The Partition Coloring Problem and its Application to Wavelength Routing and Assignment[1]

Guangzhi Li and Rahul Simha
Department of Computer Science
College of William and Mary
Williamsburg, VA 23187
{gli,simha}@cs.wm.edu

Abstract: *We consider a new vertex coloring problem, that we call* partition-coloring*, in which the vertex set of a graph $G = (V, E)$ is partitioned into k disjoint sets $V = V_1 \cup \cdots \cup V_k$, and G is said to be partition-colored if exactly one v in each V_i is colored and every two adjacent colored vertices have different colors. Analogous to the standard vertex coloring problem, we wish to partition-color a given graph with as few colors as possible. The partition-coloring problem is motivated by the wavelength routing and assignment problem in WDM (Wavelength Division Multiplexing) optical networks. In this paper, we show that this new coloring problem is as hard as standard vertex coloring and investigate heuristic algorithms. An application of these heuristics to the wavelength routing and assignment problem on optical networks is studied via simulation.*

Keywords: graph coloring, wavelength assignment, optical networks, routing.

1 Introduction

We define and study a new graph coloring problem in which the vertices of the graph are partitioned into a collection of disjoint sets and it is desired to color only one vertex in each set. This problem arises from considering the joint problem of routing and wavelength assignment in WDM optical networks, as we will show below. In that problem, we are given an optical network in which the links are optical fibers capable of carrying a specified number of wavelengths, and we are given a collection of desired connections (source-destination pairs). The goal is to find a route and wavelength for each connection

[1]This work was partially supported by grant ANI-9973106 from the National Science Foundation

so that no two connections that share a link also share a wavelength. If the routes are given as input to the problem, the wavelength assignment problem is easily transformed into the standard vertex coloring problem: represent each connection by a vertex in a new graph, and place an edge in that graph for every pair of connections that share a link in the optical network. Then, the constraint that no two link-sharing connections be assigned the same wavelength is equivalent to the graph coloring constraint that no two adjacent vertices be assigned the same color.

We take up the wavelength routing and assignment problem in which a route for each connection is not given, but rather a collection of possible "alternate" routes is given for each source-destination pair. This problem is of practical interest since route-sets are often neither restricted to a single route nor are they so large that they include every possible path from a source to a destination. The goal therefore is to find both a route (from among the given route set) and a wavelength for each connection. We observe that, in the literature, wavelength routing and assignment are usually considered separately. For example, shortest-path routing is often used to find a route and then wavelength assignment is performed using the given (shortest-path) routes. In other cases, routing and wavelength assignment are combined into an integer programming formulation. We believe our approach provides a single efficient heuristic for the joint problem of routing and wavelength assignment.

When this routing and wavelength assignment problem is transformed into a vertex coloring problem by associating a vertex in the transformed problem with a connection in the original problem, we get our partition-coloring problem as we show below. This new coloring problem is an interesting problem in its own right because it appears deceptively simpler than the standard vertex coloring problem.

In this paper, we show that partition-coloring is as hard as standard vertex coloring. We also consider extensions of well-known vertex coloring heuristics to the partition coloring problem. Finally, we apply the partition-coloring heuristics to a couple of instances of the wavelength routing and assignment problem.

We will now make some of the above notions more precise. A standard vertex coloring of a graph G can be considered as a mapping from the set of vertices V to a set S of "colors" such that no adjacent vertices have same colors. The minimum cardinality of S is called the *chromatic number* of

G, $\chi(G)$ [13]. Now, in our partition-coloring problem, we are given a partition of V into k disjoint sets $V = V_1 \cup \cdots \cup V_k$. Define G to be *partition-colored* if exactly one v in each V_i is colored and any two adjacent *colored* vertices have different colors.

The wavelength routing and assignment problem is defined as follows. We are given an undirected network graph N, a collection of source-destination pairs $C = \{(s_1, d_1), ..., (s_k, d_k) : s_i, d_i \in N\}$ and a collection of wavelengths $\Lambda = \{\lambda_1, ..., \lambda_m\}$. As usual, each vertex in the network graph represents a network node and each edge a link. Although the graph is undirected, we really assume that each edge is a full-duplex link: two separate fibers, one for each direction. Each such link is capable of carrying any subset of the wavelengths simultaneously and each network node is capable of directing a wavelength from any incoming link to any outgoing link by keeping the same wavelength. For each source-destination pair (s, d), let $\mathcal{P}_{s,d}$ denote the set of all paths between s and d in N. We are interested in the case in which a fixed set of paths $P_{s,d} \subset \mathcal{P}_{s,d}$ is given as input to the problem. An assignment consists of allocating one wavelength and one path to each source-destination pair. Thus, a path is selected from $P_{s,d}$ for each (s, d) pair and an assignment induces a mapping δ where $\delta(s, d, i, e) = 1$ if and only if e is on the path selected for (s, d) and wavelength λ_i has been assigned. Then, an assignment is feasible if no two (s, d) paths share both an edge and a wavelength, that is $\sum_{s,d} \delta(s, d, i, d) \leq 1$ for all e and i. We seek an assignment such that the cardinality of Λ is as small as possible.

Now, if $|P_{s_i,d_i}| = 1, \forall i = 1, ..., k$, which means that each source-destination pair only has one path to select (for example, the shortest path), then the wavelength assignment problem is equivalent to standard vertex coloring as mentioned above. This transformation is described as follows. We define a *path-conflict* graph $G_P = (V_P, E_P)$ from the wavelength assignment problem: a vertex is created in V_p for each (s_i, d_i) pair and an edge between two vertices v_i and v_j in V_P if the paths P_{v_i} and P_{v_j} share a link in the network, where P_{v_i} is the only path for the source-destination pair (s_i, d_i). By associating a color with each wavelength, a vertex-coloring of G_P is equivalent to a feasible solution for the wavelength assignment problem.

If $|P_{s_i,d_i}| > 1$ for some (s_i, d_i) pair, then these source-destination pairs have more than one path to choose from. In this case, we show that the routing and wavelength assignment problem is equivalent to partition-coloring. We define the *path-conflict* graph $G_P = (V_P, E_P)$ with a little difference: there are totally $\sum_{i=1}^{k} |P_{s_i,d_i}|$ vertices in V_P and each vertex represents one path; for any $u, v \in V_P$, $(u, v) \in E_P$ if and only if the two paths corresponding to u and v share at least one link in N. Because of the construction of G_P, the vertices in V_P can be partitioned into k disjoint sets $V_P = V_{s_1,d_1} \cup ... \cup V_{s_k,d_k}$ such that each partition set $V_{s,d}$ corresponds to one (s, d) pair. Since an assignment needs to select only one path for each (s, d) pair, the corresponding vertex coloring in G_P needs to color only one vertex in each $V_{s,d}$. It is easy to see that partition-coloring and multi-path routing and wavelength assignment are the same if we associate a color with each wavelength. The remaining question is how to choose the vertex from each partition and to assign a color to these selected vertices such that the number of colors used is as small as possible.

In Section 2, we review related work. Some properties of partition-coloring are investigated in Section 3 and it is shown that this new problem is as hard as the standard vertex-coloring. In Section 4, we extend the standard vertex coloring heuristics and investigate two group of approximation algorithms for partition-coloring. Detailed simulation results are given in Section 5 for these algorithms. Finally, an application on mesh WDM optical network is presented in Section 6 and concluding remarks are given in Section 7.

2 Related Work

The standard vertex coloring has a long, rich history [13] and arises in a variety of applications for example, maps [17], scheduling [3], routing [22], circuit testing [6], and sparse matrices [5]. The problem was shown to be NP-complete in 1972 [12]. This has led to several heuristics, most of which can be categorized into a few approaches. One simple heuristic algorithm, for example, is sequential-greedy: color the graph vertices one by one using any available color that is not used by its neighbors. This algorithm can perform very poorly. According to Gould [10], all existing fast standard vertex coloring algorithms use one or more of the following heuristic assumptions:

- a vertex of high degree is harder to color than a vertex of low degree;
- vertices adjacent to the same set of vertices should be colored with the same color;
- coloring as many vertices with a single color is a good idea.

We focus on the three of the most well-known heuristics: *Largest-First (LF)* by Welsh and Powell [23], *Smallest-Last (SL)* by Matula et al [13], and *Color-Degree (CD)* by Brelaz [2]. The *Largest-First* algorithm orders vertices by descending degree, and then greedily assigns the smallest available color to each vertex. *Smallest-Last* selects the vertex of lowest degree, removes it and pushes it into a stack. It repeat this process until no vertex remains in the graph. It then proceeds to color the vertices greedily by the order determined by popping from the stack, beginning from the last vertex removed and ending with the first vertex removed. The third one is different from the first two: *Color-Degree* repeatedly chooses a vertex of the largest color-degree, and assigns to it the smallest possible color (the colors are numbered), where the color-degree of vertex v is defined as the number of colors used so far on vertices adjacent to v. If two vertices have equal color-degrees, the one adjacent to the most uncolored vertices is chosen first.

For any particular instance of vertex coloring, it is difficult to determine which heuristic performs best. Indeed, no theoretical results are known that establish the superiority of any one heuristic for general graphs. However, our simulation results on randomly generated graphs show that *Color-Degree* is the best, and *Largest-First* is a little bit better than *Smallest-Last*.

In this paper, we focus on the partition-coloring problem rather than its application to wavelength assignment and routing. We refer the reader to [15, 21] for an overview of optical networks and to [1, 11, 14, 19, 20, 8, 9] for details regarding wavelength assignment problems.

3 Partition Coloring

Our partition-coloring problem is formally defined using the following notation.

- $G = (V, E)$ is a connected undirected graph.

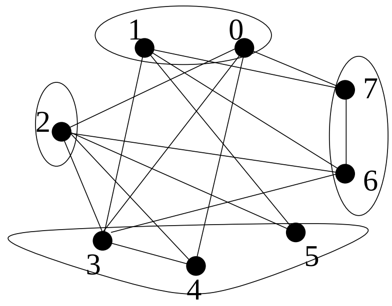

Figure 1: An example of partition-coloring

- V is partitioned into k disjointed subsets $V_1, ..., V_k$. That is, $V = V_1 \cup \cdots \cup V_k$, and $V_i \cap V_j = \emptyset, \forall i \neq j, i, j = 1, ..., k$.
- A subset of vertices $V^P \subseteq V$, is called a *feasible subset* if $|V^P \cap V_i| = 1, \forall i = 1, ..., k$.
- The subgraph induced from G by $V^P \subseteq V$ is defined as $G^P = (V^P, E^P)$, such that $e \in E^P$ if $e \in E$ and the endpoints of e are in V^P.

We are now in a position to formally define the partition-coloring problem: *find a feasible subset V^P of V, such that $\chi(G^P)$ is as small as possible.* Observe that we only need to color the feasible subset and not the entire set of vertices. Consider the example shown in Figure 1. Here, the graph $G = (V, E)$ has 8 nodes and 15 edges, where $V = \{0, 1, 2, 3, 4, 5, 6, 7\}$, $E = \{(0, 2), (0, 3), (0, 4), (0, 7), (1, 3), (1, 5), (1, 6), (1, 7), (2, 3), (2, 4), (2, 5), (2, 6), (3, 4), (3, 6), (6, 7)\}$. The vertices are partitioned into four groups. With our notation, $V_1 = \{0, 1\}, V_2 = \{2\}, V_3 = \{3, 4, 5\}, V_4 = \{6, 7\}$.

According to our formulation of partition-coloring, when the partition size is unity, the partition-coloring is equivalent to standard vertex coloring, which is NP-complete. Unfortunately, as the following claim shows, the problem remains NP-complete even if the partition sizes are forced to be larger than unity.

Proposition 1 *The partition coloring problem is NP-complete.*

Proof: Suppose the partition sizes are forced to be at least as large as some number c, the partition-coloring can be reduced to standard coloring by following construction: pick a standard coloring instance and to each vertex add $c-1$ more nodes and connect each of them to all other vertices to meet the partition size requirement. Note that the partition-coloring graph can be colored by K colors if and only if the original standard coloring instance can be colored by K colors. ∎

We now prove a transformational property that will help reduce the problem size by removing some edges from the graph. We say a graph is $k-partite$ if it is possible to partition the vertex set of G into k sets, such that every edge of G only connects two vertices in different partition sets.

Proposition 2 *Every instance of a partition-coloring graph can be reduced to a k-partite graph without changing the solution of the partition coloring problem.*

Proof: According to the partition-coloring requirement, we need to choose exactly one vertex in each group; other vertices in a group can be removed after this vertex is selected. Therefore, edges within a group cannot change the solution of the problem and thus we can remove all the edges that connect vertices in the same group. ∎

For example, in Figure 1, we can remove $(3,4)$ and $(6,7)$ two edges to simplify the graph without changing the solution of the problem. This "pre-processing" step reduces the number of edges in consideration. Once the vertex from each group is selected, another simple elimination can be performed to simplify the problem:

Proposition 3 *After a vertex is selected from a particular partition, removing the other vertices from the partition does not affect the solution to the problem with the rest of the graph.*

4 Two groups of algorithms

To design approximation algorithms for partition-coloring problem, we consider two additional heuristic assumptions beyond those already mentioned in Section 2:

- in a partition group, the vertices with lower degree are easier to color;
- the earlier the problem size is reduced, the better.

Based on the three standard vertex-coloring algorithms described in Section 2 and the above two additional heuristic assumptions, we design two groups of algorithms that we call *one-step* algorithms and *two-step* algorithms.

4.1 One step algorithms

In this group, we provide partition-coloring extensions of the three well-known standard coloring heuristics. We call these *onestepLF*, *onestepSL*, and *onestepCD*, where LF, SL, and CD stand for Largest-First, Smallest-Last, and Color-Degree, respectively, the names of the three heuristics in Section 2. In these algorithms, we choose the vertex with the smallest index for any tie-breaking. The colors are numbered arbitrarily starting with zero.

- *onestepLF*: First remove all edges whose vertices are in same group. Then, find the vertex with the smallest degree for each uncolored group. From these vertices (one for each group), chose the vertex with the largest degree. Assign the vertex with the smallest available color. Remove all other vertices in this group. Repeat until all groups are assigned one color. For example, applying this algorithm to the graph of Figure 1, we get following solution: $2(0) - 0(1) - 5(1) - 6(1)$, where we use the form $a(b)$ to represent vertex index a with color index b.
- *onestepSL*: First remove all edges whose vertices are in same group. Then find the vertex with the smallest degree in each uncolored group. From these selected vertices, find the vertex with the smallest degree. Push the vertex into a stack. Remove all vertices in this group. Repeat this process until all groups have one vertex selected. Then we color the vertices greedily by popping up vertices from the stack one by one. Applying onestepSL to the graph of Figure 1, for example, we get following solution. The vertex selection sequence is: $4-0-7-2$ and the coloring assignment is: $2(0) - 7(0) - 0(1) - 4(2)$.

- *onestepCD*: This algorithm derives from *Color-Degree.* As before, first remove all edges whose vertices are in same group. Then, find the vertex with minimal color-degree for each uncolored group. If there is a tie, choose the one with the smallest uncolored degree, where the uncolored degree of a vertex is defined as the number of uncolored adjacent vertices so far. From these vertices, find the vertex with the largest color-degree. If there is a tie, choose the one with the largest uncolored degree. Assign this vertex the smallest available color. Remove all other vertices in the same group. Repeat all these process until all groups are colored. For example, applying onestepCD to the graph in Figure 1, produces the coloring $2(0) - 1(0) - 4(1) - 7(1)$.

These three algorithms are natural extensions of *Largest-First*, *Smallest-Last*, and *Color-Degree* respectively except that we perform some pre-processing, choose the "best" vertex in each group, and remove vertices that are no longer useful in the course of the algorithm.

4.2 Two steps algorithms

The second group of algorithms is uses ideas from routing and wavelength assignment heuristics in WDM optical networks. Essentially, we first decide routes and then assign the wavelengths. In terms of the the partition coloring problem, we first choose the "best" vertex in each group, then apply the standard vertex-coloring algorithms. In keeping with the terminology of the previous section, we have *twostepLF*, *twostepSL*, and *twostepCD* respectively.

To choose the "best" vertex in each group, we use the two additional heuristic assumptions described earlier in this section. First we remove all edges whose vertices are in the same group – for example, the edges of $(3, 4)$ and $(6, 7)$ in Figure 1. Then from all un-processed groups, we choose the vertex with the smallest degree, and remove all other vertices in the same group as the selected vertex. Repeat this process until each group has exactly one vertex chosen. All these chosen vertices induce a subgraph of the original graph. For example, in Figure 1, the chosen vertices in each group are $4-6-1-2$; the induced subgraph is shown in Figure 2: Once the induced subgraph is constructed, we can use the standard coloring heuristics *Largest-First*, *Smallest-Last*, and *Color-Degree* to assign colors to each vertex. In the case of our example, these heuristics produce the following colorings: $2(0) - 6(1) - 1(0) - 4(1)$,

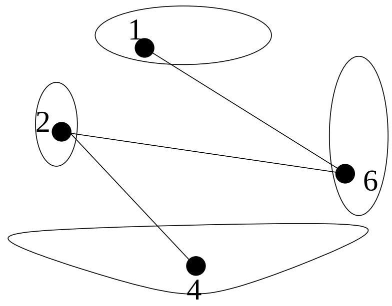

Figure 2: The induced subgraph of figure 1.

$6(0) - 2(1) - 4(0) - 1(1)$, and $2(0) - 1(0) - 4(1) - 6(1)$ respectively.

5 Numerical results

To compare the different algorithms, we performed simulations on random graphs with edge probability p (The so-called Model A graph – see [18]). Here, the vertices are given and an edge is created between every pair of vertices with probability p. Only connected graphs are selected from amongst these generated graphs.

First, we take the partition size to be $|V_i| = 2, i = 1, ..., k$, and let the number of groups (partitions) vary from from 100 to 800 in steps of size 100. Thus, the random graphs range in size from from 200 nodes to 1600 nodes, in steps of 200. The edge probability is taken to be $p = 0.5$. For each such size, we create 20 random graphs and compute the average number of colors required by the above two groups of algorithms.

Our simulation results (Figure 3) show that partition coloring follows the pattern of standard coloring: among the three most well-known approximation algorithms, on average, Color-Degree is the best, and Largest-First is a little bit better than Smallest-Last. However, for the two group of heuristics that we presented in last section, two-step works better than one-step for Largest-First and Smallest-Last approximations; but Color-Degree is different, on average: onestepCD is better than twostepCD.

Next, we fix the number of groups to be $k = 400$ and let the partition size change from 1 to 5.

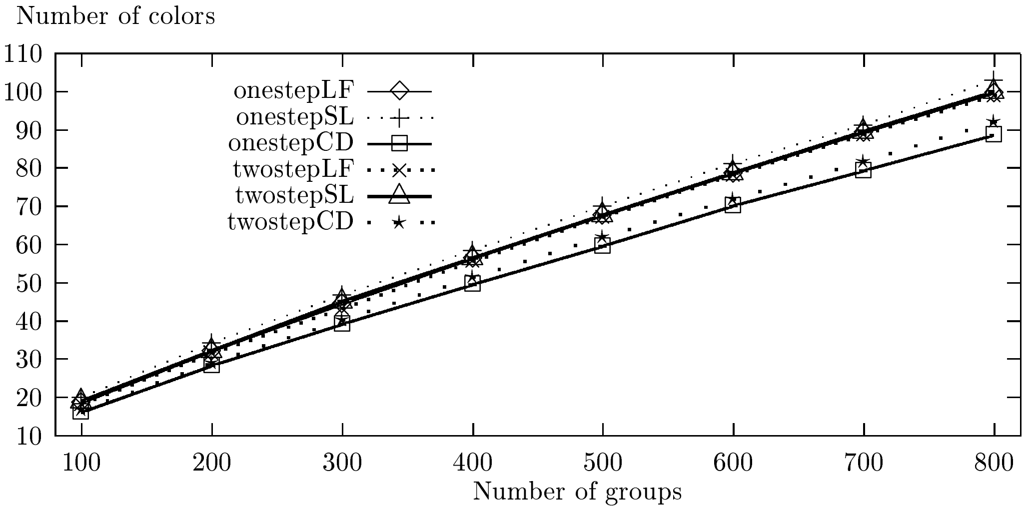

Figure 3: *Comparison of algorithms for random graphs. The partition size is* 2 *and random graphs are created with nodes ranging from* 200 *to* 1600 *with step-size* 200*; the edge probability is* 0.5*. Here, the* x*-axis represents the number of groups in random graphs and the* y*-axis is the number of required colors by the different approximation algorithms.*

Again, the random graph edge probability is $p = 0.5$. For each size, we run 20 random graphs and compute the average number of required colors. The results are shown in Figure 4.

Once again, the broad ranking of the heuristics is preserved across the different partition sizes. Since the edge probability is fixed, increasing the partition size will reduce the number of colors required, although the group size is fixed. This is because a large partition size provides more choice in selecting the vertices from groups. But the improvement decreases while the cost computation increases and so there is a tradeoff between performance gain and computation cost. For small problem sizes, such as group size less than 1000, a partition size of three or four appears sufficient.

We next consider fixing the total number of vertices in the graph and increasing only the partition size. We let the number of vertices of the random graph be 1000, the edge probability is 0.5, and partition sizes are taken to be $1, 2, 5, 10, 20$. For each instance, we run 20 random graphs and compute the average number of colors required. The results are shown in Figure 5.

As one might expect, the number of colors required decreases when we increase the partition size, across all algorithms. Amongst all the algorithms, onestepCD continues to perform the best. Also, we

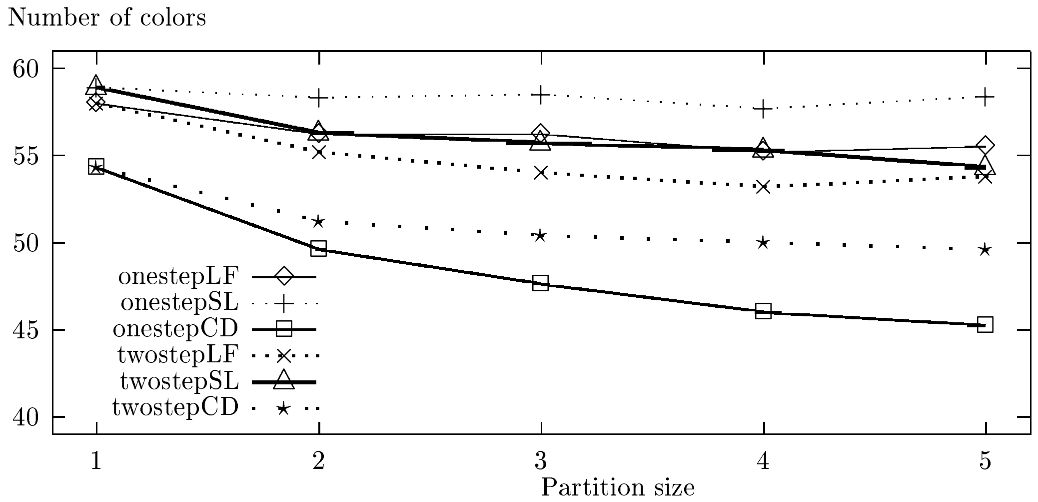

Figure 4: *Comparison of algorithms for random graphs with fixed number of groups* 400*. The x-axis represents different partition sizes and the y-axis is the number of required colors required by the different approximation algorithms.*

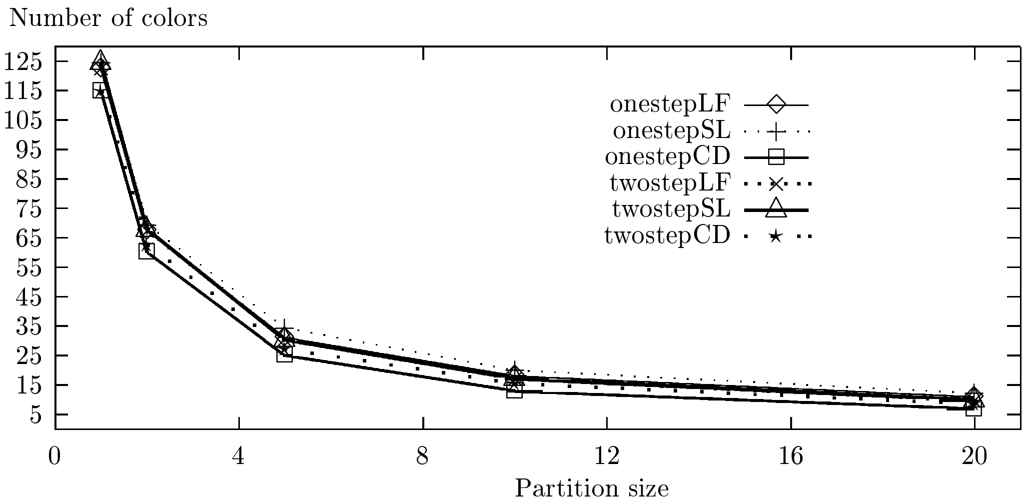

Figure 5: *Comparison of algorithms on random graphs of* 1000 *nodes and* 0.5 *edge probability. Here, the x-axis represents the partition size and the y-axis is the number of required colors required by the different approximation algorithms.*

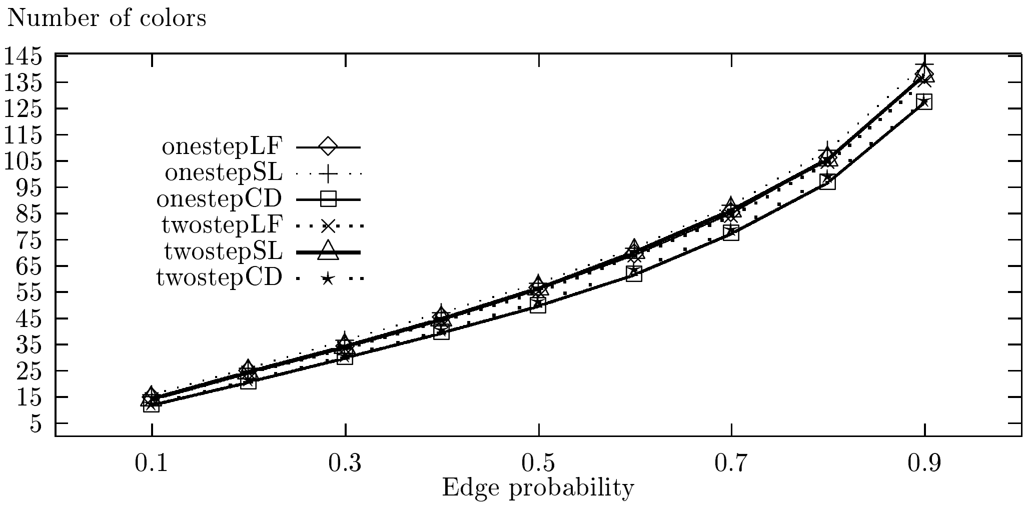

Figure 6: *Comparison of algorithms with partition size* 2 *and random graphs of* 800 *nodes. The x-axis represents edge probability in random graphs and the y-axis is the number of required colors required by the different approximation algorithms.*

observe that when the partition size is small, the gain is large. As we increase the partition size, the performance gain for the number of colors becomes small.

Finally, we consider changing the edge "density" by using various values of the edge probability. In the set of curves, we use a fixed partition size of 2 and and random graphs with 800 nodes. The edge probability is varied from 0.1 to 0.9 in steps of size 0.1. Notice that for edge probability 1, the result is trivial because each group needs one different color. Figure 6 shows the results.

For all algorithms, the number of colors required increases as we increase the random graphs' edge probability, as we might expect. When p is small, the number of colors increases slowly; when p is close to 1, the number of colors increases very fast. Although the difference of the number of colors is not large among our six algorithms, onestepCD is still the best.

6 Application to Wavelength Routing and Assignment

In this section, we apply the partition-coloring problem to solving the Routing and Wavelength Assignment Problem (RWAP) in optical networks. As mentioned earlier, most proposed solutions in the literature have separated the routing and assignment parts of RWAP. Most often, shortest path routing is used for the routing component. This creates a standard vertex coloring for the wavelength assignment part. While bounded heuristics are available for special networks such as the ring, solving the general network is equivalent to the general graph for the coloring problem. Our approach is to combine routing and wavelength assignment when the set of candidate routes between each sources destination pair is given. As we have shown earlier, this combined routing and assignment problem is equivalent to partition-coloring.

6.1 A mesh optical network

We first consider a mesh network with 10×10 nodes. As is typical, we assume that each node is capable of routing a lightpath from any incoming fiber into outgoing fiber without changing the wavelength. If two lightpaths (connections) share any one fiber, these two lightpaths must use different wavelengths. For our mesh simulation, we consider random source-destination connections. For each connection, we select routes from among a fixed set of candidate routes. This set of candidate routes is varied slightly to create different problem instances.

We assign a coordinate (x, y) to each node in the mesh: a row and a column number. Given source (x_s, y_s) and destination (x_d, y_d), we consider the following routing policies:

- *One-routing*: Here, the route set consists of only one route: first go along the row until the destination column, and then go along the column to the destination node.
- *Two-routing*: In this case, each route set has exactly two routes: one is the route above that proceeds along the row first. The second proceeds along the column first and then the row.
- *Four-routing*: In this case, each route set has four routes. First, the "median" node between the source and destination (x_m, y_m) is identified: $x_m = \lfloor (x_s + y_s)/2 \rfloor, y_m = \lfloor (y_s + y_s)/2 \rfloor$. Roughly, this is the node nearest the center of the rectangle identified with the source and destination as the corners. Then we pick four routes that traverse through the median node: row-column-row-column, row-column-column-row, column-row-row-column, column-row-column-row.

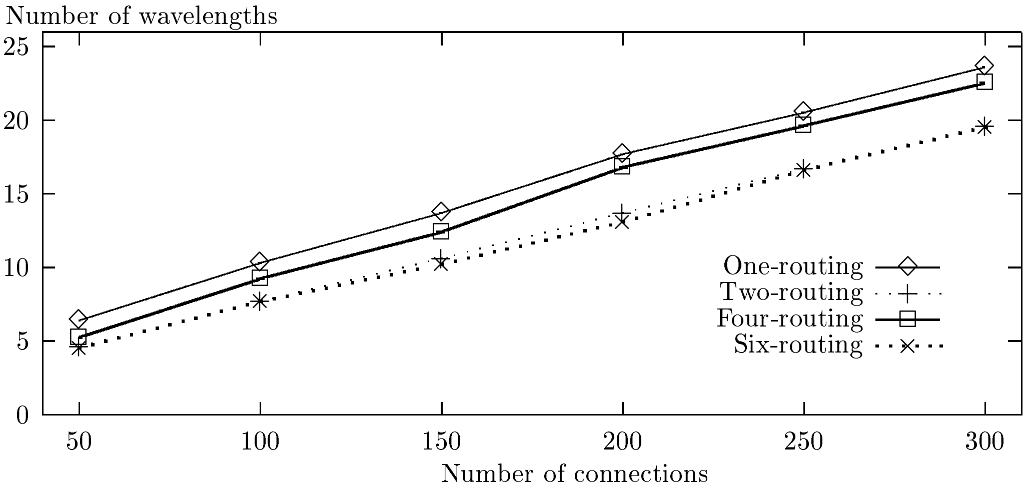

Figure 7: *A mesh network with* 10×10 *node. The x-axis represents the number of connections and the y-axis is the number of required wavelengths required by the different routing policies using onestepCD.*

- *Six-routing*: This case has six routes for each route set. These are the four routes in Four-routing combined with the two in Two-routing.

For the simulation, we first randomly create a set of connections. Then, the route sets are created according to the different routing policies described above. Following this, the instance of the optical problem is transformed into an instance of partition-coloring. Then we apply onestepCD on each partition-coloring instance to compute the number of wavelengths required because onestepCD is the best one among two groups of approximation algorithms. Again, each instance is evaluated by 20 simulation runs and the average value is reported in Figure 7.

From Figure 7, we see that routing policies *Two-routing* and *Four-routing* use much fewer wavelengths than policy *One-routing*, which is the policy (shortest-path routing) most often used in the literature. Although policy 4 is slightly better than policy 2, it requires much more computation time. What is surprising is that policy 3 performs worse than policy 2 inspite of using more paths. We believe the reason is that policy 3 has the tendency to pull all connections towards the center, such that the links in the center have higher load than links near the edges of the mesh. Then the total wavelengths required will increase. Another possible reason is that policy 3 uses more "turns" on each routing path than policy 2. It has been observed elsewhere that a routing policy that incurs more "turns" is less effective

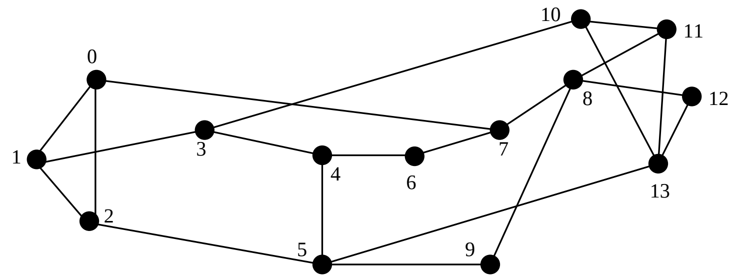

Figure 8: National Science Foundation network

[16]. From our results, we make the following conclusions:

- alternate routing is much better than single path routing;
- partition-coloring is an effective method to solve the alternate routing and wavelength assignment problem;
- a large number of routes does not necessarily improve the assignment, and in fact may increase computation cost significantly;
- convoluted routes (with many turns) may decrease the wavelength utilization on a mesh network.

6.2 NSFNET

We now apply our partition-coloring approach to the NSFNET topology, shown in Figure 8, a 14-node network with 21 edges. Each edge represents a pair of fibers, one in each direction. To construct route sets for each source destination pair, we use the k-shortest paths between each source and destination. When $k = 1$, the routing reduces to shortest path routing, and we use standard vertex-coloring to compute the number of wavelengths required; when $k > 1$, we use partition-coloring algorithms to compute the number of wavelengths.

In this example, we randomly create a set of connections and compute the k-shortest paths for each connection, where $k = 1, 2, 3$. For each k, we construct path-conflict graphs and apply partition coloring algorithms, onestepCD and twostepCD, on these path-conflict graphs to compute the number

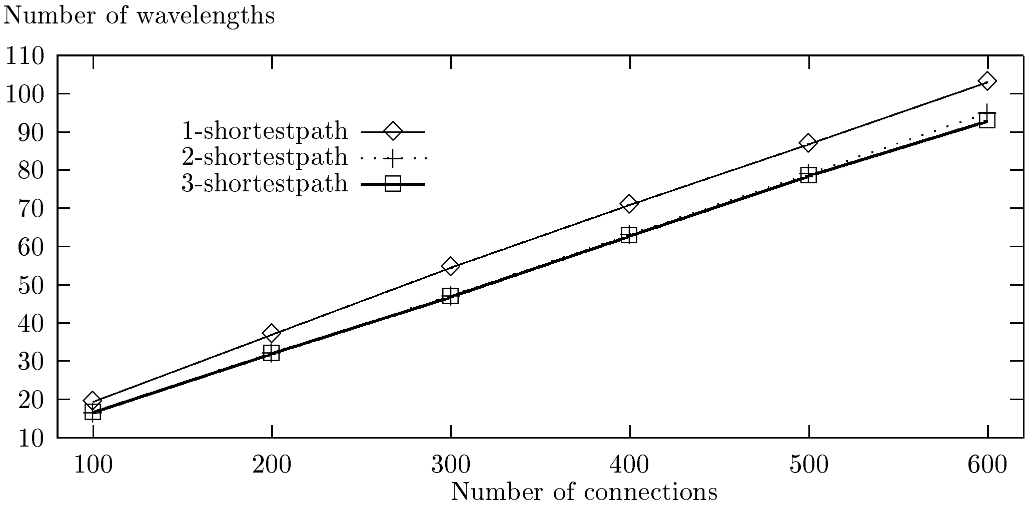

Figure 9: *NSFNET with k-shortest path routing. The x-axis represents the number of connections and the y-axis is the number of required wavelengths for* $k = 1, 2, 3$.

of wavelengths required and take the best result. Each instance is evaluated with 20 simulation runs and the average value is reported in Figure 9.

From Figure 9, we see that $k = 2$ and $k = 3$ use significantly fewer wavelengths than shortest-path routing ($k = 1$), because alternate paths provide more flexibility. For $k = 3$, the improved performance is quite small, while requiring much more computation time. Thus, in general, there is a tradeoff between performance and computation time.

7 Summary

In this paper, we defined a new coloring problem, partition-coloring, that originated from considering alternate paths in the routing and wavelength assignment for WDM all-optical networks. We reviewed the three best known approximation algorithms for standard vertex-coloring and extended them to the partition-coloring problem. Our simulation results show that *Color-degree* performs the best, in particular, our extension of it, *onestepCD*. When the size of the problem is not very large, a small partition size (2 to 4) works very well, thereby suggesting that only a few alternate routes are sufficient.

References

[1] B. Beauquier, J.C. Bermond, L. Gargano, P. Hell, S. Perennes, and U. Vaccaro, Graph problems arising from wavelength routing in all optical networks, *Proc. 2nd Workshop in Optics and Computer Science*, Geneva, Switzerland, April 1997.

[2] D. Brelaz, New methods to color the vertices of a graph, *Comm. ACM*, 22, 1979, pp251-256.

[3] J. Brown, Chromatic scheduling and the chromatic number problem, *Management Science*, 19, 1972, pp456-463.

[4] N. Christofides, An algorithm for the chromatic number of a graph, *The Computer Journal*, 14, 1971, pp38-39.

[5] A. Curtis, M. Powell, and J. Reid, On the estimation of sparse jacobian matrices, *J. Inst. Maths. Appl.*, 13, 1974, pp117-119.

[6] M. Garey, D. Johnson, and H. So, An application of graph coloring to printed circuit testing, *IEEE Trans. on Circuits and Systems*, 23, 1976.

[7] M. Garey and D. Johnson, Computers and Intractability, *Freeman Publishing*, 1979.

[8] G. Li and R. Simha, New results on wavelength assignment problems on ring and tree optical networks, *In proceeding of ACM southeast regional conference'99*, Mobile, AL, April, 1999.

[9] G. Li and R. Simha, On bounds for the wavelength assignment problem on optical ring networks, *Journal of High Speed Networks*, accepted, 1999.

[10] R. Gould, Graph Theory, *The Benjamin/Cummings Publishing Company, Inc.*, 1988.

[11] E.J. Harder, Routing and wavelength assignment in all-optical WDM wavelength-routed networks, PhD thesis, *George Washington University*, May 1998.

[12] R. Karp, Reducibility among combinatorial problems, *Complexity of computer computations*, Plenum Press, New York, 1972, pp85-104.

[13] D. Matula, G. Marble, and J. Isaacson, Graph Coloring Algorithms, *Graph Theory and Computing*, Academic Press, New York, 1972.

[14] B.Mukherjee, D. Banerjee, S. Ramamurthy, and A. Mukherjee, Some principles for designing a wide-area WDM optical network, *IEEE/ACM Trans. Networking*, vol. 4, no. 5, October 1996, pp.684-696.

[15] B. Mukherjee, Optical networks: a practical perspective, Mc-Graw-Hill, 1997.

[16] B. Narahari, S. Shende, R. Simha and S. Subramanya. Routing and Scheduling I/O Transfers on Wormhole-Routed Mesh Networks. *J. Parallel and Distributed Computing*, Vol. 57, No. 1, Apr 1999, pp. 1-13

[17] O. Ore, The four color problem, Academic Press, New york, 1967.

[18] E.M.Palmer. Graphical Evolution: An Introduction to the Theory of Random Graphs. *Wiley*, New York, 1985.

[19] P. Raghavan and E. Upfal, Efficient routing in all-optical networks, *Proc 26th ACM Symp. on Theory of Computing*, 1994, NewYork, pp.134-143.

[20] R. Ramaswami and K.N. Sivarajan, Routing and wavelength assignment in all-optical networks, *IEEE/ACM Trans. Networking*, vol. 3, no. 5, October 1995, pp.489-500.

[21] R. Ramaswami and K.N. Sivarajan, Optical communication networks, Morgan-Kaufmann, 1998.

[22] A. Tucker and L. Bodin, A model for municipal street sweeping operations, *Case studies of applied mathematics*, Math. Assoc. of America, 1976, pp251-295.

[23] D. Welsh and M. Powell, An upper bound to the chromatic number of a graph and its application to time-table problems, *Comp. J.*, no. 10, 1967, pp85-86.