IMPLEMENTATION AND ANALYSIS OF A PARALLEL ALGORITHM FOR RADIOCOLORING

Hemant Balakrishnan and Narsingh Deo
School of Computer Science
University of Central Florida
Orlando, Florida 32816-2362

{hemant, deo}@cs.ucf.edu

Abstract

A radiocoloring of a graph G is an assignment to each node, one of the colors $0, 1, 2, \ldots \lambda$ such that all pairs of adjacent nodes, get colors which differ by at least two and no pair of nodes at distance two, get the same color. The radiocoloring problem (RCP) consists of determining the minimum λ for a given graph. In this paper, we first implement the PARC (*Parallel Algorithm for Radiocoloring*) algorithm for radiocoloring which is based on the *largest-degree-first* coloring heuristic and then perform an analysis of the obtained results.

Keywords: Radiocoloring, $L(2,1)$ labelling, LF algorithm.

1 INTRODUCTION

Over the past two decades there has been considerable research and improvement in the area of wireless networks. One important application of wireless networks is in mobile telephony. The mobile phone network consists of a number of static *base stations* that serve the mobile users within their vicinity. These static base stations need to be assigned frequencies in such a way that the interference is minimal. The frequency assignment problem was introduced to assign channels to base stations in order to exploit frequency reuse. F. S. Roberts [7] proposed a variation of the frequency assignment problem in which close base stations receive different channels and very close base stations receive channels that are at least two apart. To formulate a graph theoretic approach to this problem, the base stations are represented by nodes of a graph and there exists an edge between two nodes if the corresponding transmitters are close enough such that their

frequencies could interfere. Two base stations are considered *close* if they are at a distance of two in the graph and *very close* if they are adjacent in the graph.

Definition 1. *A radiocoloring of a graph G is a function f from the vertex set $V(G)$ to the set of all non negative integers such that $|f(x) - f(y)| \geq 2$ if $d(x, y) = 1$ and $|f(x) - f(y)| \geq 1$ if $d(x, y) = 2$ where $d(x, y)$ represents the distance between the nodes x and y. The radiochromatic number, $\lambda(G)$ is the smallest number k such that G is radiocolored with $max\{f(v) : v \in V(G)\} = k$ colors.*

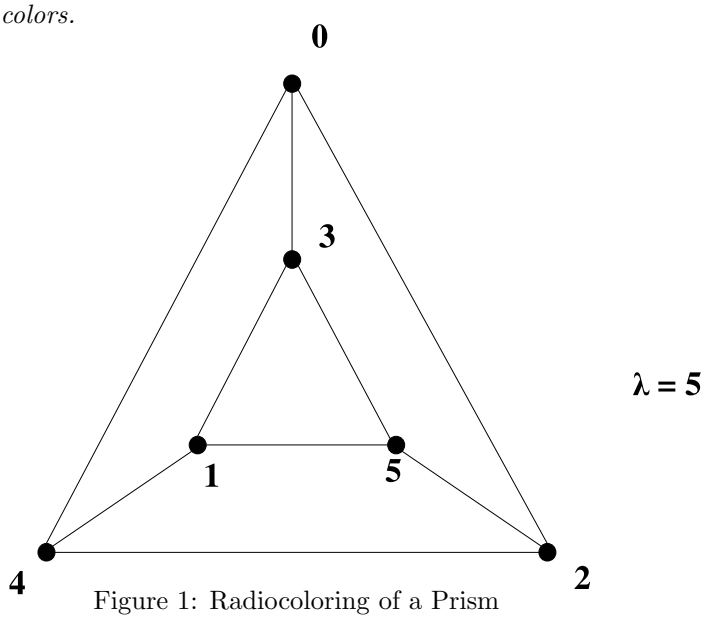

Figure 1: Radiocoloring of a Prism

If the adjacent nodes were to receive the same color this would imply that the neighboring base stations would be assigned the same frequency and they would not be able to communicate. If the adjacent nodes were to be assigned colors that were just one apart then the base stations would be assigned adjacent frequencies and it is a known fact that there is always some amount of interference between adjacent frequencies (*direct collision*). The first condition, $|f(x) - f(y)| \geq 2$ if $d(x, y) = 1$, makes sure that there is a *guard band* in between frequencies assigned to adjacent base stations.

Definition 2. *A guard band is an unused frequency band between two adjacent channels in wireless communication to minimize the interference between the channels.*

The second condition, $|f(x) - f(y)| \geq 1$ if $d(x, y) = 2$, makes sure that the base stations with a common neighbor do not communicate with the

common neighbor using the same frequency (*hidden collision*) which would lead to ambiguity.

The radiocoloring problem was first introduced by Griggs and Yeh [7] as the $L(2,1)$ labelling problem. The radiocoloring problem like the coloring problem is NP-hard [7]. However exact results have been obtained for certain special classes of graphs such as paths [12], cycles, trees [7], cacti, unicycles, bicycles [8], cliques, stars [4]; and approximate bounds have been obtained for bipartite graphs, outerplanar graphs, split graphs [2], chordal graphs [9], hypercubes [11], planar graphs [8], unigraphs [5].

The rest of the paper is organized as follows: Section 2 presents the parallel architecture that is required and the notations which would be used in the PARC pseudo code, Section 2.1 gives us the description and the pseudo code for PARC and Section 3 provides the implementation details followed by the observed results in Section 4.

2 THE PARALLEL ALGORITHM

In this paper we implement and analyze the parallel algorithm proposed by Balakrishnan and Deo in [1]. This algorithm is an approximate algorithm for computing the radiochromatic number of a given graph, G and is based on the parallel graph coloring algorithm by Das and Deo [6]. The algorithm makes use of a *concurrent-read and exclusive-write* (CREW) *parallel random access machine* (PRAM) consisting of n processors, each of which is equipped with a small amount of local memory and a processing unit. Processors are identified by a unique number and they share a global memory. The processors can simultaneously read from a location of the global shared memory. However, they cannot simultaneously write into a particular location of the shared memory. The processors communicate among themselves using shared variables, and each can perform a scalar, arithmetic, or boolean operation in unit time.

Parallel operations within the algorithm are represented using the *parallel forloop* construct. For example, the code segment

for all P_i such that $0 < i < q$ **do**
  parbegin
  statements to be executed in parallel
  parend
end for

when run on some processor P, indicates that when P executes the *forloop* it would fork into q parallel processes in q processors (corresponding to processors P_i, $0 < i < q$). The processors share a common environment and are distinguished by their unique processor *id*. Any statement between the **parbegin** and **parend** are executed simultaneously by all the q processors.

On reaching the **parend** statement the q processes again join into a single process which could again run on P.

2.1 ALGORITHM DESCRIPTION

The *largest-degree-first* (LF) algorithm for graph coloring, proposed by Walsh and Powell [10] has been extended to radiocolor general graphs. The adjacency matrix of the given graph G and the number of nodes, n, are provided as the input to the algorithm. The algorithm consists of three steps: in the first step, the given nodes are sorted in a non-increasing order on the basis of their degrees; in the second step, a distance-two binary matrix of graph G is constructed; and finally, the nodes are assigned colors in the sorted order one by one. The algorithm makes use of an $n \times n$ adjacency matrix, ADJMATRIX, which is provided as input; a linear array, DEGREE, where the degree of each node is computed and stored; a linear array, SORT, to store the nodes sorted by their degree; a $n \times n$ distance-two matrix, D2MATRIX, to show the list of nodes that are at a distance of two from each node; a $n \times 2n - 1$ (It is obvious that any graph can be radiocolored with a maximum λ of $2n - 2$ which is the radiochromatic number of a complete graph on n nodes [5].) matrix, FORBIDDEN, to record the colors that cannot be assigned to a node; and a linear array, COLOR, to store the colors assigned to each node.

First, the degree of each node is computed, degree of a node v, DEGREE$[v] = \Sigma_{0<i<n}$ ADJMATRIX$[v, i]$. The nodes are then sorted based on their degrees in a non-increasing order and stored in the array SORT. Next, the distance-two matrix is computed, D2MATRIX$[x, y] \leftarrow 1$, if and only if ADJMATRIX$[x, k] = 1$ and ADJMATRIX$[k, y] = 1$ for some k, where $0 < k < n$, and $x \neq y$. Then, a color c is assigned to node v, if and only if c is the minimum number such that FORBIDDEN$[v, c] = 0$. Now the matrix FORBIDDEN is updated, for any node u, if ADJMATRIX$[v, u] = 1$ then FORBIDDEN$[u, c-1]$, FORBIDDEN$[u, c]$, and FORBIDDEN$[u, c+1]$ are assigned 1 (*i.e.* any node u which is adjacent to v cannot receive colors that are not at least two apart from c). Similarly, for any node u, if D2MATRIX$[v, u] = 1$ then FORBIDDEN$[u, c] = 1$ (*i.e.* color c cannot be assigned to any node u which is at a distance of two from v). In each successive iteration the next uncolored node with the largest degree is colored and the matrix FORBIDDEN updated to incorporate the change.

Like the sequential algorithm, the parallel version of the algorithm has three steps and makes use of all the data structures described above.

Input : ADJMATRIX, n
Output : λ
procedure PARC
for all P_i such that $0 \leq i < n$ **do** {/* Step 1: Sort the nodes in

```
descending order based on their degree */}
  parbegin
  DEGREE[i] ← 0
  COLOR[i] ← 0
  for all l such that 0 ≤ l < n do
    if ADJMATRIX[i, l] = 1 then
      DEGREE[i] ← DEGREE[i] + 1
    end if
  end for
  Sort the nodes in parallel to obtain the array SORT
  parend
end for
for all j such that 0 ≤ j < n do {/* Step 2: Compute the
distance two matrix */}
  for all P_i such that 0 ≤ i < n do
    parbegin
    if ADJMATRIX[i, j] = 1 then
      for all m such that 0 ≤ m < n do
        if ADJMATRIX[j, m] = 1 AND m ≠ i then {/*
        concurrent read required */}
          D2MATRIX[i, m] ← 1
        end if
      end for
    end if
    parend
  end for
end for
λ ← 0 {/* assume λ to be 0 initially */}
for all j such that 0 ≤ j < n do {/* Step 3: Assign colors to
nodes */}
  for all P_i such that 0 ≤ i < n do
    parbegin
    find the smallest c such that FORBIDDEN[SORT[j], c] = 0
    parend
  end for
  COLOR[SORT[j]] ← c
  if λ < c then
    λ ← c
  end if
  for all P_i such that 0 ≤ i < n do
    parbegin
    if ADJMATRIX[SORT[j],i] = 1 AND c ≠ 0 then
      FORBIDDEN[i, c − 1] ← 1
```

```
      end if
      if ADJMATRIX[SORT[j],i] = 1 then
        FORBIDDEN[i, c] ← 1
      end if
      if ADJMATRIX[SORT[j],i] = 1 AND c ≠ 2n − 2 then
        FORBIDDEN[i, c + 1] ← 1
      end if
      if D2MATRIX[SORT[j],i] = 1 then
        FORBIDDEN[i, c] ← 1
      end if
      parend
    end for
  end for
```

2.2 COMPLEXITY OF THE ALGORITHM

In this section, we first analyze the complexity of the sequential algorithm. The sequential algorithm consists of three steps, at first we compute the degree of each node, which takes $O(n^2)$ time, followed by sorting which takes $O(n \log n)$ time. In the next step we compute the distance-two matrix which takes time $O(n^3)$ and the final step involves assigning colors to the nodes; this step could be divided into two, first we search for a color to be assigned to a node and then update the FORBIDDEN matrix. This requires $O(n^2)$ time. Thus the overall complexity for the sequential algorithm is

$$\approx O(n^2) + O(n \log n) + O(n^3) + O(n^2),$$
$$\approx O(n^3).$$

The parallel algorithm also consists of three steps. Given n processors we can compute the array DEGREE in parallel in time $O(n)$. Sorting the array DEGREE can be done in $O(\log n)$ time, if we use parallel quicksort. Then we compute the D2MATRIX which takes $O(n^2)$ time using n processors. Assigning colors to the nodes and updating the FORBIDDEN matrix requires time $O(n \log n)$ with n processors. The overall complexity for the parallel algorithm would be

$$\approx O(n) + O(\log n) + O(n^2) + O(n \log n),$$
$$\approx O(n^2).$$

The *cost* of a parallel algorithm is the product of the number of processors and time. A parallel algorithm is said to be *cost-optimal*, if the parallel cost is the same as the sequential time. The cost of the PARC is $O(n) \times O(n^2) \approx O(n^3)$, which is the same as the sequential time. Hence, the

algorithm is cost-optimal. By employing Brent's scheduling [3] to PARC, we can use fewer processors and still try to obtain the same asymptotic time complexity.

3 IMPLEMENTATION

This algorithm was implemented over a Beowulf Linux cluster consisting of 16 nodes. All the nodes as interconnected by a high speed network. The algorithm was written in C++ utilizing the MPI (Message Passing Interface) library. Jobs to the cluster are submitted via a job queue and jobs from the queue are executed one at a time, this makes sure that other processes do not interfere with the current process. The actual algorithm requires the same number of processors as the size of the graph but we have extended the algorithm to incorporate more nodes than the number of processors.

4 RESULTS AND OBSERVATION

The implementation was tested by varying the number of nodes and the number of processors used. The results are tabulated in Figure 2, The top row represents the number of processors used and the first column represents the number of nodes. Each reading was obtained by computing the average over 5 trials.

We further analyze the results by plotting a graph. The graph shows that the time taken by the sequential algorithm is better for small size graphs, there exists a threshold (minimum graph size) after which increasing the number of processors reduces the time. If we were to consider the time taken while using 2 processors and 4 processors we find that the time taken by 4 processors is initially high and this difference converges for increasing graph sizes and after a point the time taken by 4 processors would be less than the time taken by 2 processors.

The experimental results are not in congruence with the theoretical time complexity which was deduced. The experiment was conducted on a cluster utilizing the MPI library which makes use of message passing, where as the algorithm by Balakrishnan and Deo [1] was written for the PRAM model which is a virtual model that relies on shared memory for communication. The passing of messages between the processors over the network causes overhead (which would not arise if we were to be using the shared memory PRAM model). Initially the sequential algorithm out performs the parallel one because the problem size is small and the message passing overhead dominates the execution time but after the threshold value, the problem size dominates and message passing is no longer a overhead and the parallel algorithm out performs the sequential counter part. The anomaly in the

Number of nodes	1 processor	2 processors	4 processors	16 processors
32	0.00183	0.01071	0.23581	0.43975
64	0.01187	0.06321	0.36157	0.53183
128	0.08491	0.16174	0.76188	1.05582
256	0.63823	0.49245	1.52219	2.81039
512	4.95172	3.13367	4.87129	5.69074
640	9.56777	5.71789	7.09221	7.52920

Figure 2: Experimental results

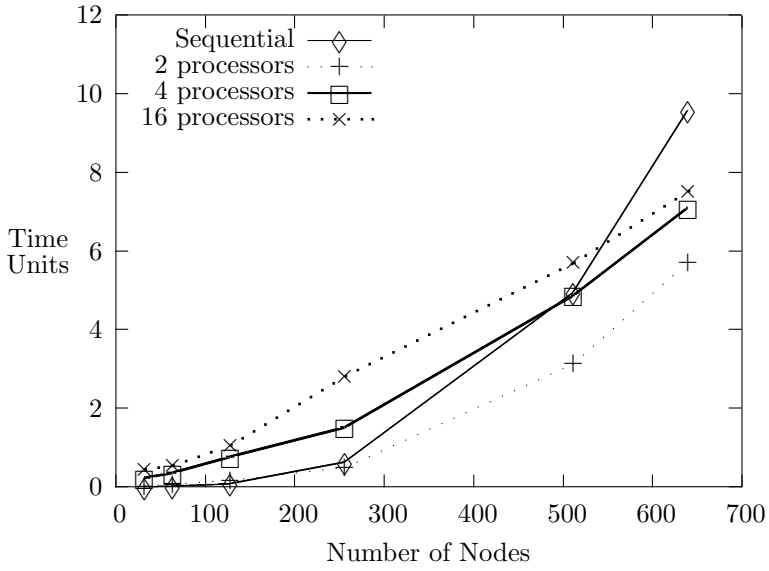

Figure 3: Number of nodes vs Time

results could also be explained because we use more than one node per processor whereas the proposed algorithm employs one node per processor.

5 CONCLUSION AND FUTURE WORK

In this paper, we have discussed the results obtained by implementing the parallel algorithm for radiocoloring graphs proposed by Balakrishnan and Deo in [1]. The results obtained were not very close to the predicted values. The reason for this anomaly was analyzed in this paper. However the parallel algorithm outperformed the sequential one after a certain threshold value of the input size. Future work could involve computing the exact value of the threshold for a given number of processors which would be dependent on the underlying machine architecture.

REFERENCES

[1] H. Balakrishnan and N. Deo. Parallel algorithm for radiocoloring a graph. *Proceedings of* 34^{th} *Southeastern International Conference on Combinatorics, Graph Theory and Computing*, 2003.

[2] H. L. Bodlaender, T. Kloks, R. B. Tan, and J. Van Leeuwen. Approximations for λ coloriongs of graphs. *Technical Report UU-CS-2000-25*, 2000.

[3] R. P. Brent. The parallel evaluation of general arithmetic expressions. *Journal of the Association for Computing Machinery*, 21:201–206, 1974.

[4] T. Calamoneri and R. Petreschi. λ-coloring of regular tiling. *Proceedings of* 1^{st} *cologne Twente Workshop (CTW)*, 8, 2001.

[5] T. Calamoneri and R. Petreschi. On the radiocoloring problem. 4^{th} *International Workshop on Distributed Computing*, 2002.

[6] S. K. Das and N. Deo. Parallel coloring of graphs: two approximate algorithms. *International Journal of Computer Mathematics*, 27:147–158, 1989.

[7] J. R. Griggs and R. K. Yeh. Labeling graphs with a condition at distance 2. *SIAM Journal on Discrete Mathematics*, 5:586–595, 1992.

[8] K. Jonas. Graph coloring analogues with a condition at distance two: L(2, 1)- labelings and list λ-labelings. 1993. Ph.D. thesis, University of South Carolina, Columbia.

[9] D. Sakai. Labeling chordal graphs: Distance two condition. *SIAM Journal on Discrete Mathematics*, 7:133–140, 1994.

[10] D. J. A. Walsh and M. B. Powell. An upper bound for the chromatic number and its application to timetabling problem. *The Computer Journal*, 10:85–86, 1967.

[11] M. A. Whittelesey, J. P. Georges, and D. W. Mauro. On the λ-number of q_n and related graphs. *SIAM Journal on Discrete Mathematics*, 8:499–506, 1995.

[12] R. Y. Yeh. Labeling graphs with a condition at distance two. 1990. Ph.D. Thesis, Dept. of Mathematics, University of South Carolina, Columbia.