingly, this time is independent of k.

6.2. U (k-COL) is NP-complete

The result will be proved using the following reduction sequence.

Theorem 6.3. NAE-3-SAT $\propto_M^P$ **U**(3-COL) $\propto_M^P$ **U**(k-COL), $k \geqslant 3$.

For the first reduction, given an instance $Q = (V, C)$ of NAE-3-SAT, we will construct a graph **G** such that Q has a satisfying assignment of its variables V if and only if **G** is 3-colorable. We will then complete the proof by showing that **G** is 3-colorable if and only if $\mathbf{G} + e$ is 3-colorable for any $e \in E(\mathbf{G}^c)$; i.e., **G** is 3-colorable if and only if $\mathbf{G} \in \mathbf{U}$(3-COL).

The second reduction is trivial, and is given in Lemma 6.4.

6.2.1. Construction of the U(k-COL) reduction graph G

As is typical, we use graph gadgets to model the clause and variable constraints of the NAE-3-SAT. The colors used to 3-color the graph will be {1, 2, 3}. Without loss of generality, a single special vertex **3** is assumed to be given color 3. If the other vertices in a gadget are given labels for reference purposes, those labels will be alphabetic.

Literal gadgets: A literal L in Q is represented in **G** as two independent vertices joined to the distinguished vertex **3**; see Fig. 2. Rather than calling the two possible states of L *true* (T) and *false* (F), we will instead denote them by *same* (**S**) and *different* (**D**). In a 3-coloring of **G**, L has *value* **S** if the independent vertices a and b are assigned the same color that is (1, 1) or (2, 2), and *value* **D** if they are assigned different colors, (1, 2) or (2, 1).

Unless specified otherwise, when we say that a literal has a certain value, we do not force any particular one of the two possible colorings of the literal. Thus, in the following

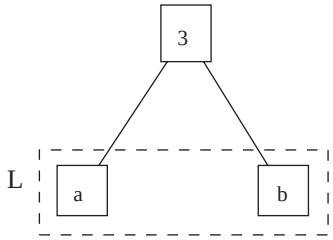

Fig. 2. A literal *L*.

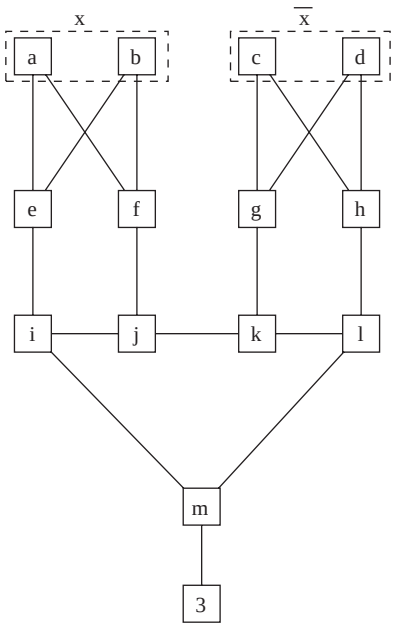

Fig. 3. Both x and $\overline{x}$ cannot have value **D**.

sections, we must show that if the literals attached to a gadget have appropriate values, then *any* coloring of the literals that encodes those values can be extended to a coloring of the entire gadget.

In the graphs that follow, a literal gadget will be denoted by two independent vertices surrounded by a box. The vertex **3** will normally not be shown. Note that for every variable x in Q, both x and $\overline{x}$ will appear as literal gadgets in **G**.

A variable and its negation: The gadgets in Figs. 3 and 4 are used to ensure that a variable x and its negation $\overline{x}$ are given different values in any valid 3-coloring of **G**.

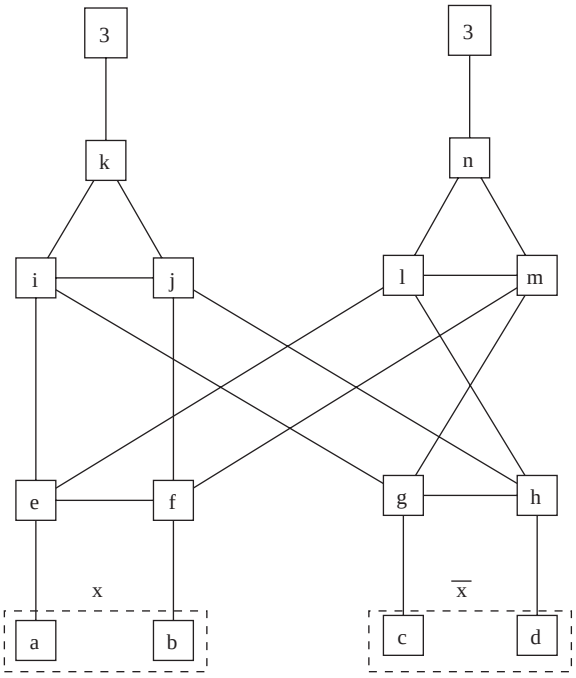

Fig. 4. Both x and $\overline{x}$ cannot have value **S**.

Fig. 3 ensures that **G** cannot be 3-colored if both x and $\overline{x}$ have value **D**. If they do, then we see that all the vertices e, f, g, h are forced to have color 3. But then the 5-cycle *ijklm* cannot be colored, since three colors are required but only two are available. On the other hand, if at least one is colored **S**, then at least one neighbor of the 5 cycle is not colored 3, and it is easy to see the cycle can be colored.

Fig. 4 ensures that x and $\overline{x}$ do not both take value **S**. If they do both have value **S**, then two of the vertices in the set $\{e, f, g, h\}$ are forced to be colored with color 3. But then one of the triangles *ijk* or *lmn* cannot be colored, as all three vertices will be adjacent to a color 3 vertex. Provided at least one of x and $\overline{x}$ has value **D**, then the gadget can be colored with three colors.

Literals in a clause do not all have value **S**: Given a clause (L_1, L_2, L_3) in Q, we prevent all three literals from taking the value **S** by using the gadget shown in Fig. 5. In this figure, each literal is adjacent to two vertices connected by an edge. Then, for every possible choice of picking one vertex from each vertex pair connected to a literal, we attach it to a "Not-All-3" gadget of the type shown in Fig. 6. That is, we attach the following eight subsets of vertices to the Not-All-3 gadgets: {*ace*, *acf*, *ade*, *adf*, *bce*, *bcf*, *bde*, *bdf*}.

In the Not-All-3 gadget, Fig. 6, *A*, *B*, and *C* are vertices outside the gadget. One Not-All-3 gadget will be attached to *ace* in Fig. 5, one will be attached to *acf*, and so on. We see that Fig. 6 can be 3-colored if and only if at least one of the vertices *A*, *B*, or *C* is not given color 3.

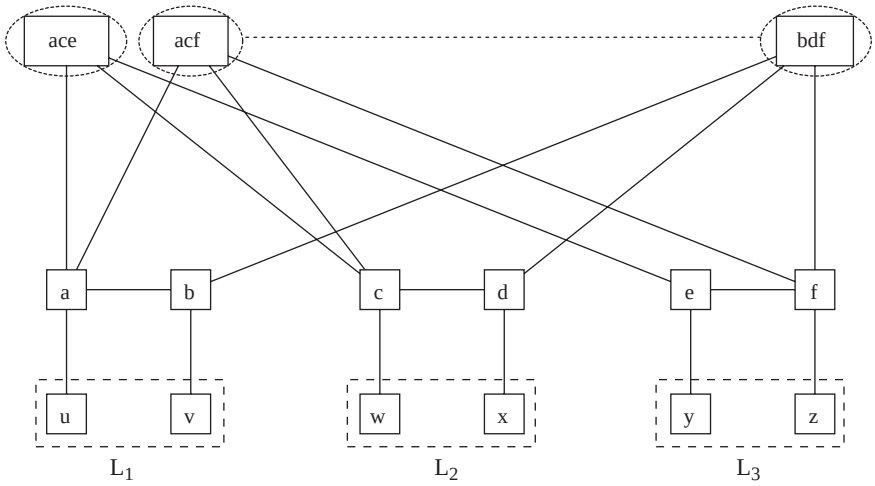

Fig. 5. All three literals L_1, L_2, L_3 cannot have value *same*. Note that there are 8 Not-All-3 gadgets: {*ace*, *acf*, *ade*, *adf*, *bce*, *bcf*, *bde*, *bdf*}. These gadgets are shown in Fig. 6.

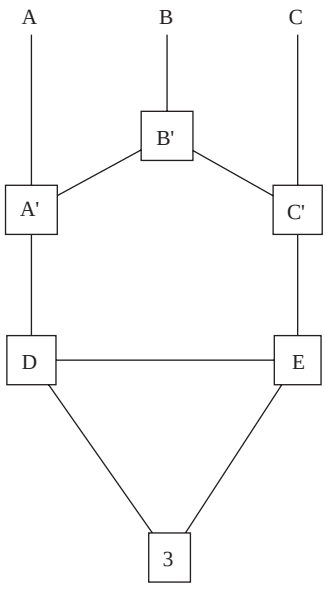

Fig. 6. Not-All-3 Gadget. Not all of A, B, C can be colored 3.

If a literal has value **S**, then one of the two vertices connected to the literal is forced to be colored 3. If all literals have value **S**, then one of the Not-All-3 gadgets will not be colorable. Provided at least one literal has value **D**, we can assign its adjacent vertices colors 1 and 2, which ensures every Not-All-3 gadget is colorable. Hence, the clause gadget in Fig. 5 is colorable if and only if at least one literal has value **D**.

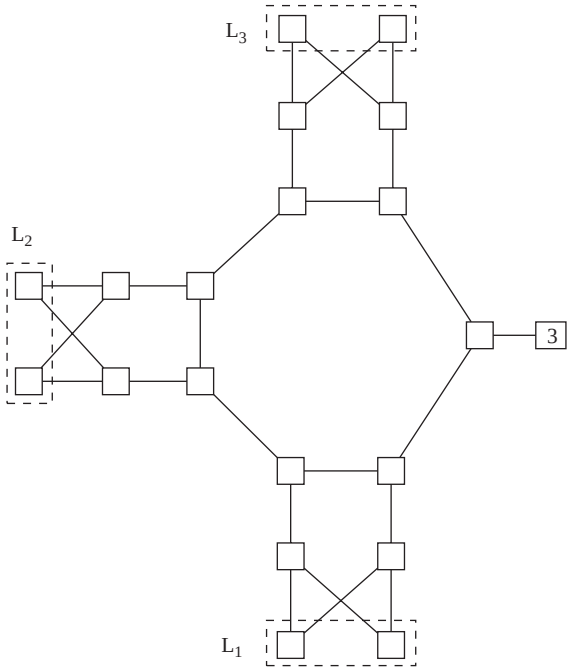

Fig. 7. All three literals L_1, L_2, L_3 cannot have value *different*.

Literals in a clause do not all have value **D**: To prevent all the literals in a clause from having value **D**, we use the gadget shown in Fig. 7. We see that if every literal is colored with two different colors, then every vertex in the 7-cycle is adjacent to a vertex of color 3. But then we only have two colors available (1 and 2) to color the cycle, which is impossible. Provided at least one literal has value **S**, we can set at least one of the vertices outside the cycle to either 1 or 2, which allows us to color the 7-cycle.

Hence, this gadget is 3-colorable if and only if not all the literals in the clause have value **D**.

Summary: Given an instance $Q = (V, C)$ of NAE-3-SAT, we create a graph **G** by constructing a literal gadget (Fig. 2) for each variable and its negation in V. We then attach complementary literals to the gadgets given in Figs. 4 and 3; this ensures they take on different values. Then, for every clause $c = (L_1, L_2, L_3) \in C$, we attach its literals to the gadgets in Figs. 5 and 7. These two gadgets force there to be literals with differing values in each clause. Altogether, we see **G** is 3-colorable if and only if Q has a satisfying assignment.

6.2.2. The colorability of **G** *is unaffected by edge addition*

We notice that provided the literal gadgets are colored in a way that corresponds to a satisfying assignment of the underlying NAE-3-SAT instance Q, then we are guaranteed to be able to 3-color all the gadgets in **G**. Furthermore, the gadgets are not affected by the particular choice of encoding. That is, if literal L is supposed to have value same (**S**), then

Table 1
Different colorings of the gadget in Fig. 3 given that the literals have certain fixed values

	$x = \mathbf{S}$		$\overline{x} = \mathbf{D}$		Valid coloring of gadget								
	a	b	c	d	e	f	g	h	i	j	k	l	m
α	1	1	1	2	2	3	3	3	3	1	2	1	2
β	1	1	1	2	3	2	3	3	2	3	1	2	1

we can color its vertices as either $(1, 1)$ or $(2, 2)$. If **G** could be colored when L was encoded as $(1, 1)$, then it can also be colored when it is encoded as $(2, 2)$.

Additionally, if Q is a satisfiable NAE-3-SAT instance, then we can invert the values of every literal (that is, exchange **S** and **D**) in **G**, and still have a colorable graph. Also, once the literals adjacent to a given gadget are colored, the gadget itself can be colored *independently* of every other gadget in **G**.

These facts give **G** a certain amount of flexibility to remain 3-colorable even when an arbitrary edge is added to the graph. Our basic strategy to fix problems introduced by a new edge can be summarized as follows:

1. If necessary, invert the values on the literals.
2. If necessary, change the encoding on a few literals: $(1, 1) \leftrightarrow (2, 2)$, $(1, 2) \leftrightarrow (2, 1)$.
3. Use the fact that gadgets can be independently colored to ensure that one endpoint of the new edge has a choice of two possible colors *independent* of the color of the other endpoint.

Item 3 tells us that for a new edge between gadgets, we will be able to color the vertices on either end of the edge different colors. For edges within gadgets, we will exhibit colorings of those gadgets that show every fixed pair of vertices within that gadget can be given different colors.

Edges connected to literals: If an edge is added between vertices a and b in a literal gadget (Fig. 2), then the literal is forced to have value **D**. Flip the value of all literals if needed to fix this problem.

If an edge is added between two literal gadgets, change the colors on one of the literals (if needed) between $(1, 1) \leftrightarrow (2, 2)$ or $(1, 2) \leftrightarrow (2, 1)$ as needed to make the edge irrelevant.

Finally, if an edge is added between either of the vertices in a literal and a vertex in any other gadget, fix the problem in the other gadget (see subsequent paragraphs).

Edges to the gadgets that enforce $x \neq \overline{x}$: Consider an edge with exactly one endpoint inside the gadget (Fig. 3) that forces at least one of x and $\overline{x}$ to have value **S**. In what follows, we can set the value of a particular literal by flipping the value of all literals. We do not consider the vertices in the literals to be part of the gadget.

By comparing colorings α and β in Table 1, we see that every vertex in the gadget except for g and h can be given one of two different colors independently of any other gadget. If we wish to give g and h the same option, we just flip the values of all literals so that $\overline{x} = \mathbf{S}$,

Table 2
Different colorings of the not both same gadget in Fig. 4 given that the literals have certain fixed values

	$x = \mathbf{S}$		$\overline{x} = \mathbf{D}$		Valid coloring of gadget									
	a	*b*	*c*	*d*	*e*	*f*	*g*	*h*	*i*	*j*	*k*	*l*	*m*	*n*
α	1	1	1	2	2	3	2	1	3	2	1	3	1	2
β	1	1	1	2	3	2	2	1	1	3	2	2	3	1

$x = \mathbf{D}$. Note that the colorings on the literals are *fixed*, so connections between the gadget and any literal vertex in **G** can also be repaired.

If both endpoints are within the gadget (Fig. 3), we notice that coloring α in Table 1 allows any edge within the gadget except for those in the set {*ek*, *em*, *fg*, *fh*, *fi*, *gh*, *gi*, *hi*, *jl*, *km*}. Of these 10, all but {*fi*, *gh*, *km*} have different colored endpoints under coloring β. These last three edges can be accommodated by swapping the values of x and $\overline{x}$ (and every other literal as well), and then using a symmetry argument. For instance, by symmetry *fi* is the same as *gl*, which means that edge *fi* can be accommodated by setting $x = \mathbf{D}$, $\overline{x} = \mathbf{S}$.

Next, consider adding an edge to **G** which has exactly one endpoint in the gadget (Fig. 4) that ensures at least one of the literals x and $\overline{x}$ has value **D**. As in the previous case, we will *fix* the colorings on the two literals by flipping the entire solution and recoloring x and $\overline{x}$ as required. We will set $x = (1, 1) = \mathbf{S}$, $\overline{x} = (1, 2) = \mathbf{D}$.

In Table 2, we see that every vertex except for *g* and *h* can be given one of two different colors independently of vertices in any other gadget once the literal colorings have been set. If we have an edge connected to *g* or *h*, we just flip the values on x and $\overline{x}$ and use symmetry to show that *g* and *h* can be given different values.

As for an edge with both endpoints in the gadget, we notice that coloring α in Table 2 allows any edge within the gadget except for those in the set $\{ej, eg, en, fl, gn, gj, hk, hm, if, il, jn, km\}$. Of these 12, all but *ej* and *fl* have different colored endpoints under coloring β. These last two edges can be accommodated by swapping the values of x and $\overline{x}$ (and every other literal as well).

Hence, if **G** is colorable, then $\mathbf{G} + e$ is also 3-colorable for any edge *e* with at least one endpoint in a gadget of the types given in Figs. 4 and 3.

Edges attached to clause gadgets: *Not-All-***D**-*Gadget*. Consider the clause gadget shown in Fig. 7 which ensures that not all of the literals in a clause (L_1, L_2, L_3) can have value *different* (**D**). We will first consider the problem of adding edges *within* the gadget; as usual, the vertices in the literals do not count as part of the gadget.

Note that if the gadget can be colored, then it is always possible to color all but one of the vertices (call the exception "v") outside the 7-cycle with color 3. In order for vertex v to have a color other than 3, it will need to be adjacent to a literal with value **S**; any of the six vertices outside the cycle that are not the **3** vertex can be v by flipping the values of all the literals (if necessary).

Case I (Edge with both endpoints inside the 7-cycle): Color one of the endpoints 3; this can always be done by ensuring that it is adjacent to the vertex v mentioned above. 2-color

Table 3
Valid colorings of the Not-All-3 gadget (Fig. 6)

Coloring of cycle given $A = 1, B = 2, C = 3$				
A'	B'	C'	D	E
3	1	2	2	1
2	3	1	1	2

the rest of the cycle with colors 1 and 2. Since the edge is incident on a vertex of color 3 and a vertex of color 1 or 2, it is not violated.

Case II (Edge with one point in the cycle, one outside): Ensure that the endpoint outside the cycle is *not* v. Then there exists a way to give the endpoint in the cycle color 1, and the endpoint outside the cycle (which is adjacent to one of the literals) color 3.

Case III (Edge with both endpoints outside the cycle): Let one of the endpoints be the special vertex v. Then that vertex will have color 1 or 2, while the other endpoint has color 3.

Case IV (Edge with one endpoint outside the gadget, one inside the 7-*cycle*). Provided the vertex in the cycle is not adjacent to the vertex v, we can give the cycle vertex either color 1 or 2 independently of the other endpoint. Thus, the graph can be colored so that the edge is not violated.

Case V (Edge with one endpoint outside the gadget, one in the gadget but outside the cycle): Notice that provided the vertex in the gadget is adjacent to a literal with value **S**, it can be given one of two colors: 3 and one other which depends on how the literal is colored. Thus, the gadget can be colored independently of the other endpoint in a way that ensures the new edge is not violated.

Edges attached to clause gadgets: *Not-All-***S**-*Gadget*. We will first investigate the properties of the Not-All-3 gadget shown in Fig. 6. Remember that our full clause gadget (Fig. 5) contains eight different Not-All-3 gadgets. Each of the Not-All-3 gadgets is attached at its connection points (A, B, C) to (in turn) one of the two vertices adjacent to each of the three literals L_1, L_2, L_3. Note that if the connection vertices are colored $A = 1, B = 2, C = 3$, then the 5-cycle within the gadget can be colored in two different ways (Table 3).

Notice that once the literal values have been fixed, any vertex in the Not-All-3 gadget can be given one of two colors independently of any other gadget. Thus, if we add an edge with one edge in the gadget and one outside it, we can still color **G** so that the edge is not violated. As for edges entirely within a Not-All-3 gadget, we see that the two colorings given above allow us to accommodate any new edge except for $C'D$. This edge can be dealt with by choosing the connection points to be colored as $A = 3, B = 2, C = 1$.

Note that it is always possible to set the connections on a Not-All-3 gadget to $A = 1$, $B = 2$, $C = 3$. C is adjacent to literal L_3; thus, set L_3 to value **S** by flipping the values of all literals if necessary. Then, we can recolor literals L_1 and L_2 while still keeping their values (**S** or **D**) constant so that A and B are colored 1 and 2, respectively. From previous results, we know that these recolorings will still allow all other gadgets in **G** to be 3-colored.

For edges with at least one endpoint in the clause gadget (Fig. 5) that is not also in one of the Not-All-3 gadgets (Fig. 6), we merely need to ensure that one endpoint is incident

on one of the vertices next to the literal with value **S**. In this case, the vertex can be given either color 3 or one of the set {1, 2}. This can be done independently of any other gadget, except for the eight Not-All-3 gadgets shown in the diagram. This technique also fixes the case where an edge is added with both endpoints inside the gadget. If the edge is connected to one of the eight Not-All-3 gadgets, we use the techniques shown above to resolve the problem.

Lemma 6.4. $\mathbf{U}(3\text{-COL}) \propto_M^P \mathbf{U}(k\text{-COL})$.

Proof. We simply add a $(k-3)$-clique K to the graph and then add all possible edges between K and the vertices of **G**. □

This completes the proof of Theorem 6.3. □

6.3. Towards tightening the boundary

On less sure footing is the conjecture that Theorem 6.1 tells us the exact point where $\mathbf{U}^i(k\text{-COL})$ becomes polynomial. Once i drops below $f(k,k) = \binom{k}{2}$, graphs in $\mathbf{U}^i(k\text{-COL})$ are not required to consist of small components. Furthermore, the existence of large girth $(k+1)$-chromatic graphs (see Jensen and Toft [15, Chapter 1.5]) tells us that checking for subsets of $k+1$ vertices that induce a clique after i edges are added is insufficient—there are graphs not in $\mathbf{U}^{\binom{k}{2}-1}(k\text{-COL})$ that pass this test. Without proof, we state our suspicions as Conjecture 6.1.

Conjecture 6.1. $\mathbf{U}^i(k\text{-COL}) \notin \mathrm{P}$ *for* $i < \binom{k}{2}$.

Proving that $\mathbf{U}^{\binom{k}{2}-1}(k\text{-COL})$ is NP-complete via a many-to-one reduction is likely to be quite difficult. Such a reduction must involve the construction of large girth components, since any component containing $k+1$ or more vertices must have girth greater than $k+1$: a smaller girth would violate the conditions of Lemma 6.1. Note that such a reduction could not be made up of only small disconnected components. Suppose every component of **G** has size $\leqslant f(k)\ln n$ for a function $f(k)$. Adding at most $\binom{k}{2}-1$ edges means the largest connected component would have at most $\binom{k}{2} f(k)\ \ln\ n$ vertices. Thus, membership in $\mathbf{U}^{\binom{k}{2}-1}(k\text{-COL})$ can be computed by brute force in

$$\mathrm{O}\left(\binom{\binom{n}{2}}{\binom{k}{2}-1} k^{\binom{k}{2} f(k)\ \ln\ n}\right) \leqslant n^{\mathrm{O}(k^2 f(k)\ \ln\ k)} \quad \text{(for fixed } k\text{)}$$

which is polynomial in n for fixed k.

Suppose $k = k(n) \in \omega(1)$, but not too large, since a k-chromatic graph must have girth at most $2\ \log\ n/\log(k-2)+2$ [11] and in our case $g \geqslant k+1$; say for example $k = \sqrt{\log\ n}$. Then $\mathbf{U}^{\binom{k}{2}}(k\text{-COL})$ is still in P by our algorithm since its running time is independent of k. But it is unclear whether $\mathbf{U}^{\binom{k}{2}-1}(k\text{-COL})$ is in NP; what polynomial certificate can we

provide, since listing all the size k subsets of edges is not polynomial? Thus, the transition may be greater than from P to NP.

7. The complexity of $\max_c$

We observe that if $\mathbf{X}$ is any of the properties in the standard list in Section 2 then the maximally constrained version of the property, $\max_c(\mathbf{X})$, is in the complexity class P. The proofs consist of verifying that the following must hold for each instance in the class (for n sufficiently large) and noticing that such instances are easily recognizable by polynomial time algorithms:

$\max_c(\mathbf{HC})$: $\mathbf{G}^c$ is an n-cycle.

$\max_c(k\text{-COL})$: $\mathbf{G}$ is k-partite complete.

$\max_c(k\text{-IS})$: the only non-edges are in the k independent set.

$\max_c(k\text{-SAT})$: every k set of variables has exactly $2^k - 1$ clauses, and the clauses force a unique solution.

$\max_c(\text{NAE-}k\text{-SAT})$: every k set of variables has $2^k - 2$ clauses, the missing clauses on each set have complementary literals, and together the clauses force a unique pair of complementary solutions.

A quick glance through [14] shows that for many other "natural" graph properties, such as graph bipartition, the corresponding $\max_c$ property is in P. However, the obvious conjecture that $\max_c(\mathbf{X})$ is in P for any monotonic $\mathbf{X} \in$ NP seems likely to be false because of the following.

Theorem 7.1. $\max_c(k\text{-}\mathbf{CSG})$ *is isomorphism complete.*

Lemma 7.1. *Let* $\mathbf{G} = \mathbf{G}_1 \times \mathbf{G}_2$ *and* $k = |V_1| = |V_2|$. If $\mathbf{G} \in k\text{-}\mathbf{CSG}$ *then the* V' *such that* $\mathbf{G}[V'] \subseteq \mathbf{G}^c[(V_1 \cup V_2)\backslash V]$ *has either* $V' \subseteq V_1$ or $V' \subseteq V_2$.

Proof. Suppose V' is composed of m $(0 < m < k)$ vertices from V_1 and $k - m$ vertices from V_2. Then $\mathbf{G}[V']$ is a connected graph while $\mathbf{G}^c[V\backslash V']$ is disconnected, so $\mathbf{G}[V'] \not\subseteq \mathbf{G}^c[V\backslash V']$. □

Lemma 7.2. *For* $\max_c(k\text{-}\mathbf{CSG})$, *if* $k < n/2$ *then the problem can be reduced in polynomial time to a case where* $k' = n'/2$.

Proof. If $\mathbf{G} \in \max_c(k\text{-}\mathbf{CSG})$, then there exist disjoint subsets of vertices $V', V'' \subseteq V$, $|V'| = |V''| = k$ such that $\mathbf{G}[V'] \cong \mathbf{G}^c[V'']$. Since $\mathbf{G}$ is maximal, every vertex in $V\backslash(V' \cup V'')$ must have degree $n - 1$. Thus, to check if a given graph $\mathbf{G}$ has the property $\max_c(k\text{-}\mathbf{CSG})$ where $k < n/2$, we can first remove $n - 2k$ vertices of degree $n - 1$ from $\mathbf{G}$ to reduce the problem to one where $k = n'/2$. If there are not that many vertices of large degree in $\mathbf{G}$, then we conclude $\mathbf{G} \notin \max_c(k\text{-}\mathbf{CSG})$. □

Definition. R is the relation on the vertices V of a given graph $\mathbf{G}$ defined by $uRv \iff uv \notin E$. Set $vRv\ \forall v \in V$. The equivalence relation $\sim$ is the transitive closure of R.

Lemma 7.3. *If* $\mathbf{G}_1 \cong \mathbf{G}_2$ *and* $\mathbf{G} = \mathbf{G}_1 \times \mathbf{G}_2^c$, *then* $\exists$ *an equivalence class* $C \subseteq V$ *under* $\sim$ *such that* $\mathbf{G}[C] = \mathbf{G}_1$ *or* $\mathbf{G}[C] = \mathbf{G}_2^c$.

Proof. Clearly, if $u \in \mathbf{G}_1$, $v \in \mathbf{G}_2^c$, then $u \not\sim v$. Thus, for any equivalence class $\overline{u}$ of $\sim$, either $\overline{u} \subseteq V_1$ or $\overline{u} \subseteq V_2$. Therefore, the equivalence classes of $\sim$ partition both V_1 and V_2. Now, suppose $\mathbf{G}_1 \neq \mathbf{G}[\overline{u}]$ for any $u \in V$. Then $\mathbf{G}_1 = \mathbf{G}[\overline{u_1}] \times \mathbf{G}[\overline{u_2}] \times \cdots \times \mathbf{G}[\overline{u_m}]$ for some vertices $u_1, u_2, \ldots, u_m \in V_1$. But then $\mathbf{G}_2^c \cong \mathbf{G}_1^c$ is a disconnected graph, which implies $V_2 = \overline{v}$ for some $v \in V_2$. □

Proof (*of Theorem 7.1*). (ISO$\propto_T^P \max_c(k\text{-}\mathbf{CSG})$) Given two graphs $\mathbf{G}_1, \mathbf{G}_2$ with $|V_1| = |V_2|$, and $|E_1| = |E_2|$, let $\mathbf{G} = \mathbf{G}_1 \times \mathbf{G}_2^c$. By Lemma 7.1 $\mathbf{G} \in \max_c(k\text{-}\mathbf{CSG}) \Longrightarrow \mathbf{G}_1 \cong \mathbf{G}_2$.

($\max_c(k\text{-}\mathbf{CSG}) \propto_T^P$ISO) By Lemma 7.2, we need only consider the case where $k = n/2$. In this case $\mathbf{G} \in \max_c(k\text{-}\mathbf{CSG}) \iff \exists V' \subseteq V$ such that $\mathbf{G} = \mathbf{G}[V'] \times \mathbf{G}[V \backslash V']$ and $\mathbf{G}[V'] \cong \mathbf{G}^c[V \backslash V']$. In polytime compute the equivalence classes of $\mathbf{G}$ with respect to the relation $\sim$. By Lemma 7.3, if $\mathbf{G} \in \max_c(k\text{-}\mathbf{CSG})$, then $\exists$ an equivalence class $\overline{v}$ of size $n/2$. Let $V' = \overline{v}$. Then $\mathbf{G} \in \max_c(k\text{-}\mathbf{CSG}) \iff$ every vertex in V' is adjacent to every vertex in $V \backslash V'$ and $\mathbf{G}[V'] \cong \mathbf{G}^c[V \backslash V']$, which requires the isomorphism test. □

We also note:

Theorem 7.2. k-**CSG** $\in$ NP-complete *for* $k = n/2$.

Proof. We already have k-**CSG** $\in$ NP, so we will complete the proof by showing that **HC**$\propto_M^P k$-**CSG**. Let $\mathbf{G}$ be any graph, and let $\mathbf{G}'$ be the complete join of a cycle on n vertices and the complement of $\mathbf{G}$; $\mathbf{G}' = C_n \times \mathbf{G}^c$. Then $\mathbf{G}$ contains a Hamiltonian cycle if and only if $\mathbf{G}' \in k$-**CSG**.

($\Longrightarrow$) If $\mathbf{G}$ contains a Hamiltonian cycle, then setting V' to the vertices in the n-cycle gives us $\mathbf{G}'[V'] \subseteq \mathbf{G}'^c[V \backslash V']$.

($\Longleftarrow$) If $\mathbf{G}' \in k$-**CSG**, then by Lemma 7.1 it follows that either $C_n \subseteq \mathbf{G}$, or $\mathbf{G}^c \subseteq C_n^c$. If $C_n \subseteq \mathbf{G}$ then we are done. On the other hand, if $\mathbf{G}^c \subseteq C_n^c$ then for every non-edge in C_n^c there is a corresponding non-edge in $\mathbf{G}^c \Longrightarrow C_n \subseteq \mathbf{G}$. □

After much searching, we found the following, rather unsatisfying, NP-hard $\max_c$ problem. Let Δ be the maximum degree of a graph.

Theorem 7.3. $\max_c$(3-COL *and* $\Delta \leqslant 4$) *is* NP-*hard*.

Proof. 3-COL with $\Delta = 4$ is known to be NP-complete [14]. To convert an instance to our $\max_c$, simply add $K_{4,4}$'s by breaking an edge and joining the free ends to vertices of degree less than 4, until all vertices are of degree 4. 3-colorability is preserved, with maximality being on degree. It is unlikely this is in NP since in general we have to prove non-3-colorability under the addition of some edge. □

We consider this example unsatisfactory from the viewpoint that we obtain a hard maximal problem by maximizing on a polynomial property. The following theorem makes this

more obvious by showing that several combinations of NP-complete properties become polynomial when taken to the extreme.

Theorem 7.4. *For the pairs* **HC** *and* k-COL, **HC** *and* j**-IS**, k-COL *and* j**-IS**, $\max_c$ *of the pair is in* P (*for fixed* k).

Proof. We only outline the proof ideas. Keep in mind that **HC** must be defined on the complement graph to be consistent with k-COL and j**-IS**. Let $\mathbf{G} \in \max_c$(3-COL and **HC**). Since **G** is k-colorable, its vertices can be split into k independent sets $C_1, \ldots, C_k$ corresponding to the color classes. As **G** is maximal, all edges possible between the color classes will be in **G**, with the exception of those non-edges (edges in $\mathbf{G}^c$) needed to have a Hamiltonian cycle in $\mathbf{G}^c$. As the C_i's are cliques in $\mathbf{G}^c$, we will have property **HC** provided we can get from one clique to another so as to complete a cycle. Since **G** is maximal, every non-edge between color classes *must* be used by every Hamiltonian cycle in $\mathbf{G}^c$, so we never require more than two edges between any pair of cliques. Since k is fixed, there are a fixed number of possible connection configurations of the k cliques, and so the conclusion. (The number of configurations is exponential in k.)

For k-COL and j**-IS** the graph must be either (1) k-bipartite complete, with the additional proviso that the largest of the independent sets is of size at least j (topping out on k-COL), or (2) have a j**-IS** and at most $k-1$ other vertices with all possible edges other than between pairs in the independent set.

For **HC** and j**-IS** the complement of the graph must be a j-clique and a set of paths each ending on points in the clique, such that the remaining vertices are covered. □

The $\max_c$(3-COL and $\Delta \leqslant 4$) problem does bear some relation to the generation of hidden solution instances and it might be worthwhile to experimentally consider hidden solution instances in which edges are added until the maximum degree bound is reached.

References

[1] D. Achlioptas, A. Chtcherba, G. Istrate, C. Moore, The phase transition in NAESAT and 1-in-k SAT, in: Proceedings of SODA 01, 2001, pp. 721–722.

[2] D. Achlioptas, C. Gomes, H. Kautz, B. Selman, Generating satisfiable problem instances, in: Proceedings of the Seventeenth National Conference on Artificial Intelligence (AAAI-2000), 2000, pp. 256–261.

[3] B. Bollobás, Modern Graph Theory, Graduate Texts in Mathematics, vol. 184, Springer, New York, 1998.

[4] B. Bollobás, C. Borgs, J.T. Chayes, J.H. Kim, D.B. Wilson, The scaling window of the 2-SAT transition, Paper in the xxx.lanl.gov e-Print archive, Available from http://xxx.lanl.gov/abs/math.CO/9909031, 1999.

[5] P. Cheeseman, B. Kanefsky, W.M. Taylor, Where the really hard problems are, in: Proceedings of the 12th IJCAI, 1991, pp. 331–337.

[7] J. Culberson, Hidden solutions, tell-tales, heuristics and anti-heuristics empirical methods in artificial intelligence, IJCAI-2001 Workshop, 2001, pp. 9–14.

[8] J. Culberson, I.P. Gent, Frozen Development in Graph Coloring, Theoret. Comput. Sci. 265 (1–2) (2001) 227–264.

[9] J.C. Culberson, F. Luo, Exploring the k-colorable landscape with iterated greedy, in: D.S. Johnson, A.M. Trick (Eds.), Cliques, Coloring, and Satisfiability: Second DIMACS Implementation Challenge, 1993, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, vol. 26, American Mathematical Society, 1996, pp. 245–284.

[11] P. Erdös, Graph theory and probability, Canad. J. Math. 11 (1959) 34–38.
[13] Y. Gao, J. Culberson, The phase transition in NK Landscapes is easy, in: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001), Morgan Kaufmann, Los Altas, CA, 2001, pp. 1227–1234.
[14] M.R. Garey, D.S. Johnson, Computers and Intractibility: A Guide to the Theory of NP-Completeness W.H. Freeman and Company, 1979.
[15] T. Jensen, B. Toft, Graph Coloring Problems, Wiley Interscience, New York, 1995.
[17] R. Monasson, R. Zecchina, Statistical mechanics of the random *k*-sat model, Phys. Rev. E 56 (1997).
[18] R. Monasson, R. Zecchina, S. Kirkpatrick, B. Selman, L. Troyansky, Determining computational complexity from characteristic 'phase transitions', Nature 400 (1998) 133–137 http://www.lpt.ens.fr/∼monasson, http://www.ictp.trieste.it/∼zecchina.
[19] C. Papadimitriou, Computational Complexity, Addison-Wesley, Reading, MA, 1994.
[20] B. Selman, D. Mitchell, H. Levesque, Generating hard satisfiability problems, Artificial Intelligence 81 (1996) 17–29.
[21] B. Vandegriend, J. Culberson, The G_{nm} phase transition is not hard for the Hamiltonian cycle problem, J. Artificial Intelligence Res. 9 (1998) 219–245 www.jair.org/contents/v9.html.