A Distributed Algorithm for a b-Coloring of a Graph

Brice Effantin, Hamamache Kheddouci

A Distributed Algorithm for a b-Coloring of a Graph

Brice Effantin and Hamamache Kheddouci

Laboratoire PRISMa, Université Lyon 1,
IUT A Département Informatique,
01000 Bourg-en-Bresse, France
{beffanti, hkheddou}@bat710.univ-lyon1.fr
http://www710.univ-lyon1.fr/~g2ap/

Abstract. A b-coloring of a graph is a proper coloring where each color admits at least one node (called *dominating node*) adjacent to every other used color. Such a coloring gives a partitioning of the graph in clusters for which every cluster has a clusterhead (the dominating node) adjacent to each other cluster. Such a decomposition is very interesting for large distributed systems, networks,... In this paper we present a distributed algorithm to compute a b-coloring of a graph, and we propose an application for the routing in networks to illustrate our algorithm.

1 Introduction

In a distributed system, a node exchanges information only with its neighborhood. Every node has a set of local variables to determine a local state of the node. The state of the entire system, called *global state*, is the union of the local states of all the nodes in the system. The objective in a distributed system is to obtain automatically a desirable global final state (called *legitimate state*) where each node is considered as a distinct entity able to compute its own state itself, with its neighborhood as only knowledge. Distributed algorithms are a very attractive topic for a lot of fields and several graph problems arise naturally in distributed systems. For example, distributed algorithms for finding spanning trees, matchings, independent sets or particular colorings have been studied [1,5,9,15,16]. Such algorithms are so very interesting and efficient for dynamic networks [3,17].

The aim of our paper is to propose a distributed method to determine a particular coloring of graphs. Thus, we propose a partitioning method under conditions of a graph based on a graph coloring called *b-coloring.* A *proper k-coloring* is a coloring using k colors such that two adjacent nodes have different colors. Then, a *b-coloring* is a proper k-coloring where for each color i, $1 \leq i \leq k$, there exists a node x, with color i, adjacent to nodes colored with every color j, $1 \leq j \neq i \leq k$. Such nodes are called *dominating nodes.* This coloring was introduced by Irving and Manlove in [10] where they presented the *b-chromatic number* as the maximum integer k such that G admits a b-coloring with k colors. In [10], they also proved that finding the b-chromatic number of any graph is a NP-hard problem and they gave a polynomial-time algorithm for finding the b-chromatic number of trees. This parameter was also studied for other classes of graphs like cartesian product of graphs [13], bipartite graphs [14], power graphs of paths and cycles [7]. More recently, Corteel et *al.* [6] proved that the b-chromatic number problem is not approximable within $120/133-\epsilon$ for any $\epsilon > 0$, unless $P = NP$. This coloring is very interesting since it partitions the nodes of G into color classes for which at least one node is adjacent to each other color class. The property of these nodes is very attractive to give a hierarchy of nodes, to exchange information with other classes,. . .

The remainder of this article is then decomposed as follows. In Section 2, we present a distributed algorithm to compute a b-coloring of a graph. Then in Section 3, we propose an application to this clustering where we study a routing method. Finally, Section 4 concludes with future works.

We first start with some definitions used in the following. A distributed system will be modeled with an undirected connected graph $G = (V, E)$, where the node set V represents the set of processors, and the edge set E represents the processor interconnexions. Throughout this paper, we assume that $|V| = n$ and $|E| = m$. For every node i, we define $N(i)$, its *open neighborhood*, as the set of nodes adjacent to i (the set of colors of $N(i)$ will be denoted $N_c(i)$). We let $d(i) = |N(i)|$, the number of neighbors of node i, or its *degree*, and we let $\Delta = \max\{d(i)|i \in V\}$. Finally, let $diam(G)$ be the diameter of the graph G, defined as the maximum distance between any pair of vertices of G. In our study we assume a synchronized model in which any process computes the same action at the same time. Then, two or more processes can be in conflict (for example, two adjacent processes colored with the same color). A central daemon selects, among all these processes, the next process to compute. Thus, if two or more processes are in conflict, we cannot predict which process will be computed next. Several protocols exist ([2,4]) that provide a scheduler. Our algorithm can be combined with any of these protocols to work under different schedulers as well.

It is very important to remark that the coloring constraint in this problem is not only on the neighborhood of each node, but on the entire graph. Indeed, the color of a node depends on the presence or not of a dominating node for its color. Note that in the following we say that a color is *emptyable* if it has no dominating nodes. Thus, the first idea for finding such a coloring is: 1- Find a proper coloring of the graph; 2- For any emptyable color c, recolor nodes with color c by other colors and discard color c. However, the b-coloring is not a continue coloring. For instance, the hypercube Q_3 admits b-colorings with 2 or 4 colors but not with 3 (see Figure 1). Thus if we remove emptyable colors, we must remove them one by one and verify, at each step, if the coloring is a b-coloring or not.

2 b-Coloring of a Graph

In this section we propose a distributed algorithm to determine a partitioning (or a clustering) of the nodes of a graph G based on a particular coloring. In

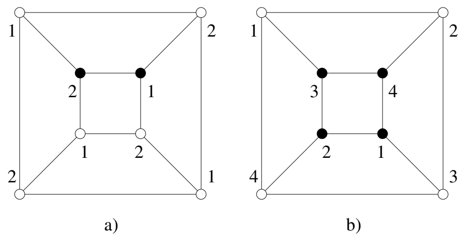

Fig. 1. b-colorings with k colors for the hypercube Q_3, where *a)* $k = 2$, *b)* $k = 4$. (black nodes are dominating nodes).

every cluster we select a clusterhead (called the dominating node in the coloring) able to join directly each other cluster.

In this approach, the nodes are represented by processes and they search to have a color with dominating node (*i.e.* the emptyable colors are removed from the coloring). Since the information on colors are dispersed in the entire graph, nodes need to exchange some messages with the remainder of the graph. Our approach enables the nodes to react to these messages to reach a legitimate state for the system. Thus nodes have different labels from the interval $[1; n]$. The principle of this algorithm is to detect emptyable colors. If such colors exist, they must be removed, but as we saw, one by one.

This method is a recoloring method. So nodes must be initialized. To put the system in an initialized state, we color a node with maximum degree by color 1 and we use Procedure 1.

Procedure 1. *Init_Coloring()*
```
begin
if $c(i) \neq \emptyset$ then
    Let $M = N_c(i) \cup \{c(i)\}$.
    $q = 0$.
    for every node $j \in N(i)$ such that $c(j) = \emptyset$ do
        $q = \min\{k | k > q, k \notin M$ and $k \notin N_c(j)\}$.
        if $q \leq \Delta + 1$ then $c(j) = q$ else $c(j) = \min\{k | k \notin N_c(j)\}$. endif.
    enddo.
endif.
end.
```

Lemma 1. *Procedure 1 is done in $O(m)$.*

Proof. Procedure 1 is applied once on every colored node. For each node, the procedure colors every of its non-colored neighbors. Since each node has $d(i)$ neighbors, Procedure 1 is computed in $\sum_{i=1}^{n} d(i) = 2m = O(m)$. □

Several studies propose fast colorings of graphs which can be used to initialized the state of nodes. In particular Hedetniemi et *al.* in [8], and Johansson in [12], propose distributed colorings using at most $\Delta + 1$ colors. Our procedure has the advantage to find exactly $\Delta + 1$ colors in a linear time.

To compute a b-coloring of a graph, any node needs some information about dominating nodes and emptyable colors. We let for any node i, $Dom_i[]$ as a table containing the label of the dominating node for each color (initialized to 0). If a color c has no dominating node we have $Dom_i[c] = 0$ and the value $Dom_i[c] = \emptyset$ means that the color c has been removed from the coloring. Then, before removing a color c, every node must verify if it is not a dominating node for this color. If it is not, it will send a message to propose to remove the color c.

Before the study of each possible message, we give the two following procedures. The first evaluates the state of a node (dominating node or not). By comparing the set of existing colors in the graph and the set of colors in its neighborhood, a node determines if it is a dominating node or not. Then, between all possible nodes to be a dominating node for a color c, we choose the node with the smallest label. If i is not a dominating node and if there does not exist a dominating node for $c(i)$, then node i supposes that its color is an emptyable color. Note that each node maintains also the color list of its neighborhood $N_c(i)$.

Procedure 2. *Processing*()

```
begin
Let $N'_c = \bigcup q$ such that $1 \leq q \neq c(i) \leq \Delta + 1$ and $Dom_i[q] \neq \emptyset$.
if $N'_c = N_c(i)$ (i.e. $i$ is a dominating node) then
    if $Dom_i[c(i)] > i$ or $Dom_i[c(i)] = 0$ then
        $Dom_i[c(i)] := i$.
        send to every $k \in N(i)$: $M_2(i, c(i))$.
    endif.
else
    if $Dom_i[c(i)] = 0$ then
        send to every $k \in N(i)$: $M_3(c(i))$.
        Execute $Waiting()$.
    endif.
endif.
end.
```

The second procedure gives some instructions computed after a delay, used to consider that a message reached every node in the graph. This procedure will be applied on a node i if it determines its color $c(i)$ as emptyable. Since the method is distributed, any node works in the same time. Thus, if there exists a dominating node j for $c(i)$, this information will be propagated to every node of G in time $diam(G)$. We will see that if the node i receive this information, it stops the execution of the Procedure 3. So, if i waits for a delay $diam(G) + 1$, it can consider that its color has no dominating node, and it will replace it.

Procedure 3. *Waiting*()

```
begin
After a time $diam(G) + 1$ do
    $col := c(i)$.
    $c(i) := \max\{q | 1 \leq q \leq \Delta + 1,\ q \notin N_c(i)$ and $Dom_i[q] \neq \emptyset\}$.
    send to every $k \in N(i)$: $M_1(i, c(i))$.
    send to every $k \in N(i)$: $M_4(col)$.
end.
```

Then, every node i will react following the received message. Messages are presented below:

– No message is received. In this case node i applies Procedure 2 to evaluate its state.
– Message $M_1(j, c)$: "NODE j HAS COLOR c."
  Every node receiving this message updates the colors of its neighborhood.
– Message $M_2(j, c)$: "NODE j IS A DOMINATING NODE FOR THE COLOR c."
  Among all nodes verifying the conditions to be a dominating node for color c, the dominating node of c will be the node with the smallest label. Then, if j is a dominating node for $c(i)$ (*i.e.* $c(i) = c$), then node i interrupts its procedure $Waiting()$. Moreover, to limit the number of messages, the node i propagates this information only if it is new.
  Then, if this message is received, node i applies the following procedure:

  Procedure 4. $M_2(j, c)$

```
begin
if $Dom_i[c] > j$ or $Dom_i[c] = 0$ then
    $Dom_i[c] := j$.
    send to every $k \in N(i)$: $M_2(j, c)$.
    if $c = c(i)$ then Interrupt $Waiting()$. endif.
endif.
end.
```

– Message $M_3(c)$: "COLOR c IS EMPTYABLE."
  Among all emptyable colors, the next color to remove will be the smallest. This message is so used by any node to determine if its color is the next color to remove. Thus, if the color received by node i is smaller than $c(i)$, then it propagates the message. Moreover, if i executes currently the procedure $Waiting()$ (Procedure 3), then it stops it since the next color to remove will not be $c(i)$.

  Procedure 5. $M_3(c)$

```
begin
if $c < c(i)$ and $Dom_i[c] = 0$ then
    send to every $k \in N(i)$: $M_3(c)$.
    Interrupt $Waiting()$.
endif.
end.
```

– Message $M_4(c)$: "DELETE COLOR c"
  If $c(i) = c$ then the node i must take another existing color and it propagates this message to remove color c from the graph. Moreover, since the color of node i perhaps changed, it stops its procedure $Waiting()$ and starts again the procedure $Processing()$.

```
Procedure 6. $M_4(c)$
begin
if $c(i) = c$ then
    $Dom_i[c] := \emptyset$.
    $c(i) := \max\{q | 1 \leq q \leq \Delta + 1,\ q \notin N_c(i)$ and $Dom_i[q] \neq \emptyset\}$.
    send to every $k \in N(i)$: $M_1(i, c(i))$.
endif.
send to every $k \in N(i)$: $M_4(c)$.
Interrupt $Waiting()$.
Execute $Processing()$.
end.
```

Proposition 1. *A legitimate state for G is reached with $O(n\Delta)$ local changes and $O(m\Delta^2)$ exchanged messages.*

Proof. First, a color is removed if it is emptyable, and only one color is removed at the same time. Thus, each node can change at most Δ times its color. Hence the total number of local changes is $O(n\Delta)$.

Then, we can evaluate the number of exchanged messages. If the color of a node changes, this node sends its new color to its $d(i)$ neighbors. Since a node can change at most Δ times its color, we have $\sum_{i=1}^{n} \Delta.d(i) = O(m\Delta)$ Messages 1 sent. For Message 2, suppose that every node is a dominating node. Then, a node can receive at most $n - 1$ messages on dominating nodes, and it can also transmit these messages to its $d(i)$ neighbors. Thus $\sum_{i=1}^{n}(n-1).d(i) = O(nm)$ messages are used to propagate these information. For the emptyable colors (Message 3), a node sends a message to its neighborhood every time it finds a smaller color to remove. Suppose that every color is emptyable. Since at most Δ colors can be removed, each node sends at most $\Delta.d(i)$ messages. Every time a color is removed, the same reasoning can be done. Thus, we deduce that $\sum_{i=1}^{n}(\sum_{q=1}^{\Delta} q.d(i)) = O(m\Delta^2)$ Messages 3 are exchanged. Finally, by the same way as for Message 2, we deduce that $O(nm)$ messages are used to remove colors (Message 4). Therefore, the number of exchanged messages to join a legitimate state is in $O(m\Delta^2)$. □

Theorem 1. *The time complexity is $O(\Delta diam(G))$.*

Proof. After the initialization step, any node determines dominating nodes in time $diam(G)$ (by the propagation of Message 2). During the same time, an emptyable color can be found (if exists). However, in the worst case, two opposite information can traverse the graph and prevent us from determining an emptyable color (since any node interrupted its Procedure 3). Nevertheless, in this case no more messages are exchanged, and from the algorithm every node applies Procedure 2. Thus, in time $diam(G)$, an emptyable color is found (if exists).

Consequently, an emptyable color starts to be removed after a delay $diam(G)$ + 1 and it is completely removed in time $diam(G)$. Since at most Δ colors are removed, a legitimate state is computed in $O(\Delta.diam(G))$. □

Figure 2 presents an example of computation of the algorithm.

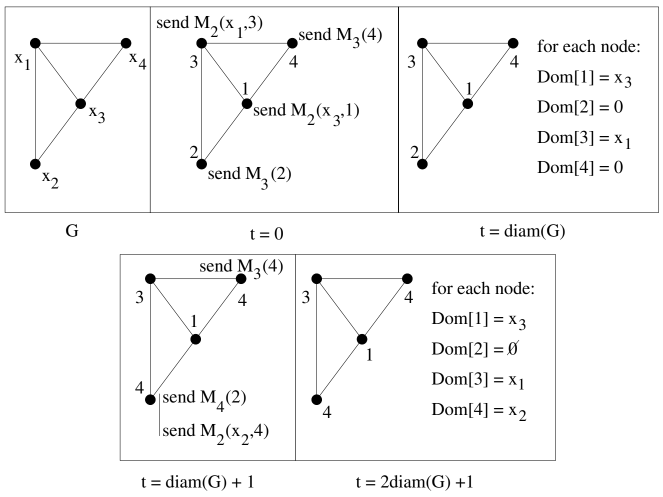

Fig. 2. Messages:$M_2(x, c)$ means "x is a dominating node for color c";$M_3(c)$ means "c is an emptyable color";$M_4(c)$ means "delete color c". Thus, at $t = 0$ the graph G is initialized and each node x_i evaluates its state (Procedure 2). At $t = diam(G)$, every node knows the dominating nodes and the next color to remove. At $t = diam(G) + 1$, Procedure 3 on node x_2 is terminated, so it changes its color. At $t = 2diam(G) + 1$, nodes know the dominating nodes and the coloring is found.

3 A b-Coloring Based Routing Method

In this section we propose an application of the above algorithm, for routing information between nodes of a weighted graph. The idea is to decompose the graph in clusters, and to use the clusterhead of the source node to join the cluster of the target node.

To adapt our algorithm to a routing application, it needs some minor modifications. Indeed, the clusterhead of each cluster will communicate with the nodes of its cluster. We just add some data to compute theses paths. Thus, we define for each node i, $Predec_i[c]$ as the predecessor of i in the path from the dominating node of color c and i, and $Dist_i[c]$ as the length (*i.e.* the total weight) of this path. Thus, Procedure 2 initializes these data for any dominated color found. Then, Message 2 can include two new parameters: the node v from which the message comes, and its distance to the dominating node j. Then, the node i can modify its path and so its distance to j in function of these parameters. These modifications will be integrated to the messages propagating the domination of j. This computation can increase the number of Message 2 sent but does not change the complexity of the algorithm. Finally, information on the path and the distance about a removed color can be destroyed in Message 4.

These modifications of the algorithm presented in the previous section enables us to compute the shortest path from a clusterhead to any node of its cluster in the same time as the b-coloring of the graph. Then, to propagate information from a source node s to a target node t, we use the dominating nodes of the clusters containing s and t. Let w_s and w_t be the dominating nodes of clusters containing respectively s and t. Thus, the steps for determining a path from s to t are:

1. Finding the path from s to w_s,
2. Joining a neighbor w' of w_s in the cluster containing node t (property of the clusterhead w_s),
3. Finding the path from w' to w_t,
4. Finding the path from w_t to t.

The Figure 3 presents the method used to determine a route between a source s and a destination t.

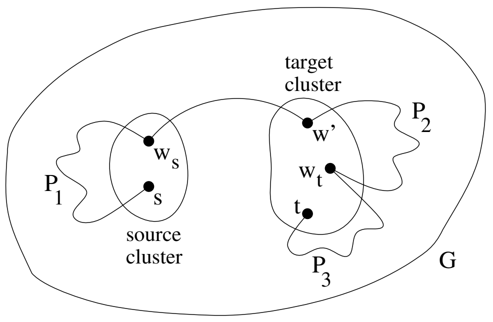

Fig. 3. Illustration of the routing from the source node s to the target node t in the graph G. Any path P_i is computed in the same time as the coloring and every path P_i is a shortest path between its endvertices.

Lemma 2. *Routing method complexity is $O(n)$.*

Proof. For the determination of a path from a clusterhead x to a member of its cluster y, we need to traverse the graph from y to x, node by node. Indeed in the algorithm, each node knows only its predecessor in the path from x to y. Thus, a path between any node and its clusterhead is done in $O(n)$ (this is the case of steps $1, 3$ and 4). Moreover, each node knows its neighborhood. So, the step 2 is done in $O(1)$. Therefore, a path from a source node to a destination node in a graph G is computed in $O(n)$. □

This method depends on the complexity of the b-coloring problem. It does not improve existing methods based on spanning trees [1,3,5] or routing table [11], it just proposes a new way to route information by a clustering of the structure.

4 Conclusion

In this article, we presented a distributed algorithm to partition a structure in clusters. Each of these clusters has a clusterhead with the property to join directly every other cluster. An interesting question completes this study: How can react vertices if the topology of the graph evolves ? Our study concerns currently the dynamic graphs.

References

1. Antonoiu, G., Srimani, P.K.: A self-stabilizing distributed algorithm for minimal spanning tree problem in a symmetric graph. Computer & Mathematics with Application **35**(10) (1998) 15-23
2. Antonoiu, G., Srimani, P.K.: Mutual exclusion between neighboring nodes in an arbitrary system graph tree that stabilizes using read/write atomicity. Proceedings Euro-par'99, LNCS **1685** (1999) 823-830
3. Baala, H., Flauzac, O., Gaber, J., Bui, M., El-Ghazawi, T.: A self-stabilizing distributed algorithm for spanning tree construction in wireless ad hoc networks. Journal of Parallel and Distributed Computing **63** (2003) 97-104
4. Beauquier, J., Datta, A.K., Gradinariu, M., Magniette, F.: Self-stabilizing local mutual exclusion and daemon refinement. Proceedings DISC 2000, LNCS **1914** (2000) 223-237
5. Bui, M., Butelle, F., Lavault, C.: A distributed algorithm for constructing a minimum diameter spanning tree. Journal of Parallel and Distributed Computing **64** (2004) 571-577
6. Corteel, S., Valencia-Pabon, M., Vera, J.-C.: On approximating the b-chromatic number. Discrete Applied Mathematics **146** (2005) 106-110
7. Effantin, B., Kheddouci, H.: The b-chromatic number of some power graphs. Discrete Mathematics and Theoretical Computer Science **6** (2003) 45-54
8. Hedetniemi, S.T., Jacobs, D.P., Srimani, P.K.: Linear time self-stabilizing colorings. Information Processing Letters **87** (2003) 251-255
9. Hsi, S-C., Huang, S-T.: A self-stabilizing algorithm for maximal matching. Information Processing Letters **43**(Issue 2) (1992) 77-81
10. Irving, R.W., Manlove, D.F.: The b-chromatic number of a graph. Discrete Applied Mathematics **91** (1999) 127-141
11. Ito, H., Iwama, K., Okabe, Y., Yoshihiro, T.: Single backup table schemes for shortest-path routing. Theoretical Computer Science **333** (2005) 347-353
12. Johansson, Ö.: Simple distributed $\Delta+1$-coloring of graphs. Information Processing Letters **70** (1999) 229-232
13. Kouider, M., Mahéo, M.: Some bounds for the b-chromatic number of a graph. Discrete Mathematics **256**(Issues 1-2) (2002) 267-277
14. Kratochvíl, J., Tuza, Z., Voigt, M.: On the b-chromatic number of graphs. 28th International Workshop on Graph-Theoretic Concepts in Computer Science, Cesky Krumlov, Czech Republic; LNCS **2573** (2002) 310-320
15. Shi, Z., Goddard W., Hedetniemi, S.T.: An anonymous self-stabilizing algorithm for 1-maximal independent set in trees. Information Processing Letters **91** (2004) 77-83
16. Šparl, P., Žerovnik, J.: 2-local distributed algorithms for generalized coloring of hexagonal graphs. Electronic Notes in Discrete Mathematics **22** (2005) 321-325
17. Srivastava, S., Ghosh, R.K.: Distributed algorithms for finding and Maintaining a k-tree core in a dynamic network. Information Processing Letters **88** (2003) 187-194