Online Graph Colouring

Avery Miller

December 6, 2004

Abstract

This paper considers the problem of online graph colouring. An algorithm for vertex-colouring graphs is said to be *online* if each vertex is irrevocably assigned a colour before later vertices are considered. We first introduce the problem formally and define a perfomance metric to evaluate the success of an online colouring algorithm. Then, several upper and lower bounds are presented on the performance ratio for *any* algorithm solving the online graph colouring problem. These general bounds are shown to hold even when we relax and vary the conditions of the problem. Next, specific algorithms are presented and analyzed on how they perform in specific cases as well as in the general case. Finally, a discussion of future directions for research is presented.

1 Introduction

Before formally defining the online graph colouring problem, some definitions and notation are needed. We denote by $G = (V, E)$ a simple graph G with vertex set V and edge set E, where $|V| = n$ and $|E| = m$. A *colouring* of a graph G is a function $\varphi : V(G) \longrightarrow \mathbb{N}$. If $\varphi(x) \neq \varphi(y)$ whenever x is adjacent to y in G, then the colouring is *proper*. In what follows, we consider proper colourings only. The *chromatic number* of a graph G is defined as the minimum number of colours required in a colouring of G and is denoted by $\chi(G)$. The problem of finding $\chi(G)$ for any graph G (MINIMUM GRAPH COLOURING) is NP-complete [1].

The definition of online graph colouring arises by placing restrictions on how an algorithm $\mathcal{A}$ is allowed to process the input graph. An *online colouring* of a graph G is a colouring assigned to G by colouring its vertices in some order $v_1, \ldots, v_n$. The colour of v_i is assigned by only looking at the subgraph of G induced by the set $\{v_1, \ldots, v_i\}$, and the assigned colour of v_i is never

changed. The formal definition of online graph colouring follows naturally.

MINIMUM ONLINE COLOURING:
INSTANCE: A simple graph $G = (V, E)$ and an ordering $v_1, \ldots, v_n$ of V.
SOLUTION: An online colouring of G using the given ordering.
MEASURE: The number of colours used in the online colouring.
GOAL: Minimization

We would like to have a way to analyze and compare algorithms that solve MINIMUM ONLINE COLOURING. If $\mathcal{A}$ is an online colouring algorithm and $\{C_1, \ldots, C_k\}$ are the colourings produced by $\mathcal{A}$ for each possible ordering of the vertices of G, we denote by $\chi_{\mathcal{A}}(G)$ the maximum number of colours used among $\{C_1, \ldots, C_k\}$. Hence $\chi_{\mathcal{A}}(G)$ measures the worst-case behaviour of $\mathcal{A}$ on G. The *performance ratio* of an online graph colouring algorithm $\mathcal{A}$ for a class of graphs $\mathcal{C}$ is $\max_{G \epsilon \mathcal{C}}\{\chi_{\mathcal{A}}(G) / \chi(G)\}$. Clearly, a small performance ratio is preferred over a large one. To evaluate the success of an online colouring algorithm, we find bounds on the performance ratio of the algorithm.

This paper first considers general upper and lower bounds on the performance ratio of any algorithm solving the online graph colouring problem and variants. Next, we present specific algorithms that solve the online graph colouring problem and some results concerning the effect of restricting the input graph to certain classes of graphs or graphs with bounded width metric. Finally, this paper outlines and suggests some further topics of research concerning the online graph colouring problem.

2 Main Discussion

2.1 General Bounds for Online Graph Colouring

Before we construct even a single online colouring algorithm, we can determine some upper and lower bounds on the performance ratio of *any* online algorithm. This preliminary analysis gives us an idea as to what kind of results we can expect when constructing an algorithm. Trivially, we have an upper bound of n on the performance ratio since any algorithm can assign at most one colour to each of the n vertices of the input graph. Similarly, we can easily see that if the input graph is restricted to be from the class of complete graphs on n vertices, any online colouring algorithm will yield an exact solution with performance ratio of 1 (since $\chi(K_n) = n$). General lower bounds are not as easy to prove. The following two theorems are due to Halldórsson and Szegedy [5]:

Theorem 1 *The performance ratio of any deterministic online colouring algorithm is at least* $2n/\log^2 n$.

Theorem 2 *The expected performance ratio of any randomized online colouring algorithm is at least* $n/(16\log^2 n)$.

These two theorems imply that there is no general absolute approximation algorithm nor any general k-approximation algorithm (for constant k) for the online colouring problem. The proofs of the above theorems are interesting applications of the adversary technique for proving lower bounds. Below is a proof of Theorem 1, which I have modified from the original.

Proof: We define an algorithm $\mathcal{A}$ and an adversary $\mathcal{B}$. Let $v_1, \ldots, v_n$ be an imposed ordering on the vertices of a graph G. A *pre-neighbour* of a vertex v_i is an adjacent vertex that precedes v_i in the given ordering. The *pre-adjacencies* of v_i is a list of its pre-neighbours, and is denoted by $Pre(v_i)$. Note that a graph is fully specified once all pre-adjacencies are determined. Also, a colouring is proper if $v_j \epsilon\ Pre(v_i)$ implies that v_i and v_j are assigned different colours. We denote by $Adm(v_i)$ the set of *admissible colours*: those colours not used by any pre-neighbour of v_i by the adversary.

As an algorithm $\mathcal{A}$ proceeds, it receives from the adversary $\mathcal{B}$ an input vertex v_i and $Pre(v_i)$. Algorithm $\mathcal{A}$ then assigns a colour, $col_{\mathcal{A}}(v_i)$, to v_i such that its colouring of G so far is proper. Adversary $\mathcal{B}$ then colours v_i with a colour, $col_{\mathcal{B}}(v_i)$, such that its colouring of G so far is proper. Clearly $col_{\mathcal{A}}(v_i)$ need not equal $col_{\mathcal{B}}(v_i)$, so we let $Hue(p) = \{col_{\mathcal{B}}(v_i) : col_{\mathcal{A}}(v_i) = p\}$ (that is, $Hue(p)$ is the set of colours used by $\mathcal{B}$ to colour all vertices coloured p by $\mathcal{A}$). A *hue collection* H is a set of all the non-empty hues, formally $\{Hue(p) : Hue(p) \neq \emptyset\}$.

The task of the adversary is to decide on the adjacencies and the colour of each vertex. A vertex v_i shall be adjacent to a previous vertex v_j if and only if the colour of v_j does not belong to the set $Adm(v_i)$. Once $Adm(v_i)$ is specified for v_i, it does not hurt to make v_i adjacent to all nodes of any colour not in $Adm(v_i)$. The following observation immediately follows since $\mathcal{B}$ can always colour v_i using a colour from $Adm(v_i)$.

Observation 1 $Hue(col_{\mathcal{A}}(v_i)) \subseteq Adm(v_i)$.

The goal of the adversary is to appropriately select $Adm(v_i)$ and choose $col_{\mathcal{B}}(v_i) \epsilon\ Adm(v_i)$ such that the algorithm $\mathcal{A}$ needs to use many colours. Our measure of the progress of the adversary is the number of distinct $(col_{\mathcal{A}}(v_j)$ $,col_{\mathcal{B}}(v_j))$ pairs for $1 \leq j \leq i$. This measure of progress is used since the number of colours used by $\mathcal{A}$ is at least the number of distinct pairs divided by the maximum number of pairs with same first value (for instance, (1,2) (1,3) (1,3) (1,4) (2,1) (3,3) (3,4) (3,5) (4,5) implies that $\mathcal{A}$ uses at least $8/4 = 2$ colours since there are 8 distinct pairs and 4 pairs have 1 as first value). But by Observation 1, the maximum number of pairs with the same first value p is bounded above by the number of admissible colours for the last vertex assigned colour p, so we get the following bound.

Observation 2 *If m distinct $(col_{\mathcal{A}}(v_j)$ $,col_{\mathcal{B}}(v_j))$ pairs have been assigned after the i^{th} vertex has been coloured, then at least $m/(\max_{j\leq i} |Adm(v_j)|)$ colours have been used by $\mathcal{A}$.*

In designing a strategy for the adversary, it is useful to have a sufficient condition for progress. If Hue$(col_{\mathcal{A}}(v_i))$ is a strict subset of $Adm(v_i)$, then we are in a situation to make progress, since then $\mathcal{B}$ can colour v_i using a different colour from those that were coloured $col_{\mathcal{A}}(v_i)$ by $\mathcal{A}$. For instance, if $\mathcal{A}$ colours v_i using 1, and $Hue(1) = \{2, 3, 4\}$ while $Adm(v_i) = \{2, 3, 4, 5\}$, then $\mathcal{B}$ can set $col_{\mathcal{B}}(v_i) = 5$ so that a distinct $(col_{\mathcal{A}}(v_i), col_{\mathcal{B}}(v_i)) = (1, 5)$ is created.

Observation 3 *If $col_{\mathcal{B}}(v_i) \epsilon\ Adm(v_i) - Hue(col_{\mathcal{A}}(v_i))$ then progress is made.*

We are now ready to define an adversary $\mathcal{B}$. We let k be a positive even number representing the number of colours sufficient to properly colour the graph constructed, and let $n = k/2 \cdot \binom{k}{k/2}$ be the number of vertices in the graph G. We denote by $[k]^{(k/2)}$ the collection of all subsets of $\{1, \ldots, k\}$ of size $k/2$. Suppose *any* is a non-deterministic selection operator that takes a set as input and returns a member from the input set. For the current vertex v_i, the adversary defines $Adm(v_i) = any([k]^{(k/2)} - H)$, where H is the current hue collection. Next, the adversary defines $Pre(v_i) = \{v_j : col_{\mathcal{B}}(v_j) \notin Adm(v_i)\ and\ j < i\}$. Algorithm $\mathcal{A}$ now uses $Pre(v_i)$ to assign $col_{\mathcal{A}}(v_i)$. Finally, $\mathcal{B}$ sets $col_{\mathcal{B}}(v_i) = any(Adm(v_i) - Hue(col_{\mathcal{A}}(v_i)))$.

Finally, we are ready to prove the claim of Theorem 1. As each vertex gets coloured, at most one element is added to a hue, namely $Hue(col_{\mathcal{A}}(v_i))$ grows by at most one. So as long as i is less than or equal to n, at least one of the $\binom{k}{k/2}$ sets of size $k/2$ is not equal to $Hue(p)$ for any p, and can therefore be assigned to $Adm(v_i)$. Thus we have shown $Adm(v_i)$ is well-defined. Next, we

show $col_{\mathcal{B}}(v_i)$ is well-defined since $Hue(col_{\mathcal{A}}(v_i))$ is a subset of $Adm(v_i)$ (by Observation 1), but by choice of $Adm(v_i)$ we observe that $Adm(v_i)$ cannot equal $Hue(p)$ for any p. But by Observation 3, our choice of $col_{\mathcal{B}}(v_i)$ makes progress for each i, so the number of colours used by algorithm $\mathcal{A}$ must be at least $n/(k/2)$ (by Observation 2). We defined k to be the number of colours sufficient to properly colour the graph, so the performance ratio of $\mathcal{A}$ is at least $2n/k^2$, where $k = \log n(1 - o(1))$.

□

Proof of Theorem 2:

A proof sketch is provided here. Full details of the proof can be found in the original paper by Halldórsson and Szegedy [5]. We carry forward any terminology or notation used in the proof of Theorem 1 above.

An *oblivious* adversary is an adversary $\mathcal{B}$ that must construct the whole input in advance before feeding it to the algorithm $\mathcal{A}$. To prove the theorem, we define an oblivious adversary $\mathcal{B}$ that constructs a k-colourable graph on which the algorithm $\mathcal{A}$ will use at least n/k colours in the expected case, where $k = O(\log n)$. By Yao's minimax principle [14], it suffices to show that there exists a distribution of k-colourable graphs for which the average number of colours used by any *deterministic* algorithm is at least n/k. Suppose that *Random* is a function that takes a finite set as input, and selects an item from the finite set with uniform probability. The adversary $\mathcal{B}$ constructs a distribution of k-colourable graphs, setting $n = 2^{k/4}$, as follows. For the current vertex, v_i, in the ordering, $\mathcal{B}$ defines $Adm(v_i) = Random([k]^{(k/2)})$. Next, the adversary defines $Pre(v_i) = \{v_j : col_{\mathcal{B}}(v_j) \notin Adm(v_i) \text{ and } j < i\}$. Finally, $\mathcal{B}$ sets $col_{\mathcal{B}}(v_i) = Random(Adm(v_i))$. Notice that the construction is oblivious, since decisions made by the algorithm $\mathcal{A}$ are never consulted.

For a set p of size $k/2$, and a collection H of subsets of $[k]$, define $dist(p, H) = \min_{h \epsilon H} |p - h|$. Then it can be shown that the probability of making progress when colouring a vertex v_i is at least

$$\frac{E[dist(Adm(v_i), H)]}{|Adm(v_i)|}$$

Lemma 1 *Let H be any collection of subsets of $[k]$, and let p be a randomly chosen subset of $[k]$ of size $k/2$. If $|H| \leq 2^{k/4}$ and k is large enough, then*

$$E[dist(p, H)] \geq k/4$$

The proof of Theorem 2 follows almost immediately. If we consider the execution of a fixed deterministic algorithm on the graphs drawn from the distribution constructed above, the probability that a vertex v_i makes progress is at least

$$\frac{E[dist(Adm(v_i), H)]}{|Adm(v_i)|} \geq \frac{k/4}{k/2} = 1/2$$

Hence the expected number of distinct $(col_{\mathcal{A}}(v_i), col_{\mathcal{B}}(v_i))$ pairs is at least $n/2$ by linearity of expectation. Using Observation 2, the expected number of colours used by $\mathcal{A}$ is at least n/k. Thus the performance ratio is at least n/k^2, where $k = 4 \log n$.

□

Other lower bounds for online graph colouring can be shown. As reported by Vishwanathan [12], Szegedy shows that for any deterministic online algorithm $\mathcal{A}$ and integer k, there is a graph on at most $k(2^k - 1)$ vertices with chromatic number k that requires at least $2^k - 1$ colours. Thus the performance ratio of any online deterministic algorithm grows at least as fast as $n/(\log n)^2$. Similar results are shown by Gyárfás and Lehel [2], as they show that for every positive integer k, there exists a tree T and a bipartite graph G with no induced subpath of length 6 such that $\chi_{\mathcal{A}}(T) \geq k$ and $\chi_{\mathcal{A}}(G) \geq k$ for every online algorithm $\mathcal{A}$. For graphs with bounded chromatic number χ, Vishwanathan reports a lower bound of $\Omega((\log n/\log\log n)^{\chi/2})$ found by Vishwanathan and Szegedy. Vishwanathan also reports that Noga Alon independently found a lower bound of $\Omega((\log n/\log\log n)^{\chi-1})$. Finally, Vishwanathan proves a lower bound of $\Omega(\frac{1}{\chi-1}(\frac{\log n}{12(\chi+1)})^{\chi-1})$, which also holds for randomized algorithms as well [12].

The online constraints posed by the online colouring problem are quite strong, and one might wonder if relaxing any of the restrictions would have an impact on the lower bounds proven above. Firstly, analyzing such relaxations may give us insight about which aspects of the problem cause the most difficulty. Such insight may help us in constructing an algorithm whose performance ratio approaches a lower bound proven above. Secondly, some relaxations of the problem may be quite natural in that the relaxed problem may correspond to a real-life problem that we wish to solve. Take for instance the "answer before next request" condition. In many real-life situations, an algorithm may be allowed to delay a response or have access to enough memory to store a buffer of requests, which would allow the algorithm to view a few (say $O(\log^2 n)$) of the subsequent requests (this is known as *lookahead*). Another relaxation may apply to the "irrevocability" condition, where we allow an algorithm to reassign the colour of a constant fraction of the vertices.

We may also allow the algorithm to pre-sort the input according to some vertex property, e.g. by vertex degree. Halldórsson and Szegedy show that all of the relaxations described above yield limited or no improvement [5].

Finally, we consider a variation of the online colouring problem where an algorithm $\mathcal{A}$ is given the entire input graph in advance (but not told which order the vertices will be presented in for colouring). In this case, all that foils an algorithm is that it is not known to which sections of the input graph the vertices to be coloured belong. At first, it may seem clear that *a priori* knowledge of the input graph would surely allow an algorithm $\mathcal{A}$ to beat the general lower bound presented in Theorems 1 and 2 above. However, Halldórsson demonstrates a lower bound of $\Omega(n/\log^2 n)$ on the performance of any such online algorithm by defining appropriate adversaries [3]. But this lower bound matches the lower bound for the case of an unknown input graph! Therefore, advance knowledge of the entire input graph does not provide enough power to perform better than the previously proven lower bound of $\Omega(n/\log^2 n)$ on the performance ratio.

As presented above, several general bounds have been determined that apply to any algorithm $\mathcal{A}$ solving the online graph colouring problem. In finding these bounds, we have developed an idea as to what we can expect from any algorithm we construct, and at the same time we have gained some insight into what makes the problem difficult and how we can address these difficulties. Although knowing these general bounds is helpful to some degree, we need to analyze specific algorithms to determine how close we are to achieving the general bounds. There also may be applications where we do not care how well an algorithm performs on *every* graph, so we may find algorithms that perform extremely well for a given subset of graphs while we ignore the performance on graphs that will never appear as input.

2.2 Algorithms for Online Graph Colouring

2.2.1 The First Fit Algorithm

As a starting point, we can consider the consequences of using our intuition to formulate an online colouring algorithm. If we pretend that we are an online colouring algorithm and we are presented with a new vertex to colour, which colour do we assign to the vertex if our ultimate goal is to use the smallest number of colours possible? Perhaps we should be greedy and use the lowest colour available to us, and hope that doing so will minimize the total number used for the whole input graph G. This is precisely what the *first fit algorithm (FF)* does: assign to the current vertex v the least possible

colour that has not been assigned to any previously coloured vertex adjacent to v. It turns out that our intuitive approach performs well for specific classes of graphs, graphs with a bounded width metric, and asymptotically in general. Unfortunately, there also exist cases where FF performs very badly.

The FF algorithm performs very well in cases where the input graph belongs to a certain class of graphs. The following theorems (whose proofs can be found in the original paper) are due to Gyárfás and Lehel [2]. The *clique number of G*, denoted by $\omega(G)$, is the maximum number of vertices in a complete subgraph of G.

Theorem 3 *If G is a split graph (a* split graph *is a graph whose vertices can be partitioned into a clique and a stable set), then*

$$\chi_{\mathcal{FF}}(G) \leq \omega(G) + 1$$

Theorem 4 *If G is the complement of a bipartite graph, then*

$$\chi_{\mathcal{FF}}(G) \leq \frac{3}{2} \cdot \omega(G)$$

Theorem 5 *If G is the complement of a chordal graph (a* chordal graph *is a simple graph possessing no chordless cycles), then*

$$\chi_{\mathcal{FF}}(G) \leq 2 \cdot \omega(G) - 1$$

Theorem 6 *If G does not contain a subgraph isomorphic to P_4 (a path of four vertices), then*

$$\chi_{\mathcal{FF}}(G) = \chi(G)$$

The graphs considered in Theorems 3, 4, and 5 are subfamilies of perfect graphs, so $\chi(G) = \omega(G)$. Therefore the bounds given above show that the performance ratio of FF colouring is a constant for the specified classes of graphs.

Another class of graphs on which FF performs well is the class of d-inductive graphs. A graph G is *d-inductive* if the vertices of G can be ordered in such a way that each vertex has at most d edges to higher-numbered vertices. Such an ordering of the vertices of G is called an *inductive ordering*. In particular, planar graphs are 5-inductive, and chordal graphs are $\chi(G)$-inductive [7]. Irani shows that FF uses $O(d \log n)$ colours on G if G belongs to the class of d-inductive graphs [7]. Thus the performance ratio of FF on chordal and planar graphs is bounded above by $O(\log n)$. Irani also shows

a lower bound of $\Omega(d \log n)$ colours holds for any online graph colouring algorithm $\mathcal{A}$ on d-inductive graphs and that FF performs as well as any other online graph colouring algorithm [7]. In the same paper [7], Irani shows that if $\ell < \frac{n}{\log n}$, the lower bound of $d \log n$ colours still holds for any online colouring algorithm that is allowed to look ahead to ℓ subsequent vertices in the ordering.

In real-world applications, we are not always guaranteed that the input graph will belong to a certain class of graphs. Some real-life online situations often give rise to input patterns that seem to be "long" and "narrow", that is, "path-like". A *path decomposition* of a graph G is a tree decomposition of G which is simply a path (i.e. the nodes of the tree decomposition have degree at most two). The *pathwidth* of G is the minimum width over all possible path decompositions of G, that is, the pathwidth intends to measure how "path-like" G is. We consider a parameterized version of online graph colouring where the input graph has pathwidth bounded above by k. McCartin and Downey show that FF will use at most $3k + 1$ colours to colour any tree with pathwidth at most k [10]. In the same paper, the authors show that for each $k \geq 0$, there exists a tree T of pathwidth k such that FF will use $3k + 1$ colours to colour T. For general graphs of pathwidth at most k, the situation for FF is less well understood. McCartin and Downey refer to several results that bound the performance ratio of FF on graphs of pathwidth of at most k between the constants 4.4 and 25.72. Therefore, in certain situations where the pattern of inputs appear to be "path-like" (as measured by pathwidth), we know that the first fit algorithm will be satisfactory for solving the problem.

We say that *almost all graphs* satisfy a property $\mathcal{P}$ if the proportion of n-vertex graphs that satisfy $\mathcal{P}$ tends to 1 as n tends to infinity. Thus, asymptotically speaking, McDiarmid shows that $\chi_{\mathcal{FF}}(G) \leq (2 + \varepsilon) \cdot \chi(G)$ for almost all graphs G [11].

The general story for FF is rather disappointing if we consider examples of graphs that yield very high performance ratios. The proof of the following theorem is my own.

Theorem 7 *The performance ratio of FF can be as bad as $n/4$.*

Proof:
We construct a bipartite graph G on $2t$ vertices and an ordering of $V(G)$ that forces the first fit algorithm to use t colours. We define G by letting $V(G) = \{a_1, \ldots, a_t, b_1, \ldots, b_t\}$ and $E(G) = \{a_i b_j : i \neq j\}$. Now consider

running FF using the ordering $\{a_1, b_1, a_2, b_2, \ldots, a_t, b_t\}$. We show by induction on i that both a_i and b_i will be assigned the colour i, and thus FF uses $\{1, \ldots, t\}$ to colour G.
Base Case: FF will assign colour 1 to both a_1 and b_1 since they are not adjacent and 1 is the lowest colour available.
Induction Hypothesis: For each $i \in \{1, \ldots, k-1\}$, both a_i and b_i are assigned colour i for $k \geq 2$.
Inductive Step: The vertex a_k is adjacent to each of $\{b_1, \ldots, b_{k-1}\}$ by definition of $E(G)$, and by the induction hypothesis these use the colours $\{1, \ldots, k-1\}$. So FF must assign colour k to a_k since k is the least possible colour that has not been assigned to a previously coloured vertex adjacent to a_k. The vertex b_k is adjacent to each of $\{a_1, \ldots, a_{k-1}\}$ by definition of $E(G)$, and by the induction hypothesis these use the colours $\{1, \ldots, k-1\}$. Also, b_k is not adjacent to a_k by definition of $E(G)$. So FF must assign colour k to b_k since k is the least possible colour that has not been assigned to a previously coloured vertex adjacent to b_k.

The induction is complete, so we have shown that FF will use t colours to colour G, a graph on $2t$ vertices. Thus if $|V(G)| = n$, we have $\chi_{\mathcal{FF}}(G) = n/2$. But G is bipartite with bipartition (A, B) where $A = \{a_1, \ldots, a_t\}$ and $B = \{b_1, \ldots, b_t\}$, so $\chi(G) = 2$. Therefore the performance ratio of FF for the constructed graph G is $n/4$, as required.

□

The majority of results seem to concentrate on the FF algorithm, surely due to its simplicity and proven power in many cases. However the story does not end with FF, as there are still several areas where we would like to improve, for instance the poor performance ratio of $n/4$ in the general case. Like with most problems, we must turn away from our intuitive solution and attempt to find better algorithms that improve on the weaknesses of FF.

2.2.2 An Algorithm with Sublinear Performance Ratio

The main weakness of FF is the poor performance ratio of at $n/4$ over all possible input graphs. Considering that the trivial upper bound on the performance ratio of *any* algorithm is n (i.e. linear) and the tightest lower bound proven thus far is $\Omega(n/\log n)$ (i.e. sublinear), we would expect to at least be able to find an algorithm solving online graph colouring with a sublinear performance ratio. Lovász et al. [9] have shown that such an algorithm does exist by presenting an algorithm with performance ratio $(2n/\log^* n)(1+o(1))$, where $\log^* n$ is the smallest k for which the value of taking the logarithm of n recursively k times is at most 1. The details of the actual algorithm and

the proof of the performance ratio are rather complex, but they can be found in the original paper [9].

Perhaps of particular interest is the general framework of the algorithm proposed by Lovász et al. [9]. The authors define an algorithm **Color** that is constructed recursively from an algorithm **Partition*** that partitions the vertex set into subsets each having clique number strictly smaller than the input graph. This same kind of recursive construction was used by Wigderson [13] in defining a polynomial (but not online) approximate colouring algorithm with performance ratio $n(\log\log n)^2/(\log n)^2$ [13]. The fact that an offline technique became useful for the corresponding online problem reminds us that perhaps we should remain open and aware of progress made in the offline problem and perhaps online/offline versions of other problems.

Now we know that a sublinear performance ratio is achievable, so in cases where we desire the best performance over *all* possible input graphs, we know there is a better alternative to FF. However, so far we have only considered deterministic algorithms. We know that in many cases, allowing randomization in the design of an algorithm can give us extra power. Although we have already shown that the lower bound for randomized algorithms is no better than the deterministic lower bound, perhaps there is an algorithm that shows a general improvement in performance in the expected case.

2.2.3 A Randomized Algorithm

We can often use the technique of randomization to design an algorithm yielding better performance. The randomization technique has proven to be useful, as Vishwanathan [12] presents a polynomial-time algorithm that achieves an expected performance ratio bound of $O(n/\sqrt{\log n})$. This performance ratio can be improved to $O(n/\log n)$ using the polynomial-time randomized online colouring algorithm devised by Halldórsson [4]. A basic overview of the algorithm created by Vishwanathan is provided here; of course full details of the algorithm and analysis can be found in the original paper [12].

The algorithm is very similar in style to the sublinear performance ratio algorithm by Lovász et al. [9], except randomization is added. The general technique employed by the algorithm is to reduce the problem and then recursively apply the algorithm to solve the smaller problems until an easy base case is encountered. The algorithm exploits a crucial property of k-colourable graphs, that is, if a graph G is k-colourable then the subgraph induced by the neighbours of any vertex in G is $(k-1)$-colourable. Let us

assume that the number of vertices, n, in the graph and $\chi(G)$ are known in advance. In each iteration, the algorithm chooses a limited set of colours, say $\mathcal{S}$, which it uses to colour the vertices of G. All vertices assigned the colour i are placed in a bin B_i. The set $\mathcal{S}$ was chosen to be relatively small, so there may exist vertices of G that cannot be coloured properly. These non-coloured vertices are placed in a residue bin, $\mathcal{R}$. Now the algorithm wishes to partition the vertices that are found in the residue bin $\mathcal{R}$, so the algorithm randomly chooses a B_i where $i \in \{1, \ldots, |\mathcal{S}|\}$. Say it picks bin B_j. Each vertex in $\mathcal{R}$ must be adjacent to a vertex in B_j, or else the vertex could have been coloured j and placed in B_j, a contradiction to the construction of $\mathcal{R}$. So for each vertex $v \in \mathcal{R}$, find a vertex w in B_j adjacent to v and place v in the bin $\mathcal{R}_w$ (creating this bin if it doesn't already exist). Each $\mathcal{R}_w$ is $(k-1)$-colourable by the property stated above, so recursively call the algorithm on each $\mathcal{R}_w$. The algorithm terminates the recursion when a bin is 2-colourable, and it uses the algorithm by Lovász et al. [9] to colour the bin with at most $2 \log n$ colours.

The above algorithm and the improvement by Halldórsson show that the online graph colouring problem is yet another problem where randomization can give us more power in designing algorithms. However, we are still unable to make upper and lower bounds meet for either the randomized or the deterministic case. Either the lower bounds are correct and we just haven't found good enough algorithms to match them, or we have found the best algorithms but we have not yet been clever enough to prove a tighter lower bound. Clearly this is a topic for further study.

2.3 Further Study

As with any algorithmic analysis of a problem, the ultimate goal is to find the best possible algorithm that will solve the problem. We can stop once we have found a tight lower bound for *any* algorithm solving the problem that matches the upper bound determined by a well-defined algorithm. As the above discussion suggests, we are not finished with the online graph colouring problem.

There is still much territory to be explored in the parameterized problem domain, as one can delve deeper into the bounded pathwidth problem, or investigate other graph width metrics of interest (treewidth, for instance). One can either analyze how the previously proposed algorithms perform on various parameterized instances of the problem, or a new algorithm can be designed to specifically exploit the bounded parameter presented in the input. Both avenues present a vast amount of previously unexplored territory.

The most popular investigation always seems to involve the analysis of certain classes of graphs as input. This approach is generally the most fruitful, since there is usually a significant amount of graph theoretic literature outlining various properties of certain classes of graphs. I was surprised, however, at the absence of Kneser graphs in the discussion of online graph colouring algorithms. Kneser graphs have been quite popular in graph theoretic investigations of graph colouring. The following definition and discussion of Kneser graphs is taken from the CO442 Course Notes [6]. For n, k positive integers with $k \leq n$, the *Kneser Graph* $KN(n,k)$ is defined as follows: The vertices of $KN(n,k)$ are all the k-element subsets of $\{1,2,\ldots,n\}$, and an edge joins vertices S and T if and only if $S \cap T = \emptyset$. The class of Kneser graphs contains many familiar classes of graphs. If $k > n/2$, then $KN(n,k)$ is the empty graph. If $k = 1$, then $KN(n,1) = K_n$, the complete graph on n vertices. Finally, $KN(5,2)$ is the Petersen graph. It can be shown that the chromatic number of $KN(2k+\ell,k)$ is $\ell + 2$. One could investigate the behaviour of previously proposed algorithms on a set of Kneser graphs, or one could devise a clever algorithm that performs well for a set of Kneser graphs. Better yet would be to find a practical reason to study the online colouring problem on Kneser graphs!

3 Conclusion

The online graph colouring problem has been studied extensively over many years, but the analysis of the problem is not complete. General lower bounds of $\Omega(n/(\log n)^2)$ have been proven for the the deterministic and randomized models of computation, however these do not match the current upper bounds that have been found. It is also important to note that online graph colouring does not seem to get easier after relaxing some of the restrictions placed on the algorithm by the problem.

Several algorithms have been proposed to solve online graph colouring, each successful in their own way. The First Fit algorithm is the simplest and most intuitive, and also tends to work very well for a variety of specific graph classes, for graphs with bounded pathwidth, and is effective in the asymptotic case. The weak point for the FF algorithm is the disastrous potential performance ratio of $n/4$ which arises for certain bipartite graphs. Thus the FF algorithm cannot claim to have sublinear performance ratio. On the other hand, Lovász et al. [9] created an algorithm whose performance ratio is sublinear. The algorithm is deterministic, so one wonders if randomizing it would increase its power. Such a randomization was performed by Vishwanathan [12], and the resulting randomized algorithm came even closer to

matching the lower bound. Halldórsson [4] improved the randomized algorithm to yield the current best randomized upper bound of $O(n/\log n)$.

Much work is left to be done before the book can be closed on the online graph colouring problem. We have yet to match upper and lower bounds, delve deeper into parameterized problems, and investigate such graph classes as the Kneser graphs. Perhaps the day will come when mankind has fully analyzed the online graph colouring problem as well as the other difficult NP-complete problems, but when that day comes, life will be just a little less fun.

References

[1] M. R. Garey and D. S. Johnson. Computers and Intractability: A Guide to the Theory of NP-Completeness. W.H. Freeman and Company, San Francisco, 1979.

[2] A. Gyárfás and J. Lehel. On-line and First Fit Colorings of Graphs. *Journal of Graph Theory*, 12(2) : 217-227, 1988.

[3] M. M. Halldórsson. Online Coloring Known Graphs. *Journal of Combinatorics*, 7(1) : #R7 1-9, 2000.

[4] M. M. Halldórsson. Parallel and On-line Graph Coloring. *Journal of Algorithms*, 23 : 265-280, 1997.

[5] M. M. Halldórsson and M. Szegedy. Lower Bounds for On-line Graph Coloring. *Theoretical Computer Science*, 130 : 163-174, 1994.

[6] P. E. Haxell, H. R. Hind, R. B. Richter, N. C. Wormald and D. H. Younger. C&O 442 Graph Theory Course Notes, 6-7, Fall 2004.

[7] S. Irani. Coloring Inductive Graphs On-line. In *Proceedings of the 31st Annual IEEE Symposium on Foundations of Computer Science*, 470-479, 1990.

[8] H. A. Kierstead and W. T. Trotter. On-line Graph Coloring. *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, 7 : 85-92, 1992.

[9] L. Lovász, M. Saks, and W. T. Trotter. An On-line Graph Coloring Algorithm with Sublinear Performance Ratio. *Discrete Mathematics*, 75 : 319-325, 1989.

[10] C. McCartin and R. G. Downey. Online Problems, Pathwidth, and Persistence. To appear in *Proceedings of the International Workshop on Parameterized and Exact Computation 2004*.

[11] C. McDiarmid. Coloring Random Graphs Badly. *Graph Theory and Combinatorics*. Pitman Advanced Publishing Program, San Francisco, 76-86, 1979.

[12] S. Vishwanathan. Randomized Online Graph Coloring. In *Proceedings of the 31st Annual IEEE Symposium on Foundations of Computer Science*, 464-469, 1990.

[13] A. Wigderson. Improving the Performance Guarantee for Approximate Graph Coloring. *Journal of the ACM*, 30 : 729-735, 1983.

[14] A. C.-C. Yao. Probabilistic Computations: Towards a Unified Measure of Complexity. In *Proceedings of the 18th Annual IEEE Symposium on Foundations of Computer Science*, 535-542, 1977.