DIMACS Technical Report 95-01
January 1995

Analysis of approximate algorithms for constrained and unconstrained edge-coloring of bipartite graphs

by

Ravi Jain[1,2,3]
Applied Research
Bellcore

John Werth
Dept of Computer Sciences
Univ. of Texas at Austin

[1]Permanent Member
[2]Part of this research was performed while the author was at the University of Texas and supported by the IBM Corporation through grant 61653 and by the State of Texas through TATP Project 003658-237.
[3]Address correspondence to Ravi Jain, Applied Research, Bellcore, 445 South St, Morristown, NJ 07960. e-mail: `rjain@thumper.bellcore.com`

ABSTRACT

The problem of edge-coloring a bipartite graph is to color the edges so that adjacent edges receive different colors. An optimal algorithm uses the minimum number of colors to color the edges. We consider several approximation algorithms for edge-coloring bipartite graphs and show tight bounds on the number of colors they use in the worst case. We also briefly consider the constrained edge-coloring problem where each color may be used to color at most k edges, and obtain bounds on the number of colors used by the approximation algorithms in the worst case.

1 Introduction

Many applications can be modeled as edge-colorings of bipartite graphs, such as the scheduling of data transfers in parallel computers and communications switches [10]; vertices represent communicating entities, edges represent the data transfers, and edges with the same color represent data transfers that can occur in parallel. For a bipartite graph of degree Δ, a minimum edge-coloring requires Δ colors [2] and can be obtained in polynomial time [6, 7, 8, 4].

Scheduling applications, such as the scheduling of parallel I/O operations, motivate the development of faster algorithms for approximate edge-coloring of bipartite graphs [9, 14, 10]. We analyze the worst-case behavior of several greedy approximation algorithms for edge-coloring bipartite graphs. Experimental studies have shown that these algorithms can generate minimum or near-minimum edge colorings in much less execution time than exact algorithms [14, 12]. However, previous studies [14, 1] do not provide tight theoretical bounds on the worst-case behavior of these algorithms.

For the approximation algorithms we will discuss, different runs of an algorithm may well edge color a given graph with different numbers of colors, since the algorithm may make arbitrary choices of which edge to color next. We will need notation for the maximum number of colors used by an algorithm for edge-coloring a particular graph as well as for coloring any graph of given degree.

Def. For a given graph G let $A(G)$ denote the maximum number of colors used by algorithm A to edge-color G in any execution of A.

Def. Let $B(A, \Delta) = \max\{A(G) : G \text{ has degree } \Delta\}$. We say the positive integer x is a *bound* on A for all graphs of degree Δ iff $B(A, \Delta) \leq x$. We say the bound x is *tight* iff $B(A, \Delta) = x$.

In the rest of this paper, a graph is understood to be bipartite, a coloring to be an edge-coloring, and the degree of a graph to be a positive integer, unless otherwise specified. $G(\Delta)$ denotes a bipartite graph of degree Δ.

In sec. 2 we define a class of greedy approximation algorithms. We discuss in sec. 3 the worst-case analysis of approximation algorithms for the *unconstrained* edge-coloring problem, where the objective is to obtain an edge-coloring of a bipartite graph using as few colors as possible. We derive relationships between the bounds on the number of colors used by two approximation algorithms for unconstrained edge-coloring, and show that the bounds are tight. In sec. 4 we will consider the following *constrained* edge-coloring problem: Find a minimum edge-coloring of a bipartite graph where no more than k edges can have the same color. This constraint arises frequently in data transfer scheduling applications as a limitation in the capacity of the channel used for the data transfers [10]. For the constrained edge-coloring problem we consider the worst-case behavior of two approximation algorithms also, presenting bounds on the number of colors used, and the algorithms' time complexities. In sec. 5 we end with a discussion.

2 The Greedy algorithm

Since we will be presenting several greedy algorithms, we establish a template for describing them using pseudo-code as follows. The *Order()* function will be specified below. The **break** statement exits the smallest enclosing loop. A sequence is denoted by angle brackets.

```
Algorithm Greedy
Input: Bipartite graph $G = (A, B, E)$, Coloring constraint $k$,
$0 < k \leq \min(|A|, |B|)$
Output: An edge-coloring of $G$, $color : E \to \{1, 2, ...\}$

   $F := Order(A, B, E)$;
      /* $F$ is some ordered sequence of all the edges in $E$ */
   $i := 1$;
   while $F \neq \langle\ \rangle$ {
      $M := \{\ \}$;
         /* $M$ is edges which are assigned color $i$ */
      for each $e$ read in sequence from $F$ {
         if neither endpoint of $e$ is colored $i$ {
            $color(e) := i$;
            $E := E - \{e\}$;
            $M := M \cup \{e\}$;
         }
         if $|M| = k$, break;
      }
      $i := i + 1$;
      $F := Order(A, B, E)$;
         /* Re-order the remaining edges in the graph */
   }
```

We call all algorithms described by the template above, regardless of the function used for $Order()$, greedy algorithms. The approximation algorithms we investigate in this section are defined by the function used for $Order()$.

1. First-Come First-Served, **FCFS**. $Order()$ is the identity function.
2. Highest Degree First, **HDF**. $Order()$ sorts the edges in descending order of the maximum of the degrees of their endpoints. Ties between edges are broken arbitrarily.
3. Highest Combined Degree First, **HCDF**. $Order()$ sorts the edges in descending order of the sum of the degrees of their endpoints. Ties between edges are broken arbitrarily.

Note that the greedy algorithm above examines the list of edges. In [13] we have also considered a greedy algorithm which examines the list of vertices.

3 The unconstrained edge-coloring problem

In this section we derive bounds on the behavior of the greedy algorithm for the unconstrained case, i.e., $k \geq \min(|A|, |B|)$.

Lemma 3.1 *[10, 1] For all greedy edge-coloring algorithms A where the edge-coloring is unconstrained, i.e., $k \geq \min(|A|, |B|)$, $\forall\, \Delta, B(A, \Delta) \leq 2\Delta - 1$.*

Lemma 3.1 implies that there are $O(\Delta)$ iterations of the **while** loop. The **for** loop takes time $O(m)$, and a bucket sort taking time $O(m)$ can be used for $Order()$ [14], so that **FCFS**, **HDF** and **HCDF** all run in time $O(m\Delta)$. (In [13] we have described versions of **FCFS** and **HDF** which operate on lists of vertices rather than edges, and hence can be implemented to run in time $O(m + n\Delta)$). Note that Lemma 3.1 also implies, for instance, that $\forall\, \Delta, B(\mathbf{FCFS}, \Delta) \leq 2\Delta - 1$ and $\forall\, \Delta, B(\mathbf{HDF}, \Delta) \leq 2\Delta - 1$, but does not imply that these bounds are tight. Bar-Noy et al [1] have shown that $\forall\, \Delta$, $B(\mathbf{FCFS}, \Delta) = 2\Delta - 1$. Their proof is by construction of a set of graphs, $\{G'(\Delta) : 1 \leq \Delta\}$, where each $G'(\Delta)$ is a full Δ-ary tree of three levels (i.e., $G'(\Delta)$ consists of a root with Δ children, each of degree Δ). We might expect that **HDF** and **HCDF** perform better than **FCFS**. In fact, we can show that they can perform substantially better.

Lemma 3.2 $\forall\, \Delta > 1 : \mathbf{HCDF}(G'(\Delta)) = \Delta$ *and* $\mathbf{HDF}(G'(\Delta)) = \Delta + 1$ *and* $\mathbf{FCFS}(G'(\Delta)) = 2\Delta - 1$

Proof. Left to the reader.

We have also found experimentally that **HDF** and **HCDF** can perform substantially better than **FCFS** when presented with graphs generated pseudo-randomly [12]. Further, in our experiments we found that in no case do they perform any worse; in Theorem 3.1 we show a theoretical justification for this. Let $G = (V, E)$ be a bipartite graph with vertex set V, edge set E, degree Δ, and let $d(u)$ denote the degree of vertex $u \in V$.

Lemma 3.3 *If a vertex $u \in V$ is not colored during an iteration of* **HCDF**, *then either $d(u) = 0$ at the start of that iteration, or $\forall\,(u,v) \in E$, $\exists\,(v,w) \in E$: (v,w) is colored during that iteration and $d(u) \le d(w)$.*

Proof. Clearly u will not be colored if $d(u) = 0$ at the start of the iteration. Assume $d(u) > 0$. Then, since **HCDF** is greedy, u is not colored during the iteration iff $\forall\,(u,v) \in E$, $\exists\,(v,w) \in E$ such that (v,w) is colored in this iteration. From the criterion used by **HCDF** to choose edges, edge (v,w) is colored in this iteration iff $d(u) + d(v) \le d(v) + d(w)$, i.e., $d(u) \le d(w)$. □

Notation. We partition E into subsets by degree, i.e., let $E(i) = \{(u,v) : \max(d(u), d(v)) = i\}$, for $1 \le i \le \Delta$. Similarly, let $V(i) = \{u : d(u) = i\}$, for $1 \le i \le \Delta$. (Clearly **HDF** examines edges in $E(\Delta)$ followed by edges in $E(\Delta - 1)$, etc.) Consider a coloring of G by some execution of **HCDF**. Let E' denote the set of edges, and V' the set of vertices, colored by the first iteration of **HCDF**. Let $E'(i) = E' \cap E(i)$, and let $V'(i) = V' \cap V(i)$, i.e., let $V'(i)$ be the vertices of degree i included in E'.

Lemma 3.4 *There exists an execution of* **HDF** *on input graph G such that the set of edges colored during the first iteration is identical to E'.*

Proof. By definition of **HDF** all the edges in $E(i+1)$ are examined before any edges in $E(i)$, for $i \in \{1, ..., \Delta - 1\}$. For each i, let the edges in $E'(i)$ be ordered before any of the edges in $E(i) - E'(i)$, so that **HDF** will examine the edges in $E'(i)$ first. We now show by induction on the maximum degree of vertices in E' that **HDF** colors exactly the edges in E' on its first iteration.

base. $(i = \Delta)$. Since $E'(\Delta)$ is a matching and its edges are examined before any edges in $E(\Delta) - E'(\Delta)$, **HDF** will color all the edges in $E'(\Delta)$. We show that, having done so, **HDF** cannot color any other edge in $\in E(\Delta) - E'(\Delta)$. Suppose $u \in V(\Delta) - V'(\Delta)$. Then from Lemma 3.3, none of the edges (u,v) adjacent to u are colorable.

hyp. Of the edges in $E(\Delta) \cup \ldots \cup E(i{+}1)$, **HDF** colors exactly the edges in $E'(\Delta)\ \cup E'(\Delta - 1)\ \cup \ldots \cup E'(i+1)$ on the first iteration.

ind. (i). Clearly edges in $E'(i)$ cannot be adjacent to those in $E'(\Delta) \cup \ldots \cup E'(i+1)$. However, edges in $E'(i)$ can be adjacent to the set of edges $E(\Delta) \cup \ldots \cup E(i{+}1) - E'(\Delta) \cup \ldots \cup E'(i+1)$. But by the induction hypothesis, **HDF** has not colored any of the edges in this set during its first iteration. Thus **HDF** will color all the edges in $E'(i)$. As for the base case, consider $u \in V(\Delta) - V'(\Delta)$. By Lemma 3.3, $\forall\,(u,v) \in E,\ \exists\,(v,w) \in E'(j)$ for some $j \geq i$. Hence **HDF** cannot color any edge in $E(i) - E'(i)$. □

Theorem 3.1 $\forall\, G$, **HCDF**$(G) \leq$ **HDF**$(G) \leq$ **FCFS**(G).

Proof. First note that since any greedy schedule can be produced by the arbitrary ordering **FCFS**, it follows that **HCDF**$(G)\ \leq$ **FCFS**(G) and **HDF**$(G)\ \leq$ **FCFS**(G). To show **HCDF**(G) $\leq$ **HDF**(G), we show the stronger result that the set of edges colored during every iteration of **HCDF** can be colored by the same iteration of **HDF**. The proof is sketched as follows. From Lemma 3.4, after one iteration, the remaining graph $G' = (V, E - E')$ for both algorithms is identical; repeated application of Lemma 3.4 gives the result. □

From the foregoing we might expect that the bound given by Lemma 3.1 could be tightened further for **HDF** and **HCDF**. In the following sections we show that this is not the case.

3.1 A tight bound for HDF

In this section we prove that for any Δ a bipartite graph $G(\Delta)$ can be constructed such that **HDF**$(G) = 2\Delta - 1$. In fact, we will show that G is a tree. To describe the construction precisely we introduce some notation.

Notation. (See Fig. 1 for examples.) In the following, upper-case italic letters denote vertices or subtrees of a tree; they may be subscripted. If A and B are vertices, $A;B$ denotes that they are siblings, and $A\langle B\rangle$ denotes that A is the parent of B. R is used to distinguish the root of a (sub)tree, and C for its child. Thus $R\langle C_1;C_2\rangle$, where the C_i are vertices which can be distinguished from each other, denotes a binary tree of two levels. A set of siblings which need not be distinguished from each other is denoted using an array notation: thus $R\langle C[2]\rangle$ also denotes a binary tree with two levels. Angle brackets have higher precedence than semi-colons. Thus $R\langle C_1;C_2\rangle\ ;A$ denotes a forest with two trees, and $R\langle C_1;C_2;A\rangle$ denotes a ternary tree with two levels.

Def. Two trees S and T with roots R_S and R_T, respectively, are *root-merged* by deleting R_S and R_T (along with any incident edges), introducing a vertex R, and adding edges from R to every child of R_S and R_T. Letting $+$ denote root-merging, if $S = R_S\langle S_1; S_2; ...; S_i\rangle$ and $T = R_T\langle T_1; T_2; ...; T_j\rangle$ then

$$S + T = R\langle S_1; ...; S_i; T_1; ...; T_j\rangle$$

We now construct two families of trees to be used later in the construction of $G(\Delta)$, and consider how they could be colored.

Def. The trees $F_{i,\Delta}$ and $H_{i,\Delta}$ are defined by mutual recursion as follows. (See Fig. 1 for examples). The operator $\langle\rangle$ has precedence over $+$, which has precedence over ; and [].

1. $F_{0,\Delta}$ consists of a single vertex, S.
2. $H_{1,\Delta} = R_1\langle C_1\langle F_{0,\Delta}[\Delta - 1]\rangle$
3. $F_{1,\Delta} = H_{1,\Delta}$
4. For $1 < i < \Delta$, $H_{i,\Delta} = R_i\langle C_i\ \langle F_{i-1,\Delta}[\Delta - 1]\rangle\rangle$
5. For $1 < i < \Delta$, $F_{i,\Delta} = H_{1,\Delta} + H_{2,\Delta} + ... + H_{i,\Delta}$

Def. A vertex is *critical* if it has maximal degree.

Observe that for every tree $H_{i,\Delta}$, the child of the root, C_i, is critical. Also note that for every $F_{i,\Delta}$, the root is not critical while all the children of its root are critical. Thus, by construction, at every alternate level of $F_{i,\Delta}$, all the vertices are critical.

Def. A vertex is *covered* if some edge incident upon it is colored. A vertex is *colored with color i* if some edge incident upon it is colored i.

We will now show that the construction of the tree $F_{i,\Delta}$ enables **HDF** to make a sequence of choices such that i colors are consumed before every critical vertex in the tree is covered. As an example, in Fig. 1(c), edges $(R_2, C_1), (a, b), (c, d)$ and (e, f) would be colored with color 1, necessitating the use of color 2 to cover the critical vertex C_2.

Lemma 3.5 *For every tree $F_{i,\Delta}$, $0 < i < \Delta$, there exists a sequence of choices made by* **HDF** *such that i colors are required to cover all the critical vertices in $F_{i,\Delta}$, and the root of $F_{i,\Delta}$ is colored with every color in the set $\{1, ..., i\}$.*

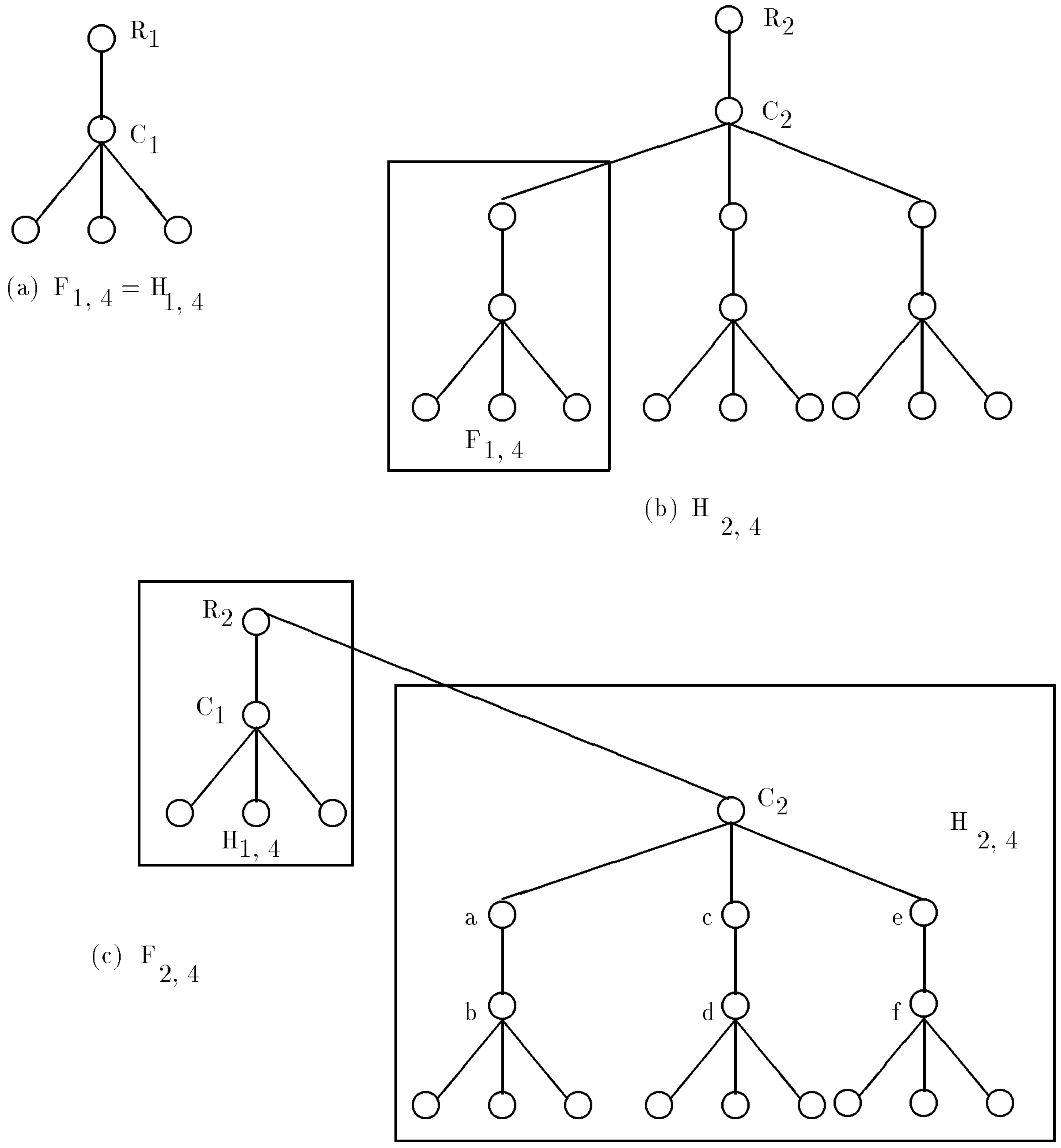

Figure 1: Example construction to show **HDF** takes up to $2\Delta - 1$ colors to color a graph of degree Δ

Proof. By induction over i.

base. $i = 1$. For $F_{1,\Delta} = H_{1,\Delta} = R_1\langle C_1\langle S[\Delta - 1]\rangle\rangle$, the choice of coloring edge (R_1, C_1) with color 1 suffices.

hypothesis. For all $F_{j,\Delta}$, $0 < j < i < \Delta$, there exists a sequence of choices made by **HDF** such that j colors are required to cover all the critical vertices in $F_{j,\Delta}$, and the root of $F_{j,\Delta}$ is colored with all colors in $\{1, ..., j\}$.

induction. Consider the coloring of $F_{i,\Delta}$. By definition,

$$\begin{aligned} F_{i,\Delta} &= H_{1,\Delta} + H_{2,\Delta} + ... + H_{i,\Delta} \\ &= R_i\langle C_1\langle F_{0,\Delta}[\Delta - 1]\rangle; C_2\langle F_{1,\Delta}[\Delta - 1]\rangle; ...C_i\langle F_{i-1,\Delta}[\Delta - 1]\rangle; \rangle \end{aligned}$$

Recall that by construction, for all $j, 1 \leq j \leq i$, C_j in the expression above is critical. First, we will show that there is a sequence of choices made by **HDF** such that color i is required to cover C_i. It will follow that i colors are required in order to cover all critical vertices in $F_{i,\Delta}$, and that R_i is colored with all colors $\{1, ..., i\}$.

Observe that color i is required to cover C_i only if all the neighbors of C_i are colored with all colors $\{1, ..., i - 1\}$. The neighbors of C_i consist of its parent and its children.

Considering the children of C_i, it can be seen from the expression above that $F_{i,\Delta}$ is constructed so that the children of C_i are the roots of $F_{i-1,\Delta}$. By the induction hypothesis, there is a sequence of choices made by **HDF** such that the root of $F_{i-1,\Delta}$ is colored with all colors $\{1, ..., i - 1\}$.

Consider R_i, the parent of C_i, which is the root of $F_{i,\Delta}$. By definition, $F_{i,\Delta} = F_{i-1,\Delta} + H_{i,\Delta}$. That is, R_i along with all the subtrees rooted at its children C_1, ..., C_{i-1} form the tree $F_{i-1,\Delta}$. By the induction hypothesis, R_i is already colored with all colors $\{1, ..., i - 1\}$.

Clearly, the sequences of choices given by the induction hypothesis can be merged appropriately so that every child of C_i, and also C_i's parent, is colored with all colors $\{1, ..., i - 1\}$. Covering the critical vertex C_i thus requires color i. In addition, once the colors $\{1, ..., i - 1\}$ have been used, every critical vertex in the subtrees rooted at C_1, ..., C_{i-1} is covered, as is every critical vertex in the subtrees rooted at the children of C_i. Thus coloring C_i covers all critical vertices in $F_{i,\Delta}$.

Now we need to show that the root of $F_{i,\Delta}$ is colored with all colors in $\{1, ..., i\}$. It follows directly from the arguments above that every C_j is colored j, for $1 \leq j \leq i - 1$. For C_i, we let **HDF** choose to color edge (R_i, C_i) with color i. □

Theorem 3.2 $\forall\, \Delta, B(\mathbf{HDF}, \Delta) = 2\Delta - 1.$

Proof. By construction of the graph. Let

$G(\Delta) = R\langle F_{\Delta-1,\Delta}[\Delta]\rangle$

Clearly, R is a critical vertex. From Lemma 3.5 there exists a sequence of choices made by **HDF** such that the root of every $F_{\Delta-1,\Delta}$ is colored with all colors in $\{1, ..., \Delta-1\}$. In addition, it is possible to use $\Delta - 1$ colors to color every critical vertex in $F_{\Delta-1,\Delta}$, while still leaving R uncolored. Therefore, each of the links incident to R will have to be colored with a color not in the set $\{1, ..., \Delta-1\}$, and each will require a distinct color. Therefore Δ additional colors are required to color the links incident to R, i.e., at least $2\Delta - 1$ colors are required to color $G(\Delta)$. **HDF**, being a greedy algorithm, requires at most $2\Delta - 1$ colors to color $G(\Delta)$. It follows that there exists a sequence of choices for which **HDF** uses exactly $2\Delta - 1$ colors to color $G(\Delta)$. □

3.2 A tight bound for HCDF

We now show that **HCDF**'s bound given by Lemma 3.1 is also tight, by modifying the construction used for **HDF**.

Theorem 3.3 $\forall\, \Delta, B(\mathbf{HCDF}, \Delta) = 2\Delta - 1.$

Proof. Construct $G(\Delta)$ as described for Theorem 3.2. We will construct a new tree $T(\Delta)$ by modifying $G(\Delta)$ so that **HCDF** will be able to make the same sequence of choices to color $T(\Delta)$ as made by **HDF** in coloring $G(\Delta)$. Initially set $T(\Delta) = G(\Delta)$.

We call an edge *critical* in $G(\Delta)$ or $T(\Delta)$ if the combined degree of its endpoints is maximal. Clearly, before any edges have been colored, a critical edge in $T(\Delta)$ has combined endpoint degree 2Δ. As before, we need to ensure that the critical edges in $F_{i,\Delta}$ are such that i colors are consumed before all of them are colored. Now the root of $F_{i,\Delta}$ has i children and one parent; thus for all i, $1 \leq i < \Delta - 1$, the edges connecting the root of $F_{i,\Delta}$ to its children are not critical. To make each such edge critical, for all i, $1 \leq i < \Delta - 1$, we add $\Delta - i - 1$ children to every root of $F_{i,\Delta}$. Thus we add $\Delta - i - 1$ vertices, each connected by one edge to the root of $F_{i,\Delta}$.

Now the same sequence of choices used by **HDF** to color the root of $F_{i,\Delta}$ can be used by **HCDF**. Thus it takes $\Delta - 1$ colors before the root of $T(\Delta)$ is colored, and an additional Δ colors to color all the edges incident to the root. Since **HCDF** is greedy, all other edges can be colored using $2\Delta - 1$ colors. $\square$

4 The constrained edge-coloring problem

We now consider the following *constrained* edge-coloring problem: Find a minimum edge-coloring of a bipartite graph where no more than k edges have the same color. Clearly, a minimum constrained edge-coloring requires max(Δ, $\lceil m/k \rceil$) colors, where m is the number of edges. Bongiovanni et al [3] presented an $O(m^2 n + mn^2)$ time algorithm for minimum constrained edge-coloring of bipartite graphs, where n is the number of vertices. Algorithms taking time $O(mn^{0.5} \log n)$ and $O((m+n)n \log m))$ have subsequently been developed [14, 10, 11].

The approximation algorithm Modified-**HDF** (**MHDF**) for edge-coloring bipartite graphs when a color may be used to color at most $k < n$ edges consists of invoking the greedy algorithm for **HDF** with input k where $1 \leq k <$ min$(|A|, |B|)$.

Lemma 4.1 MHDF *produces a coloring using at most* $\lfloor m/k \rfloor + (2\Delta - 1)$ *colors for a graph of* n *vertices,* m *edges, and degree* Δ*, if at most* $k \leq n$ *edges may be colored with a single color.*

Proof. Suppose **HDF** was used on the graph instead of **MHDF**. In the worst case it uses $2\Delta - 1$ colors, with color i coloring m_i edges, and $\sum_{i=1}^{2\Delta-1} m_i = m$. If **MHDF** uses the same sequence of choices, coloring m_i edges may require up to $\lfloor m_i/k \rfloor + 1$ colors. Thus **MHDF** may require at most $\sum_{i=1}^{2\Delta-1}(\lfloor m_i/k \rfloor + 1) = (2\Delta - 1) + \sum_{i=1}^{2\Delta-1} \lfloor m_i/k \rfloor$ colors. Clearly, $\sum_{i=1}^{2\Delta-1} \lfloor m_i/k \rfloor \leq \lfloor m/k \rfloor$, so the result follows. $\square$

Note that this bound is not tight for the class of graphs with $k = n$ and $k = 1$; in the former case, **MHDF** is simply **HDF**, so at most $2\Delta - 1$ colors are needed, and in the latter case, exactly $m/k = m$ colors are needed.

Finally, we call the algorithm **HCDF** invoked with $k <$ min$(|A|, |B|)$ the Modified-HCDF algorithm, **MHCDF**. The proof of Lemma 4.1 can be used to show that, like **MHDF**, algorithm **MHCDF** produces a coloring using at most $\lfloor m/k \rfloor + (2\Delta - 1)$ colors.

5 Discussion

We observe that our constructive proofs for obtaining tight bounds involve constructing trees for which the degree is much less than the number of vertices. Let $N(\Delta)$ be the number of vertices in the graph $G(\Delta)$ of Theorem 3.2, and $M(i,\Delta)$ the number of vertices in graph $F_{i,\Delta}$. Then, by the definitions we have $N(\Delta) = 1 + \Delta\ M(\Delta-1,\Delta)$ and $M(i,\Delta) = 1\ + i +\ (\Delta-1)\ \sum_{j=0}^{i-1} M(j,\Delta)$. A simple calculation shows that $N(\Delta) = O(\Delta^{\Delta})$. In contrast, showing that $B(\mathbf{FCFS},\Delta) = 2\Delta - 1$ involves constructing graphs for which the number of vertices is $O(4^{\Delta})$ [1].

This observation may support the intuition that the approximation algorithms which are greedy but not arbitrary in their examination of vertices and edges are likely to provide minimum or near-minimum colorings for a larger range of graphs than the arbitrary greedy algorithm.

We are currently investigating the behavior of various greedy approximation algorithms in terms of the probability that they use the number of colors given by their bounds. We are also investigating the problem of edge-coloring the graphs given that certain edges must receive the same color, and of developing distributed edge-coloring algorithms [5].

Acknowledgements. We thank Bill Aiello of Bellcore as well as an anonymous referee for helpful comments.

References

[1] A. Bar-Noy, R. Motwani, and J. Naor. The greedy algorithm is optimal for on-line edge coloring. *Inf. Proc. Lett.*, pages 251–253, Dec. 1992.

[2] Claude Berge. *Graphs.* North-Holland, 1985.

[3] G. Bongiovanni, D. Coppersmith, and C. K. Wong. An optimum time slot assignment algorithm for an SS/TDMA system with variable number of transponders. *IEEE Trans. Comm.*, 29(5):721–726, May 1981.

[4] R. Cole and J. Hopcroft. On edge coloring bipartite graphs. *SIAM J. Comput.*, 11(3):540–546, 1982.

[5] Dannie Durand, Ravi Jain, and David Tseytlin. Distributed scheduling algorithms to improve the performance of parallel data transfers. Technical Report 94-38, DIMACS, July 1994.

[6] H. Gabow. Using euler partitions to edge color bipartite multigraphs. *Intl. J. Computer and Inf. Sci.*, 5:345–355, 1976.

[7] H. Gabow and O. Kariv. Algorithms for edge coloring bipartite multigraphs. *ACM Symp. Th. of Comp.*, pages 184–192, 1978.

[8] H. Gabow and O. Kariv. Algorithms for edge coloring bipartite graphs and multigraphs. *SIAM J. Comput.*, 11(1):117–129, 1982.

[9] D. S. Hochbaum, T. Nishizeki, and D. B. Shmoys. A better than "best possible" algorithm to edge color multigraphs. *SIAM J. Comput.*, 7:79–104, 1986.

[10] Ravi Jain. *Scheduling data transfers in parallel computers and communications systems.* PhD thesis, Univ. Texas at Austin, Dept. of Comp. Sci., 1992. Available as Tech. Rept. TR 93-03.

[11] Ravi Jain, Kiran Somalwar, John Werth, and J. C. Browne. Scheduling parallel I/O operations in multiple-bus systems. *J. Par. and Distrib. Comp.*, Dec. 1992. Special Issue on Scheduling and Load Balancing.

[12] Ravi Jain, Kiran Somalwar, John Werth, and J. C. Browne. Scheduling parallel I/O operations. In *Proc. Workshop on I/O in Par. Comp. Sys.*, Apr. 1993.

[13] Ravi Jain, Kiran Somalwar, John Werth, and J. C. Browne. Heuristics for scheduling parallel I/O operations. 1994. Submitted for publication.

[14] Kiran Somalwar. Data transfer scheduling. Technical Report TR-88-31, Univ. Texas at Austin, Dept. of Comp. Sci., 1988.