Analysis of approximate algorithms for edge-coloring bipartite graphs★

Ravi Jain

Bellcore, 445 South St, Morristown, NJ 07960

John Werth

Dept of Computer Sciences, University of Texas at Austin, Austin, TX 78712

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 present results on the constrained edge-coloring problem where each color may be used to color at most k edges.

Keywords: Analysis of algorithms, edge coloring, approximation algorithms, scheduling, parallel data transfers, parallel I/O, parallel processing.

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 [5]; 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, it is well known that a minimum edge-coloring requires Δ colors and can be obtained in polynomial time [2].

Scheduling applications, such as the scheduling of parallel I/O operations, motivate the development of faster algorithms for approximate edge-coloring

★ This research was supported by the IBM Corporation through grant 61653 and by the State of Texas through TATP Project 003658-237.

of bipartite graphs [4,10,5]. 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 [10,7]. However, previous studies [10,1] 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 the following notation.

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 present in sec. 3 worst-case bounds on 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. In sec. 4 we show these bounds are tight. In sec. 5 we briefly consider the *constrained* edge-coloring problem, where no more than k edges can have the same color, and end with some 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 $G = (A, B, E)$, Constraint k, $0 < k \leq \min(|A|, |B|)$
Output: An edge-coloring of G, $color : E \rightarrow \{1, 2, ...\}$

$F := Order(A, B, E)$;
/* F is some ordered sequence of all the edges in E */

```
i := 1;
while F ≠ ⟨ ⟩{
  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 ∪ {e};
    }
    if |M| = k, break;
  }
  i := i + 1;
  F := Order(A, B, E); /* Re-order the remaining edges */
}
```

The approximation algorithms[1] we investigate are defined by the function used for $Order()$.

(i) First-Come First-Served, **FCFS**. $Order()$ is the identity function.
(ii) Highest Degree First, **HDF**. $Order()$ sorts the vertices by descending degree, and for each vertex in turn, orders edges incident upon it arbitrarily. Ties between vertices are broken arbitrarily.
(iii) 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.

3 Bounds on greedy edge-coloring algorithms

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 1 *[5,1] 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 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 $Order()$ [10], so that **FCFS**, **HDF** and **HCDF** all run in time $O(m\Delta)$. Note that Lemma 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

[1] Note that the greedy algorithm above examines the list of edges. In [8] we have also considered a greedy algorithm which examines the list of vertices.

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 can show that **HDF** and **HCDF** can perform significantly better than **FCFS**.

Lemma 2 $\forall\, \Delta >$ *1:* **HCDF**$(G'(\Delta)) = \Delta \wedge$ **HDF**$(G'(\Delta)) = \Delta + 1 \wedge$ **FCFS**$(G'(\Delta)) = 2\Delta - 1$.

We have also found experimentally that **HDF** and **HCDF** can perform substantially better than **FCFS** when presented with graphs generated pseudo-randomly [7]. Further, in our experiments we found that in no case do they perform any worse; in Theorem 5 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 *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 $d(u) \leq d(w)$.*

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 $d(u) + d(v) \leq d(u) + d(w)$, i.e., $d(u) \leq d(w)$. □

Notation. We partition E into subsets by degree, i.e., let $E(i) = \{(u, v) : \max(d(u), d(v)) = i\}$, for $1 \leq i \leq \Delta$. Similarly, let $V(i) = \{u : d(u) = i\}$, for $1 \leq i \leq \Delta$. (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 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. The proof is by induction on the maximum degree of vertices in E'.

base. ($i = \Delta$). Since $E'(\Delta) \subseteq E(\Delta)$ and $E'(\Delta)$ is a matching, **HDF** can color all the edges in $E'(\Delta)$. We show that, having done so, **HDF** cannot color any other edge $(u, v) \in E(\Delta) - E'(\Delta)$. Two cases arise: either both u and v have been colored, or at least one of them has not. The first case holds iff there exist some $(t, u) \in E'(\Delta)$ and $(v, w) \in E'(\Delta)$, in which case **HDF** cannot color (u,v). Suppose the second case holds, and say $u \in V(\Delta) - V'(\Delta)$. Then from Lemma 3, $\forall\ (u,v) \in E,\ \exists\ (v,w) \in E : d(u) \leq d(w)$, i.e., $v \in V'(\Delta)$ and $(v,w) \in E'(\Delta)$. Therefore, having colored $E'(\Delta)$, **HDF** cannot color any vertex in $V(\Delta) - V'(\Delta)$, and hence any edge in $E(\Delta) - E'(\Delta)$.

hyp. The set of edges colored by **HDF** on the first iteration includes $E'(\Delta)$ $\cup\ E'(\Delta - 1)\ \cup ... \cup E'(i+1)$.

ind. (i). As for the base case, let **HDF** color all the edges in $E'(i)$. Consider some $(u,v) \in E(\Delta) - E'(\Delta)$. As before, if both u and v have been colored then **HDF** cannot color (u,v). Otherwise, say $u \in V(\Delta) - V'(\Delta)$. By Lemma 3 this occurs iff $\forall\ (u,v) \in E,\ \exists\ (v,w) \in E$ such that for some $j \geq i$, $(v,w) \in E'(j)$. If $j > i$ then by the induction hypothesis **HDF** has colored (v,w) and cannot color (u,v). If $j = i$, then **HDF** cannot color (u,v) having colored $E'(i)$. □

Theorem 5 $\forall\ G,$ **HCDF**$(G) \leq$ **HDF**$(G) \leq$ **FCFS**(G).

Proof. Clearly, **HCDF**$(G)\ \leq$ **FCFS**(G) and **HDF**$(G)\ \leq$ **FCFS**(G). To show **HCDF**(G) $\leq$ **HDF**(G), we apply Lemma 4 repeatedly (see [9]). □

From the foregoing we might expect that the bound given by Lemma 1 could be tightened further for **HDF** and **HCDF**. We will show that this is not the case.

4 Tight bounds on greedy algorithms

In this section we show that for any Δ a bipartite graph $G(\Delta)$ can be constructed such that **HDF**$(G) = 2\Delta - 1$.

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

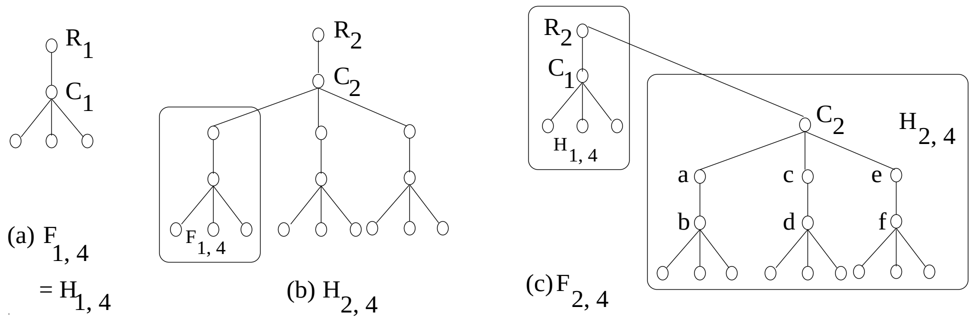

Fig. 1. Example construction to show **HDF** takes up to $2\Delta - 1$ colors to color a graph of degree Δ

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 [].

(i) $F_{0,\Delta}$ consists of a single vertex, S.
(ii) $H_{1,\Delta} = R_1\langle C_1\langle F_{0,\Delta}[\Delta - 1]\rangle$
(iii) $F_{1,\Delta} = H_{1,\Delta}$
(iv) For $1 < i < \Delta$, $H_{i,\Delta} = R_i\langle C_i\ \langle F_{i-1,\Delta}[\Delta - 1]\rangle\rangle$
(v) 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 6 *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* $\{1, ..., i\}$*.*

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. $\square$

Theorem 7 $\forall\, \Delta, B(\mathbf{HDF}, \Delta) = 2\Delta - 1$.

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 6 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 $G(\Delta)$. The theorem follows. $\square$

Theorem 8 $\forall\, \Delta, B(\mathbf{HCDF}, \Delta) = 2\Delta - 1$.

Proof. By extension of the construction above; see [9]. $\square$

5 Discussion

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. This constraint arises frequently in data transfer scheduling applications as a limitation in the capacity of the data channel [5]. Clearly, a minimum constrained edge-coloring requires $\max(\Delta, \lceil m/k \rceil)$ colors, where m is the number of edges. For a survey of optimal algorithms for this problem, see [6].

The approximation algorithms Modified-**HDF** (**MHDF**) and Modified-HCDF (**MHCDF**) consist respectively of the greedy algorithm for **HDF** and **HCDF** with input $k < \min(|A|, |B|)$.

Lemma 9 **MHDF** *and* **MHCDF** *produce 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. See [9]. □

We observe that the graphs $G(\Delta)$ constructed in our proof for obtaining tight bounds for the unconstrained problem in Theorem 7 have degree Δ much less than the number of vertices, $N(\Delta)$. It can be shown [9] that $N(\Delta) = O(\Delta^\Delta)$. In contrast, the graphs $G'(\Delta)$ constructed for showing $B(\mathbf{FCFS}, \Delta) = 2\Delta - 1$ have $O(4^\Delta)$ vertices [1].

We are currently investigating the problem of edge-coloring the graphs given that certain edges must receive the same color, and developing distributed edge-coloring algorithms [3].

Acknowledgement. We thank Bill Aiello of Bellcore and the 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] Dannie Durand, Ravi Jain, and David Tseytlin. Distributed scheduling algorithms to improve the performance of parallel data transfers. Technical Report 94-38, DIMACS, Rutgers Univ., July 1994.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

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

[9] Ravi Jain and John Werth. Analysis of approximate algorithms for constrained and unconstrained edge-coloring of bipartite graphs. Technical report 95-01, DIMACS, Rutgers Univ., Jan. 1995.

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