Journal Title: Annals of Pure and Applied Logic

Volume: 45
Issue:
Month/Year: 1989
Pages: 1-38

Article Author: Richard Beigel and William Gasarch

Article Title: On the complexity of finding the chromatic number of a recursive graph I: The bounded case

Imprint:

CUSTOMER HAS REQUESTED:
Mail to Address

William Gasarch (000000350541)
College Park, MD 20742

38 pages

Annals of Pure and Applied Logic 45 (1989) 1–38
North-Holland

ON THE COMPLEXITY OF FINDING THE CHROMATIC NUMBER OF A RECURSIVE GRAPH I: THE BOUNDED CASE

Richard BEIGEL
Department of Computer Science, The Johns Hopkins University, Baltimore, MD 21218, USA

William I. GASARCH
Department of Computer Science, University of Maryland, College Park, MD 20742, USA

Communicated by A. Nerode
Received July 1988

We classify functions in recursive graph theory in terms of how many queries to K (or $\emptyset''$ or $\emptyset'''$) are required to compute them. We show that (1) binary search is optimal (in terms of the number of queries to K) for finding the chromatic number of a recursive graph and that no set of Turing degree less than $\mathbf{0}'$ will suffice, (2) the problem of determining if a recursive graph has a finite chromatic number is Σ_2-complete, and (3) binary search is optimal (in terms of the number of queries to $\emptyset'''$) for finding the recursive chromatic number of a recursive graph and that no set of Turing degree less than $\mathbf{0}'''$ will suffice. We also explore how much help queries to a weaker set may provide. Some of our results have analogues in terms of asking p questions at a time, but some do not. In particular, $(p+1)$-ary search is not always optimal for finding the chromatic number of a recursive graph. Most of our results are also true for highly recursive graphs, though there are some interesting differences when queries to K are allowed for free in the computation of a recursive chromatic number.

1. Introduction

We examine the complexity of several graph coloring problems in recursive graph theory. All the problems we deal with are unsolvable, but are recursive in either K (the halting set), $\emptyset''$ (the jump of the halting set, see [29] or [34]) or $\emptyset'''$ (the jump of the jump of the halting set). We measure the complexity of these problems in two ways: the Turing degree of the oracle and the number of queries to that oracle. In most cases we pin down both quantities exactly. Henceforth 'graph' means 'recursive or highly recursive graph,' terms we define in Section 2.

We will be concerned with finding the chromatic number of a graph when that number is *a priori* bounded above by a constant. Unbounded versions of problems in this paper are studied in [11]. In Section 2 we rigorously define the class of functions that can be computed with bounded access to an oracle for set A. We state a theorem about how many queries the function

$$F_k^A(x_1,\ldots,x_k)=\langle\chi_A(x_1),\ldots,\chi_A(x_k)\rangle,$$

(χ_A is the characteristic function of the set A) may require to be computed. This theorem is used to establish lower bounds. In Section 3 we show that finding the

0168-0072/89/$3.50

chromatic number of a graph requires an oracle of degree at least $\mathbf{0}'$ and that a binary search algorithm uses the minimal number of queries. This result is tight in two ways: the lower bound on the number of queries holds even is a different (e.g., more powerful) oracle is used, and no matter how many queries are used, the oracle must be of degree at least $\mathbf{0}'$. In Section 4 we show that determining whether the chromatic number of a graph is finite is Σ_2-complete, and hence requires a $\emptyset''$ oracle. In Section 5 we look at the harder question of determining the recursive chromatic number of a graph, i.e. the minimum number of colors required to achieve a recursive coloring. We show that finding the recursive chromatic number of a graph requires an oracle of degree at least $\mathbf{0}'''$ and that a binary search algorithm uses the minimal number of queries. This result is tight in the same ways the result in Section 3 was tight. In Section 6 we show that determining if a recursive graph has a finite recursive chromatic number is Σ_3-complete, but determining if a highly recursive graph has a finite recursive chromatic number is Σ_2-complete. In Section 7 we examine how much the number of queries can be reduced if an auxiliary (but weaker) oracle is allowed to be used free of charge. In Section 8 we examine parallel versions of the questions raised in Sections 3 and 5. We examine how hard it is to find the chromatic number (recursive chromatic number) of a graph in terms of the number of queries to F_p^K $(F_p^{\emptyset'''})$ that are required. Some of the results obtained in Sections 3 and 5 have analogs in this new setting. In particular, when using K, $(p+1)$-ary search [25, 33] is optimal for finding the chromatic number of a graph. If other oracles can be used, then $(p+1)$-ary search is not optimal. In Section 9 we examine using parallel queries and an auxiliary (weaker) oracle. Section 10 contains a summary of our results and some open questions.

Other work on bounded queries in a recursion–theoretic context has been done by Beigel, Gasarch, Gill, Hay and Owings [7, 10, 12, 13, 14, 28]. In a polynomial framework, work on bounded queries has been done by Amir, Beigel, and Gasarch [1, 2, 5, 7, 8, 9, 17], Goldsmith, Joseph, and Young [19], Kadin [20], Krentel [24], Rosier and Yen [30], and Wagner and Wechsung [36, 37]. Other work on recursive graph theory has been done by Bean [3, 4], Burr [15], Carstens and Pappinghaus [16], Gasarch and Lockwood [18], Kierstead [21, 22, 23], Manaster and Rosenstein [26, 27] Schmerl [31, 32] and Tverberg [35].

2. Notation, conventions and useful known results

All logarithms in this paper are base two, and all graphs are undirected. Throughout this paper $\{0\}^{()}, \{1\}^{()}, \ldots$ is a list of all oracle Turing machines. A subscript s on any of these computations means that the computation only runs for s steps. Let $\{e\}$ denote $\{e\}^{\emptyset}$. Let W_e denote the domain of $\{e\}$, hence the set $\{W_e \mid e \in \mathbb{N}\}$ is the set of all recursively enumerable sets. Let $W_{e,s}$ be W_e after s stages, i.e. $\{0, 1, 2, \ldots, s\} \cap \{x \mid \{e\}_s(x)\downarrow\}$. K represents the halting set, K_s denotes $\{x \mid x \in W_{x,s}\}$. FIN represents the set of indices of functions that are only defined finitely often, i.e. $\{e \mid W_e \text{ is finite}\}$. TOT represents the set of indices of functions that are always defined, i.e. $\{e \mid W_e = \mathbb{N}\}$. COF represents the set of indices of cofinite sets, i.e. $\{e \mid \mathbb{N} - W_e \text{ is finite}\}$. $\bar{K}$ is Π_1-complete, FIN is Σ_2-complete, TOT is Π_2-complete, and COF is Σ_3-complete [34, p. 65–66].

Let A be any set of natural numbers. The function χ_A, called the characteristic function of A, is defined by

$$\chi_A(x) = \begin{cases} 1 & \text{if } x \in A, \\ 0 & \text{if } x \notin A. \end{cases}$$

We identify a set with its characteristic function. A^i denotes $A \times A \times \cdots \times A$ (i times), the set of all i-tuples of elements of A. The set of *unordered pairs* of elements of A is denoted $[A]^2$. $A[\omega]$ denotes $A \cap \{0, 1, 2, \ldots, w\}$.

Let $\mathbb{N}$ denote the set of natural numbers. We denote a fixed recursive pairing (tripling, etc.) bijection from $\mathbb{N} \times \mathbb{N}$ onto $\mathbb{N}$ ($\mathbb{N} \times \mathbb{N} \times \mathbb{N}$ onto $\mathbb{N}$, etc.) by $\langle -, - \rangle$ ($\langle -, -, - \rangle$, etc). We denote a fixed recursive bijection from the set $[\mathbb{N}]^2$ onto $\mathbb{N}$ by '$[-, -]$', so the symbol '$[x, y]$' is a natural number which represents the unordered pair $\{x, y\}$. Since these functions are recursive and onto they have recursive inverses.

If A and B are sets, then $A \oplus B$ is the set

$$\{2x \mid x \in A\} \cup \{2x + 1 \mid x \in B\}.$$

An oracle machine using oracle $A \oplus B$ can essentially ask either A or B questions. If an even number is queried, we say that a query to A has been made, and when an odd number is queried, we say that a query to B has been made.

If A is a finite set, then $|A|$ denotes the cardinality of A.

Definition. A graph $G = (V, E)$ is *recursive* if every node of G has a finite number of neighbors and both $V \subseteq \mathbb{N}$ and $E \subseteq [\mathbb{N}]^2$ are recursive.

Definition. A graph $G = (V, E)$ is *highly recursive* if G is recursive and the function that produces all the neighbors of a given node is recursive.

Note. *Most of the theorems in this paper will be stated and proven for recursive graphs, but are also true for highly recursive graphs unless otherwise noted.*

If G is a graph, then $\chi(G)$ (the chromatic number of G) is the minimal number of colors required to color the vertices of G such that no two adjacent vertices have the same color (called a 'proper coloring'). By convention the empty graph $(\emptyset, \emptyset)$ has chromatic number 0.

We need a representation for recursive graphs. We will represent graphs by the Turing machines that determine their vertex and edge sets. An index for a graph will be an ordered pair, the first component of which is an index for a Turing machine which decides the vertex set, the second the edge set.

Definition. If $\{e_1\}$ and $\{e_2\}$ are total, then the number $e = \langle e_1, e_2\rangle$ *determines the recursive graph* $G_e^{\mathrm{r}} = (V, E)$, where

$$V = \{x \mid \{e_1(x) = 1\},$$
$$E = \{\{x, y\} \mid x, y \in V \text{ and } \{e_2\}([x, y]) = 1\}.$$

If $\{e_1\}$ or $\{e_2\}$ is not total, then e does not determine a recursive graph. (The 'r' in G_e^{r} stands for 'recursive').

Definition. A number $\langle e_1, e_2\rangle$ *determines a highly recursive graph* if $\{e_1\}$ and $\{e_2\}$ are total, and when $\{e_2\}$ is interpreted as a mapping from $\mathbb{N}$ to finite subsets of $\mathbb{N}$, if $\{e_2\}(x) = Y$ then for all $y \in Y$, $x \in \{e_2\}(y)$ (Y is the set of vertices that x is adjacent to). If e determines a highly recursive graph, then the *highly recursive graph determined by* e is $G_e^{\mathrm{hr}} = (V, E)$ where

$$V = \{x \mid \{e_1(x) = 1\},$$
$$E = \{\{x, y\} \mid x \in \{e_2\}(y)\}.$$

If $\{e_1\}$ or $\{e_2\}$ is not total, then e does not determine a highly recursive graph. (The 'hr' in G_e^{hr} stands for 'highly recursive'.)

Note. Another valid representation would be to only insist that $\{e_2\}([x, y])\downarrow$ when $\{e_1\}\{x\} = \{e_1\}(y) = 1$, instead of demanding that $\{e_2\}$ be total. All of our results would also hold using that representation.

In this paper we will classify, in the arithmetic hierarchy, many sets of indices of recursive graphs (henceforth called just 'indices'). Our concern is **not** with determining if a number is an index of a recursive graph, hence we will actually classify 0–1 valued partial recursive functions that are associated to sets of indices. Determining if e determines either a recursive or highly recursive graph is equivalent to TOT, so it is Π_2-complete.

Definition. A 0–1 valued partial function f is in Σ_n if there exists a partial recursive function g such that

$$f(x) = \begin{cases} 1 & \text{if } (\exists y_1)(\forall y_2)\cdots(\cdots y_n)\, g(y_1, y_2, \ldots, y_n, x)\downarrow = 1 \text{ and } x \in \mathrm{Domain}(f), \\ 0 & \text{if } (\forall y_1)(\exists y_2)\cdots(\cdots y_n)\, g(y_1, y_2, \ldots, y_n, x)\downarrow = 0 \text{ and } x \in \mathrm{Domain}(f). \end{cases}$$

A 0–1 valued partial function f is in Π_n if there exists a partial recursive function g such that

$$f(x) = \begin{cases} 1 & \text{if } (\forall y_1)(\exists y_2)\cdots(\cdots y_n)\, g(y_1, y_2, \ldots, y_n, x)\downarrow = 1 \text{ and } x \in \mathrm{Domain}(f), \\ 0 & \text{if } (\exists y_1)(\forall y_2)\cdots(\cdots y_n)\, g(y_1, y_2, \ldots, y_n, x)\downarrow = 0 \text{ and } x \in \mathrm{Domain}(f). \end{cases}$$

Note. The function $g(y_1, \ldots, y_n, x)$ in the above definition need not be defined when $x \notin \mathrm{Domain}(f)$, though it can be.

Definition. A 0–1 valued partial function f is Σ_n*-complete* if $f \in \Sigma_n$ and f is Σ_n-hard, i.e. if X is a Σ_n set, then there exists a recursive function g such that

$$x \in X \quad \text{iff} \quad f(g(x)) = 1.$$

A 0–1 valued partial function f is Π_n*-complete* if $f \in \Pi_n$ and f is Π_n-hard, i.e. if X is a Π_n set, then there exists a recursive function g such that

$$x \in X \quad \text{iff} \quad f(g(x)) = 1.$$

Let $I = \{e \mid e$ is the index of a recursive graph$\}$. Then for $A \subseteq I$, we think of A as being the 0–1 valued partial function which is 1 on A, 0 on $I - A$, and undefined otherwise. Most of the functions that we are concerned with are only defined on I or some subset of I. If the value of a function at a point is not stated, then it is assumed to be undefined there. We will use the term 'function' even if we mean a partial function defined on I or a subset of I.

We need to approximate infinite graphs by how they look after some finite time, so we make the following definition:

Definition. Let $e = \langle e_1, e_2\rangle$ be a number that determines a recursive graph. We define the *approximation to* G_e^{r} *by stage* s ($G_{e,s}^{\mathrm{r}}$) to be the subgraph of G_e^{r} formed by taking all nodes in the set $\{1, 2, 3, \ldots, s\}$ that are in the graph and connecting them as they are connected in the graph. Formally, $G_{e,s}^{\mathrm{r}} = (V, E)$ where

$$V = \{1, 2, 3, \ldots, s\} \cap \{x \mid \{e_1\}(x) = 1\},$$
$$E = [V]^2 \cap \{\{u, v\} \mid \{e_2\}([u, v]) = 1\}.$$

We will often exhibit many finite graphs and take their union, in a way so that all vertices are distinct. We formalize this:

Definition. If $G_1 = (V_1, E_1)$, $G_2 = (V_2, E_2), \ldots$ are graphs, then the *disjoint union of* $G_1, G_2, \ldots$ is the union of the G_i's with all vertices relabeled to be distinct. Formally it is the graph (V, E) where

$$V = \bigcup_{i=1}^{\infty} \{i\} \times V_i,$$

$$E = \bigcup_{i=1}^{\infty} \{(i, u), (i, v)\} \mid u, v \in V_i \text{ and } \{u, v\} \in E_i\}.$$

We formally define the class of functions which can be computed by an oracle Turing machine, with a bound on the number of queries it can make.

Definition. Let g be a total function and $n \geq 0$ be a number. A partial function f is in FQ(n, g) if $f \leq_{\mathrm{T}} g$ via an oracle Turing machine which uses oracle g, and never makes more than n queries. If g is the characteristic function of a set A, then we use the notation FQ(n, A). (This will usually be the case in this paper.)

Definition. Let A and B be sets, and $n \geq 0$ be a number. A partial function f is in $\mathrm{FQ}^B(n, A)$ if $f \leq_T A \oplus B$ via an oracle Turing machine, which uses oracle $A \oplus B$ and never makes more than n queries to A, though it may make arbitrarily many queries to B. A similar definition can be made when A is a function instead of a set.

The following function will be useful to us.

Definition. Let A be any set and $k \geq 1$ be a number. The function F_k^A is defined by

$$F_k^A(x_1, \ldots, x_k) = \langle \chi_A(x_1), \ldots, \chi_A(x_k) \rangle,$$

where χ_A is the characteristic function of A.

The following lemmas are proven in [12].

Lemma 1. *If A and X are sets, A is nonrecursive, and n is any number, then*

$$F_{2^n}^A \notin \mathrm{FQ}(n, X).$$

Lemma 2. *For any numbers $x_1, \ldots, x_n$, given the value of $|K \cap \{x_1, \ldots, x_n\}|$, the value of $F_n^K(x_1, \ldots, x_n)$ can be computed.*

Proof. Let $m = |K \cap \{x_1, \ldots, x_n\}|$. Run all the machines $\{e\}(e)$ for $e \in \{x_1, \ldots, x_n\}$ until exactly m of them halt. Output the information that those m are in K, and the rest are not. □

3. Chromatic number of recursive graphs

We apply the theorems stated in Section 2 to classify graph colorability problems in the $\mathrm{FQ}(n, K)$ hierarchy. We show that if c is any constant, then the function that determines $\chi(G)$ (where G is a recursive graph and $\chi(G) \leq c$) can be computed in $\lceil \log(c+1) \rceil$ queries to K, but cannot be computed in $\lceil \log(c+1) \rceil - 1$ queries to *any* oracle.

Lemma 3. *Let $k \geq 0$ be a fixed natural number. Let A_k be the partial recursive function defined by*

$$A_k(e) = \begin{cases} 1 & \text{if } G_e^{\mathrm{r}} \text{ exists and } \chi(G_e^{\mathrm{r}}) \leq k, \\ 0 & \text{if } G_e^{\mathrm{r}} \text{ exists and } \chi(G_e^{\mathrm{r}}) > k, \\ \text{undefined} & \text{if } G_e^{\mathrm{r}} \text{ does not exist.} \end{cases}$$

A_k is Π_1-complete.

Proof. Since a graph is k-colorable iff all its finite subgraphs are k-colorable, $\chi(G_e^{\mathrm{r}}) \leq k$ iff for all s, $\chi(G_{e,s}^{\mathrm{r}}) \leq k$. Therefore

$$A_k(e) = \begin{cases} 1 & \text{if } (\forall s)\, \chi(G_{e,s}^{\mathrm{r}}) \leq k, \\ 0 & \text{if } (\exists s)\, \chi(G_{e,s}^{\mathrm{r}}) > k, \\ \text{undefined} & \text{otherwise.} \end{cases}$$

The function that, for given e and s, checks whether $\chi(G_{e,s}^{\mathrm{r}}) \leq k$, is partial recursive and is defined when G_e^{r} exists. Hence A_k is in Π_1 (using the definition of a 0–1 valued partial function being in Π_n given in Section 2).

We show that A_k is Π_1-hard by showing that $\bar{K} \leq_m A_k$. Given a number x, let G_s be a clique of size $k+1$, if $x \in W_{x,s}$ and $(\emptyset, \emptyset)$ otherwise; and let G be the disjoint union of $G_1, G_2, \ldots$. For the G so constructed

G is k-colorable iff G contains no clique of size $k+1$ iff $(\forall s)\, x \notin W_{x,s}$ iff $x \notin K$. □

Note. if $k \geq 3$, then the graphs reduced to in the above lemma can be made connected.

Lemma 3 shows that determining the chromatic number of a graph *requires* an oracle of degree at least $\mathbf{0}'$. Theorem 4 gives an exact bound on *how many* queries to K are required to actually find $\chi(G_e^{\mathrm{r}})$.

Theorem 4. *Let $c \geq 1$ by any number. Let g be the function*

$$g(e) = \begin{cases} \chi(G_e^{\mathrm{r}}) & \text{if } \chi(G_e^{\mathrm{r}}) \leq c, \\ c & \text{if } \chi(G_e^{\mathrm{r}}) \geq c. \end{cases}$$

The function g is in $\mathrm{FQ}(\lceil \log(c+1) \rceil, K)$. If X is any set, then

$$g \notin \mathrm{FQ}(\lceil \log(c+1) \rceil - 1, X).$$

Proof. Using the previous lemma and a binary search on $[0, c]$ for the proper number of colors, one obtains that g is in $\mathrm{FQ}(\lceil \log(c+1) \rceil, K)$. First ask if the graph is $\lfloor c/2 \rfloor$-colorable, and keep cutting the current interval of possible chromatic numbers in half until it only has one element in it.

Let X be any set. To establish that g is not in $\mathrm{FQ}(\lceil \log(c+1) \rceil - 1, X)$ we show that if it is then $F_{2^n}^K \in \mathrm{FQ}(n, X)$ (where $n = \lceil \log(c+1) \rceil - 1$), which contradicts Lemma 1.

We describe an algorithm to determine $F_{2^n}^K(x_1, \ldots, x_{2^n})$ that will use one call to the function g; hence if g is in $\mathrm{FQ}(n, X)$ then the function $F_{2^n}^K$ is in $\mathrm{FQ}(n, X)$.

For $s = 1, 2, \ldots, 2^n$ let

$$G_s = \begin{cases} \text{the empty graph} & \text{if } |K \cap \{x_1, \ldots, x_{2^n}\}| < s, \\ \text{the complete graph on } s \text{ vertices} & \text{otherwise.} \end{cases}$$

Let G_e^r be the disjoint union of $G_1, G_2, G_3, \ldots, G_{2^n}$. Then $\chi(G_e^r) = |K \cap \{x_1, \ldots, x_{2^n}\}|$, which is $\leq 2^n \leq c$. By Lemma 2, $F_{2^n}^K$ can be computed from $|K \cap \{x_1, \ldots, x_{2^n}\}| = \chi(G_e^r)$. Note that an index for G_e^r can be computed (uniformly) from $\{x_1, \ldots, x_{2^n}\}$. Hence $F_{2^n}^K$ can be computed from a single query to g. $\square$

Lemma 3 and Theorem 4 show that the binary search algorithm for g, which gives $g \in \mathrm{FQ}(\lceil \log(c+1) \rceil, K)$ is optimal in terms of both Turing degree and number of queries. Even if an oracle of larger Turing degree is used, the number of queries must be at least $\lceil \log(c+1) \rceil$; and even if more queries were allowed, a set of Turing degree at least $\mathbf{0}'$ is required. There is no tradeoff between the number of queries and Turing degree which would allow a reduction in either one.

4. Finiteness of chromatic number

In this section we show that determining if a graph has a finite chromatic number is Σ_2-complete.

Theorem 5. *The partial function*

$$A(e) = \begin{cases} 1 & \text{if } G_e^r \text{ exists and } \chi(G_e^r) < \infty, \\ 0 & \text{if } G_e^r \text{ exists and } \chi(G_e^r) = \infty, \\ \text{undefined} & \text{if } G_e^r \text{ does not exist} \end{cases}$$

is Σ_2-complete.

Proof. The partial recursive function A is

$$A(e) = \begin{cases} 1 & \text{if } G_e^r \text{ exists and } (\exists k)(\forall s)\, \chi(G_{e,s}^r) \leq k, \\ 0 & \text{if } G_e^r \text{ exists and } (\forall k)(\exists s)\, \chi(G_{e,s}^r) > k, \\ \text{undefined} & \text{if } G_e^r \text{ does not exist.} \end{cases}$$

The function that, given e and s, determines whether if $\chi(G_{e,s}^r) \leq k$, is partial recursive and is defined when G_e^r exists. Hence A is in Σ_2 (using the definition of a 0–1 valued partial function being in Σ_n given in Section 2).

We show that A is Σ_2-hard by showing $FIN \leq_m A$. For a given x, let G_s ($s \in \{1, 2, 3, \ldots\}$) be a clique of size $|W_{x,s}|$; and let G_e^r be the disjoint union of $G_1, G_2, \ldots$. If $x \in FIN$, then W_x is finite, so $\chi(G_e^r) < \infty$ and $e \in A$. If $x \notin FIN$, then W_x is infinite, so G_e^r will contain arbitrarily large cliques, and $e \notin A$. $\square$

5. Recursive chromatic number

In Section 3 we considered the problem of finding the minimal number of colors needed to color a graph. In this section we consider the problem of finding the minimal number of colors needed to *recursively* color a graph.

Definition. Let k be a nonnegative integer. If $G = (V, E)$ is any graph such that $V \subseteq N$ then G is *recursively k-colorable* if there exists a Turing machine $\{m\}$ such that for all x, $\{m\}(x)\downarrow \in \{1, 2, \ldots, k\}$; and if x and y are two nodes in V such that $\{x, y\} \in E$, then $\{m\}(x) \neq \{m\}(y)$. The empty graph is recursively 0-colorable by convention.

Definition. If G is a graph, then the *recursive chromatic number* of G (denoted $\chi^r(G)$) is the least number of colors required to recursively color G.

Note. the definition of a recursive k-coloring can be changed to only requiring that for x a vertex, $\{m\}(x)\downarrow \in \{1, 2, 3, \ldots, k\}$ without effecting any of our results.

It is known [3] that there are recursive graphs that are 3-colorable but not *recursively* k-colorable for any k. Highly recursive graphs are better behaved in that every k-colorable highly recursive graph is *recursively* $2k-1$-colorable [16, 32], although there exist k-colorable highly recursive graphs that cannot be *recursively* $2k-2$ colored [32].

We show that finding the recursive chromatic number of a graph is harder (in terms of Turing degree) than finding the chromatic number. The problem of determining if G is k-colorable is Π_1-complete; however, we show that determining if G is recursively k-colorable is Σ_3-complete.

The next lemma gives us a way to show that the problem of determining whether or not a graph is recursively k-colorable is Σ_3-hard. It gives us more information than we need at present, however, we will need its full strength in Section 7. We state it for highly recursive graphs because a stronger form of it is true for recursive graphs (see the next section).

Definition. Two r.e. sets X and Y are *recursively separable* if there is a recursive set that contains X and is disjoint from Y. If two r.e. sets are not recursively separable, they are called *recursively inseparable*. Define SEP to be the set

$$SEP = \{\langle y, z\rangle \mid W_y \text{ and } W_z \text{ are recursively separable}\}.$$

Note. SEP is Σ_3-complete [34].

Lemma 6. *For any $k \geq 2$ and any m such that $k \leq m \leq 2k-1$, there exists a recursive function $f_{k,m}$ such that for all x, $\chi(G^{hr}_{f_{k,m}(x)}) = k$ and*

$$x \in COF \Rightarrow \chi^r(G^{hr}_{f_{k,m}(x)}) = k,$$
$$x \notin COF \Rightarrow \chi^r(G^{hr}_{f_{k,m}(x)}) = m.$$

A similar function for recursive graphs also exists.

Proof. Assume m is odd, $m = 2a - 1$. Schmerl [32] (or alternatively Appendix A of this paper) showed how to construct a highly recursive graph G such that $\chi(G) = a$ and $\chi^r(G) = 2a - 1$. Schmerl's construction uses two r.e. sets X and Y that are recursively inseparable. If the sets X and Y are not recursively inseparable, then the graph constructed has recursive chromatic number a. Hence the construction can be modified to include a parameter x such that if $x \in SEP$, then the graph has recursive chromatic number a, and if $x \notin SEP$, then the graph has recursive chromatic number m. Since both COF and SEP are Σ_3 complete (see [34]), the parameterized construction can be modified to let COF take the place of SEP. Let $G(x)$ denote the graph constructed with parameter x. For all x, $\chi(G(x)) = a$ and

$$x \in COF \Rightarrow \chi^r(G(x)) = a,$$
$$x \notin COF \Rightarrow \chi^r(G(x)) = m.$$

(Alternatively, one can modify the version of Schmerl's construction in Appendix A using the techniques of Theorem 9.)

Let K_k denote the complete graph on k vertices. Let $f_{k,m}(x)$ be such that

$$G^{hr}_{f_{k,m}(x)} = G(x) \cup K_k.$$

Note that $\chi(G^{hr}_{f_{k,m}(x)}) = k$. Hence

$$x \in COF \Rightarrow \chi^r(G(x)) = a \Rightarrow \chi^r(G^{hr}_{f_{k,m}(x)}) = k,$$
$$x \notin COF \Rightarrow \chi^r(G(x)) = m \Rightarrow \chi^r(G^{hr}_{f_{k,m}(x)}) = m.$$

Assume m is even, $m = 2a - 2$ ($a \geq 3$). By the modification of Schmerl's construction in Appendix A, there is a highly recursive graph G such that $\chi(G) = a$ and $\chi^r(G) = 2a - 2 = m$. This construction can be modified (using the techniques of Theorem 9 of this paper) to include a parameter x such that if $G(x)$ denotes the graph constructed, then

$$x \in COF \Rightarrow \chi^r(G(x)) = a,$$
$$x \notin COF \Rightarrow \chi^r(G(x)) = m.$$

The rest of the proof is analogous to the case where m is odd.

It is easy to pass from an index of a highly recursive graph G to an index of G as a recursive graph. Hence the functions $f_{k,m}$ exist for recursive graphs as well. □

Theorem 7. *Let $k \geq 2$ be a fixed natural number. Let $A_k = \{e \mid \chi^r(G^r_e) \leq k\}$. The set A_k is Σ_3-complete.*

Proof. To determine the membership of $e = \langle e_1, e_2 \rangle$ in A_k we need to know if there exists a Turing machine $\{m\}$ such that

(1) For all x, $\{m\}(x)\downarrow \in \{1, 2, 3, \ldots, k\}$.

(2) The function computed by $\{m\}$ restricted to the nodes of G^r_e (i.e. the numbers x such that $\{e_1\}(x) = 1$) is a proper coloring of G^r_e.

This can be phrased as a Σ_3 set:

$$A_k = \{\langle e_1, e_2 \rangle \mid \exists m\, \forall x,y\, \exists s\, [[\{e_1\}_s(x)\downarrow] \wedge [\{e_1\}_s(y)\downarrow] \wedge [\{e_2\}_s([x,y])\downarrow] \wedge [\{m\}_s(x)\downarrow \in \{1, 2, \ldots, k\}] \wedge [\{m\}_s(y)\downarrow \in \{1, 2, \ldots, k\}] \wedge [\{e_2\}_s([x,y]) = 1 \Rightarrow \{m\}_s(x) \neq \{m\}_s(y)]]\}.$$

Hence A_k is in Σ_3.

If $k \geq 2$, we show that A_k is Σ_3-hard. Let $f_{k,k+1}$ be the function defined in Lemma 6, for recursive graphs. By the properties of $f_{k,k+1}$

$$x \in COF \Rightarrow \chi^r(G^{hr}_{f_{k,k+1}(x)}) = k \Rightarrow f_{k,k+1}(x) \in A_k,$$
$$x \notin COF \Rightarrow \chi^r(G^{hr}_{f_{k,k+1}(x)}) = k + 1 \Rightarrow f_{k,k+1}(x) \notin A_k.$$

This shows that $COF \leq_m A_k$. Hence A_k is Σ_3-complete. □

Note. Theorem 7 did not need to use the conventions associated with 0–1 valued partial recursive functions; it states that a set is Σ_3-complete.

Theorem 7 shows that determining the recursive chromatic number of a graph *requires* an oracle of degree at least $\mathbf{0}'''$. Theorem 8 gives an exact bound on *how many* queries to $\emptyset'''$ are required to actually find $\chi^r(G^r_e)$.

Theorem 8. *Let $c \geq 1$ be any number. Let h be the function*

$$h(e) = \begin{cases} \chi^r(G^r_e) & \text{if } \chi^r(G^r_e) \leq c, \\ c & \text{if } \chi^r(G^r_e) \geq c. \end{cases}$$

The function h is in $\mathrm{FQ}(\lceil \log(c+1) \rceil, \emptyset''')$. *If X is any set, then*

$$h \notin \mathrm{FQ}(\lceil \log(c+1) \rceil - 1, X).$$

Proof. We determine $\chi^r(G^r_e)$ by performing binary search on the interval $[0, c]$. Since $\emptyset'''$ is Σ_3-complete and Theorem 7 shows that $A_k \in \Sigma_3$, we can determine if $\chi^r(G^r_e) \leq k$ by making a single query to $\emptyset'''$. Binary search requires only $\lceil \log(c+1) \rceil$ queries. If $\chi^r(G^r_e) > c$, then binary search will give the answer c.

To obtain the lower bound, note that in the proof of Theorem 4 all the graphs G constructed were such that $\chi(G) = \chi^r(G)$. Therefore, that proof establishes that if X is any set, then $h \notin \mathrm{FQ}(\lceil \log(c+1) \rceil - 1, X)$. □

6. Finiteness of recursive chromatic number

In this section we show that determining if a recursive graph has a finite recursive chromatic number is Σ_3-complete; and that the same problem for highly recursive graphs is Σ_2-complete. These results are surprising for two reasons: (1) all problems encountered so far in this paper have been equally difficult for recursive and highly recursive graphs; and (2) by Theorem 7, the problem of determining if the recursive chromatic number of a highly recursive graph is $\leqslant k$ (fixed k) is Σ_3-complete, hence one would naively conjecture that adding a '$\exists k$' to the predicate would keep the problem Σ_3-complete.

Theorem 9. *The set $A = \{e \mid \chi^r(G_e^r) < \infty\}$ is Σ_3-complete.*

Proof. Note that $A = \{e \mid (\exists k)\, \chi^r(G_e^r) \leqslant k\}$. By Theorem 7 this can be written as a Σ_3 predicate, hence A is in Σ_3.

To show that A is Σ_3-hard, we show $COF \leqslant_m A$. Given x, we construct a recursive graph $G(x) = G$ such that

$$\chi^r(G) < \infty \quad \text{iff} \quad W_x \text{ is cofinite.}$$

We use a modification of Bean's construction of a recursive graph which is 3-colorable but not recursively colorable [3]. In our modification the recursive graph is 2-colorable (but not connected) and we weave the set W_x into the construction in such a way that if W_x is cofinite, then the construction fails and $\chi^r(G) = 2$; and if W_x is not cofinite, then the construction succeeds and, because the graph is not recursively colorable, $\chi^r(G)$ does not exist.

We 'try' to satisfy the following requirements:

$$R_{\langle e,i\rangle}: \quad \{e\} \text{ is not an } i\text{-coloring of } G.$$

The following claim is implicit in Bean [3]. It will henceforth be referred to as 'Bean's Claim'. It is proven in Appendix B.

Bean's Claim. *Let $\hat{L}_0$ be the graph consisting of 2^i isolated vertices, and let $\{e\}$ be a Turing machine. There exists a finite sequence of finite graphs $\hat{L}_0, \hat{L}_1, \ldots, \hat{L}_r$ such that the following hold.*

(a) *For every i, $1 \leqslant i \leqslant r$, $\hat{L}_i$ is an extension of $\hat{L}_{i-1}$, i.e. $\hat{L}_0 \subseteq \hat{L}_1 \subseteq \hat{L}_2 \subseteq \cdots \subseteq \hat{L}_r$.*

(b) *For every i, $1 \leqslant i \leqslant r$, $\hat{L}_i$ can be obtained recursively from $\hat{L}_{i-1}$ and the values of $\{e\}(x)$ for every $x \in \hat{L}_{i-1}$. If there is a vertex in $\hat{L}_{i-1}$ on which $\{e\}$ diverges, then $\hat{L}_{i-1} \doteq \hat{L}_r$.*

(c) *The function $\{e\}$ is not an i-coloring of $\hat{L}_r$.*

(d) *$\hat{L}_r$ is 2-colorable.*

We assign to each $R_{\langle e,i\rangle}$ an infinite set of graphs, each consisting of 2^i isolated vertices. At any single stage the construction tries to satisfy $R_{\langle e,i\rangle}$ by working on a particular graph (in the manner specified by Bean's Claim) with which we associate a marker. The marker may change as W_x grows. Our intention is the following: if W_x is cofinite, then almost all the markers will go to infinity, so almost all requirements will not be able to work with any particular graph long enough to be satisfied, which will make the graph recursively colorable; and if W_x is not cofinite, then all the markers will approach limits, so eventually all requirements will have a graph to work with permanently, and will be satisfied.

Recursively partition the set of natural numbers into an infinite set of infinite sets. We index the parts of the partition by the numbers $-1, 0, 1, 2, \ldots$. Let the partition be denoted by

$$\{X_{\langle e,i\rangle} \mid (e, i) \in \mathbb{N} \times \mathbb{N}\} \cup \{X_{-1}\}.$$

For each $\langle e, i\rangle \in \mathbb{N}$ recursively partition $X_{\langle e,i\rangle}$ into an infinite number of sets of size 2^i. Let this partition be denoted by

$$\{L_{\langle e,i\rangle}(j) \mid j \geqslant \langle e, i\rangle\}.$$

The construction proceeds in stages. G^s is the graph at the end of stage s. G is the graph $\bigcup_{s=0}^{\infty} G^s$. In the construction we will, for each $\langle e, i\rangle$, connect up the elements of $L_{\langle e,i\rangle}(j)$ into a graph, and then add auxiliary vertices and edges to that graph as indicated in Bean's Claim, to force $\{e\}$ not to be an i-coloring. $L^s_{\langle e,i\rangle}(j)$ denotes $L_{\langle e,i\rangle}(j)$ together with all vertices and edges added to it by stage s.

For a fixed requirement $R_{\langle e,i\rangle}$, and a fixed stage s, we will have a unique j such that we work only on $L_{\langle e,i\rangle}(j)$ during stage s. We use a marker $m^s_{\langle e,i\rangle}$ to denote the value of j. As a function of s, $m^s_{\langle e,i\rangle}$ is nondecreasing.

Construction

Stage 0. For all $\langle e, i, j\rangle \in \mathbb{N} \times \mathbb{N} \times \mathbb{N}$ let $L^0_{\langle e,i\rangle}(j)$ be a graph that has isolated vertices $L_{\langle e,i\rangle}(j + \langle e, i\rangle)$; and let the markers be defined by $m^0_{\langle e,i\rangle} = \langle e, i\rangle$. Let

$$G^0 = \bigcup_{e,i,j=0}^{\infty} L^0_{\langle e,i\rangle}(j).$$

Stage $s+1$. For each $\langle e, i\rangle \leqslant s$ such that

(a) $R_{\langle e,i\rangle}$ is not satisfied, and

(b) for all vertices z in $L^s_{\langle e,i\rangle}(m^s_{\langle e,i\rangle})$ the computation $\{e\}_s(z)$ halts,

take whatever action is necessary to help satisfy $R_{\langle e,i\rangle}$ using $L = L^s_{\langle e,i\rangle}(m^s_{\langle e,i\rangle})$. In particular, in terms of Bean's Claim, if L is $\hat{L}_k$ then add vertices and edges to L to form $\hat{L}_{k+1}$. Formally let $L^{s+1}_{\langle e,i\rangle}(m^s_{\langle e,i\rangle})$ be $\hat{L}_{k+1}$. All extra vertices added are the last unused vertices in X_{-1}.

For each $\langle e, i\rangle < s$ adjust the markers as follows: $m^{s+1}_{\langle e,i\rangle}$ is the maximum element in the set

$$\{y \mid \{m^s_{\langle e,i\rangle}, m^s_{\langle e,i\rangle}+1, m^s_{\langle e,i\rangle}+2, \ldots, y\} \subseteq W_{x,s+1}\} \cup \{m^s_{\langle e,i\rangle}\}.$$

For all $\langle e, i\rangle$ and j such that no action is taken on $L^s_{\langle e,i\rangle}(j)$ let $L^{s+1}_{\langle e,i\rangle}(j) = L^s_{\langle e,i\rangle}(j)$. Let

$$G^{s+1} = \bigcup_{e,i,j=0}^{\infty} L^{s+1}_{\langle e,i\rangle}(j). \qquad \textbf{End of Construction}$$

We show that W_x is cofinite iff G has a finite recursive chromatic number.

Assume W_x is not cofinite. for each $\langle e, i\rangle$ we claim that

$$\lim_{s\to\infty} m^s_{\langle e,i\rangle} < \infty.$$

If $\langle e, i\rangle \notin W_x$, then the marker never moves so $\lim_{s\to\infty} m^s_{\langle e,i\rangle} = \langle e, i\rangle$. If $\langle e, i\rangle \in W_x$, then let b be the largest element such that $\{\langle e, i\rangle, \langle e, i\rangle + 1, \ldots, b\} \subseteq W_x$ (note that $b+1$ is not in W_x). Such a b exists since W_x is not cofinite. By the nature of how the markers move $\lim_{s\to\infty} m^s_{\langle e,i\rangle} = b$.

Since $\lim_{s\to\infty} m^s_{\langle e,i\rangle} < \infty$, for s large all attempts to satisfy $R_{\langle e,i\rangle}$ use the same graph. By Bean's Claim these efforts succeed, hence all requirements are satisfied, and G is not recursively colorable.

Assume W_x is cofinite. Then for almost all $\langle e, i\rangle$

$$\lim_{s\to\infty} m^s_{\langle e,i\rangle} = \infty.$$

This fact can be used to recursively 2-color G. Let $S = \{\langle e, i\rangle \mid \lim_{s\to\infty} m^s_{\langle e,i\rangle} < \infty\}$. S is a finite set. If $\langle e, i\rangle \in S$, then only a finite number of vertices and edges are ever added to any $L_{\langle e,i\rangle}(j)$. This finite information is hardwired into the following algorithm.

Algorithm to 2-color G

(a) Input(z).

(b) Run the construction of G until z appears as a vertex. Let e, i, j and s_0 be such that $z \in L^{s_0}_{\langle e,i\rangle}(j)$ and s_0 is the least such number.

(c) If $\langle e, i\rangle \in S$, then the graph $L = \lim_{s\to\infty} L^s_{\langle e,i\rangle}(j)$ is hardwired. Let c be the least (in some ordering) 2-coloring of L. Output $c(z)$.

(d) If $\langle e, i\rangle \notin S$, then $\lim_{s\to\infty} m^s_{\langle e,i\rangle} = \infty$. Run the construction to the least stage $t \geq s_0$ such that $m^t_{\langle e,i\rangle} > j$. Note that $L^t_{\langle e,i\rangle}(j) = \lim_{s\to\infty} L^s_{\langle e,i\rangle}(j)$. Let c be the least (in some ordering) 2-coloring of $L^t_{\langle e,i\rangle}(j)$. Output $c(z)$. **End of Algorithm** □

Note. Many sets that arise in recursive graph theory can be shown to be Σ_3-complete using the techniques of the above theorem. In particular, determining the existence of a recursive matching in either a recursive or highly recursive bipartitie graph is Σ_3-complete [18].

The following theorem can be proved using the techniques of the last theorem, hence we only sketch the proof. It is stated for recursive graphs, and does not hold for highly recursive graphs if $k \geq 4$. It will be of use in Section 7.

Lemma 10. *For every $k \geq 3$ there exists a recursive function f_k such that for all x, $\chi(G^{\mathrm{r}}_{f_k(x)}) = 2$ and*

$$x \in COF \Rightarrow \chi^{\mathrm{r}}(G^{\mathrm{r}}_{f_k(x)}) = 2,$$
$$x \notin COF \Rightarrow \chi^{\mathrm{r}}(G^{\mathrm{r}}_{f_k(x)}) = k.$$

Proof. Note that in the construction in the proof of Theorem 9, all the graphs G constructed had $\chi(G) = 2$.

Given x, take the construction in the proof of Theorem 9 but modify it to try to satisfy only the requirements that make the graph not recursively $(k-1)$-colorable. Call the resulting graph G. If $x \in COF$, then $\chi^{\mathrm{r}}(G) = 2$ (i.e. the construction will fail); if $x \notin COF$, then $\chi^{\mathrm{r}}(G) \geq k-1$ (i.e. the construction will succeed). In the second case we need to show that $\chi^{\mathrm{r}}(G) = k$.

The graph G is the disjoint union of graphs L that are produced in the manner of the last theorem. Each L is the last element of a sequence of graphs

$$L_0 \subseteq L_1 \subseteq \cdots \subseteq L_r$$

where L_0 is the graph with 2^i isolated vertices. Since we only try to satisfy the requirements that make the graph not recursively $(k-1)$-colorable, when a vertex becomes part of the graph $k-1$ of its neighbors are known, and at most 1 more will eventually be discovered.

The following algorithm recursively k-colors G.

Algorithm

(a) Input(x).

(b) Run the construction until x appears in the graph. If it appears in some L_0, then color it 1, and halt.

(c) If x appears in $L_{j+1} - L_j$, then (recursively) color the L_j graph with the colors $\{1, 2, \ldots, k\}$. Now color x with a color that was not used by any of its neighbors in L_j. This is possible since x has at most $k-1$ neighbors in L_j. **End of Algorithm** □

Note. The graphs constructed above are not connected. If we insist they be connected, we get a slightly weaker result, namely that for every $k \geq 4$ there exists a function f_k such that

$$x \in COF \Rightarrow \chi^{\mathrm{r}}(G^{\mathrm{r}}_{f_k(x)}) = 3,$$
$$x \notin COF \Rightarrow \chi^{\mathrm{r}}(G^{\mathrm{r}}_{f_k(x)}) = k.$$

Theorem 11. *The partial recursive function*

$$B(e) = \begin{cases} 1 & \text{if } G^{\mathrm{hr}}_e \text{ exists and } \chi^{\mathrm{r}}(G^{\mathrm{hr}}_e) < \infty, \\ 0 & \text{if } G^{\mathrm{hr}}_e \text{ exists and } \chi^{\mathrm{r}}(G^{\mathrm{hr}}_e) = \infty, \\ \text{undefined} & \text{if } G^{\mathrm{hr}}_e \text{ does not exist} \end{cases}$$

is Σ_2-complete.

Proof. Since $\chi(G^{\mathrm{hr}}_e) \leq \chi^{\mathrm{r}}(G^{\mathrm{hr}}_e) \leq 2\,\chi(G^{\mathrm{hr}}_e) - 1$ (see [16] or [32]),

$$\chi^{\mathrm{r}}(G^{\mathrm{hr}}_e) < \infty \Leftrightarrow \chi(G^{\mathrm{hr}}_e) < \infty.$$

Hence

$$B(e) = \begin{cases} 1 & \text{if } G_e^{\text{hr}} \text{ exists and } \chi(G_e^{\text{hr}}) < \infty, \\ 0 & \text{if } G_e^{\text{hr}} \text{ exists and } \chi(G_e^{\text{hr}}) = \infty, \\ \text{undefined} & \text{otherwise} \end{cases}$$

which is Σ_2-complete by Theorem 5. □

7. Mixed queries

We have seen that $\lceil \log(c+1) \rceil$ queries to K ($\emptyset'''$) are *required* to compute $\chi(G_e^{\text{r}})$ ($\chi^{\text{r}}(G_e^{\text{r}})$) when this quantity is bounded by c. If we allow queries to a set Y such that $K \not\leq_{\text{T}} Y$ ($\emptyset''' \not\leq_{\text{T}} Y$) 'for free', then perhaps the number of queries to K ($\emptyset'''$) can be reduced. In this section we will see that for finding $\chi(G_e^{\text{r}})$ or $\chi(G_e^{\text{hr}})$ queries to such a Y do not help; however for finding $\chi^{\text{r}}(G_e^{\text{r}})$ or $\chi^{\text{r}}(G_e^{\text{hr}})$ they do. We also exhibit lower bounds on how much help queries to Y can provide.

Lemma 1 relativizes to yield the following.

Lemma 12. *If A, X and Y are sets, A is nonrecursive, $A \not\leq_{\text{T}} Y$, and n is any number, then*

$$F_{2^n}^A \notin \text{FQ}^Y(n, X).$$

Theorem 4 realtivizes, with the help of Lemma 12, to yield the following.

Theorem 13. *Let Y be any set such that $K \not\leq_{\text{T}} Y$. The function g in Theorem 4 is not in*

$$\text{FQ}^Y(\lceil \log(c+1) \rceil - 1, K).$$

Theorem 14. *Let $c \geq 2$ be any number. Let h be the function*

$$h(e) = \begin{cases} \chi^{\text{r}}(G_e^{\text{r}}) & \text{if } \chi^{\text{r}}(G_e^{\text{r}}) \leq c, \\ c & \text{if } \chi^{\text{r}}(G_e^{\text{r}}) \geq c. \end{cases}$$

The function h is in $\text{FQ}^K(\lceil \log(c+1) \rceil, \emptyset''')$.

Proof. Note that if $\chi(G_e^{\text{r}}) \in \{0, 1\}$, then $\chi(G_e^{\text{r}}) = \chi^{\text{r}}(G_e^{\text{r}})$. Given e, determine (recursively in K) whether $\chi(G_e^{\text{r}}) \in \{0, 1\}$; if it is then find its value (recursively in K) and output it. If not, then a binary search on $[2, c]$, using $\lceil \log(c+1) \rceil$ queries to $\emptyset'''$, will locate $h(e)$. □

Note. If the graphs being considered are connected, then $\chi(G_e^{\text{r}}) \in \{1, 2\} \Rightarrow \chi(G_e^{\text{r}}) = \chi^{\text{r}}(G_e^{\text{r}})$. This can be used to obtain an $\text{FQ}^K(\lceil \log(c-2) \rceil, \emptyset''')$ algorithm for h.

We show that Theorem 14 is optimal in that if $c \geq 3$, X is any set, and Y is such that $\emptyset''' \not\leq_{\text{T}} Y$, then h is not in $\text{FQ}^Y(\lceil \log(c-1) \rceil - 1, X)$.

Definition. Let A be any set and n be any number. The function $\#_n^A$ is defined by

$$\#_n^A(x_1, \ldots, x_n) = |\{i : x_i \in A\}|.$$

Note. Owings [28] has studied the function $\#_n^A$ and has shown that if there exists an X such that $\#_{2^n}^A \in \text{FQ}(n, X)$ then $A \leq_{\text{T}} K$.

Lemma 15. *Let X and Y be any sets. Let n and i be any numbers. If $\#_{2^n}^{\emptyset^{(i)}} \in \text{FQ}^Y(n, X)$, then $\emptyset^{(i)} \leq_{\text{T}} Y$.*

Proof. Assume $\#_{2^n}^{\emptyset^{(i)}} \in \text{FQ}^Y(n, X)$. Since for all $j \leq i$, $\emptyset^{(j)} \leq_{\text{m}} \emptyset^{(i)}$, we have

$$(\forall j \leq i)[\#_{2^n}^{\emptyset^{(j)}} \in \text{FQ}^Y(n, X)].$$

By a relativized version of Lemma 2 we have that

$$(\forall j)[F_{2^n}^{\emptyset^{(j)}} \in \text{FQ}^{\emptyset^{(j-1)}}(1, \#_{2^n}^{\emptyset^{(j)}})].$$

We show (inductively) that for all $j \leq i$, $\emptyset^{(j)} \leq_{\text{T}} Y$. For $j = 0$ this is trivial. Assume that $\emptyset^{(j-1)} \leq_{\text{T}} Y$. Hence

$$F_{2^n}^{\emptyset^{(j)}} \in \text{FQ}^{\emptyset^{(j-1)}}(1, \#_{2^n}^{\emptyset^{(j)}}) \subseteq \text{FQ}^Y(1, \#_{2^n}^{\emptyset^{(j)}}) \subseteq \text{FQ}^Y(n, X).$$

By Lemma 12, $\emptyset^{(j)} \leq_{\text{T}} Y$. Therefore we have, in the $j = i$ case $\emptyset^{(i)} \leq_{\text{T}} Y$. □

The second part of the following lemma is false for highly recursive graphs.

Lemma 16. *Let $b \geq 1$. Let h_1 be the function*

$$h_1(e) = \begin{cases} 2 & \text{if } \chi^{\text{r}}(G_e^{\text{r}}) \leq 2 \text{ and } \chi(G_e^{\text{r}}) = 2, \\ \chi^{\text{r}}(G_e^{\text{r}}) & \text{if } 2 \leq \chi^{\text{r}}(G_e^{\text{r}}) \leq b+2 \text{ and } \chi(G_e^{\text{r}}) = 2, \\ b+2 & \text{if } \chi^{\text{r}}(G_e^{\text{r}}) \geq b+2 \text{ and } \chi(G_e^{\text{r}}) = 2. \end{cases}$$

The function h_1 is in $\text{FQ}(\lceil \log(b+1) \rceil, \emptyset''')$. Let Y be any set such that $\emptyset''' \not\leq_{\text{T}} Y$. Let X be any set. Then

$$h_1 \notin \text{FQ}^Y(\lceil \log(b+1) \rceil - 1, X).$$

Proof. The function h_1 is in $\text{FQ}(\lceil \log(b+1) \rceil, \emptyset''')$ by a binary search algorithm.

We show $h_1 \notin \text{FQ}^Y(\lceil \log(b+1) \rceil - 1, X)$. Assume, by way of contradiction, that $h_1 \in \text{FQ}^Y(\lceil \log(b+1) \rceil - 1, X)$. We show that $\#_b^{\emptyset'''} \in \text{FQ}(1, h_1)$ and hence that $\#_b^{\emptyset'''} \in \text{FQ}^Y(\lceil \log(b+1) \rceil - 1, X)$. By Lemma 15 this implies that $\emptyset''' \leq_{\text{T}} Y$, contrary to the hypothesis.

To simplify notation, in the following algorithm if x_i is a number then 'run x_i' or 'x_i halts' refers to the computation of $\{x_i\}^{\emptyset''}(x_i)$.

Algorithm to compute $\#_b^{\emptyset''}$ in $FQ(1, h_1)$

(1) Input$(x_1, \ldots, x_b)$. Each x_i is an oracle Turing machine with oracle $\emptyset''$.

(2) Create $\emptyset''$-oracle Turing machines y_i, for $1 \le i \le b$, such that y_i halts iff at least $b-i+1$ of $x_1, \ldots, x_b$ halt. (Note that if $j = |\emptyset''' \cap \{x_1, \ldots, x_b\}|$ then $y_1, \ldots, y_{b-j} \notin \emptyset'''$ and $y_{b-j+1}, \ldots, y_b \in \emptyset'''$. In all future comments j will always be $|\emptyset''' \cap \{x_1, \ldots, x_b\}|$.)

(3) Using the fact that COF is Σ_3-complete, compute z_i, for $1 \le i \le b$ such that

$$z_i \in COF \quad \text{iff} \quad y_i \in \emptyset'''.$$

(Note that $z_1, \ldots, z_{b-j} \notin COF$ and $z_{b-j+1}, \ldots, z_b \in COF$.)

(4) Let $f_3, f_4, \ldots, f_{b+2}$ be the functions defined in Lemma 10. For $1 \le i \le b$, let $e_i = f_{i+2}(z_i)$. By the nature of the f_i, $\chi(G^{\mathrm{r}}_{e_i}) = 2$ and

$$z_i \notin COF \;\Rightarrow\; \chi^{\mathrm{r}}(G^{\mathrm{r}}_{e_i}) = i+2,$$
$$z_i \in COF \;\Rightarrow\; \chi^{\mathrm{r}}(G^{\mathrm{r}}_{e_i}) = 2.$$

(Note that $\chi^{\mathrm{r}}(G^{\mathrm{r}}_{e_i}) = i+2$ for $1 \le i \le b-j$, and $\chi^{\mathrm{r}}(G^{\mathrm{r}}_{e_i}) = 2$ for $b-j+1 \le i \le b$.)

(5) Let e be the index for the recursive graph formed by taking the disjoint union of the graphs $G^{\mathrm{r}}_{e_i}$ for $1 \le i \le b$. (Note that $\chi^{\mathrm{r}}(G^{\mathrm{r}}_e)$ is the maximum of $\chi^{\mathrm{r}}(G^{\mathrm{r}}_{e_i})$ as $1 \le i \le b$, which is $\chi^{\mathrm{r}}(G^{\mathrm{r}}_{e_i}) = b-j+2$. Also note that $2 \le \chi^{\mathrm{r}}(G^{\mathrm{r}}_e) \le b+2$ and $\chi(G^{\mathrm{r}}_e) = 2$, so $h_1(e) = \chi^{\mathrm{r}}(G^{\mathrm{r}}_e)$.)

(6) Compute the quantity $j = b + 2 - h_1(e)$. By the commentary throughout this construction $j = \#_b^{\emptyset'''}(x_1, \ldots, x_b)$. Output this value.

End of Algorithm □

Note. The condition that $\chi(G^{\mathrm{r}}_e) = 2$ for e in the domain of h_1 is not used in this paper, but is used in [11].

Theorem 17. *Let Y be any set such that $\emptyset''' \not\le_T Y$. Let X be any set, and $c \ge 3$. The function h (from Theorem 14) is not in* $FQ^Y(\lceil \log(c-1) \rceil - 1, X)$.

Proof. Assume $h \in FQ^Y(\lceil \log(c-1) \rceil - 1, X)$. Let h_1 be the function in Lemma 16 with $b = c-2$. Since $h_1 \in FQ(1, h)$, we obtain

$$h_1 \in FQ^Y(\lceil \log(c-1) \rceil - 1, X) = FQ^Y(\lceil \log(b+1) \rceil - 1, X).$$

This contradicts Lemma 16. □

Note. If h is restricted to operate on connected recursive graphs, then $h \in FQ^K(\lceil \log(c-2) \rceil, \emptyset''')$ (by the note after Theorem 14). Let h_1' be just like h_1 except that its upper and lower bounds are 3 and $b+3$, and it must operate on connected graphs. By using the note following Lemma 10 to modify the proof of Lemma 16, one can obtain that for all $b \ge 1$, all Y such that $Y \not\le_T \emptyset'''$, and all X, $h_1' \notin FQ^Y(\lceil \log(b+1) \rceil - 1, X)$. This can be used to show that h restricted to connected graphs is not in $FQ^K(\lceil \log(c-2) \rceil - 1, \emptyset''')$. Thus we have matching

upper and lower bounds for the case when h is restricted to connected recursive graphs.

For highly recursive graphs we can obtain a greater saving of queries to $\emptyset'''$. This is because if G is highly recursive, then [16, 32]

$$\chi(G) \le \chi^{\mathrm{r}}(G) \le 2\chi(G) - 1.$$

We use this to obtain an algorithm that asks one less $\emptyset'''$ query than the algorithm in Theorem 8; however this algorithm will ask many K queries.

The statement of the following theorem is false for recursive graphs.

Theorem 18. *Let $c \ge 1$ be any number. Let h be the function*

$$h(e) = \begin{cases} \chi^{\mathrm{r}}(G^{\mathrm{hr}}_e) & \text{if } \chi^{\mathrm{r}}(G^{\mathrm{hr}}_e) \le c, \\ c & \text{if } \chi^{\mathrm{r}}(G^{\mathrm{hr}}_e) \ge c. \end{cases}$$

If c is odd, then h is in $FQ^K(\lceil \log(c+1) \rceil - 1, \emptyset''')$. *If c is even, then h is in* $FQ^K(\lceil \log c \rceil - 1, \emptyset''')$.

Proof. Given e, first determine $\chi(G^{\mathrm{hr}}_e)$ by the binary search algorithm in Theorem 4. This only requires queries to K. We now use the fact that

$$\chi(G^{\mathrm{hr}}_e) \le \chi^{\mathrm{r}}(G^{\mathrm{hr}}_e) \le 2\,\chi(G^{\mathrm{hr}}_e) - 1.$$

Since we only care about $\chi^{\mathrm{r}}(G^{\mathrm{hr}}_e)$ if it is $\le c$, we do a binary search for $\chi^{\mathrm{r}}(G^{\mathrm{hr}}_e)$ on the interval $[\chi(G^{\mathrm{hr}}_e), \min\{2\,\chi(G^{\mathrm{hr}}_e) - 1, c\}]$ using queries to $\emptyset'''$. The length of this interval is

$$\begin{cases} \chi(G^{\mathrm{hr}}_e) & \text{if } 2\,\chi(G^{\mathrm{hr}}_e) - 1 \le c, \\ c - \chi(G^{\mathrm{hr}}_e) + 1 & \text{if } c \le 2\,\chi(G^{\mathrm{hr}}_e) - 1. \end{cases}$$

It can be shown that if c is odd, then the length of the interval is at most $(c+1)/2$; and if c is even, then the length of the interval is at most $c/2$. Hence the binary search on this interval takes at most $\lceil \log(c+1) \rceil - 1$ queries to $\emptyset'''$ when c is even; and at most $(\lceil \log c \rceil - 1)$ queries to $\emptyset'''$ when c is odd. □

We show that if Y is any set such that $\emptyset''' \not\le_T Y$, X is any set, and $c \ge 2$, then if c is odd, h is not in $FQ^Y(\lceil \log(c+1) \rceil - 2, X)$; and if c is even, h is not in $FQ^Y(\lceil \log c \rceil - 2, X)$. This is easily seen to be true for $c = 2, 3$.

Lemma 19. *Let $b \ge 1$. Let h_2 be the function*

$$h_2(e) = \begin{cases} b & \text{if } \chi^{\mathrm{r}}(G^{\mathrm{hr}}_e) \le b \text{ and } \chi(G^{\mathrm{hr}}_e) = b, \\ \chi^{\mathrm{r}}(G^{\mathrm{hr}}_e) & \text{if } b \le \chi^{\mathrm{r}}(G^{\mathrm{hr}}_e) \le 2b-1 \text{ and } \chi(G^{\mathrm{hr}}_e) = b, \\ 2b-1 & \text{if } \chi^{\mathrm{r}}(G^{\mathrm{hr}}_e) \ge 2b-1 \text{ and } \chi(G^{\mathrm{hr}}_e) = b. \end{cases}$$

Let h_3 be the function

$$h_3(e) = \begin{cases} b & \text{if } \chi^{r}(G_e^{hr}) \leqslant b \text{ and } \chi(G_e^{hr}) = b, \\ \chi^{r}(G_e^{hr}) & \text{if } b \leqslant \chi^{r}(G_e^{hr}) \leqslant 2b-2 \text{ and } \chi(G_e^{hr}) = b, \\ 2b-2 & \text{if } \chi^{r}(G_e^{hr}) \geqslant 2b-2 \text{ and } \chi(G_e^{hr}) = b. \end{cases}$$

The function h_2 (h_3) is in $\mathrm{FQ}(\lceil \log b \rceil, \emptyset''')$ ($\mathrm{FQ}(\lceil \log(b-1) \rceil, \emptyset''')$). Let Y be any set such that $\emptyset''' \not\leqslant_T Y$. Let X be any set. The function h_2 (h_3) is not in $\mathrm{FQ}^Y(\lceil \log b \rceil - 1, X)$ ($\mathrm{FQ}^Y(\lceil \log(b-1) \rceil - 1, X)$).

Proof. The upper bound for both h_2 and h_3 are obtained by binary search. The lower bound for h_2 (h_3) is obtained by showing that $\#_{b-1}^{\emptyset'''} \in \mathrm{FQ}(1, h_2)$ ($\#_{b-2}^{\emptyset'''} \in \mathrm{FQ}(1, h_3)$) in a manner similar to the proof of Lemma 16, except that we use the functions $f_{b,b+1}, f_{b,b+2}, \ldots, f_{b,2b-1}$ ($f_{b,b+1}, f_{b,b+2}, \ldots, f_{b,2b-2}$). Lemma 15 is then used to derive a contradiction. □

Note. The condition that $\chi(G_e^r) = b$ for e in the domain of h_2 is not used in this paper, but is used in [11].

Theorem 20. *Let Y be any set such that $\emptyset''' \not\leqslant_T Y$, X be any set, and $c \geqslant 4$. If c is odd, then h (from Theorem 18) is not in $\mathrm{FQ}^Y(\lceil \log(c+1) \rceil - 2, X)$; if c is even, then h is not in $\mathrm{FQ}^Y(\lceil \log c \rceil - 2, X)$.*

Proof. Assume, by way of contradiction, that Y, b and an algorithm exists as specified above.

If c is odd, $c = 2b-1$, then using the $\mathrm{FQ}^Y(\lceil \log(c+1) \rceil - 2, X)$ algorithm for h, the function h_2 (with parameter b) can be computed in

$$\mathrm{FQ}^Y(\lceil \log(c+1) \rceil - 2, X) = \mathrm{FQ}^Y(\lceil \log 2b \rceil - 2, X) = \mathrm{FQ}^Y(\lceil \log b \rceil - 1, X).$$

This contradicts Lemma 19.

If c is even, $c = 2b-2$, then using the $\mathrm{FQ}^Y(\lceil \log c \rceil - 2, X)$ algorithm for h, the function h_2 (with parameter b) can be computed in

$$\mathrm{FQ}^Y(\lceil \log c \rceil - 2, X) = \mathrm{FQ}^Y(\lceil \log 2b-2 \rceil - 2, X) = \mathrm{FQ}^Y(\lceil \log(b-1) \rceil - 1, X).$$

This contradicts Lemma 19. □

8. Parallel queries

In this section we look at machines that can ask p (a fixed constant) queries to a set simultaneously. This notion is formalized by considering queries to the function F_p^X (where X is some oracle). Binary search can be replaced by $(p+1)$-ary search [25, 33] in many of our theorems, but this is not always optimal. The constant p is fixed throughout this section.

The following lemma will be useful in establishing a lower bound on the number of queries to F_p^K that are needed to find the chromatic number of a recursive graph.

Lemma 21. *If A is a nonrecursive set and X is an r.e. set, then*

$$F^A_{(p+1)^n} \notin \mathrm{FQ}(n, F_p^X).$$

Proof. Since X is r.e., $\mathrm{FQ}(n, F_p^X) \subseteq \mathrm{FQ}(n, F_p^K)$. Beigel [7] has shown that

$$\mathrm{FQ}(n, F_p^K) \subseteq \mathrm{FQ}(1, F^K_{(p+1)^n-1}).$$

If $F^A_{(p+1)^n} \in \mathrm{FQ}(n, F_p^X)$ then

$$F^A_{(p+1)^n} \in \mathrm{FQ}(n, F_p^X) \subseteq \mathrm{FQ}(n, F_p^K) \subseteq \mathrm{FQ}(1, F^K_{(p+1)^n-1}).$$

This violates the separation theorem of Beigel (in [7] and [6]) which states that if A is a nonrecursive set, B is an arbitrary set, and $a \geqslant 1$, then $F_a^A \notin \mathrm{FQ}(1, F^B_{a-1})$ □

We now look at finding the chromatic number of a graph in terms of queries to F_p^K.

Lemma 22. *Let g be the function*

$$g(e, \langle k_1, \ldots, k_p \rangle) = \begin{cases} 0 & \text{if } \chi(G_e^r) < k_1, \\ i & \text{if } k_i \leqslant \chi(G_e^r) < k_{i+1} \ (1 \leqslant i < p), \\ p & \text{if } k_p \leqslant \chi(G_e^r). \end{cases}$$

Then g is complete for $\mathrm{FQ}(1, F_p^K)$, that is, $g \in \mathrm{FQ}(1, F_p^K)$ and $F_p^K \in \mathrm{FQ}(1, g)$. Hence every function in $\mathrm{FQ}(1, F_p^K)$ is in $\mathrm{FQ}(1, g)$.

Proof. We show that $g \in \mathrm{FQ}(1, F_p^K)$. On input $(e, \langle k_1, \ldots, k_p \rangle)$, to compute g just pose the p questions "$\chi(G_e^r) \leqslant k_i$?" ($1 \leqslant i \leqslant p$). All these questions can be phrased as questions to K (by Lemma 3), so asking them can be phrased as one query to F_p^K. From the answers we can obtain $g(e, \langle k_1, \ldots, k_p \rangle)$.

We show $F_p^K \in \mathrm{FQ}(1, g)$. On input $\langle z_1, \ldots, z_p \rangle$, create e (using methods similar to Theorem 4) such that $\chi(G_e^r) = |K \cap \{z_1, \ldots, z_p\}|$. Compute $g(e, \langle 1, 2, 3, \ldots, p \rangle)$, and from this compute $\chi(G_e^r)$. By Lemma 2, from $\chi(G_e^r) = |K \cap \{z_1, \ldots, z_p\}|$ we can compute $F_p^K(z_1, \ldots, z_p)$. □

Theorem 23. *Let $c \geqslant 1$ be any number. Let g be the function*

$$g(e) = \begin{cases} \chi(G_e^r) & \text{if } \chi(G_e^r) \leqslant c, \\ c & \text{if } \chi(G_e^r) \geqslant c. \end{cases}$$

Then

$$g \in \mathrm{FQ}\left(\left\lceil \frac{\log(c+1)}{\log(p+1)} \right\rceil, F_p^K\right).$$

If $X = \emptyset'$, or any other recursively enumerable set, then

$$g \notin \mathrm{FQ}\left(\left\lceil \frac{\log(c+1)}{\log(p+1)} \right\rceil - 1, F_p^X\right).$$

Proof. Using Lemma 22 and a $(p+1)$-ary search on $[0, c]$ for the proper number of colors, one obtains that

$$g \in \mathrm{FQ}\left(\left\lceil \frac{\log(c+1)}{\log(p+1)} \right\rceil, F_p^K\right).$$

First ask 'evenly spaced questions' to get the graphs's chromatic number in an interval of length $(c+1)/(p+1)$, then $(c+1)/(p+1)^2$, etc.

Let X be an r.e. set. To establish

$$g \notin \mathrm{FQ}\left(\left\lceil \frac{\log(c+1)}{\log(p+1)} \right\rceil - 1, F_p^X\right)$$

we show that

$$\text{if } g \in \mathrm{FQ}(n, F_p^X) \quad \text{then } F^K_{(p+1)^n} \in \mathrm{FQ}(n, F_p^X) \qquad \left(\text{where } n = \left\lceil \frac{\log(c+1)}{\log(p+1)} \right\rceil - 1\right)$$

which contradicts Lemma 21.

Assume $g \in \mathrm{FQ}(n, F_p^X)$. To compute $F^K_{(p+1)^n}(x_1, \ldots, x_{(p+1)^n})$ create (using the technique of Theorem 4) a recursive graph G_e^r whose chromatic number is $|K \cap \{x_1, \ldots, x_{(p+1)^n}\}|$. Compute $g(e)$. Note that

$$\chi(G_e^r) \leqslant (p+1)^n = (p+1)^{\lceil \log(c+1)/\log(p+1) \rceil - 1}.$$

the quantity $(p+1)^{\lceil \log(c+1)/\log(p+1) \rceil - 1}$ is $(p+1)^{\log(c+1)/\log(p+1)+\epsilon}$ for some $\epsilon < 1$. Hence

$$\begin{aligned}(p+1)^{\lceil \log(c+1)/\log(p+1) \rceil - 1} &= (p+1)^{\log(c+1)/\log(p+1)+\epsilon} \\ &= (c+1)/(p+1)^{1-\epsilon} < c+1.\end{aligned}$$

Since $\chi(G_e^r)$ is an integer we have $\chi(G_e^r) \leqslant c$. Hence

$$g(e) = \chi(G_e^r) = |K \cap \{x_1, \ldots, x_{(p+1)^n}\}|.$$

By Lemma 2 we can compute $F^K_{(p+1)^n}$ from this quantity. Since $g \in \mathrm{FQ}(n, F_p^X)$, we obtain

$$F^K_{(p+1)^n} \in \mathrm{FQ}\left(\left\lceil \frac{\log(c+1)}{\log(p+1)} \right\rceil - 1, F_p^X\right). \qquad \square$$

The above theorem shows that if F_p^K (or F_p^X for any r.e. X) is used as an oracle, then $(p+1)$-ary search is optimal. The question arises: "Are there sets A such that by using F_p^A the number of queries needed can be reduced?" The answer is "Yes".

Theorem 24. *Let g be as in the last theorem. There exists a set $A \equiv_{\mathrm{T}} K$ such that*

$$g \in \mathrm{FQ}\left(\left\lceil \frac{\log(c+1)}{p} \right\rceil, F_p^A\right).$$

For all sets X, the function

$$g \notin \mathrm{FQ}\left(\left\lceil \frac{\log(c+1)}{p} \right\rceil - 1, F_p^X\right).$$

Proof. Let $\mathscr{A}$ be the algorithm in Theorem 4 that computes g with $\lceil \log(c+1) \rceil$ queries to K. Note that it always halts, even if the input is not the index of a recursive graph. Let

$$A = \{\langle e, i \rangle \mid \text{when } \mathscr{A} \text{ is run on } e, \text{ the } i\text{th query to } K \text{ is answered "Yes"}\}.$$

Since $\mathscr{A}$ is recursive in K, $A \leqslant_{\mathrm{T}} K$. Since from A we can compute the chromatic number of a graph (if it is $\leqslant c$), by Lemma 3, $K \leqslant_{\mathrm{T}} A$. Hence $A \equiv_{\mathrm{T}} K$.

The value of $g(e)$ can be deduced from the $\lceil (\log(c+1))/p \rceil$ questions

$$\begin{aligned}
&F_p^A(\langle e, 1\rangle, \langle e, 2\rangle, \langle e, 3\rangle, \ldots, \langle e, p\rangle),\\
&F_p^A(\langle e, p+1\rangle, \langle e, p+2\rangle, \langle e, p+3\rangle, \ldots, \langle e, 2p\rangle),\\
&\quad\vdots\\
&F_p^A\left(\left\langle e, \left(\left\lceil \frac{\log(c+1)}{p} \right\rceil - 1\right)p + 1\right\rangle, \ldots, \left\langle e, \left(\left\lceil \frac{\log(c+1)}{p} \right\rceil\right)p\right\rangle\right).
\end{aligned}$$

The answers to these questions provide the correct query answers that are needed for running $\mathscr{A}$ on e. Once obtained, run $\mathscr{A}$ on e with the correct query answers, and $g(e)$ can be found.

If X is any set and if $g \in \mathrm{FQ}(\lceil (\log(c+1))/p \rceil - 1, F_p^X)$ then g is in $\mathrm{FQ}(\lceil \log(c+1) \rceil - 1, X)$, which contradicts Theorem 4. $\square$

We now look at finding recursive chromatic numbers.

Theorem 25. *Let h be the function*

$$h(e, \langle k_1, \ldots, k_p \rangle) = \begin{cases} 0 & \text{if } \chi^r(G_e^r) < k_1, \\ i & \text{if } k_i \leqslant \chi^r(G_e^r) < k_{i+1} \ (1 \leqslant i < p), \\ p & \text{if } k_p \leqslant \chi^r(G_e^r). \end{cases}$$

Then h is in $\mathrm{FQ}(1, F_p^{\emptyset''})$.

Proof. This result is obtained by combining the technique of Lemma 22 ($(p+1)$-ary search) with the result of Theorem 7 (determining if $\chi^r(G) \leqslant c$ can be done with a $\emptyset'''$ oracle). □

Theorem 26. *Let $c \geqslant 1$ be any number. Let h be the function*

$$h(e) = \begin{cases} \chi^r(G_e^r) & \text{if } \chi^r(G_e^r) \leqslant c, \\ c & \text{if } \chi^r(G_e^r) \geqslant c. \end{cases}$$

Then h is in

$$\mathrm{FQ}\left(\left\lceil \frac{\log(c+1)}{\log(p+1)} \right\rceil, F_p^{\emptyset'''}\right).$$

The function h as presented (applying to recursive graphs) is not in

$$\mathrm{FQ}\left(\left\lceil \frac{\log(c-1)}{\log(p+1)} \right\rceil - 1, F_p^{\emptyset'''}\right).$$

If c is odd then h, when modified to apply to highly recursive graphs, is not in

$$\mathrm{FQ}\left(\left\lceil \frac{\log(c+1)-1}{\log(p+1)} \right\rceil - 1, F_p^{\emptyset'''}\right);$$

if c is even, then h is not in

$$\mathrm{FQ}\left(\left\lceil \frac{\log(c)-1}{\log(p+1)} \right\rceil - 1, F_p^{\emptyset'''}\right).$$

Proof. The upper bound is obtained by $(p+1)$-ary search. The lower bounds are the $Y = \emptyset$ cases of Theorems 33 and 36. □

Improving the lower bounds in Theorem 26 remains an open question.
If an oracle other than $\emptyset'''$ is used, then the number of queries can be reduced.

Theorem 27. *There exists a set $A \equiv_T \emptyset'''$ such that*

$$h \in \mathrm{FQ}\left(\left\lceil \frac{\log(c+1)}{p} \right\rceil, F_p^A\right).$$

For all X,

$$h \notin \mathrm{FQ}\left(\left\lceil \frac{\log(c+1)}{p} \right\rceil - 1, F_p^X\right).$$

Proof. Let $A = \{\langle e, i\rangle \mid G_e^r$ exists, $\chi^r(G_e^r) \leqslant c$, and the ith bit of $\chi^r(G_e^r)$, expressed in binary, is $1\}$.

Since determining if G_e^r exists is recursive in $\emptyset''$ and determining if $\chi^r(G_e^r) \leqslant c$ is recursive in $\emptyset'''$, $A \leqslant_T \emptyset'''$. Since from A we can find $\chi^r(G_e^r)$, $\emptyset''' \leqslant_T A$. The rest of this proof is analogous to Theorem 24. □

9. Parallel and mixed queries

In this section we explore the questions raised in Section 7 in a parallel setting. Most of the proofs use a combination of techniques from the last two sections and hence will be omitted.

Lemma 28. *If A, X, and Y are sets, A is nonrecursive, X is r.e., and $A \not\leqslant_T Y$, then*

$$F_{(p+1)^n}^A \notin \mathrm{FQ}^Y(n, F_p^X).$$

Proof. Relativize the proof of Lemma 21. □

Theorem 29. *Let Y be any set such that $K \not\leqslant_T Y$. The function g in Theorem 4 (and 23) is not in*

$$\mathrm{FQ}^Y\left(\left\lceil \frac{\log(c+1)}{\log(p+1)} \right\rceil - 1, K\right).$$

Proof. The proof of Theorem 23 relativizes, with the help of Lemma 28. □

Theorem 30. *Let h be the function in Theorem 14 (and 26). The function h is in*

$$\mathrm{FQ}^K\left(\left\lceil \frac{\log(c-1)}{\log(p+1)} \right\rceil, F_p^{\emptyset'''}\right).$$

Proof. Combine the techniques of Theorems 14 and 23. □

To prove analogs of Lemmas 16 and 19, and Theorems 17 and 20, we use the following lemma.

Lemma 31. *Let $b, p \geqslant 1$. Let Y be any set. Then*

$$\mathrm{FQ}^Y\left(\left\lceil \frac{\log(b+1)}{\log(p+1)} \right\rceil - 1, F_p^{\emptyset'''}\right) \subseteq \mathrm{FQ}^Y(1, F_{b-1}^{\emptyset'''}).$$

Proof. Let

$$n = \left\lceil \frac{\log(b+1)}{\log(p+1)} \right\rceil - 1 = \lceil \log_{p+1}(b+1) \rceil - 1.$$

Beigel [7] has shown that for all n and p

$$\mathrm{FQ}(n, F_p^K) \subseteq \mathrm{FQ}(1, F_{(p+1)^n-1}^K).$$

This result relativizes (in two ways) to show that for any Y and A,

$$\mathrm{FQ}^Y(n, F_p^{A'}) \subseteq \mathrm{FQ}^Y(1, F_{(p+1)^n-1}^{A'}).$$

In particular,

$$\mathrm{FQ}^Y(n, F_p^{\emptyset'''}) \subseteq \mathrm{FQ}^Y(1, F_{(p+1)^n-1}^{\emptyset'''}).$$

Since

$$(p+1)^n - 1 = (p+1)^{\lceil \log_{p+1}(b+1)\rceil - 1} - 1 < b$$

it follows that

$$\mathrm{FQ}^Y(1, F_{(p+1)^n-1}^{\emptyset'''}) \subseteq \mathrm{FQ}^Y(1, F_{b-1}^{\emptyset'''}).$$

Combining the last two inclusions yields the desired result. $\square$

The statement of the following lemma is not known to be true for highly recursive graphs.

Lemma 32. *Let Y be any set such that $\emptyset''' \not\leq_T \emptyset'' \oplus Y$. Let h_1 be the function in Lemma 16. Then*

$$h_1 \in \mathrm{FQ}\left(\left\lceil \frac{\log(b+1)}{\log(p+1)} \right\rceil, F_p^{\emptyset'''}\right) \quad \text{but } h_1 \notin \mathrm{FQ}^Y\left(\left\lceil \frac{\log(b+1)}{\log(p+1)} \right\rceil - 1, F_p^{\emptyset'''}\right).$$

Proof. The function h_1 is in

$$\mathrm{FQ}\left(\left\lceil \frac{\log(b+1)}{\log(p+1)} \right\rceil, F_p^{\emptyset'''}\right)$$

by $(p+1)$-ary search. Assume

$$h_1 \in \mathrm{FQ}^Y\left(\left\lceil \frac{\log(b+1)}{\log(p+1)} \right\rceil - 1, F_p^{\emptyset'''}\right).$$

By Lemma 31,

$$h_1 \in \mathrm{FQ}^Y(1, F_{b-1}^{\emptyset'''}).$$

By the proof of Lemma 16,

$$F_b^{\emptyset'''} \in \mathrm{FQ}^{\emptyset''}(1, h_1).$$

Hence

$$F_b^{\emptyset'''} \in \mathrm{FQ}^{\emptyset'' \oplus Y}(1, F_{b-1}^{\emptyset'''}).$$

Since $\emptyset''' \not\leq_T \emptyset'' \oplus Y$, this violates the relativized version of the Separation Theorem proven in [6], which states that if $b \in \mathbb{N}$, and A and B are sets such that $A \not\leq_T B$, then $F_b^A \notin \mathrm{FQ}^B(1, F_{b-1}^A)$. $\square$

Note. The condition $\emptyset''' \not\leq_T \emptyset'' \oplus Y$ does not imply $\emptyset''' \not\leq_T Y$ since by a relativized form of the Friedberg Jump Theorem [29] there exist sets Y such that $\emptyset''' \not\leq_T \emptyset'' \oplus Y$ but $\emptyset''' \not\leq_T Y$.

Theorem 33. *Let h be the function in Theorem 14 (and 26). Let Y be any set such that $\emptyset''' \not\leq_T \emptyset'' \oplus Y$. If $c \geq 4$, then h is not in*

$$\mathrm{FQ}^Y\left(\left\lceil \frac{\log(c-1)}{\log(p-1)} \right\rceil - 1, F_p^{\emptyset'''}\right).$$

Theorem 34. *Let h be the function in Theorem 18. If c is odd, then h is in*

$$\mathrm{FQ}^K\left(\left\lceil \frac{\log(c+1)-1}{\log(p+1)} \right\rceil, F_p^{\emptyset'''}\right);$$

if c is even, then h is in

$$\mathrm{FQ}^K\left(\left\lceil \frac{\log(c)-1}{\log(p+1)} \right\rceil, F_p^{\emptyset'''}\right).$$

Lemma 35. *Let h_2 and h_3 be the functions in Lemma 19. Let Y be such that $\emptyset''' \not\leq_T \emptyset'' \oplus Y$. The function h_2 (h_3) is in*

$$\mathrm{FQ}\left(\left\lceil \frac{\log b}{\log(p+1)} \right\rceil, F_p^{\emptyset'''}\right) \quad \left(\mathrm{FQ}\left(\left\lceil \frac{\log(b-1)}{\log(p+1)} \right\rceil, F_p^{\emptyset'''}\right)\right)$$

but not in

$$\mathrm{FQ}^Y\left(\left\lceil \frac{\log b}{\log(p+1)} \right\rceil - 1, F_p^{\emptyset'''}\right) \quad \left(\mathrm{FQ}^Y\left(\left\lceil \frac{\log(b-1)}{\log(p+1)} \right\rceil - 1, F_p^{\emptyset'''}\right)\right).$$

Theorem 36. *Let h be the function in Theorem 18. Let Y be such that $\emptyset''' \not\leq_T \emptyset'' \oplus Y$. If c is odd, then h is not in*

$$\mathrm{FQ}^Y\left(\left\lceil \frac{\log(c+1)-1}{\log(p+1)} \right\rceil - 1, \emptyset'''\right);$$

is c is even, then h is not in

$$\mathrm{FQ}^Y\left(\left\lceil \frac{\log(c)-1}{\log(p+1)} \right\rceil, \emptyset'''\right).$$

If we do not insist that $\emptyset'''$ be the oracle we use, then we can reduce the number of queries substantially.

Theorem 37. *Let h be the function in Theorem 14 (and 26). There exists a set $A \equiv_T K$ such that*

$$h \in \mathrm{FQ}^K\left(\left\lceil \frac{\log(c-1)}{p} \right\rceil, F_p^A\right).$$

For all Y such that $\emptyset''' \not\leq_T \emptyset'' \oplus Y$, and for all sets X,

$$h \notin \mathrm{FQ}^Y\left(\left\lceil \frac{\log(c-1)}{p} \right\rceil - 1, F_p^X\right).$$

Theorem 38. *Let h be the function in Theorem 18. If c is odd, then there exists a set $A \equiv_T \emptyset'''$ such that*

$$h \in \mathrm{FQ}^K\left(\left\lceil \frac{\log(c+1)-1}{p} \right\rceil, F_p^A\right);$$

if c is even, then there exists a set $A \equiv_T \emptyset'''$ such that

$$h \in \mathrm{FQ}^K\left(\left\lceil \frac{\log(c)-1}{p} \right\rceil, F_p^A\right).$$

For all Y such that $\emptyset''' \not\leq_T Y$, and for all sets X, if c is even, then

$$h \notin \mathrm{FQ}^Y\left(\left\lceil \frac{\log(c+1)-1}{p} \right\rceil - 1, F_p^X\right);$$

if c is odd, then

$$h \notin \mathrm{FQ}^Y\left(\left\lceil \frac{\log(c)-1}{p} \right\rceil - 1, F_p^X\right).$$

Proof. The upper bound comes from combining the techniques of Theorem 18 and Theorem 27. The lower bound comes directly from Theorem 20. □

10. Summary and open problems

We summarize our results in the following table. Let $c, p \geq 1$ be fixed natural numbers. The function χ returns the chromatic number of a graph if it is $\leq c$. The function χ^r returns the recursive chromatic number of a graph if it is $\leq c$. Unless otherwise specified, a result holds for both recursive and highly recursive graphs. If X is used in a statement of a result then that result holds when X is replaced by any set. If Y is used in a statement about chromatic number, then the intention is that the statement holds for any Y such that $K \not\leq_T Y$. If Y is used in a statement about recursive chromatic number, then the intention is that the statement holds for any Y such that $\emptyset''' \not\leq_T Y$; unless it is a statement about parallel queries to $\emptyset'''$ in which case the intention is that the statement holds for all Y such that $\emptyset''' \not\leq_T \emptyset'' \oplus Y$. If A is used in a statement about chromatic number, then we are saying that a set A, $A \equiv_T K$, exists; if A is used in a statement about recursive chromatic number, then we are saying that a set A, $A \equiv_T \emptyset'''$, exists.

In some cases our lower bounds do not (numerically) match our upper bounds. These lower bounds are marked with **. We conjecture that the lower bounds can be improved to match the upper bound. In some cases we have the condition $\emptyset''' \not\leq_T Y \oplus \emptyset''$ instead of $\emptyset''' \not\leq_T Y$. These cases are marked with *. We conjecture that the lower bound with the condition $\emptyset''' \not\leq_T Y$ can be obtained. The following conjecture would imply that $\emptyset''' \not\leq_T Y$ would suffice: "If $\#_n^{\emptyset'''} \in \mathrm{FQ}^Y(1, F_{n-1}^{\emptyset'''})$, then $\emptyset''' \leq_T Y$."

I. Serial queries without help

$\chi \in \mathrm{FQ}(\lceil \log(c+1) \rceil, K)$

$\chi \notin \mathrm{FQ}(\lceil \log(c+1) \rceil - 1, X)$

$\chi^r \in \mathrm{FQ}(\lceil \log(c+1) \rceil, \emptyset''')$

$\chi^r \notin \mathrm{FQ}(\lceil \log(c+1) \rceil - 1, X)$

II. Serial queries with help

(a) *Recursive graphs*

$\chi \in \mathrm{FQ}(\lceil \log(c+1) \rceil, K)$

$\chi \notin \mathrm{FQ}^Y(\lceil \log(c+1) \rceil - 1, X)$

$\chi^r \in \mathrm{FQ}^K(\lceil \log(c-1) \rceil, \emptyset''')$

$\chi^r \notin \mathrm{FQ}^Y(\lceil \log(c-1) \rceil - 1, X)$

(b) *Highly recursive graphs*

$\chi \in \mathrm{FQ}(\lceil \log(c+1) \rceil, K)$

$\chi \notin \mathrm{FQ}^Y(\lceil \log(c+1) \rceil - 1, X)$

$\chi^r \in \mathrm{FQ}^K(\lceil \log(c+1) \rceil - 1, \emptyset''')$ c odd

$\chi^r \notin \mathrm{FQ}^Y(\lceil \log(c+1) \rceil - 2, X)$ c odd

$\chi^r \in \mathrm{FQ}^K(\lceil \log c \rceil - 1, \emptyset''')$ c even

$\chi^r \notin \mathrm{FQ}^Y(\lceil \log c \rceil - 2, X)$ c even

III. Parallel queries without help

(a) Using queries to F_p^K ($F_p^{\emptyset'''}$) to compute χ (χ^r)

$\chi \in \mathrm{FQ}\left(\left\lceil \frac{\log(c+1)}{\log(p+1)} \right\rceil, F_p^K\right)$

$\chi \notin \mathrm{FQ}\left(\left\lceil \frac{\log(c+1)}{\log(p+1)} \right\rceil - 1, F_p^K\right)$

$\chi^r \in \mathrm{FQ}\left(\left\lceil \frac{\log(c+1)}{\log(p+1)} \right\rceil, F_p^{\emptyset'''}\right)$

$\chi^r \notin \mathrm{FQ}\left(\left\lceil \frac{\log(c-1)}{\log(p+1)} \right\rceil - 1, F_p^{\emptyset'''}\right)$ ** for recursive graphs

$\chi^r \notin \mathrm{FQ}\left(\left\lceil \frac{\log(c+1)-1}{\log(p+1)} \right\rceil - 1, F_p^{\emptyset'''}\right)$ ** for highly recursive graphs and c odd

$\chi^r \notin \mathrm{FQ}\left(\left\lceil \frac{\log(c)-1}{\log(p+1)} \right\rceil - 1, F_p^{\emptyset'''}\right)$ ** for highly recursive graphs and c even

(b) Using queries to F_p^A, any A, to compute χ and χ^r

$$\chi \in \mathrm{FQ}\left(\left\lceil \frac{\log(c+1)}{p} \right\rceil, F_p^A\right)$$

$$\chi \notin \mathrm{FQ}\left(\left\lceil \frac{\log(c+1)}{p} \right\rceil - 1, F_p^X\right)$$

$$\chi^r \in \mathrm{FQ}\left(\left\lceil \frac{\log(c+1)}{p} \right\rceil, F_p^A\right)$$

$$\chi^r \notin \mathrm{FQ}\left(\left\lceil \frac{\log(c+1)}{p} \right\rceil - 1, F_p^X\right)$$

IV. Parallel queries with help

(a) Using queries to F_p^K ($F_p^{\emptyset'''}$) to compute χ (χ^r), but allowing unlimited queries to a set Y where $K \not\leq_T Y$ ($\emptyset''' \not\leq_T Y$ or $\emptyset''' \not\leq_T Y \oplus \emptyset''$ when noted)

(i) *Recursive graphs*

$$\chi \in \mathrm{FQ}\left(\left\lceil \frac{\log(c+1)}{\log(p+1)} \right\rceil, F_p^K\right)$$

$$\chi \notin \mathrm{FQ}^Y\left(\left\lceil \frac{\log(c+1)}{\log(p+1)} \right\rceil - 1, F_p^K\right)$$

$$\chi^r \in \mathrm{FQ}^K\left(\left\lceil \frac{\log(c-1)}{\log(p+1)} \right\rceil, F_p^{\emptyset'''}\right)$$

$$\chi^r \notin \mathrm{FQ}^Y\left(\left\lceil \frac{\log(c-1)}{\log(p+1)} \right\rceil - 1, F_p^{\emptyset'''}\right) * \quad (\emptyset''' \not\leq_T \emptyset'' \oplus Y)$$

(ii) *Highly recursive graphs*

$$\chi \in \mathrm{FQ}\left(\left\lceil \frac{\log(c+1)}{\log(p+1)} \right\rceil, F_p^K\right)$$

$$\chi \notin \mathrm{FQ}^Y\left(\left\lceil \frac{\log(c+1)}{\log(p+1)} \right\rceil - 1, F_p^K\right)$$

$$\chi^r \in \mathrm{FQ}^Y\left(\left\lceil \frac{\log(c+1)-1}{\log(p+1)} \right\rceil, F_p^{\emptyset'''}\right) \quad c \text{ odd}$$

$$\chi^r \notin \mathrm{FQ}^Y\left(\left\lceil \frac{\log(c+1)-1}{\log(p+1)} \right\rceil - 1, F_p^{\emptyset'''}\right) * \quad \emptyset''' \not\leq_T \emptyset'' \oplus Y \text{ and } c \text{ odd}$$

$$\chi^r \in \mathrm{FQ}^K\left(\left\lceil \frac{\log(c)-1}{\log(p+1)} \right\rceil, F_p^{\emptyset'''}\right) \quad c \text{ even}$$

$$\chi^r \notin \mathrm{FQ}^Y\left(\left\lceil \frac{\log(c)-1}{\log(p+1)} \right\rceil - 1, F_p^{\emptyset'''}\right) * \quad \emptyset''' \not\leq_T \emptyset'' \oplus Y \text{ and } c \text{ even}$$

(b) Using queries to F_p^A, any A, to compute χ and χ^r

(i) *Recursive graphs*

$$\chi \in \mathrm{FQ}\left(\left\lceil \frac{\log(c+1)}{p} \right\rceil, F_p^A\right)$$

$$\chi \notin \mathrm{FQ}^Y\left(\left\lceil \frac{\log(c+1)}{p} \right\rceil - 1, F_p^X\right)$$

$$\chi^r \in \mathrm{FQ}^K\left(\left\lceil \frac{\log(c-1)}{p} \right\rceil, F_p^A\right)$$

$$\chi^r \notin \mathrm{FQ}^Y\left(\left\lceil \frac{\log(c-1)}{p} \right\rceil - 1, F_p^X\right)$$

(ii) *Highly recursive graphs*

$$\chi \in \mathrm{FQ}\left(\left\lceil \frac{\log(c+1)}{p} \right\rceil, F_p^A\right)$$

$$\chi \notin \mathrm{FQ}^Y\left(\left\lceil \frac{\log(c+1)}{p} \right\rceil - 1, F_p^X\right)$$

$$\chi^r \in \mathrm{FQ}^K\left(\left\lceil \frac{\log(c+1)-1}{p} \right\rceil, F_p^A\right) \quad c \text{ odd}$$

$$\chi^r \notin \mathrm{FQ}^Y\left(\left\lceil \frac{\log(c+1)-1}{p} - 1 \right\rceil, F_p^X\right) \quad c \text{ odd}$$

$$\chi^r \in \mathrm{FQ}^K\left(\left\lceil \frac{\log(c)-1}{p} \right\rceil, F_p^A\right) \quad c \text{ even}$$

$$\chi^r \notin \mathrm{FQ}^Y\left(\left\lceil \frac{\log(c)-1}{p} \right\rceil - 1, F_p^X\right) \quad c \text{ even}$$

Appendix A

We show that for every $n \geq 3$ there is a highly recursive graph G such that $\chi(G) = n$ and $\chi^r(G) = 2n - 3$. Techniques used here are a variation on Schmerl's construction [32] of a graph G such that $\chi(G) = n$ and $\chi^r(G) = 2n - 2$. A less complicated version of our construction yields Schmerl's result. At the end of the proof we will indicate how to accomplish this.

Notation. If G and G' are graphs, then $G \cong G'$ means that G is isomorphic to G'.

Definition. Let $n \geq 3$. Let $G^n = (V, E)$ where

$$V = \{(i, j) \mid 1 \leq i, j \leq n\},$$
$$E = \{\{(i, j), (r, s)\} \mid i \neq r \text{ and } j \neq s\}.$$

If $1 \leq i \leq n$, then the set of vertices $\{(i, j) \mid 1 \leq j \leq n\}$ is called the ith *column of* G^n. The jth *row of* G^n is defined similarly. The *basic row coloring* of G^n assigns color i to every vertex in the ith row. The *basic column coloring* of G^n assigns color i to every vertex in the ith column. Note that both are valid vertex colorings of G^n using only n colors.

Definition. Let $n \geq 3$. Let $G^{n,n-1} = (V, E)$ where

$$V = \{(i, j) \mid 1 \leq i \leq n, 1 \leq j \leq n-1\},$$
$$E = \{\{(i, j), (r, s)\} \mid i \neq r \text{ and } j \neq s\}.$$

Rows (columns) of $G^{n,n-1}$, and the basic row (column) coloring of $G^{n,n-1}$ are defined in a manner similar to those of G^n. Note that the basic row coloring only needs $n-1$ colors.

Definition. If χ is a coloring of G^n or $G^{n,n-1}$, then χ *induces a colorful column* (*row*) if χ assigns to each vertex in a particular column (row) a different color. If the coloring being referred to is obvious, we may say "G has a colorful column (row)" to mean that the coloring induces a colorful column (row).

Lemma 39. *If χ is a $2n-2$ ($2n-3$) coloring of G^n ($G^{n,n-1}$), then χ either induces a colorful row or induces a colorful column, but not both.*

Proof. See [32, Lemma 2.1] for a proof of the unparenthesized version of this theorem. The parenthesized version can be established in a similar manner. □

We now define a way to connect two graphs such that if in some coloring one of them has a colorful row (column) the other will have a colorful column (row).

Definition. Let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ be two graphs such that either $G_1 \cong G_2 \cong G^n$, or $G_1 \cong G_2 \cong G^{n,n-1}$, or $G_1 \cong G^n$ and $G_2 \cong G^{n,n-1}$. In any case assume the vertices of G_k are of the form (k, i, j) in such a way that (k, i, j) corresponds to (i, j). The following graph is the *2-element chain of* G_1 *and* G_2, denoted $\mathrm{CH}(G_1, G_2)$:

$$V = V_1 \cup V_2,$$
$$E = E_1 \cup E_2 \cup E_{12},$$
$$E_{12} = \{\{(1, i, j), (2, r, s)\} \mid i \neq s \text{ and } r \neq j\}.$$

The edges in E_{12} are said to *link together* G_1 *and* G_2. Let $G_1, \ldots, G_s$ be graphs of type G^n or $G^{n,n-1}$. The *s-element chain of* $G_1, \ldots, G_s$, denoted $\mathrm{CH}(G_1, \ldots, G_s)$, can be defined by linking G_1 to G_2, G_2 to $G_3, \ldots, G_{s-1}$ to G_s.

In $\mathrm{CH}(G_1, G_2)$ the rth row of G_2 acts like the rth column of G_1 in terms of which vertices of G_1 it is connected to. This intuition underlies the next lemma.

Lemma 40. *Let χ be a $2n-3$ partial coloring of $\mathrm{CH}(G_1, G_2)$ that induces a colorful row (column) of the G_1 part. Any extension of χ to a $2n-3$-coloring of $\mathrm{CH}(G_1, G_2)$ must induce a colorful column (row) in the G_2 part.*

Proof. We only consider the case where $G_1 \cong C^n$ and $G_2 \cong G^{n,n-1}$. The other cases are similar.

Let χ and i be such that χ is a $2n-3$ partial coloring of $\mathrm{CH}(G_1, G_2)$ that induces the ith column of G_1 to be colorful. Assume, by way of contradiction, that there exists χ' and j such that χ' is a $2n-3$-coloring of $\mathrm{CH}(G_1, G_2)$ that is an extension of χ which does not induce a colorful row of G_2. By Lemma 39, χ' induces a colorful column of G_2, which we call the rth column of G_2. For $1 \leq j \leq n$ let $\chi'((1, i, j)) = c_j$ and for $1 \leq s \leq n-1$ let $\chi'((2, r, s)) = d_s$.

We show $|\{c_1, \ldots, c_n, d_1, \ldots, d_{n-1}\}| > 2n-3$. We know all the c_j's are distinct and all the d_s's are distinct. Let j be such that $1 \leq j \leq n$ and $j \neq r$; and let s be such that $1 \leq s \leq n-1$ and $s \neq i$. By the definition of $\mathrm{CH}(G_1, G_2)$ the vertices $(1, i, j)$ and $(2, r, s)$ are connected by an edge, hence $c_j \neq d_s$. The only possible equality of a c_j and a d_s is $c_i = d_r$. Hence $|\{c_1, \ldots, c_n, d_1, \ldots, d_{n-1}\}| = 2n-1 > 2n-3$ which contradicts χ' being a $2n-3$-coloring.

In the analogous proof for $G_1 \cong G_2 \cong G^{n,n-1}$ the last step is $|\{c_1, \ldots, c_{n-1}, d_1, \ldots, d_{n-1}\}| = 2n-2 > 2n-3$. This is the only case that needs χ' to be a $2n-3$-coloring. □

Lemma 41. *Let χ be a $2n-3$ partial coloring of $\mathrm{CH}(G_1, \ldots, G_s)$ that induces a colorful row (column) of the G_1 part. If s is even, then any extension χ' of χ to a $2n-3$-coloring of $\mathrm{CH}(G_1, \ldots, G_s)$ must induce a colorful column (row) of the G_s part; if s is odd, then χ' must induce a colorful row (column).*

Proof. This follows from the previous lemma and induction. □

Theorem 42. *Let $n \geq 3$. There exists a highly recursive graph $\tilde{G}$ such that*

$$\chi(\tilde{G}) = n \quad \text{and} \quad \chi^{r}(\tilde{G}) = 2n-2.$$

Proof. Fix e. We show how to construct a highly recursive graph G such that

(a) $\chi(G) = n$,
(b) $\{e\}$ is not a $2n-3$-coloring of G,
(c) $\chi^{r}(G) \leq 2n-2$.

The graph $\tilde{G}$ is formed by taking the disjoint union over e of all the graphs G as described above.

We construct G in stages. To avoid confusion we **do not** use 'G_s', we merely speak of 'G at stage s.'

Construction

Stage 0. At this stage G consists of two graphs G_1 and G_2 such that $G_1 \cong G_2 \cong G^n$.

Stage $s+1$. (At the end of stage s, G consists of $\mathrm{CH}(G_1, G_3, \ldots, G_{2s+1})$ and $\mathrm{CH}(G_2, G_4, \ldots, G_{2s+2})$, where each G_i is isomorphic to G^n.) Run $\{e\}_s$ on all the vertices of G_1 and G_2. There are several cases.

Case 1. There exists a vertex in G_1 or G_2 where $\{e\}_s$ does not converge. Let G_{2s+3} and G_{2s+4} be graphs isomorphic to G^n that use the least numbers not already in G for vertices. Extend the $s+1$-chains to $s+2$-chains using G_{2s+3} for the odd chain, and G_{2s+4} for the even chain.

Case 2: $\{e\}_s$ converges on all the vertices of G_1 and G_2, and either uses more than $2n-3$ colors, or is not a coloring. Proceed as in Case 1.

Case 3: $\{e\}_s$ converges on all the vertices in G_1 and G_2, uses $\leq 2n-3$ colors, is a coloring, and both G_1 and G_2 have colorful rows (columns). By the previous lemma any extension of $\{e\}_s$ to a coloring of G will induce G_{2s+1} and G_{2s+2} to either both have a colorful column or both have a colorful row. If we linked G_{2s+1} and G_{2s+2}, then the coloring could not be extended (as two adjacent G^n graphs would have the same type of file induced) but G might not be recursively $2n-2$ colorable. Instead we do the following: if s is odd, then extend both chains with graphs isomorphic to $G^{n,n-1}$, and then link the two new $G^{n,n-1}$ graphs; if s is even, then first extend both chains by a G^n graph before extending with a $G^{n,n-1}$ graph and linking. In either case there are an odd number of G^n graphs before the $G^{n,n-1}$ graph and the two $G^{n,n-1}$ graphs are linked. Stop the construction.

Case 4: $\{e\}_s$ converges on all the vertices in G_1 and G_2, uses $\leq 2n-3$ colors, is a coloring, and G_1 has a colorful row (column) while G_2 has a colorful column (row). Link both G_{2s+1} and G_{2s+2} to a graph isomorphic to $G^{n,n-1}$. The coloring $\{e\}_s$ cannot be extended to a $2n-3$-coloring of G since in such a coloring the $G^{n,n-1}$ graph would have to have both a colorful row and a colorful column. **End of Construction**

By comments made during the construction $\{e\}_s$ is not a $2n-3$-coloring of G. We show that $\chi(G) = n$ and $\chi^r(G) = 2n-2$.

No matter which case happens, G is a union of a chain of graphs of type G^n or $G^{n,n-1}$. The chromatic number of G is n since a chain can be n-colored by coloring (say) G_1 with a basic row coloring, G_3 with a basic column coloring, etc.

To recursively $2n-2$ color G, we will try to follow the strategy of coloring each G^n graph with a basic row or column coloring, in an alternating fashion, starting off by giving G_1 and G_2 a basic column coloring. If case 3 or 4 never occurs, then this will result in a recursive n coloring of G. If case 4 occurs, then note that the two graphs linked to the $G^{n,n-1}$ graph are either both basic column colored, or are both basic row colored. In either case, giving the $G^{n,n-1}$ graph a basic coloring of the opposite type will suffice and only n colors are used to color G. If case 3 occurs (the hard case), then we will proceed as follows. Note that the G^n graphs linked to the two $G^{n,n-1}$ graphs will both be basic column colored since G_1 and G_2 are basic column colored, and there are an odd number of G^n graphs in the chain (we made sure of this in the construction). Color one of the $G^{n,n-1}$ graphs with a basic row coloring. *This will only need $n-1$ colors, say* $\{1, 2, \ldots, n-1\}$. The other $G^{n,n-1}$ graph can be basic row colored with colors $\{n, n+1, \ldots, 2n-2\}$, making sure that the row colored with n is the one row that the link with the G^n graph will allow to be colored n (the first row). Even though both $G^{n,n-1}$ graphs are basic row colored they use disjoint sets of colors, hence the coloring is valid. □

Note. If in the above construction G^n is used instead of $G^{n,n-1}$, then with recursive chromatic number $2n-1$ is obtained. The upper bound in this case is easier since every highly recursive graph with chromatic number n has recursive chromatic number at most $2n-1$ [16, 32].

Appendix B

To establish Bean's Claim as stated in Section 6 we actually prove something stronger. The techniques we use appear in Bean's paper [3], but in a different form.

Definition. If $\{e\}$ is a Turing machine and W is a set on which $\{e\}$ is defined, then

$$\{e\}(W) = \{\{e\}(w) \mid w \in W\}.$$

Theorem 43. *Let L_0 be the graph consisting of 2^i isolated vertices, and let $\{e\}$ be a Turing machine. There exists a finite sequence of finite graphs $L_0, L_1, \ldots, L_r$ such that the following conditions hold.*

(a) *For every i, $1 \leq i \leq r$, L_i is an extension of L_{i-1}, i.e., $L_0 \subseteq L_1 \subseteq L_2 \subseteq \cdots \subseteq L_r = (V, E)$.*

(b) *For every i, $1 \leq i \leq r$, L_i can be obtained recursively from L_{i-1} and the values of $\{e\}(x)$ for every $x \in L_{i-1}$. If there is a vertex in L_{i-1} on which $\{e\}$ diverges, then $L_{i-1} = L_r$.*

(c) *There exists a set $W \subseteq V$ of outerplanar vertices such that either*

(1) *$\{e\}$ is not total on W, or*

(2) *there exists $v \in V$, $w \in W$ such that $\{v, w\} \in E$ and $\{e\}(v) = \{e\}(w)$, or*

(3) *$|W| = i+1$ and $\{e\}$ maps every element of W to a different value.*

(d) *L_r is planar.*

(e) *There is a 2-coloring of L_r in which W is 1-colored.*

The set W witnesses the fact that $\{e\}$ is not an i-coloring of L_r. We call W a witness of type 1, 2, or 3 depending on which subcase of (c) it falls under. If it falls under more then one, then we take the least such subcase.

Proof. We prove this by induction on i. We consider the $i=0$ case. Let

$$L_0 = (\{1\}, \emptyset), \qquad W = \{1\}.$$

If $\{e\}(1)\uparrow$, then W is a witness of type 1. If $\{e\}(1)\downarrow$, then W is a witness of type 3. In either case conditions (a)–(e) are easily seen to be satisfied.

Assume the theorem is true for i. We show it is true for $i+1$. Let L_0 be a graph consisting of 2^{i+1} isolated vertices. Let L_{01} be the first 2^i vertices of L_0 and L_{02} be the second 2^i vertices of L_0. By the induction hypothesis there exists graphs

$$L_{01} \subseteq L_{11} \subseteq L_{21} \subseteq \cdots \subseteq L_{r_11} = (V_1, E_1),$$
$$L_{02} \subseteq L_{12} \subseteq L_{22} \subseteq \cdots \subseteq L_{r_22} = (V_2, E_2);$$

and sets $W_1 \subseteq V_1$, $W_2 \subseteq V_2$ such that W_1 is a witness set for L_{r_11}, and W_2 is a witness set for L_{r_22}. Assume $r_1 \leqslant r_2$. We define graphs $L_0, L_1, L_2, \ldots, L_{r'}$ that satisfy the theorem (r' will either be r_2 or r_2+1). For $0 \leqslant j \leqslant r_1$ let

$$L_j = L_{j1} \cup L_{j2}.$$

For $r_1 + 1 \leqslant j \leqslant r_2$ let

$$L_j = L_{r_11} \cup L_{j2}.$$

If W_j ($j \in \{0, 1\}$) is a witness of type 1 or 2, then L_{r_2} is our final graph and $W = W_j$. The 2-coloring of the final graph with the witnesses 1-colored can be obtained by combining such colorings from L_{r_11} and L_{r_22}. It is easy to see that the sequence of graphs and the witness set W all satisfy requirements (a)–(e).

If both W_1 and W_2 are witnesses of type 3 then there are two cases:

Case 1: If $\{e\}(W_1) \neq \{e\}(W_2)$, then either there is some element $w \in W_1$ such that $\{e\}(w) \notin \{e\}(W_2)$; or there is some element $w \in W_2$ such that $\{e\}(w) \notin \{e\}(W_1)$. We examine the latter case, the former is similar. Our final graph is L_{r_2} and we let $W = W_1 \cup \{w\}$. By the induction hypothesis and the fact that W_1 is of type 3, $|W_1| = |\{e\}(w_1)| = i+1$. Since $w \notin W_1$ and $\{e\}(w) \notin \{e\}(W_1)$, $|W_1 \cup \{w\}| = |\{e\}(W_1 \cup \{w\})| = i+2$. Hence W is a witness of type 3. The 2-coloring of the final graph with the witnesses 1-colored can be obtained by combining such colorings from L_{r_11} and L_{r_22}. Therefore the sequence of graphs and the witness set W satisfy requirements (a)–(e).

Case 2: If $\{e\}(W_1) = \{e\}(W_2)$, then let w be a new vertex that is not in L_{r_11} or L_{r_22}. Let

$$L_{r_2+1} = L_{r_2} \cup \{\{u, w\} \mid u \in W_1\}, \qquad W = W_2 \cup \{w\}.$$

If $\{e\}(w)\uparrow$, then W is a witness of type 1. If $\{e\}(w)\downarrow \in \{e\}(W_1)$, then since w is connected to all vertices in W_1, W is a witness of type 2. If $\{e\}(w)\downarrow \notin \{e\}(W_1)$ (and hence $\{e\}(w) \notin \{e\}(W_2)$), then $\{e\}(W) = \{e\}(W_2 \cup \{w\}) = \{e\}(W_2) \cup \{e\}(w)$ which has cardinality $i+2$; hence W is a witness of type 3. Hence W is a witness set. A 2-coloring of L_{r_2+1} with W 1-colored can easily be obtained from the 2-coloring of L_{01} (that 1-colors W_1) and the 2-coloring of L_{02} (that 1-colors W_2). □

Acknowledgements

We would like to thank Gary Benson, Mark Berman, Susan Flynn, Valentina Harizanov, Larry Herman, Mike Lockwood, Dave Mount, and Jim Owings for proofreading and commentary; and Clyde Kruskal for suggesting that we look at parallel versions of some of the problems here. We would also like to thank Steven Lempp for useful discussions of Σ_3 complete sets which helped us to obtain Theorem 7; and Hal Kierstead for discussion of recursive chromatic numbers which helped us to obtain the results in Appendix A. The first author thanks NSF grant CCR-8808949, and second author thanks NSF-DCR84-05079, for financial support, Lastly, the second author wishes to thank Carolyn Woolf, whose suggestion that he work on more practical matters inspired much of this work.

References

[1] A. Amir and W. I. Gasarch, Polynomial terse sets, Information and Computation 77 (1988) 37–56.

[2] A. Amir, W. I. Gasarch and R. J. Beigel, Cheatable, P-terse, and P-superterse sets, University of Maryland, Dept. of Computer Science, Tech. Rep. 2090.

[3] D. R. Bean, Effective coloration, J. Symbolic Logic 41 (1976) 469–480.

[4] D. R. Bean, Recursive Euler and Hamiltonian paths, Proc. Amer. Math. Soc. 55 (March 1976) 385–394.

[5] R. J. Beigel, SAT^A is terse with probability 1, Tech. Rep. 4, The Johns Hopkins University, Dept. of Computer Science (1987).

[6] R. J. Beigel, Functionally supportive sets, Tech. Rep. 10, The Johns Hopkins University, Dept. of Computer Science (1987).

[7] R. J. Beigel, Query limited reducibilities. Ph.D thesis, Stanford University. Also The Johns Hopkins University Tech. Rep. 5, Dept. of Computer Science (1987).

[8] R. J. Beigel, A structural theorem that depends quantitatively on the complexity of SAT, Theoret. Comput. Sci., to appear.

[9] R. J. Beigel, Bounded queries to SAT and the boolean hierarchy, Tech. Rep. 7, The Johns Hopkins University, Dept. of Computer Science.

[10] R. J. Beigel and W. I. Gasarch, Supportive and parallel supportive sets. University of Maryland at College Park, Dept. of Computer Science, Tech. Rep. 1805.

[11] R. J. Beigel and W. I. Gasarch, On the complexity of finding the chromatic number of a recursive graph II: the unbounded case, Ann. Pure Appl. Logic, to appear.

[12] R. J. Beigel, W. I. Gasarch, J. T. Gill and J. Owings, Terse, superterse, and verbose sets, University of Maryland at College Park, Dept. of Computer Science, Tech. Rep. 1806 (1987).

[13] R. J. Beigel, W. I. Gasarch and L. Hay, Bounded queries classes and the difference hierarchy, Arch. Math. Logik.

[14] R. J. Beigel, W. I. Gasarch and J. Owings, Nondeterministic bounded query reducibilities, Ann. Pure Appl. Logic 41 (1989) 107–118.

[15] S. A. Burr, Some undecidable problems involving the edge-coloring and vertex coloring of graphs, Discrete Math. 50 (1984) 171–177.

[16] H. G. Carstens and P. Pappinghaus, Recursive coloration of countable graphs, Ann. Pure Appl. Logic 25 (1983) 19–45.

[17] W. I. Gasarch, The complexity of optimization functions, University of Maryland, Dept. of Computer Science, Tech. Rep. 1652.

[18] W. I. Gasarch and M. Lockwood, The existence of matchings for recursive and highly recursive bipartite graphs, University of Maryland at College Park, Dept. of Computer Science, Tech. Rep. 2029.

[19] J. Goldsmith, D. Joseph and P. Young, Self-reducible, P-selective, near-testable, and P-cheatable sets: the effect of internal structure on the complexity of a set. Proc. 2nd Annual Conf. on Structure in Complexity Theory (1987) 50–59.

[20] J. Kadin, $P^{NP[\log n]}$ and sparse Turing-complete sets for NP. Proc. 2nd Annual Structure in Complexity Theory Conference (1987) 33–40.

[21] H. A. Kierstead, An effective version of Dilworth's theorem, Trans. Amer. Math. Soc. 268 (1981) 63–77.

[22] H. A. Kierstead, Recursive colorings of highly recursive graphs, Canad. J. Math. 33 (1981) 1279–1290.

[23] H. A. Kierstead, An effective version of Hall's theorem, Proc. Amer. Math. Soc. 88 (1983) 124–128.

[24] M. W. Krentel, The complexity of optimization problems. J. Comput. Systems Sci., 36 (1988) 490–509.

[25] C. P. Kruskal, Searching, merging, and sorting in parallel computation, IEEE Trans. Comput. 32 (October 1983) 942–946.

[26] A. Manaster and J. Rosenstein, Effective matchmaking, Proc. London Math. Soc. 25 (1972) 615–654.

[27] A. Manaster and J. Rosenstein, Effective matchmaking and k-chromatic graphs, Proc. Amer. Math. Soc. 39 (1973) 371–378.

[28] J. Owings, A cardinality version of Beigel's nonspeedup theorem, J. Symbolic Logic (1989).

[29] H. Rogers Jr., Theory of Recursive Functions and Effective Computability (McGraw-Hill, New York, 1967).

[30] L. E. Rosier and H. Yen, Logspace hierarchies, polynomial time, and the complexity of fairness problems concerning ω-machines, SIAM J. Comput. 16 (1987) 779–807.

[31] J. H. Schmerl, The effective version of Brooks theorem, Canad. J. Math. 34 (1982) 1036–1046.

[32] J. M. Schmerl, Recursive colorings of graphs, Canad. J. Math. 32 (1980) 821–830.

[33] M. Snir, On parallel searching, SIAM J. Comput. 14 (August 1985) 688–708.

[34] R. I. Soare, Recursively Enumerable Sets and Degrees, Omega Series (Springer, Berlin, 1987).

[35] H. Tverberg, On Schmerl's effective version of Brooks' theorem, J. Combinatorial Theory (Series B) 37 (1984) 27–30.

[36] K. Wagner, More complicated questions about maxima and minima and some closures of NP, Proc. 13th ICALP, Lecture Notes in Computer Science 226 (Springer, Berlin, 1986) 434–443.

[37] G. Wechsung, On the boolean closure of NP, Proc. 1985 FCT, Lecture Notes in Computer Science 199 (Springer, Berlin, 1986) 485–493. This paper was actually co-authored by K. Wagner.

Annals of Pure and Applied Logic 45 (1989) 39–101
North-Holland

SOME PRINCIPLES RELATED TO CHANG'S CONJECTURE

Hans-Dieter DONDER
Institut für Mathematik II, Freie Universität Berlin, Arnimallee 3, 1000 Berlin, 33, West Germany

Jean-Pierre LEVINSKI
Department of Mathematics, Dartmouth College, Hanover, NH 03755, USA

Communicated by T. Jech
Received 10 March 1988; revised 1 August 1988

We determine the consistency strength of the negation of the transversal hypothesis. We also study other variants of Chang's conjecture.

0. Introduction

The transversal hypothesis TH is the statement that there exist ω_2 many almost disjoint functions from ω_1 to ω. This seems to be a basic principle because it implies the negation of Chang's conjecture, that no ω_1-complete uniform filter on ω_1 is ω_2-saturated and that every uniform ultrafilter on ω_1 is regular. The main aim of this paper is to determine the consistency strength of the negation of TH. This will be done in Section 7 where we show that ZFC + $\neg$TH is equiconsistent to ZFC + "there exists a $(<\omega_1, <\omega_1)$-Erdös cardinal". Since the definition of this type of partition cardinal is quite complicated we do not give it here.

Good lower and upper bounds for the strength of $\neg$TH have been known for some time because of the implication

$$\mathrm{CC} \to \neg\mathrm{TH} \to \mathrm{wCC}.$$

Here CC denotes the well-known Chang conjecture and wCC is the weak Chang conjecture which says that there is no family $\langle f_\nu \mid \nu < \omega_2 + 1\rangle$ of functions from ω_1 to ω_1 which is strictly increasing with respect to the club filter on ω_1. The strength of the principles above was determined in [7] and [8]. Our results in Section 7 especially show that the two implications above are strict in the sense of consistency strength.

A key to our main result is a model-theoretic equivalent of $\neg$TH which is given in Section 2 (see Theorem 2.14). This shows that $\neg$TH is really a variant of CC. We show that $\neg$TH is rather close to CC whereas there is a large gap between wCC and $\neg$TH. This gap can be filled with a family of game principles which we introduce in Section 2. We think that these principles are very natural. So we