Graph Ramsey Theory and the Polynomial Hierarchy

Extended Abstract

Marcus Schaefer
Department of Computer Science
University of Chicago
1100 East 58th Street
Chicago, Illinois 60637, USA
schaefer@cs.uchicago.edu

Abstract

In the Ramsey theory of graphs $F \to (G, H)$ means that for every way of coloring the edges of F red and blue F will contain either a red G or a blue H. The problem ARROWING of deciding whether $F \to (G, H)$ lies in $\Pi_2^{\mathrm{P}} = \mathbf{coNP}^{\mathbf{NP}}$ and it was shown to be **coNP** hard by Burr [5]. We prove that ARROWING is actually Π_2^{P}-complete, simultaneously settling a conjecture of Burr and providing a natural example of a problem complete for a higher level of the polynomial hierarchy. We also show that STRONG ARROWING, the version for induced subgraphs, is Π_2^{P}-complete.

1 Introduction

Party mathematics is an important tool in the repertoire of the socially gifted mathematician, and one of the all-time favorite stories tells us that at a party of six people there are at least three people who know each other, or three people who do not know each other. As mathematicians started to get invited to larger parties, they began working on the general case: how large a group of people do you need to have at least k people who know each other, or l people who do not know each other. Frank Ramsey [20, 12] showed in 1930 that the size of the group needed is finite. Since then Ramsey theory has developed into an active and rich field.

Looking at the party example again, we see that it could also have been phrased thus: every edge-coloring of the complete graph on six vertices K_6 in red and blue contains either a red or a blue triangle. Graphs offer a generalized approach to classical Ramsey theory which has turned out to be quite fruitful in the last twenty years [12]. The basic notion of Graph Ramsey theory is *arrowing*: we say that a graph F *arrows* (G, H) and write $F \to (G, H)$ if for every edge-coloring of F with colors red and blue, a red G or a blue H occurs as a subgraph. We say F *strongly arrows* (G, H) and write $F \rightarrowtail (G, H)$ if for every edge-coloring of F with colors red and blue, a red G or a blue H occurs as an induced subgraph.

Take for example $G = H = P_4$, a path on four vertices. One quickly verifies that $K_5 \to (P_4, P_4)$ [8]. For the induced case the complete graph will obviously not do: in a monochromatic coloring K_5 will only have complete induced subgraphs. The graph obtained from the Möbius ladder M_8 (a C_8 in which all pairs of opposite vertices are connected) by removing one of the spokes, howver, strongly arrows P_4 [22]. See Figure 1 for a picture of the graph. Harary, Nešetřil and Rödl [13] claimed that M_8 itself would do, but this is not the case.

Given that arrowing is the central notion of

STOC '99 Atlanta GA USA
Copyright ACM 1999 1-58113-067-8/99/05...$5.00

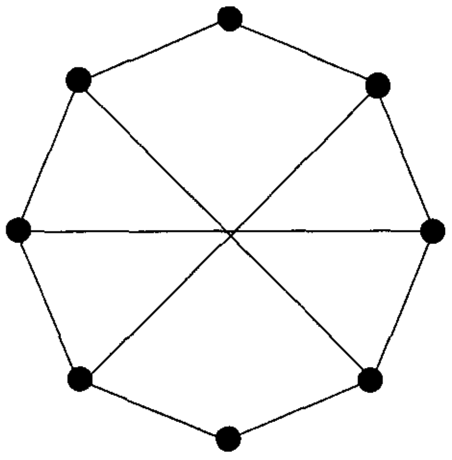

Figure 1: A variant of the Möbius ladder M_8

Graph Ramsey theory, it is natural to ask what its computational complexity is, that is, how hard is it to determine whether $F \to (G, H)$? In the notation of Garey and Johnson we can define two problems:

ARROWING
Instance: (Finite) graphs F, G and H.
Question: Does $F \to (G, H)$?

STRONG ARROWING
Instance: (Finite) graphs F, G and H.
Question: Does $F \rightarrowtail (G, H)$?

The contribution of this paper is to show that both problems are complete for the second level of the polynomial hierarchy (settling a conjecture by Burr [5]), simultaneously determining their computational complexity and giving examples of natural problems complete for higher levels of the polynomial hierarchy (in addition to Π_2^p for $\mathbf{coNP^{NP}}$ we will also use Σ_2^p for $\mathbf{NP^{NP}}$, and Σ_3^p for $\mathbf{NP}^{\Sigma_2^p}$).

As a consequence several other Ramsey type questions can be reduced (by a polynomial time algorithm) to a single question $F \to (G, H)$, including Ramsey problems with several colors, or problems of the type $F \to (\mathcal{G}, \mathcal{H})$ (where $\mathcal{G}$ and $\mathcal{H}$ are families of graphs). On the other hand there cannot be a polynomial time algorithm converting a question of the form $F \not\to (G, H)$ to a question $F' \to (G', H')$ unless the polynomial hierarchy collapses to the second level.

Let us have a look at the history of the ARROWING problem. Fixing the graphs G and H makes $F \to (G, H)$ a **coNP** problem in input F since checking for fixed subgraphs can be done in polynomial time for any coloring. Stefan Burr [5] showed that this problem is complete for **coNP** for any fixed three-connected graphs G and H (a graph is *three-connected* if it remains connected when any two vertices are removed from it). An earlier version of this result (again by Burr) where $G = H = K_3$ is already mentioned by Garey and Johnson [10, GT6]. Compared to this restricted version ARROWING immediately appears stronger, since $F \to (K_{1,1}, K_n)$ decides whether F has a clique of size n and is therefore **NP**-hard.

It is a widely accepted view that a complexity class is justified by its complete problems. Both the number of problems and their naturalness play a role. The classes **P** and **NP** excel on both accounts and have become the most popular classes of computational complexity even outside the field. Nothing similar is true for the higher levels of the polynomial hierarchy, and Garey and Johnson [10, Section 7.2] went so far as to say that the interest of the hierarchy was mainly theoretical and not practical: would we really care if the hierarchy collapsed to the second level as long as $\mathbf{P} \neq \mathbf{NP}$? There is some evidence, however, that even Σ_2^p and Π_2^p are natural classes of more than theoretical interest.

Probably the first natural problem to be shown Σ_2^p-complete was INTEGER EXPRESSION INEQUIVALENCE by Stockmeyer [10, AN18] in 1976 shortly after the polynomial hierarchy was defined. A couple of problems related to INTEGER EXPRESSION INEQUIVALENCE were later shown to be Π_2^p-complete and Σ_2^p-complete by Wagner [24] (he even mentions one Σ_3^p-complete problem). The basic structure of INTEGER EXPRESSION INEQUIVALENCE of testing whether two representations denote the same object is at the root of at least two other completeness results for the second level: Sagiv and Yannakakis [21] proved that deciding whether two monotonic relational expressions are equivalent is Π_2^p-complete (monotonic expressions contain only the operators select, project, join and union), and a result by Huynh [16] shows that deciding whether two context-free grammars with only one terminal letter generate the same language is Π_2^{p}-complete. In a similar spirit Lin [19] recently showed a problem related to pattern matching (and program optimization) to be Π_2^{p}-complete. Ko and Tzeng [17] exhibited three Σ_2^{p}-complete problems that belong to the realm of computational learning theory: PATTERN CONSISTENCY, GRAPH RECONSTRUCTION and GENERALIZED 3-CNF CONSISTENCY. The last problem asks whether for two sets of Boolean formulas there is a 3-CNF formula which is consistent with each formula from the first set, but with no formula from the second set.

Some famous problems remain candidates for Σ_2^{p}-completeness. In the sixties and seventies the question of size of machines and programs was investigated in detail, and in parallel to research done in computability, Meyer and Stockmeyer defined the MINIMAL problem of deciding whether for a given formula φ there is no formula of smaller size that is equivalent to it. MINIMAL lies in Π_2^{p} but only recently did Hemaspaandra and Wechsung [15] show that MINIMAL is **coNP**-hard. Stockmeyer also defined the problem MINIMUM EQUIVALENT EXPRESSION$_{\mathrm{DNF}}$ of deciding whether for a given DNF formula φ there is a DNF formula of size at most k which is equivalent to it. This problem was recently shown to be Σ_2^{p}-complete by Umans [23].

Finally there is an analogue of the GRAPH ISOMORPHISM problem, the BOOLEAN ISOMORPHISM problem in Π_2^{p} which asks whether two formulas are equivalent up to a permutation of the variables. BOOLEAN ISOMORPHISM is Π_1^{p}-hard, but not known to lie in Π_1^{p}. Agrawal and Thierauf [1] recently showed that the complement of BOOLEAN ISOMORPHISM lies in $\mathrm{AM}^{\mathrm{NP}}$ which means that we do not expect it to be complete for Σ_2^{p} since this would collapse the polynomial hierarchy to Σ_3^{p}.

2 Arrowing

Theorem 2.1 *ARROWING is Π_2^{p}-complete.*

The result will be immediate from Corollary 2.8 which shows that deciding $F \to (T, K_n)$ for any tree T is Π_2^{p}-complete.

Definition 2.2 *A coloring of F is called (G,H)-*good *if F does not contain a red G or a blue H in this coloring.*

In all of the following constructions we will use the implicit convention that if we take several graphs and identify some of their edges or vertices, then only these edges or vertices are identical and the remainders of the graphs still distinct.

Definition 2.3 *A graph F is called a (G,H)-*enforcer with signal vertex v *if the graph obtained from F by attaching a free edge in v has the property that this edge is colored blue in all (G,H)-good colorings, and there is a (G,H)-good coloring of the graph.*

We can use enforcers to force edges leaving the signal vertex to be blue (assuming that H is two-connected). We will sometimes talk of a blue enforcer to make this point explicit. Red enforcers are defined analogously.

Our main result in this section is the next theorem. It will follow from a series of lemmata we will prove later.

Theorem 2.4 *Let G be a fixed connected graph of size at least two for which we can compute a (G,K_n)-enforcer in time polynomial in n. Then deciding $F \to (G,K_n)$ is Π_2^{p}-complete.*

Before we prove the theorem we show how to apply it in case G is a tree. For this we need a well-known fact from the Ramsey theory of graphs. The *Ramsey number $r(G,H)$* of two graphs is defined as the least n such that $K_n \to (G,H)$.

Lemma 2.5 (Chvátal [7]) *If T is a tree on t vertices, then $r(T,K_n) = (t-1)(n-1)+1$.*

The lower bound of Chvátal's result is established by coloring a subgraph $(n-1)K_{t-1}$ of $K_{(t-1)(n-1)}$ red and its complement blue. Note that in this coloring the red degree of each vertex is $t-2$. This observation can be generalized.

Lemma 2.6 *The red degree of every vertex in $K_{(t-1)(n-1)}$ in any (T, K_n)-good coloring is at least $t-2$, where t is the order of T.*

Proof Let $m = (t-1)(n-1)$. Assume for a contradiction that there is a (T, K_n)-good coloring of K_m in which some vertex x has red degree at most $t-3$. Let $y_1, \ldots, y_{t-3}$ be its red neighbors. Look at the graph $K_m - \{x, y_1, \ldots, y_{t-3}\}$ under the induced coloring. This graph has $m-(t-2) = (t-1)(n-1)-(t-2) = (t-1)(n-2)+1$ vertices and hence (by Chvátal's result) contains a red T or a blue K_{n-2}. Neither, however, is possible, since a red T would also be contained in the original graph, and a blue K_{n-2} together with x would form a blue K_{n-1} in the original graph. □

It is a graph-theoretical folklore fact (a proof can be found in Chvátal's note [7]) that a graph of minimum degree $t-1$ contains every tree on t points. More precisely the tree can be embedded node by node, starting at an arbitrary node of the tree and the graph. Hence if we remove from T an edge leading to a leaf, then the resulting tree T' will occur as a red subgraph in any (T, K_n)-good coloring of $K_{(t-1)(n-1)}$ with any particular vertex of T' in any particular vertex of $K_{(t-1)(n-1)}$. In other words $K_{(t-1)(n-1)}$ is an enforcer, and each of its vertices can act as a signal vertex.

Lemma 2.7 *For any tree T a (T, K_n)-enforcer can be constructed in polynomial time in n (namely take a $K_{(|T|-1)(n-1)}$ and any of its vertices).*

Hence we obtain the following corollary of the above theorem, in turn implying Theorem 2.1.

Corollary 2.8 *Deciding $F \rightarrow (T, K_n)$ is Π_2^{p}-complete (where T is a tree of size at least two).*

Before we can prove Theorem 2.4 we need to construct a particular kind of graph which works like a switch between two edges: at least one of the edges will be blue in a good coloring.

Definition 2.9 *A graph F is called a (G, H)-*switch *if it contains an induced path of length two such that for all (G, H)-good colorings at most one of the edges of the path is red, and there are (G, H)-good colorings of F in which each of the two edges is blue while the other is red.*

The construction of switches requires yet another kind of graph which like the enforcer determines a particular edge to have a certain color.

Definition 2.10 *A graph is called a (G, H)-*determiner *with signal edge e if e is colored red in all (G, H)-good colorings, and there is a (G, H)-good coloring of the graph.*

For later reference note that we will sometimes call these determiners red determiners to distinguish them from blue determiners which are defined analogously.

Lemma 2.11 *Suppose that G is a connected graph of size at least two and that we can compute a (G, K_n)-enforcer in polynomial time in n. Then a (G, K_n)-switch can be constructed in polynomial time.*

Proof The proof will be in two steps. We first show how to obtain a red determiner from the blue enforcer and then how to use that to obtain a switch.

Build a graph F by taking a copy of K_n and identifying all of its vertices except for two (say u and v) with the signal vertex of a (G, K_n)-enforcer. We claim that F is a (G, K_n)-determiner. Fix a (G, K_n)-good coloring of F. The enforcers force all edges of K_n except for uv to be blue. This determines uv to be red since the coloring does not contain a blue K_n. Furthermore coloring all the enforcers with a good coloring and letting all the edges in K_n be blue except for uv which is red shows that there is a good coloring of F (this relies on the fact that G is connected).

With the determiner we can now construct the switch. The fact that there is a (G, K_n)-enforcer implies that G must have a vertex of degree one. Let e be the edge attached to that vertex, and let f be an edge adjacent to e (remember that G is connected and has size at least two).

Construct the switch by taking a copy of G and making all of its edges except for e and f signal edges of determiners. That forces one of e or f to be blue. It is straightforward to construct two good colorings of the switch in which either e or f is blue and the other red (again we need that G is connected).

Obviously all of these constructions can be performed in polynomial time. □

In view of the preceding result the next lemma is sufficient to establish Theorem 2.4

Lemma 2.12 *Let G be a connected graph of size at least two such that we can compute a (G, K_n)-enforcer and a (G, K_n)-switch in polynomial time in n. Then for any Π_2 sentence ψ we can construct a graph F and compute a number n in polynomial time such that ψ is true if and only if $F \to (G, K_n)$.*

Proof Fix G, and let ψ be the formula $(\forall x_1 \ldots x_k)(\exists y_1 \ldots y_k)[\varphi(x_1, \ldots, x_k, y_1, \ldots, y_k)]$ which is to be coded, the $2k$ variables ranging over $\{0, 1\}$. We will refer to the x_i as the *x-variables* and the y_i as the *y-variables*. Without loss of generality φ is in conjunctive normal form with at most three literals per clause and each literal occurs at most twice in the whole formula. We can also assume that each variable occurs at most once per clause. Finally using the fact that $(\forall x)[\varphi(x, \overline{x})] \equiv (\forall x)(\exists y_1, y_2)[(\overline{x} \vee y_1) \wedge (x \vee \overline{y_2}) \wedge \varphi(y_1, y_2)]$ we can assume that all the x-variables occur only twice, once positive and once negative with one y variable each (thanks to Dieter Van Melkebeek for pointing out this important simplification).

Let $\varphi(a, b)$ consist of m clauses $C_1, \ldots, C_m$.

We will now mimic the construction that shows that CLIQUE is **NP**-complete. We can already compute n: it is going to be m, the number of clauses.

Construct the graph F as follows. For each clause C that does not contain an x-variable take a triangle or an edge (if the clause contains only two literals), and label the vertices by the literals in the clause. Call this a y-gadget. For each pair $(\overline{x_i} \vee y_j)$ and $(x_i \vee y_l)$ take a (G, K_m)-switch with adjacent signal edges e_i and f_i say. Since these two edges are different there is a vertex u of e_i which does not belong to f_i and a vertex v of f_i which does not belong to e_i. Label u with y_j and v with y_l and the common vertex of e_i and f_i with x_i. Call this an x-gadget.

Include all edges between vertices which are not labeled by contradictory literals and belong to different gadgets. Furthermore make each vertex of a y-gadget the signal vertex of a (G, K_m)-enforcer.

This completes the construction of F. We claim that $F \to (G, K_m)$ if and only if ψ is true (see Figure 2 for an example).

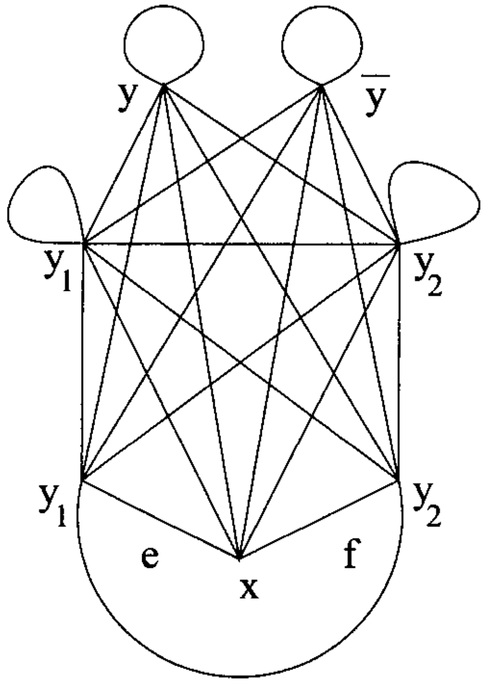

Figure 2: F for $(\forall x)(\exists y)[x \not\leftrightarrow y]$

Assume first that ψ is true. Suppose for a contradiction that there is a (G, K_m)-good coloring of F. Fix that coloring. In this coloring all the enforcers and switches work, namely all the edges within the y-gadgets and from the y-gadgets to the x-gadgets are blue, and in the x_i-gadget at least one of e_i and f_i is blue. Call x_i true if e_i is blue, and false otherwise. For this truth assignment to the x-variables there is a truth assignment to the y variables such that $\varphi(x_1, \ldots, x_k, y_1, \ldots, y_k)$ is true. Fix this assignment. For each y-gadget pick a vertex whose label is assigned true. In each x-gadget choose the two vertices belonging to e_i if x_i is true and f_i otherwise. By the construction of F these m vertices form a blue clique contradicting the assumption.

To show the other direction assume that ψ is false, i.e. there is an assignment of truth-values to $x_1, \ldots, x_k$ such that for all assignments of truth-values to $y_1, \ldots, y_k$ the formula $\varphi(x_1, \ldots, x_k, y_1, \ldots, y_k)$ is false. Fix such $x_1, \ldots, x_k$. The enforcers and switches were constructed in such a way that they have good colorings. Fix a good coloring for each enforcer. For the x_i-gadget we choose a coloring of the switch in which e_i is blue and f_i is red if x_i is true and vice versa otherwise.

Finally color all edges between gadgets blue. We claim that this coloring is a (G, K_m)-good coloring. It obviously does not contain a red G since none of the components contains a red G, and only the x-gadgets contain isolated red edges (we use that G is connected). Suppose, for a contradiction, that there is a blue K_m. Each y-gadget can share at most one vertex with this K_m and each x-gadget at most two. Hence every y-gadget has exactly one vertex in common with K_m and every x_i-gadget exactly two (from e_i if x_i is true or from f_i otherwise). Call a y-literal true if it one of the K_m vertices is labeled with that literal. By the construction this yields a well-defined partial assignment of truth-values to y-variables such that $\varphi(x_1, \ldots, x_k, y_1, \ldots, y_k)$ is true, contradicting the choice of the assignment to the x variables. □

3 Strong Arrowing

At the heart of the completeness proof for ARROWING was the construction of enforcers, building on the computation of $r(T, K_n)$ by Chvátal. Exact *induced* Ramsey numbers are only known for small graphs. Even finding some F with $F \rightarrowtail (G, H)$ for a given G and H is hard. The existence of such an F for every pair (G, H) was only shown in 1973, independently, by Deuber, by Erdős, Hajnal and Pósa, and by Rödl [9]. Though two of these proofs are constructive, the graph F they construct will be at least of doubly exponential size in the input graphs. Since we are only allowed polynomial time we cannot use them. Though there are better bounds for induced Ramsey numbers in the literature (see for example the forthcoming paper by Kohayakawa, Prömel and Rödl [18]) these bounds are obtained by the use of random graphs, and therefore of no use in polynomial time.

Theorem 3.1 STRONG ARROWING *is* Π_2^p-*complete.*

Since we are dealing with induced graphs we will (in this section) say that a coloring of a graph F is (G, H)-*good* if it contains neither a red G nor a blue H as an induced subgraph. This, of course, changes the notions of enforcers and determiners as defined above, and to avoid confusion we will call them strong enforcers and strong determiners.

Theorem 3.2 *Let G be a fixed connected graph of size at least two for which we can compute a strong (G, K_n)-enforcer in polynomial time. Then deciding $F \rightarrowtail (G, K_n)$ is Π_2^p-complete.*

The proof is the same as in the noninduced case, with enforcers, etc. substituted by their strong variants. We only need some additional reasoning to show that the enforcers, determiners, switches, and the final graph forces the colors in a (G, K_n)-good coloring as it did before. This, however, follows from the modularity of the construction: all the gadgets are induced subgraphs of the final graph F (the only interesting case to verify are switches since they have two signal edges; by construction the two signal edges of a switch form an induced path, and the construction of F does not change this property.) Hence the same arguments as before apply.

Corollary 3.3 *Deciding whether $F \rightarrowtail (K_{1,p}, K_n)$ is Π_2^p-complete.*

The proof of the corollary reduces (by the preceding theorem) to the construction of a strong

$(K_{1,p}, K_n)$-enforcer. As in the non-induced case there is a connection to the construction of graphs F for which $F \rightarrowtail (K_{1,p}, K_n)$. For the special case at hand we use a construction based on an idea of Thomas Hayes [14]: a complete n-partite graph with ip vertices in the ith partition class will do.

Lemma 3.4 *We can construct a strong $(K_{1,p}, K_n)$-enforcer in polynomial time in p and n.*

Proof We will show how to construct F such that $F \not\rightarrowtail (K_{1,p}, K_n)$, and F has a vertex v such that in any $(K_{1,p}, K_n)$-good coloring v is at the center of a red induced $K_{1,p-1}$. Let $F = K_{1,p-1,2p,3p,\dots,(n-1)p}$, i.e. a complete n-partite graph where the ith partition class contains $\max\{1, (i-1)p\}$ vertices with the exception of the second partition which only contains $p-1$ vertices. Coloring the $p-1$ edges between the first and second partition red and all the other edges blue shows that $F \not\rightarrowtail (K_{1,p}, K_n)$. Fix any $(K_{1,p}, K_n)$-good coloring of F. Suppose there is a blue edge from v to a vertex in the second partition. Using the fact that there is no red $K_{1,p}$ one easily shows by induction that the first i partition classes contain a blue K_i, finishing the proof. □

4 Other Graphs

How can we build enforcers for other graphs? We will sketch an example which employs a new idea. Our modest goal is to show that $F \rightarrowtail (P_4, K_n)$ is Π_2^{P}-complete. We do not quite achieve this goal, the best we can do at present is to show completeness for Π_2^{P} under truth-table reductions. A *truth-table reduction* is a reduction that makes all its queries simultaneously at once, and then decides acceptance without being able to make more queries.

We can in polynomial time compute a graph F such that $F \rightarrowtail (P_4, K_n)$ (for a proof see [22]). Fix an ordering of the edges of F and remove the edges in order one by one until finding a graph F' for which $F' \not\rightarrowtail (P_4, K_n)$ (use the oracle to do this). The search terminates since $\emptyset \not\rightarrowtail (P_4, K_n)$. Let the last edge that was removed be between vertices v and w. That is $F' \cup \{v, w\} \rightarrowtail (P_4, K_n)$ and $F' \not\rightarrowtail (P_4, K_n)$.

Consider a good coloring of F'. Adding a red edge vw obviously cannot create a blue clique, hence vw must be part of a red P_4. There are two cases: either (i) there is a good coloring of F' in which neither v nor w are incident to a red P_3 (and hence they are both incident to a red edge), or (ii) at least one of v or w is incident to a red induced P_3 in any good coloring.

In the first case we take three copies F_1, F_2, F_3 of F' and identify $v := v_1 = v_2 = v_3$ and $w := w_1 = w_2 = w_3$ to get a new graph G. In any good coloring v and w will not be incident to a P_3, but they will be incident to a P_2. Take two copies of G and connect their v vertices. This graph will function as a blue-determiner. The signal edge is the edge connecting the two copies of G. As a switch we take a copy of G in which we connect v and w by a path of length two. In the second case we take two copies F_1, F_2 of F' and identify $v := v_1 = w_2$ and $v_2 = w_1$ to get a strong enforcer G with signal vertex v. Note that we can effectively decide which case we are in: case (i) applies if and only if $G \not\rightarrowtail (P_4, K_n)$. For the remainder of the proof we can apply Theorem 3.2 in the second case. For the first case we have to carefully check the construction to see that the blue determiners can be used instead of the enforcers, and that they do not interfere with each other.

What have we shown? For the construction of the enforcer we made queries of the form $F \rightarrowtail (P_4, K_n)$. Since there are at most polynomially many queries and they can be determined ahead of time, this amounts to a truth-table reduction.

Theorem 4.1 *Deciding $F \rightarrowtail (P_4, K_n)$ is truth-table complete for Π_2^{P}.*

This example is meant to illustrate the use of ideas related to Ramsey-minimal graphs. These ideas might also be fruitful in the non-induced case.

Another question to be asked is whether the complete graph can be substituted by other families of graphs. As we transformed CLIQUE into ARROWING it should be possible to lift similar $\mathbf{NP}$-completeness results to $\mathbf{\Pi}_2^{\mathbf{P}}$. In the induced case we can indeed show that $F \rightarrowtail (P_3, P_n)$ is $\mathbf{\Pi}_2^{\mathbf{P}}$-complete, where P_n is a path on n vertices (we do not include a proof). This result could be viewed as a lifting of Yannakakis's result that finding induced paths is $\mathbf{NP}$-complete [10, GT23] (namely he showed that $F \rightarrowtail (P_2, P_n)$ is $\mathbf{NP}$-complete).

An interesting open problem is settling the complexity of $F \rightarrowtail (K_{1,n}, K_{1,m})$. In the non-induced case deciding $F \rightarrow (K_{1,n}, K_{1,m})$ can be done in polynomial time by reducing it to a matching problem [5]. This is not true for the induced case (unless $\mathbf{P} = \mathbf{NP}$): a graph G has an independent set of size at least n iff the graph obtained from G by adding a vertex and connecting it to all vertices of G contains an induced $K_{1,n}$ (for $n > 4$, remember that we can assume that G has maximal degree 3). Hence $F \rightarrowtail (K_{1,n}, K_{1,1})$ is an $\mathbf{NP}$-complete problem. It seems likely that $F \rightarrowtail (K_{1,n}, K_{1,m})$ is $\mathbf{\Pi}_2^{\mathbf{P}}$-complete.

5 Conclusion

We have shown several completeness results for both ARROWING and STRONG ARROWING and presented some general results which reduce the problem to the effective construction of graphs like (strong) enforcers and (strong) determiners. These are combinatorial problems of some trickiness. In a first step the paper [22] collects effective constructions of graphs that strongly arrow other graphs, but more work is to be done.

As a particular problem let us mention the complexity of $F \rightarrow (K_3, K_n)$. The enforcer approach will not work here. A modification (using a lemma from a paper by Burr, Nešetřil and Rödl [6]) will give the result that if a (K_3, K_n)-determiner can be computed in polynomial time, then the problem is $\mathbf{\Pi}_2^{\mathbf{P}}$-complete. This leaves open as an interesting task the construction of a (K_3, K_n)-determiner (with or without oracle help). Another question is whether we can show completeness for diagonal arrowing, i.e. a problem of the form $F \rightarrow (G, G)$ or $F \rightarrowtail (G, G)$. The main difficulty, of course, is that we cannot construct enforcers or determiners in this case, and would have to base the whole construction on switches (if this is possible). A good candidate to investigate more closely should be $F \rightarrowtail (K_{1,n}, K_{1,n})$.

The effectiveness of general Ramsey theory (as opposed to Graph Ramsey theory) is covered in a forthcoming survey by Bill Gasarch [11]. Most of the results in this area belong to computability rather than complexity. There is one exception though which also establishes a link to Graph Ramsey Theory, and that is the computation of Ramsey numbers. Remember the definition of the generalized Ramsey number of two graphs G and H as $r(G, H) = \min\{n : K_n \rightarrow (G, H)\}$. The usual Ramsey numbers can be defined from this as $r(k, l) = r(K_k, K_l)$.

The question of how hard it is to compute $r(k, l)$ is probably not a good one, since $r(k, l)$ might be exponentially large compared to k and l, so that an algorithm in the polynomial hierarchy might fail just because of the size of the output, whereas we do not expect the problem to be hard for exponential time. This problem is not an issue in the graph variant. Burr [3] considered the question of how hard it is to decide whether $r(G, H) < m$, where m is part of the input. He showed that this problem is $\mathbf{NP}$-hard. Since the best upper bound we have is $\mathbf{\Pi}_2^{\mathbf{P}}$ again this leaves us with a gap which will be more difficult to close than the one for the ARROWING problem, since we rely heavily on the structure of the graph F we are constructing. The situation for $r(k, l)$ seems worse. Burr's proof requires one of G or H to be a path, so the restriction to complete graphs necessitates new ideas. There do not seem to be any lower bounds on the complexity of computing $r(k, l)$.

In conclusion let us point out some other results inquiring into the complexity of Ramsey Theory. Burr showed that deciding whether a finite set of points in the plane can be colored with three colors such that no two points within distance K have the same color is $\mathbf{NP}$-complete [2]. This seems to be the only computational result in Euclidean Ramsey theory.

We should finally mention an analogue of $F \rightarrow (G, H)$ in computability. If the edge set of (the possibly infinite graph) F is computably enumerable, and G and H are finite, then ARROWING m-reduces to $\overline{\emptyset'}$, the complement of the halting problem (if $e_1, \ldots, e_n, \ldots$ is an enumeration of the edges of F ask whether there is a (G, H)-good coloring of F restricted to edges $e_1, \ldots, e_n$ for all n). This observation is complemented by a result of Burr's [4] who showed that $\overline{\emptyset'}$ can be m-reduced to ARROWING even if F is restricted to be a highly computable (essentially finite and periodic) graph. Hence this infinite version of ARROWING is $\mathbf{\Pi_1}$-complete.

Acknowledgments My thanks go to Gerry Brady, Janos Simon, Amber Settle and Pradyut Shah for listening to earlier and more complicated versions of the proofs. I am also grateful to Lance Fortnow for pointing out flaws in the original argument. Further thanks go to Stuart Kurtz, Laci Babai, and to Tom Hayes for coming up with the n-partite construction. Dieter Van Melkebeek deserves special thanks for several remarks that greatly simplified the proofs of the main theorems. Finally, thanks go to Martin Kummer for getting me interested in Ramsey Theory in the first place.

References

[1] Manindra Agrawal and Thomas Thierauf. The Boolean isomorphism problem. In *37th Annual Symposium on Foundations of Computer Science*, pages 422–430, Burlington, Vermont, 14–16 October 1996. IEEE.

[2] Stefan A. Burr. An NP-complete problem in euclidean ramsey theory. *Congressus Numerantium*, 35:131–138, 1982.

[3] Stefan A. Burr. Determining generalized ramsey numbers is NP-hard. *Ars Combinatoria*, 17:21–25, 1984.

[4] Stefan A. Burr. Some undecidable problems involving the edge-coloring and vertex-coloring of graphs. *Discrete Mathematics*, 50:171–177, 1984.

[5] Stefan A. Burr. On the computational complexity of ramsey-type problems. In Nesetril & Rodl, editor, *Mathematics of Ramsey Theory*. Springer-Verlag, 1990.

[6] Stefan A. Burr, Jaroslav Nešetřil, and Vojtech Rödl. On the use of senders in generalized ramsey theory of graphs. *Discrete Mathematics*, 54:1–13, 1985.

[7] Václav Chvátal. Tree-complete graph ramsey numbers. *Journal of Graph Theory*, page 93, 1977.

[8] Václav Chvátal and Frank Harary. Generalized ramsey theory for graphs, iii. small off-diagonal numbers. *Pacific Journal of mathematics*, 41(2):335–345, 1972.

[9] Reinhard Diestel. *Graph Theory*. Springer, New York, 1997.

[10] Michael R. Garey and David S. Johnson. *Computers and Intractability*. Freeman, San Francisco, 1979.

[11] William Ian Gasarch. A survey of recursive combinatorics. In Ershov, Goncharov, Nerode, and Remmel, editors, *Handbook of Recursive Algebra*. North–Holland Publishing Co.

[12] Ronald L. Graham, Bruce L. Rothschild, and Joel H. Spencer. *Ramsey Theory*. Wiley, 1990.

[13] Frank Harary, Jaroslav Nešetřil, and Vojtech Rödl. Generalized ramsey theory for graphs xiv: Induced ramsey numbers. In Miroslav Fiedler, editor, *Graphs and Other Combinatorial Topics*, pages 90–100. Teubner, 1983.

[14] Thomas Hayes. Personal communication.

[15] Edith Hemaspaandra and Gerd Wechsung. The minimization problem for Boolean formulas. In *38th Annual Symposium on Foundations of Computer Science*, pages 575–584,

Miami Beach, Florida, 20–22 October 1997. IEEE.

[16] D. T. Huynh. Deciding the inequivalence of context-free grammars with 1- letter terminal alphabet is Σ_2^p-complete. *Theoretical Computer Science*, 33(2-3):305–326, October 1984.

[17] Ker-I Ko and Wen-Guey Tzeng. Three Σ_2^P-complete problems in computational learning theory. *Computational Complexity*, 1:269–310, 1991.

[18] Y. Kohayakawa, H. J. Prömel, and Vojtech Rödl. Induced ramsey numbers. To appear in *Combinatorica*.

[19] Chih-Long Lin. Optimizing TRIEs for ordered pattern matching is Π_2^P-complete. In *Proceedings of the 10th Annual Conference on Structure in Complexity Theory (SCTC '95)*, pages 238–245, Los Alamitos, CA, USA, June 1995. IEEE Computer Society Press.

[20] Frank Plumpton Ramsey. On a problem of formal logic. In *Proceedings of the London Mathematical Society*, volume xxx of *2*, pages 264–286. 1930.

[21] Yehoshua Sagiv and Mihalis Yannakakis. Equivalences among relational expressions with the union and difference operators. *Journal of the Association for Computing Machinery*, 27(4):633–655, October 1980.

[22] Marcus Schaefer and Pradyut Shah. Induced graph ramsey theory. Unpublished manuscript.

[23] Christopher Umans. The minimum equivalent DNF problem and shortest implicants. In *IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 556–563, 1998.

[24] Klaus W. Wagner. The complexity of combinatorial problems with succinct input representation. *Acta Informatica*, 23(3):325–356, 1986.