A Greedy Approach to Replicated Content Placement Using Graph Coloring

Bong-Jun Ko and Dan Rubenstein

Department of Electrical Engineering
Columbia University
New York, NY
bk384@columbia.edu,danr@ee.columbia.edu

ABSTRACT

Connectivity within ad-hoc and peer-to-peer networks undergoes constant change. One approach to reducing the cost of finding information within these networks is to replicate the information among multiple points within the network. A desirable replication approach should cache copies of all pieces of information as close to each node as possible without exceeding the storage resources of the nodes within the network. In addition, the approach should require minimum communication overhead among participating nodes and should adjust the locations of stored content as connectivity within the network changes. Here, we formulate this caching problem as a graph coloring problem, where the color of the node determines the content that the node should store. We present a distributed algorithm where each node chooses its color in a greedy manner, minimizing its own distance to the color furthest from it. We demonstrate convergence of this algorithm and evaluate its performance in the context of its ability to place information near all nodes in the network.

1. INTRODUCTION

Often, it is not possible for every user to communicate directly with every other user. In the wireless ad hoc environment, this may be due to limitations in the distances that can be reached by a device's transmissions. In peer-to-peer (P2P) wired environments, there may simply be too many participants for an individual user to track. Wireless ad hoc and wired P2P networks facilitate communication among sets of end-users. To enable communication between all pairs of users, participating users form network *overlays*: a virtual networking substrate in which the users are the nodes of the network and the direct communication paths between pairs of users (on top of IP or via wireless transmission) are the virtual network links. The participants in these overlays perform routing functions on the behalf of their fellow participants, providing communication paths that carry information from a source to its desired destination. In addition, popular information that is generated at a small number of participants can be replicated among various users (nodes) inside the overlay. By doing so, the distance that a search for this content must travel within the ad hoc or P2P network is significantly reduced, as is the load placed on the originating server of the popular content.

In this paper, we explore the problem of placing copies of content to reduce the distance that a query must travel within the network to retrieve the copy. We assume that there is a large enough body of content such that it would not be feasible to expect every node in the network to store every content object. Therefore, each node is somehow apportioned a subset of the content objects. It is desirable to perform the assignment of objects to nodes so that every node can "easily" reach every object. We assume that a path that proceeds through a fewer number of hops (intermediate participants) is "easier" than one that proceeds through more hops.

In a distributed overlay, there are two challenges in performing such placement. The first challenge is to construct a placement strategy that puts copies of all objects near to each node. The second challenge is to design

This research was supported in part by NSF grants ANI-0117738 and CAREER Award No. ANI-0133829. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

algorithms that can implement the placement strategy in a distributed fashion with minimal communication overhead (i.e., the passing of messages between nodes). We present preliminary work here that addresses both of these concerns. Our idea is to divide nodes into k classes, and to divide the objects among the k classes. Each node is then assigned to one of the classes. To reach any object, a node must simply reach another node that was assigned to the class that contains the desired object.

By thinking of each class as a distinct *color*, the above mapping induces a coloring upon a graph whose node/edge structure mimics the node/link structure of the overlay network, and where a node's color indicates the class of objects that it stores. Our goal is to construct a coloring for graphs that allows each node to reach each color in a minimal number of hops, and to create a distributed algorithm that realizes this coloring. We assume each node wishes to minimize the distance to the color furthest from it, but is only permitted to change its own color to do so. We present a simple greedy, distributed algorithm that operates under these assumptions and prove the following results about the algorithm for a graph of more than k nodes that is colored with k colors:

- We prove that our distributed, greedy algorithm always converges to a particular coloring. In other words, a coloring will be reached where no node benefits any further from changing its own color.
- We prove that in this final coloring, any node can reach any color in the graph by traversing fewer than k hops.

In addition to these theoretical results, we evaluate the algorithm via simulation by comparing the maximum distance from a node to a color achieved by the algorithm to what this distance would be if the entire graph were colored simply to minimize the distance for this particular node. We find that the algorithm converges quickly, and on average comes within 1.5 times the distance of this bound.

The remainder of the paper proceeds as follows. We begin by discussing related work in Section 2. In Section 3, we present a more formal description of the distributed, greedy coloring algorithm. Section 4 presents theoretical results that prove convergence of the algorithm and bound the maximal distance from each node to nodes of all colors achieved by the algorithm. Section 5 presents simulation results that quantify expected convergence times and distances to colors within the final coloring. Section 6 discusses open issues and future directions, and Section 7 concludes the paper.

2. RELATED WORK

Several recent works have explored the topic of content replication in mobile ad hoc networks. Reference 1 provides a decentralized and dynamic mechanism where each node decides how many replicas of each file should be created and where these replicas should be placed. This mechanism relies on a resource discovery service that is used by each node to locate available storage and network resources throughout the entire overlay. Reference 2 proposes a scheme that increases data availability in mobile ad hoc networks when the original data is inaccessible due to network partitioning. It predicts group partitioning based on nodes' mobility and replicates data to the other partition before partitioning occurs. Reference 3 describes the 7DS system which provides mobile users with a means to share and cache their contents via either a pull-based or push-based sharing scheme, or a combination of the two. It utilizes periodic broadcasting of queries and advertisements, but does attempt to optimize the location of these replicas within the network other than through simple cache replacement policies.

Another topic of recent interest that relates to our work here is the placement of caches or replicas inside a content distribution network. Reference 4 studies the optimal web server replication problem incorporating clients' request (read) workload, and proposes centralized heuristic algorithms to approximate this NP-hard problem. Reference 5 examines optimal replica placement to minimize the workload caused not only by searching and retrieving information but also by updates of the contents at the nodes. Both of these works consider the problem in the context of placing only a single object. Reference 6 proposes a centralized, greedy heuristic algorithm that replicates objects in an attempt to minimize the average distance client queries must traverse to

obtain objects as a function of the object's popularity. The work also shows that the complexity of an algorithm that minimizes this average is NP-complete.

The problem of resource location and discovery in mobile ad hoc networks is explored in Reference 7, which presents a distributed location service that can connect two mobile nodes that wish to communicate. It employs geographic forwarding and predefined ordering of node identifiers and geographic hierarchy to update and find a node serving as a location server for the node looked for. However, the goal is to provide a forwarding mechanism to reach resources, and not to move resources near the querying location. Hence, the work makes no attempt to place replicas near the node that initiates the search.

In both References 8 and 9, the locations of the resources (or nodes) are registered in and maintained by a set of location (directory) servers. These nodes exchange registration and query messages, and each server responds to queries from clients in its nearby domain. These hierarchical schemes relying on some special nodes are different from our coloring scheme where all nodes have flat, peer-to-peer relationship with one another.

While this work does not explicitly focus on a comparison of our algorithm to algorithms that approximate optimal solutions to facility location problems,[10] there are some striking similarities between the two objectives. Reference[10] and more recent work[11–14] has produced polynomial-time, constant-factor approximation algorithms for this class of problems. In addition, some earlier work exploring facility location uses a "local search" approach in which a placement scheme is modified one location at a time in a search for a good approximation for the optimal solution.[15] Also, Reference 16 employs a greedy approach to further improve the performance of the heuristic proposed in Reference 10. Surprisingly, we are unable to find work in the area of facility location that considers distributed algorithms to solve these problems. We plan to evaluate whether our coloring approach can be of some use in this regard.

3. ALGORITHM DESCRIPTION

In this section, we present a formal description of the distributed algorithm, named `GreedyMinColor`, that we evaluate throughout the rest of the paper. The algorithm assumes that each node has an accurate snapshot of the current coloring of the graph. We also provide a simple distributed solution for distributing the current color information of the nodes throughout the graph. Last, we describe the metric we will use to evaluate the performance of the coloring algorithm.

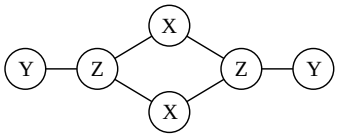
We view the overlay network as a connected graph, G, containing n nodes, where each node represents an overlay participant that is to be colored one of k colors. Each color maps to a set of objects that should be stored by nodes of that color. We assume in this section that the value of k is determined in advance and remains fixed for the duration of the execution of the algorithm. Execution of `GreedyMinColor` changes the color of nodes within the graph. Each time a node in the graph changes its color, we say that the graph has a new *coloring*. We define C_ℓ to be the ℓth coloring of the graph G (i.e., ℓ equals the sum over all nodes of number of times each node changes its color), with $C_\ell(j)$ being the color of node j in the ℓth coloring. We assume that simultaneous changes in color by different nodes are sequentialized in an arbitrary manner, such that only one node is colored differently between any pair of adjacent colorings C_ℓ and $C_{\ell+1}$.

We define a *hop* to be the traversal of an edge from one node in the overlay to another, and $d_{i,j}(\ell)$ to be the minimum number of hops from node j to color i within the ℓth coloring, i.e.,

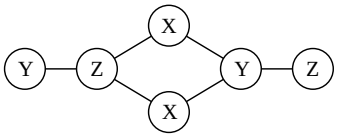
$$d_{i,j}(\ell) \quad = \quad \min_{1 \le m \le n} \{D(j,m) : C_\ell(m) = i\}$$

where $D(j,m)$ is the minimum (hop) distance in the graph between nodes j and m. If no node is assigned a particular color i, then $d_{i,j}(\ell) = \infty$ for all nodes j under coloring C_ℓ. We also, define $\delta_\ell(m)$ be the minimum number of hops from node m to a node of the same color in coloring C_ℓ. For the remainder of the paper, we define the *distance* between two nodes to be the minimum number of hops that must be traversed to reach one of the nodes from the other. We say node j is *further* from node j' than it is from j'' if the distance from j to j' is larger than the distance from j to j''.

We now describe the distributed algorithm, `GreedyMinColor`. We assume that each node j obtains accurate values values of $d_{i,j}(\ell)$ for all colors i before changing its color (a preliminary method to implement the



(a) non-optimal



(b) optimal

Figure 1. Two non-isomorphic stable colorings.

distribution of this information is discussed in subsection 3.1), and that only one node changes its color at a time. Node j computes $d = \delta_\ell(j)$, the distance to the next closest node that is assigned the same color. If $d < \max_{1 \le i \le k} d_{i,j}(\ell)$, then j changes its color to a color y that satisfies $d_{y,j}(\ell) = \max_{1 \le i \le k} d_{i,j}(\ell)$. In other words, j changes to color y only if prior to its change in color, no color is further from j than y, and y is further from j than its current color $C_\ell(j)$. If, by applying `GreedyMinColor`, no node changes its color from coloring C_ℓ, then we say that the algorithm *converges* to a coloring and we call this final coloring a *stable* coloring. The initial coloring can be chosen arbitrarily. For instance, all nodes can choose to be color 1 when they first start, or can choose their color at random from the set of k colors. We note that there is not a unique stable coloring, even under isomorphic mappings. For instance, Figure 1 presents two stable colorings (the reason why Figure 1(a) is called non-optimal and Figure 1(b) is called optimal will be explained later in this section).

3.1. Accurate Estimates of Distances to Colors

The above description of the Algorithm assumes that each node m, when considering changing its color, knows the values of $\delta_\ell(m)$ and $d_{m,j}(\ell)$ for all colors j. One simple protocol that allows each node to obtain these values is to have each node broadcast its best known distance to all colors to each of its neighboring nodes. This distance is 0 for the color of the node, and is one plus the minimum value reported by any neighbor for any other color. A node only retransmits distance information to a neighbor when the distance is lower than what was previously reported. This simple algorithm does not take into account increases in distance to a color i when a node in the graph changes from color i to another color. For the purposes of this paper, we assume that, before each new coloring, all nodes discard the previous distance information and rerun this algorithm to recompute the distances under the current coloring.

Clearly, such an algorithm has higher communication overhead than is necessary and requires a certain amount of synchronization among the distributed nodes to allow them all to perform this computation prior to the next color change. Determining a better algorithm for distributing this information is one area for future work.

3.2. Measuring Algorithm `GreedyMinColor` Performance

There are several variations of objective functions whose minimization could be a target of networks that seek to reduce distance to nodes of various colors. For example, one could argue that a "best" coloring C_ℓ is one that minimizes one of the following:

- $\sum_{1 \le i \le k} \sum_{1 \le j \le n} d_{i,j}(\ell)$
- $\max_{1 \le i \le k} \max_{1 \le j \le n} d_{i,j}(\ell)$
- $\sum_{1 \le i \le k} \max_{1 \le j \le n} d_{i,j}(\ell)$
- $\max_{1 \le i \le k} \sum_{1 \le j \le n} d_{i,j}(\ell)$

Applying the sum within the objective function is useful when evaluating directed searches, where the location of the color being sought is known, such that the cost of a node reaching a given color is proportional to the distance to that color. Applying maxima within the objective function is useful when evaluating undirected searches, where the location of the color being sought is unknown such that the search must take random paths, but needs only go as far as the maximum distance required to reach all colors.

Not only are these objective functions likely to lie in the class of NP-hard problems (given their relation to the k-median problem), but they also fail to capture the "sacrifice" that each node makes on the behalf of the entire graph of nodes. To more directly evaluate this "sacrifice" for an algorithm, A, we compare each node's distance to the color furthest from it using algorithm A to the minimum such distance over the set of all possible colorings. This measure, which we call the *per-node-best coloring (pnbc)* is calculated by allowing each node to independently color its nearest neighbors so as to minimize the maximum distance to a color. We then average these minimized distances over all nodes. More formally, let $h_i(j)$ be the number of hops that node j must be allowed to traverse in order to reach $i-1$ neighbors (and up to i colors, including its own color). Clearly, the minimum distance for which a coloring exists where j can reach all colors is $h_i(j)$. For instance, if j connects directly to $k-1$ other neighbors, then $h_k(j)=1$. Our measure of a coloring, C_ℓ is

$$pnbc(\ell) \quad = \quad \sum_{1\leq j\leq n} \frac{\max_{1\leq i\leq k} d_{i,j}(\ell)}{nh_k(j)}.$$

This measure cannot be greater than one, as $\max_{1\leq i\leq k} d_{i,j}(\ell) \geq h_k(j)$. However, there are topologies for which no coloring exists for which the measure equals one. A simple example is four nodes connected in a cycle where the graph is to be colored with three colors. Clearly, $h_3(j)=1$ for all four nodes j, since each node itself would be colored one color and its immediate neighbors would respectively be colored the remaining two colors. However, for any coloring C_ℓ, at least two nodes in the graph will have $h_3(j)=2$, such that the lowest possible value of $pnbc(\ell)$ is $3/2$.

Last, we note that Algorithm `GreedyMinColor` does not always converge to a coloring that minimizes $pnbc(\ell)$. For instance, Figure 1(a) presents a graph in which the coloring ℓ is stable and $pnbc(\ell)=4/3$. However, Figure 1(b) presents the same graph on the right, but with a different coloring, where the coloring ℓ' is stable but where $pnbc(\ell')=1$.

4. CONVERGENCE AND BOUNDS ON COLOR DISTANCE

In this section, we prove that (as long as the graph contains at least k nodes where k is the number of colors being used to color the graph) Algorithm `GreedyMinColor` converges to a stable coloring, and that the maximum distance from any node to a node of a fixed color is $k-1$. We begin by proving that if the algorithm converges, then the stable coloring requires no node to travel further than distance $k-1$ to reach all colors, and we then show that `GreedyMinColor` converges to a stable coloring.

LEMMA 1. *If Algorithm* `GreedyMinColor` *converges to a stable coloring C_ℓ, then for all nodes m, $d_{i,m}(\ell) < k$ for all colors i.*

Proof. We prove this result by contradiction. Assume `GreedyMinColor` converges to a stable coloring C_ℓ where $d = d_{y,m}(\ell) \geq k$ for some node m and some color y. Clearly, m is some color $x \neq y$, since $d_{x,m}(\ell)=0$ when $C_\ell(m)=x$. Since the coloring is stable, $\delta_\ell(m) \geq d$ or else m would change its color (e.g., from x to y), contradicting stability of the coloring. Consider a shortest path from m to a node m' where m' is a node of color y of distance d from m (i.e., a "closest" node to m of color y). The intermediate nodes on this path cannot be colored x or y or else these colors would lie less than distance d from m. Since this path contains $d-1 \geq k-1$ nodes between m and m' whose colors are drawn from the remaining $k-2$ colors, at least two nodes on the path are the same color, z. To assist the reader, this path is depicted in Figure 2. The colored circle indicates the distance around node m where no other node can be colored x or y.

Let m'' be the node colored z that is closest to m along the path to m' and let d_z be its distance from m. Clearly, m'' can be no closer than $d-d_z$ to a node colored y (or else m would be closer than d to a node of color y), so $d_{y,m''}(\ell) = d-d_z$. Furthermore, since there is another node of color z on the path of length $d-d_z$

from m'' to m', $\delta_\ell(m'') < d - d_z$. Thus, $\delta_\ell(m'') < d - d_z = d_{y,m''}(\ell) < d$. This contradicts the stability of C_ℓ, since m'' is further from color y than it is from another node of color z. Hence, application of Algorithm `GreedyMinColor` by m'' would change its color. □

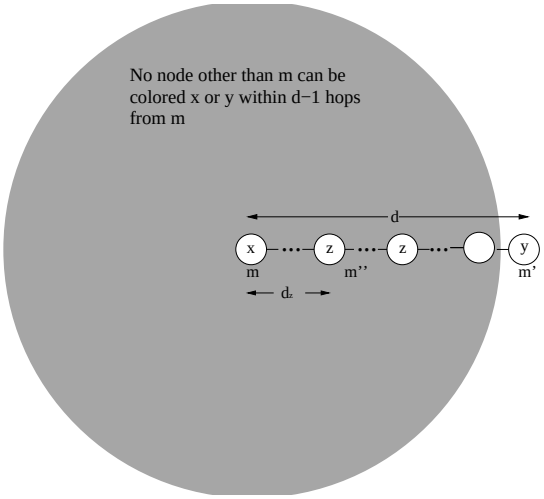


Figure 2. Counterexample of stable coloring for Lemma 1. There are no nodes colored x or y between m and m', but there are two nodes colored z. Hence, m'' would change its color from z.

4.1. Demonstrating Convergence

We now explore convergence properties of `GreedyMinColor`. Such a distributed algorithm is most useful if it eventually yields a stable coloring.

Let V_ℓ be an (increasing) ordered n-component vector, $(v_1, v_2, \cdots, v_n)$ with $v_i \leq v_j$ for $i < j$ that is formed by ordering the members of $\{\delta_\ell(m) : 1 \leq m \leq n\}$ in increasing order. Define the lexicographic ordering relation "$<$" on the set vectors for different colorings $V_\ell = (v_1, v_2, \cdots, v_n)$ and $V_{\ell'} = (w_1, w_2, \cdots, w_n)$ such that $V_\ell < V_{\ell'}$ whenever there exists an i (possibly $i = 1$) where $v_i < w_i$ and $v_j = w_j$ for all $0 < j < i$.

LEMMA 2. *For any coloring C_ℓ, if a node changes color by applying Algorithm* `GreedyMinColor` *to produce coloring $C_{\ell+1}$, then $V_\ell < V_{\ell+1}$.*

Proof. Let m be the node that, using Algorithm `GreedyMinColor`, changes from color x to color y, yielding coloring $C_{\ell+1}$ from coloring C_ℓ. Letting $d = \delta_{\ell+1}(m)$, we have that $\delta_\ell(m) < d$, or else m would not have changed color. Furthermore, for any node m', if $\delta_{\ell+1}(m') < \delta_\ell(m')$, then m' and m must be the same color in coloring $\ell + 1$ with m being the closest node to m' that is of the same color (or else $\delta_{\ell+1}(m')$ would equal $\delta_\ell(m')$). Since $\delta_{\ell+1}(m) = d$, nodes m and m' must be at least distance d apart, such that $\delta_{\ell+1}(m') \geq d$.

Thus, for any node m', if $\delta_{\ell+1}(m') < \delta_\ell(m')$, then $\delta_{\ell+1}(m') \geq d$. This plus $\delta_{\ell+1}(m) = d > \delta_\ell(m)$ gives us that there are fewer components in V_ℓ that are less than or equal to d than there are in $V_{\ell+1}$. Also, since every component of V_ℓ that is less than d maps to a node j whose distance to a node of the same color can only decrease, we have that $V_\ell < V_{\ell+1}$. □

COROLLARY 1. *Algorithm* `GreedyMinColor` *converges.*

Proof. Each component of V_ℓ is either less than n or is infinite (for the case where the node is uniquely colored). Since $V_\ell < V_{\ell+1}$, there is a finite number of colorings $\ell + 1, \ell + 2, \cdots, \ell + \ell'$ that satisfy $V_\ell < V_{\ell+1} < \cdots < V_{\ell+\ell'}$. Thus, the sequence cannot continue indefinitely, so some coloring in the sequence must be stable. □

THEOREM 1. *When coloring a graph with k colors, Algorithm* `GreedyMinColor` *converges to a coloring C_ℓ where $d_{i,j}(\ell) < k$ for all colors i and all nodes j.*

Proof. The proof follows directly from Lemma 1 and Corollary 1. □

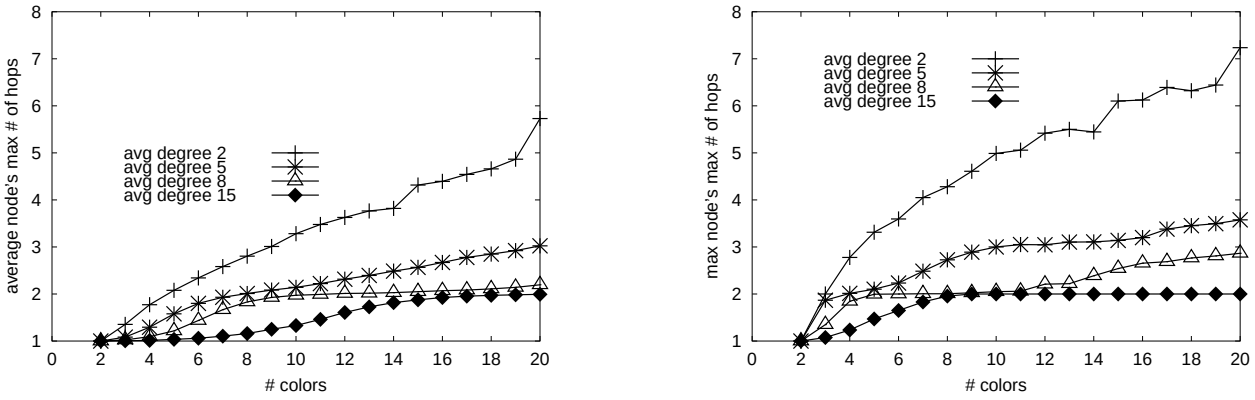
We note that the above bound is tight for the general class of graphs by considering the graph of k nodes connected as a chain. Clearly, each node must have a distinct color, placing the node at one end of the chain a distance of $k-1$ from the color of the node at the other end of the chain.

5. SIMULATION RESULTS

In this section, we present an evaluation of Algorithm `GreedyMinColor` via simulation. Each simulation run is performed upon a 20-node graph where we vary the number of colors used to color the graph and the set of edges that connect nodes within the graph. The number of edges to be added is determined by a *connectivity value*, a. We choose edges to add at random until the graph is fully connected and there are at least $10a$ edges. Note that if exactly $10a$ edges are added, then the average number of edges extending from a node is a. We define the *average degree* of a run, r_i, to be $a_i = E_i/10$, where E_i is the number of edges in the graph. Since the number of edges that are added to connect all nodes in the graph can be more than $10a$, the average degree for a graph generated with connectivity value a may be greater than a.

For each value of a, we randomly generate 3,600 different graphs, and perform one run on each graph for values of k ranging from 2 and 20. In each run, we cycle through the set of nodes repeatedly,* applying Algorithm `GreedyMinColor` to each node in the cycle, allowing it to change its color if doing so reduces its distance to its furthest color. We terminate the algorithm once we complete a cycle through the set of all nodes and no node changes color, since at that point we have a stable coloring.

Figures 3 and 4 depict results from our simulation runs. On the x-axis of each of these figures, we vary the number of colors used to color the 20-node graph. We group together results into the same curve whose floor of the average degrees are the same (i.e., results from runs r_i and r_j are averaged within the same curve when $\lfloor a_j \rfloor = \lfloor a_i \rfloor$).



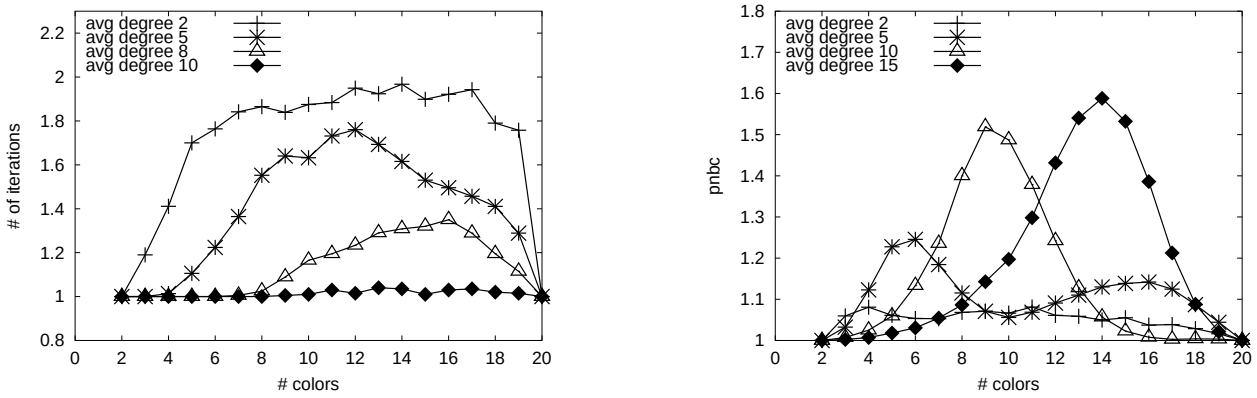
(a) Average over all nodes.

(b) Maximum over all nodes.

Figure 3. Maximum distance to a color.

Figure 3(a) depicts the distance, averaged over all nodes in the graph, to the closest node of the furthest color, i.e., $(\sum_{1 \le j \le n} \max_{1 \le i \le k} d_{i,j}(\ell))/n$. Figure 3(b) depicts the maximum of the maximum distances to the closest nodes of the furthest colors, i.e., $(\max_{1 \le j \le n} \max_{1 \le i \le k} d_{i,j}(\ell))$. Not surprisingly, these distances grow larger as the average degree reduces and as the number of colors increases. We see that even when the average degree is low, the average distance to the furthest color from a node is often much less than the upper bound of $k-1$, given by Theorem 1.

*The cycle that we take through the nodes does not necessarily exist within the graph. It is simply an arbitrary ordering of the nodes.



(a) iterations to convergence

(b) $pnbc()$

Figure 4. Average number of iterations and $pnbc(\ell)$ of the stable coloring.

Figure 4(a) plots the average number of iterations needed by a run to converge to a stable coloring. During each iteration, we cycle through the set of nodes in a fixed but arbitrarily chosen order, applying Algorithm `GreedyMinColor` to each node in each iteration. We only count those iterations in which at least one node changes its color. We see that the number of iterations is very small: on average fewer than two iterations are necessary to reach a stable coloring. As the average degree of a node increases, the number of iterations needed is reduced. Intuitively, this is because the set of colors that are far from a node from which a node might select its color shrinks quickly as the average degree grows. We also note that the number of iterations needed also reduces when the number of colors is very small or very large (i.e., where almost every node is assigned its own color). Last, as the average degree increases, the number of colors that requires the most iterations to reach a stable coloring increases.

Figure 4(b) plots the average $pnbc(\ell)$ for graphs with the same average degree and same number of colors. We see that Algorithm `GreedyMinColor` comes very close to achieving a distance to the furthest color per node that is close to what is optimal if the coloring were selected to minimize this distance for that node. For higher degrees, there is a range where $pnbc(\ell)$ rises to a value that is slightly higher than 1.5. Thus, our experimental results suggest that the distance to the furthest color from a node is on average less than double what can be achieved by optimizing the coloring for that particular node. We see that $pnbc(\ell)$ can grow larger for graphs with higher degrees of average connectivity, and that the respective peaks of $pnbc(\ell)$ occur roughly where the average connectivity matches the number of colors. One intuitive reason is that at this point, $h_k(j)$ is 1 for many nodes, but that the number of neighbors is low enough such that, when using Algorithm `GreedyMinColor`, a few neighbors will likely choose the same color forcing some $d_{i,j}(\ell) > 1$.

6. DISCUSSION AND FUTURE WORK

We have shown that the `GreedyMinColor` Algorithm converges to a coloring where all colors are "close" to a node and where nodes only change their colors a handful of times. However, we made several assumptions about the abilities of nodes to coordinate the times at which they change their colors, and their ability to correctly assess the minimum distances to the different colors when making a decision to change color. There are several directions for future work before these ideas can be deployed in practice. We point out several possible directions in this section.

Distributed methods for obtaining C_ℓ: In Section 3, we presented a simple distributed algorithm that required a large amount of communication between neighbors (up to $n\nu$ transmissions where n is the number of nodes in the graph and ν is the number of neighbors) between each change in coloring. This method also

required a fair amount of synchronization between nodes. One direction for future work is to modify this part of the distributed algorithm. We are currently exploring variants of the Bellman-Ford algorithm[17] for this purpose.

Shifting in the Underlying Topology: Our goal is to develop an algorithm that can be used in ad hoc and peer-to-peer networks where graph membership and connectivity can vary quickly in time. Hence, our distributed coloring algorithm should adapt quickly to small changes in topology. We hope that simple modifications to a Bellman-Ford type algorithm will also prove fruitful as a low-overhead means of adapting to such changes. The fact that on average a small number of iterations are needed within the entire graph to converge to a coloring is an indication that Algorithm `GreedyMinColor` will adapt quickly to these types of changes.

Race Conditions: Our proof of convergence assumes that at most one node changes its color at a time. We also assume that the information regarding the change in coloring is propagated to each node before that node runs Algorithm `GreedyMinColor`. The algorithm need not converge when multiple nodes change colors simultaneously. For instance, consider an arbitrary graph we wish to color with 2 colors in which all nodes are color 1. If all nodes were to simultaneously change color, they would all change to color 2, and then back to color 1, repeating this process indefinitely. We conjecture that there are simple mechanisms that can break the adverse effects of such synchronization, such that convergence might be slower, but that the coloring will eventually converge.

Selecting the number of colors: The amount of information that is stored at a node is inversely proportional to the number of colors. However, increasing the number of colors also increases the expected distance to reach an arbitrary color. We leave it as future work to both pose and solve an optimization problem that determines the appropriate number of colors that should be used. It may also be of interest to dynamically vary the number of colors used within the graph.

Varying density of an object's location with popularity: Our initial model assumes that the density of nodes that contain a particular object is a function only of the number of colors, and that the popularity of the object has no bearing on its placement. One approach to adjusting the coloring as a function of object popularity is to add weights to colors, such that a node's color choice is the one that minimizes the distance to a color times the color's weight. Another possibility is to map popular objects to several colors.

Moving Information During Color Changes: We have assumed that when a node changes color, the information that the node stores is adjusted to match its new color. Such an adjustment can entail a fair amount of transfer overhead. The node must retrieve the information associated with the new color, which, presumably is further from the node than the information associated with any other color. If node colorings change frequently, for instance due to joins, leaves, node movement, or as a result of race conditions, it will likely be useful to create a lag between the time the node changes color and the time it decides to receive the information associated with the color.

Additional Applications of `GreedyMinColor`: We have described using Algorithm `GreedyMinColor` to decide where information should be replicated within an overlay network. This information can be anything from arbitrary data files to information that describes configuration information of the overlay (e.g., where to locate printers). We believe that Algorithm `GreedyMinColor` can also provide a method for locally replicating information within structured search approaches for peer-to-peer networks such as CHORD[18] and CAN.[19]

7. CONCLUSION

We have presented a distributed algorithm to place content within an overlay network such that any piece of content can be reached within a small number of hops. We pose the placement problem as a graph coloring problem and prove two results. First, we prove that a synchronized version of the algorithm where one node changes its color at a time will converge. Second, we show that when the algorithm converges, the distance from a node to any color is less than the number of colors used to color the graph. Simulations demonstrate that the algorithm converges quickly, with each node typically undergoing fewer than 2 color changes. Also, the algorithm produces colorings of the graph that are on average less than 1.5 times greater than the distance from a node to the furthest color that can be achieved by coloring the graph to minimize this distance for that node.

There are numerous issues that still need to be explored, however, before the algorithm can be deployed in practical networking environments. In addition, distributed coloring algorithms of this type reveal some interesting computational complexity issues. We plan to explore these two directions as future work.

REFERENCES

1. K. Ranganathan, A. Iamnitchi, and I. Foster, "Improving data availability through dynamic model-driven replication in large peer-to-peer communities," in *Global and Peer-to-Peer Computing on Large Scale Distributed Systems Workshop*, May 2002.
2. K. Chen and K. Nahrstedt, "An integrated data lookup and replication scheme in mobile ad hoc networks," in *Proc. of SPIE International Symposium on the Convergence of Information Technologies and Communications (ITCom 2001)*, August 2001.
3. M. Papadopouli and H. Schulzrinne, "Seven degrees of separation in mobile ad hoc networks," in *Proc. of IEEE Global Telecommunications Conference 2000 (GLOBECOM '00)*, November 2000.
4. L. Qiu, V. Padmanabham, and G. Voelker, "On the placement of web server replicas," in *Proc. 20th IEEE INFOCOM 2001*, August 2001.
5. S. A. Cook, J. K. Pachl, and I. S. Pressman, "The optimal location of replicas in a network using a read-one-write-all policy," *Distributed Computing 15(1)* **7**, April 2002.
6. J. R. J. Kangasharju and K. Ross, "Object replication strategies in content distribution networks," in *Proceedings of WCW'01: Web Caching and Content Distribution Workshop, Berlin*, June 2001.
7. J. Li, J. Jannotti, D. De Couto, D. Karger, and R. Morris, "A scalable location service for geographic ad-hoc routing," in *Proceedings of the 6th ACM International Conference on Mobile Computing and Networking (Mobicom '00)*, August 2000.
8. Z. Haas and B. Liang, "Ad-hoc mobility management with uniform quorum systems," *IEEE/ACM Transactions on Networking* **7**, April 1999.
9. J. Liu, Q. Zhang, W. Zhu, J. Zhang, , and B. Li, "A novel framework for qos-aware resource discovery in mobile ad hoc networks," in *Proceedings of IEEE ICC'02*, April 2002.
10. D. B. Shmoys, E. Tardos, and K. Aardal, "Approximation algorithms for facility location problems (extended abstract)," in *ACM Symposium on Theory of Computing (STOC)*, pp. 265–274, (El Paso, TX), May 1997.
11. M. Charikar and S. Guha, "Improved Combinatorial Algorithms for the Facility Location and k-Median Problems," in *IEEE Symposium on Foundations of Computer Science (FOCS)*, pp. 378–388, 1999.
12. M. Charikar, S. Guha, E. Tardos, and D. B. Shmoys, "A constant-factor approximation algorithm for the k -median problem (extended abstract)," in *ACM Symposium on Theory of Computing (STOC)*, pp. 1–10, (Atlanta, GA), May 1999.
13. A. Goel, P. Indyk, and K. R. Varadarajan, "Reductions among high dimensional proximity problems," in *Symposium on Discrete Algorithms (SODA)*, pp. 769–778, (Washington DC), 2001.
14. K. Jain and V. V. Vazirani, "Primal-Dual Approximation Algorithms for Metric Facility Location and k-Median Problems," in *IEEE Symposium on Foundations of Computer Science (FOCS)*, pp. 2–13, 1999.
15. M. R. Korupolu, C. G. Plaxton, and R. Rajaraman, "Analysis of a local search heuristic for facility location problems," Tech. Rep. 98-30, University of Texas, Austin, 24, 1998.
16. Guha and Khuller, "Greedy strikes back: Improved facility location algorithms," in *Symposium on Discrete Algorithms (SODA)*, (San Francisco, CA), Jan. 1998.
17. T. Cormen, C. Leiserson, and R. Rivest, *Introduction into Algorithms*, MIT Press, Cambridge, MA, 1990.
18. I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan, "Chord: A Scalable Peer-To-Peer Lookup Service for Internet Applications," in *Proceedings of ACM SIGCOMM'01*, (San Diego, CA), August 2001.
19. S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, "A Scalable Content-Addressable Network," in *Proceedings of ACM SIGCOMM'01*, (San Diego, CA), August 2001.