Would Diversity Really Increase the Robustness of the Routing Infrastructure against Software Defects?

Juan Caballero, Theocharis Kampouris, Dawn Song, Jia Wang

February 6, 2007
CMU-CyLab-07-002

CyLab
Carnegie Mellon University
Pittsburgh, PA 15213

Would Diversity Really Increase the Robustness of the Routing Infrastructure against Software Defects?

Juan Caballero*, Theocharis Kampouris*, Dawn Song*, Jia Wang†
*Carnegie Mellon University
{juanca@ece, tkampouris@ece, dawnsong@ece}.cmu.edu
†AT&T Labs-Research
jiawang@research.att.com

Abstract

Today's routing infrastructure exhibits high homogeneity. This constitutes a serious threat to the resilience of the network, since a bug or security vulnerability in an implementation could make all routers running that implementation to become simultaneously unusable. This situation could arise as a defective software upgrade or a denial-of-service attack.

Diversity has been proposed as a solution to increase the resilience to software defects, but the benefits have not been clearly studied. In this paper, we use a graph theoretic approach to study those benefits, addressing three fundamental questions: 1) how to measure the robustness of the network to such failures; 2) how much diversity would be needed for a certain degree of robustness; and 3) how to best use the available diversity.

We find that a small degree of diversity can significantly increase the robustness of the network against software defects. We observe that for small networks, partitioning the network into contiguous regions that use the same implementation, works best. For large ISP networks, where routers usually have roles, the best approach is to apply diversity to each role separately. We evaluate our approach on multiple real ISP topologies, including the topology of a Tier-1 ISP.

1 Introduction

Today's routing infrastructure exhibits high homogeneity. A few router vendors dominate the market [20] and each ISP relies on a very limited number of vendors to build their infrastructure, often a single one. Simplified network operation, uniform operator training, reduced complexity and interoperability are common reasons behind this homogeneity. However, such homogeneity raises serious questions about the resilience of the routing infrastructure against software defects. Computer systems are notoriously known for being laid with bugs and vulnerabilities, and routers are no exception.

Software defects in router software are not uncommon. Markopoulou et al. [21] found that 12% of all failures in a Tier-1 ISP network were router related, and a NIST-funded survey [24] in 2002 found that in financial services approximately 16% of reported major bugs were attributed to router and switch problems. Most vendors acknowledge these problems and provide search tools or toolkits to find the flaws in a specific software version [2, 6, 14]. Vulnerabilities in router software are also frequent and can allow denial-of-service attacks [9, 10, 15], remote execution of system-level commands with no authentication [5], unauthorized privileged access [30], or possible remote shell execution [11].

Given the reliance of ISPs on single vendors and the common use of a shared code base among different implementations from the same vendor, a real threat lurks. A serious bug in a router software implementation could make all routers running that implementation to become simultaneously unusable[1]. Due to the router homogeneity in ISPs and enterprise networks, this would effectively disconnect their customers/sites and halt their operation, dealing a major blow to their business and their perceived quality. This dramatic scenario could appear as a result of a defective software upgrade, or a security vulnerability that opens the door for a denial-of-service attack on the routing infrastructure.

Network administrators thus face a dilemma on introducing router software diversity in their networks, i.e.

[1] A router is a combination of hardware and software. In this paper we focus on software failures, which are more likely to simultaneously affect all routers running an implementation, but our approach can also protect against similar hardware failures.

using different software implementations from different code bases on different routers. Diversity could increase the overall resiliency of a network against software defects but it would also increase the complexity and cost associated with network deployment and management. Thus, in order to take this step, we need to understand to what extent diversity can increase the robustness of a network and what is the best way to apply the available diversity.

In this paper when we refer to *network robustness* or simply to *robustness* we mean the property of the network staying connected under a software failure that simultaneously disables all routers running a specific implementation. To the best of our knowledge, this is the first paper which systematically studies the effectiveness of using software diversity to increase the resiliency of the router infrastructure against software defects and to address the following three fundamental questions: 1) How do we measure the robustness of a network under such failures? 2) How much diversity is needed to guarantee a certain degree of robustness? 3) How should the available implementations be used to maximize robustness?

There are many factors contributing to the resiliency of a network. As a first step, in this paper we consider the most fundamental property to guarantee the resiliency of a network, the *connectivity*. Connectivity is the most essential property that needs to be guaranteed for a network to operate, without connectivity no routing is possible and no quality-of-service can be guaranteed. A good example was the recent Sprint network incident on January 9, 2006 when failure of two links led to a network partition [35]. Other properties such as network capacity, which will be reduced due to the failure and might lead to an increase in delay and packet loss, are left as future work.

To study whether the network topology will stay largely connected or will be partitioned into small connected components, when all routers running a certain implementation become unusable, we propose a graph theoretic approach and convert the problem into a version of a graph coloring problem, where routers are colored based on their implementations and connectivity needs to be maximized when any color is removed from the graph. A color failure (i.e. removing a color) is equivalent to disconnecting all nodes using that color and has two impacts: the disconnection of nodes themselves, which cannot be avoided, and the disconnection of other nodes who relied on those nodes to connect to the rest of the graph. Our goal is to minimize this second factor so that a network is as robust as a fully connected topology (i.e. full mesh), since removing any number of nodes from a fully connected topology does not affect the connectivity of the remaining nodes.

In this paper, we propose a number of coloring algorithms and evaluate them over real topologies, including the topologies obtained from a large Tier-1 ISP[2] and from the Rocketfuel project [31]. Our results demonstrate that we can significantly enhance the robustness of an ISP network against software defects, using our coloring algorithms, without increasing the amount of existing diversity in the network. Furthermore, we show that under certain constraints it is possible to apply diversity to achieve robustness near to the optimal one that can be achieved with a fully connected topology. Our contributions are summarized below.

Convert the introduction of diversity into a graph theoretic problem: We show how to convert the problem of introducing router diversity to increase the network resiliency against software defects into a version of a graph coloring problem, where the goal is to maximize a robustness metric when faced with a color removal in the graph.

Measure the robustness of the network topology upon color failures: There has been significant work on graph robustness and related metrics. Our contribution here is to lay out a step by step process to calculate the robustness of the network upon color failures, eliminating the confusion about multiple metrics and procedures. When different metrics are possible we state the properties we should require of those metrics.

Design and implement coloring algorithms: We have designed and implemented multiple coloring algorithms, grouped in five different families. Our algorithms take advantage of existing techniques to provide resiliency to the network such as router and geographical redundancy, and aim to maximize the robustness of the network to color failures, given the available diversity.

Identify the degree of diversity needed to achieve a certain degree of robustness: We have evaluated our coloring algorithms over multiple real ISP topologies. We find that a good coloring algorithm allows us to restrict the impact of a color failure to the disconnection of the routers running that implementation, with minimal impact on other routers. Thus, if a color failure disconnects half the routers from a network that uses two implementations, we can still expect to have the remaining half connected in a single component.

Account for real network constraints: We have extended the basic problem to take into account different router roles that might limit the implementations that can be run by each router. For example in large ISP networks, backbone routers might not be able to run the same implementation as access routers. We have also accounted

[2]The name of the ISP is omitted for anonymization reasons.

for different node importance in the form of node weights that can be assigned by network administrators and can for example represent the number of customers connected through the router or the amount of traffic it carries.

The remainder of the paper is organized as follows. Section 2 defines our graph theoretic approach. In Section 3 we present the metrics needed to evaluate the robustness of the network. Then, in Section 4 we propose different graph coloring algorithms and in Section 5 we evaluate them over different topologies. Section 6 presents the related work. Finally, we discuss extensions and future work in Section 7 and conclude in Section 8.

2 Problem Statement

In this section, we convert the problem of using router software diversity to increase network robustness into a graph coloring algorithm. We represent a given network topology as a graph $G = (V, E)$, where V is the set of nodes representing routers in the network, and E is the set of undirected edges each corresponding to a link between two routers in the network. Let $n = |V|$ be the number of nodes, and $m = |E|$ be the number of edges in G. Let $C = \{c_1, c_2, \ldots, c_k\}$ be a set of distinct colors representing the available implementations that any router can utilize. We say a *vertex-coloring algorithm* takes as input the graph G and the color set C with k distinct colors, and outputs a *colored graph* $G_k = (V, E)$ where each node in V has been tagged with a color from C. Thus, the coloring algorithm determines which implementation should be run by each router, by assigning a color to each router in the network.

A *color failure* is the simultaneous failure of all the routers running a specific implementation, which makes those routers unusable. This color failure can be caused by a defective router software upgrade, or a denial of service attack on the routing infrastructure. A color failure is represented as a *color removal*, which takes as input the colored graph G_k and a color C_i and returns a *color-removed subgraph* $G_k^i = (V, E^*)$ with $E^* \subseteq E$, which is the subgraph of G_k generated by *disconnecting* all nodes of color C_i from G_k, where disconnecting a node means removing all edges connected to that node while leaving the node in the graph. Intuitively, the color-removed subgraph represents the remaining network topology after a color failure.

Problem definition: Given a graph $G = (V, E)$ with n nodes and m edges, a set C with k distinct colors, and a robustness function $\phi(\cdot)$, find a k-color vertex-coloring of graph G that maximizes $\phi(\cdot)$.

Our problem is different from the most common graph coloring problem, which tries to find a *proper coloring*, meaning no two adjacent nodes are assigned the same color [12]. We do not require adjacent nodes to have different colors. We try to find a coloring of G that maximizes a robustness function ϕ, where ϕ is computed on the subgraphs of G obtained by removing the colors, one at a time.

Note that the probability of finding a software defect in one implementation should be independent of the probability of finding a defect in any other implementation. Thus, the implementations used to add diversity to the network should come from different code bases. This often means different vendors, since the different versions from the same vendor usually share a significant code base.

To summarize, our problem is composed of two main parts. First, how to define the function ϕ, such that it measures the robustness of the graph upon a color removal. We address this in Section 3. Second, how to design an effective vertex-coloring algorithm, that achieves high robustness for a given graph with a limited number of colors. We address this in Section 4.

3 Measuring Network Robustness

In this section we introduce the metrics used to describe the robustness of a graph under color failures. First, we describe connectivity metrics that can be used on any graph, and then we define the robustness metric on a colored graph, as a function of a connectivity metric.

3.1 Connectivity Metrics for a Graph

We use the connectivity of the graph as a measure of the robustness of the network topology. There have been numerous previously proposed connectivity metrics for a graph such as: number of components in the graph, normalized size of the largest component [28], pair connectivity [27], average size of the components (excluding the largest component) [28], average distance [28], number of biconnected components [37], minimum cut-set size for a balanced bi-partition of the graph [33], and effective diameter [26].

But we are not aware of any in-depth comparison of these connectivity metrics that allows to select one of them. Thus, a significant part of our work has been to identify the important characteristics that a connectivity metric should possess. In this paper, we require the following characteristics to be satisfied by a connectivity metric: 1) it has to be non-negative; 2) it should monotonically decrease as more nodes are disconnected; 3) it needs to be normalized; 4) it should represent only binary

connectivity, that is, if two nodes are connected or not, rather than try to measure the degree of connectivity between them, and 5) it should be as intuitive as possible.

Note that some of the above metrics, such as those using shortest path computations or hop counts, attempt to measure not only if two nodes are connected, but also the degree of connectivity between them, but in doing so introduce an assumption about shortest path routing being used in the network, which is not desirable. We believe the proper approach is to first guarantee the physical connectivity of the network before investigating the impact on routing and end-to-end performance. We select the following two connectivity metrics because they satisfy the above requirements. Note that, the robustness function is independent of the connectivity metrics used and other metrics can thus be selected.

The *normalized size of the largest component (NSLC)* [28] is the size of the largest component over the total number of nodes. Ideally, when removing a color from the graph, the remaining nodes should all form a single component and thus be all reachable to each other.

The *pair connectivity (PC)* [27] is the fraction of connected node pairs in a graph over the total number of distinct pairs of nodes. It measures for each node, how many other nodes it can reach. This metric takes into account all the components in the graph rather than just the largest one and it is useful when the graph is partitioned into multiple large components where each component connects a significant number of nodes and should not be ignored.

$$PC = \frac{\sum_{i=1}^{comp} \frac{1}{2}|C_i|(|C_i|-1)}{\binom{n}{2}}$$

where $comp$ is the number of weak components in the graph and $|C_i|$ is the number of nodes in each component. Note that we need to first extract the weak components of the graph before calculating any of the two metrics, which takes $O(n+m)$.

Modeling node importance: So far, our connectivity metrics consider each node in the graph to be equally important. However, for a given graph, some nodes can be more important than others. For example, a node which carries more traffic would be considered more important than those carrying less traffic. Alternatively, a node which connects to more customers or has more capacity can be considered important. In this paper, we assign different weights to different nodes in the graph to reflect their importance.

[2]A weak component is a set of nodes that are connected, regardless of the direction of the edges that connect them, as opposed to a strong component that requires a directed path between the nodes.

In our setting, we do not need to use link weights because we only deal with node, rather than individual link, failures and in this case node weights allow dealing with link weights in a straightforward way. Since a node failure implies the failure of all the links connected to the node, then a node can be assigned a weight proportional to the sum of the weights of all the links it connects to. We now extend our connectivity metrics to the case where nodes are associated with weights.

The *weighted normalized size of the largest component (W-NSLC)* is the sum of the weights of all nodes that belong to the largest component, over the total weight of all nodes in the graph.

The *weighted pair connectivity (W-PC)* takes into account the sum of the products of all the node weights that belong to the same weak component.

$$\textit{W-PC} = \frac{\sum_{i=1}^{comp} \sum_{j \in C_i} w_j (\sum_{k \neq j; k \in C_i} w_k)}{\sum_{i=1}^{n} w_i (\sum_{j \neq i} w_j)}$$

where $comp$ is the number of weak components in the graph and C_i is the ith weak component of the graph.

Disconnecting a node: To generate G_k^i from G_k, we disconnect all nodes with color i by removing its edges but we do not remove the nodes themselves from the graph. The reason is that this allows a fair comparison between the connectivity of G_k and G_k^i. Figure 1 illustrates an example. There are two alternative ways to generate G_2^1 from G_2, disconnecting the black node (case A) or removing it (case B). Table 1 shows the connectivity metrics for both cases. Note that the connectivity for case B is the same as the connectivity for the original colored graph, which is misleading. However, in case A the connectivity reflects the node removal, and as a result, the normalized size of the largest component is 0.66.

3.2 Robustness Metrics for a Colored Graph

Intuitively, the robustness of a colored graph measures the remaining connectivity of the colored graph when a color is removed. Given a colored graph G_k and a connectivity metric, we define two robustness metrics: 1) the average of the connectivity of the subgraphs created by disconnecting all nodes of a specific color, and 2) the minimum connectivity among all the subgraphs created by disconnecting all nodes of a specific color. Thus, given the colored graph G_k and a connectivity metric $f(\cdot)$, we first obtain all color-removed subgraphs G_k^i by removing each color C_i in turn, where $i \in [1,k]$. Then, we compute the robustness of G according to the connectivity measure f as follows.

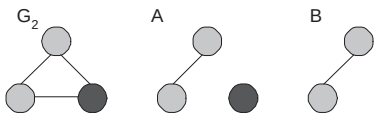


Figure 1: Two alternatives to generate G_2^1 from G_2: disconnecting the black node (A) or removing it (B)

Metric	G_2	**A**	**B**
Pair Connectivity	1	0.33	1
Normalized Size of Largest Component	1	0.66	1

Table 1: Connectivity metrics for both possible alternatives to generate G_2^1 from G_2

$$\phi_{G,f}^{avg} = \frac{1}{k} \sum_{1 \le i \le k} f(G_k^i) \quad \phi_{G,f}^{min} = \min_{1 \le i \le k} f(G_k^i)$$

The average robustness is the most straightforward measure of robustness, representing the expected robustness under a color failure. But, if taken alone, it can sometimes be misleading because some colored graphs can present good average robustness but bad minimum robustness. Since, a priori we do not know what the probability of failure is for each of the implementations, we need to try to protect against failures of *any* of the implementations. That is precisely what the minimum robustness gives us: the worst case.

One scenario in which we might get good average robustness and bad minimum robustness is when the coloring is unbalanced, that is, most nodes in the graph use one color and the rest split the remaining colors. Figure 2 shows that scenario in the Abilene backbone network with two colors. The average robustness is quite good, 0.5 for the normalized size of the largest component, but the minimum robustness is only 0.18. The average is good because the removal of the black color gives a very high connectivity that compensates the poor connectivity when the gray color is removed. Such an unbalanced coloring creates a situation where if the grey color fails the graph becomes almost unusable. This situation is clearly undesirable and we will favor coloring algorithms that produce balanced colorings.

4 Coloring Algorithms

In this section, we propose coloring algorithms which assign a color, from a set of k available colors, to each node in the graph. Finding an optimal coloring of a graph is known to be NP-complete. We have devised a total of 10 approximation algorithms which can be classified into 5 different families: *randomized coloring*, *importance coloring*, *redundancy coloring*, *region coloring*, and *role coloring*.

4.1 Randomized Coloring

The randomized coloring works as follows: given a color set C with k different colors, for each node in the graph, choose a color from C at random and assign it to the node. Randomized coloring is simple and fast. Its main problem is that it usually has a bad minimum robustness. In the worst case, every node will choose the same color, although the probability that something like this will happen is extremely low (i.e., $(1/k)^{n-1}$). More generally, it can produce very unbalanced colored graphs and that results in bad minimum robustness as shown in Section 3.2. Appendix B shows theoretical results for randomized coloring when applied to the Internet AS-level topology, rather than the router-topology within an AS. More importantly, randomized coloring does not take advantage of the topology information of the graph. We now investigate three different families of deterministic algorithms that use topological properties of the graph to color their nodes.

4.2 Importance Coloring

Importance coloring is based on the following premise: identify important nodes in the topology (e.g., nodes that will affect the connectivity of the graph the most) and assign them different colors. The intuition behind it is that even if some important nodes may be taken out due to a color failure, other important nodes will stay unaffected and hence keep the network largely connected. The algorithm works as follows: 1) order the set of colors randomly, 2) order the set V of nodes according to their importance, and 3) iterate through nodes in descending order of importance till all nodes have been colored, assigning the first color c_1 to the most important node, c_2 to the second most important node, etc. When all k colors have been assigned, c_1 will be assigned again to the (k+1)th most important node.

There have been several previously proposed metrics for topological node importance: node degree, betweeness centrality [8], router utilization [17], likelihood [17], and assortativity coefficient [23]. In this paper, we select *node degree* and *betweenness centrality* as representatives of the family. Node degree orders nodes by the number

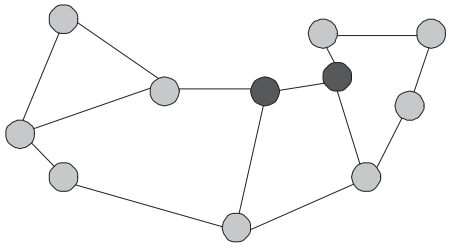


Figure 2: An unbalanced coloring of the Abilene backbone using two colors

Connectivity Metric	G_2	G_2^1	G_2^2
Pair Connectivity (PC)	1	0.018	0.65
Normalized Size of Largest Component (NSLC)	1	0.18	0.82

Robustness Metric	G_2
Ave. Robustness (NSLC)	0.5
Min. Robustness (NSLC)	0.18
Ave. Robustness (PC)	0.34
Min. Robustness (PC)	0.018

Table 2: Connectivity and robustness metrics for the colored graph of the Abilene topology

of outgoing edges from a node, while betweeness centrality orders nodes by the number of shortest paths passing through a node [8]. We stress that using betweeness centrality as importance metric does not mean that shortest path routing is assumed. Our approach is independent of the routing policies and routing protocols used in the network. One important difference between the two importance metrics is the computation complexity for ordering the nodes. Computing the node degree in a graph with n nodes and m edges takes $O(n+m)$, while computing the betweeness centrality takes $O(n^2+nm)$.

Furthermore, we have considered two different versions of these algorithms: *static* and *dynamic*. In the static version, the algorithm is as described above and the node ordering is computed once. The dynamic algorithm is similar but every time a node is colored, it is removed from the graph and the node ordering is recomputed using only the remaining nodes. Thus, the dynamic versions are more expensive, requiring $O(n^2+nm)$ and $O(n^3+n^2m)$ for computing node degree and betweeness centrality, respectively.

4.3 Redundancy Coloring

Redundancy coloring is based on a backup premise. A *backup node* of node A is a node B that would take over the responsibilities of A if A fails. Redundancy coloring works as follows: 1) order nodes based on their importance, and 2) iterate through the nodes in decreasing order of importance, assigning a backup node to each node and coloring the node and its backup with different colors.

More precisely, we define the backup node of a node A as the most *similar* node $B \neq A$ in the graph. To measure similarity between nodes, we use a similarity metric employed in social networks: *Structural Equivalence* [18]. Two nodes are said to be structurally equivalent if they connect to the exact same set of neighbor nodes. Intuitively, equivalent nodes are perfectly substitutable, thus we want to assign them different colors. Note that, this exactly reflects a common practice used in today's network design: nodes that are considered more important, are sometimes made redundant. However, the structural equivalence between two nodes is a boolean function. Since we need to find the most similar node in the graph to use as backup node, we propose a generalization of the structural equivalence metric to model the case where two nodes are not completely similar. We define *Node Redundancy NR(A,B)* as

$$NR(A,B) = \frac{|N_A \cap N_B|}{|N_A|}$$

where N_A and N_B are the sets of neighbors for nodes A and B. Note that, the node redundancy function is asymmetric, i.e. $NR(A,B) \neq NR(B,A)$.

In this paper, we have implemented two versions of this algorithm: *Redundancy-Next* and *Redundancy-Random*. The difference is that, when the most similar node for the current node is already colored, we can either select the next node in similarity (Redundancy-Next) or color the current node with a random color different from the one already used by the most similar node (Redundancy-Random). The pseudocode for both algorithms is provided in Appendix A.

4.4 Region Coloring

Region coloring tries to divide the network into contiguous regions and color each region with the same color. This is somewhat similar to a geographic coloring but with the difference that the regions are found automatically, i.e., no geographical information needs to be provided with the graph. Thus, the intuition behind region coloring is to try to isolate the failure to a contiguous region of the graph. Since networks are designed with geography in mind, then removing a particular region of the graph has little additional effect on the rest of the graph,

other than the disconnection of the nodes themselves.

In fact, this problem is related to the *graph clustering* and *graph partitioning* problems. Graph clustering algorithms try to find peninsulas of connectivity, i.e., regions with a high density of intra-regional links and a few inter-regional links connecting them with other regions. Graph partitioning algorithms usually employ a similar approach, but the goal is to split the network into balanced partitions, i.e., partitions with a similar number of nodes. As we will see in Section 5, a good algorithm from this family should generate balanced and contiguous partitions. Unbalanced or discontiguous partitions would provide a less robust coloring of the graph.

4.5 Role Coloring

So far, all our coloring algorithms have assumed that all nodes in the network can use any of the k colors in C. However, the role that a router plays in a network will pose further constraints on the implementation the router may use. For example, backbone routers and access routers might be built with different design goals and using very different technology, i.e. a backbone router might not be able to use the implementation of an access router and vice versa. Role Coloring assumes that each node in the graph is appended with a role tag. We consider two main roles: *access routers* and *backbone routers*. Each role has a different color set: *access color set* C_A and *backbone color set* C_B. and a node can only be assigned a color from the color set corresponding to its role. Role coloring works as follows: 1) color all backbone routers, since they provide connectivity to the access routers, and 2) color the access routers, which provide connectivity to the customers.

Role Coloring can be combined with any of the previous coloring algorithms. Here, we present it in conjunction with the Partition algorithm since our experiments have shown this to be the best combination.

Coloring backbone routers: We create the *backbone graph* by removing all access routers and their corresponding links from the original graph. Then, we consider three cases in coloring the backbone graph: 1) if every backbone router has a structurally equivalent neighbor, then color each router in a structurally equivalent pair differently, 2) if no backbone router has a structurally equivalent neighbor, then use region coloring, and 3) if only some backbone routers have structurally equivalent neighbors, then color each node in a structurally equivalent pair differently and use region coloring but imposing the constraint that two nodes previously colored differently need to belong to different regions.

Coloring access routers: We create the *access graph* by collapsing all the backbone nodes into a single node, that connects to all access routers. We consider two cases: 1) if the probability of a failure in C_A is independent from the probability of a failure in C_B, for example because they come from different vendors or code bases, then we evenly split the colors in C_A among all access routers to create a balanced colored graph, and 2) if failures in access and backbone colors cannot be assumed independent, then for each access router, if all backbone routers connecting to it have the same color, then the access router is assigned a different color, if available. Otherwise, there is no constraint on the color assigned to the access router.

Note that ISPs might sell, at an additional cost, redundant connections to their network, that is, a client could connect to two different access routers for redundancy reasons and be charged an extra fee for the service. In this case, we might have another constraint when coloring the access routers as the access routers to which that client connects should be colored differently for additional protection.

5 Evaluation

5.1 Experimental Setup and Methodology

The network topologies used in the evaluation are presented in Table 3. All the topologies are router-level topologies, where each node represents a router and each edge represents a link between routers. The table is divided into three parts. The top part shows real ISP topologies that we have access to, including the topology of a Tier-1 ISP; the middle shows Rocketfuel topologies [31] also used in our experiments; and at the bottom, as a base case, we present a synthetic topology which is a full mesh of 100 nodes. All topologies have a single connected component. If there are multiple edges (e.g., parallel links) between a pair of nodes, we collapse them into a single edge.

The Cenic and Abilene topologies are obtained from the public information available on their websites [1, 4]; the Tier-1 ISP topology is generated from a snapshot of router configurations taken by the ISP and the Rocketfuel topologies are taken from [29]. These topologies were obtained by the authors through external probing. Though they are known to contain some inaccuracies, we consider them a good approximation of a router-level ISP topology and accurate enough to test our algorithms. A noteworthy case is the Level3 topology, which is highly interconnected. As explained in [31] this is likely due to the use of virtual circuits. We mainly use the Level3 and Mesh topologies to understand how the coloring algo-

Topology	Date	Nodes	Edges
Tier-1 ISP	Oct 2006	A few hundreds	A couple of thousands
Cenic	Aug 2006	51	91
Abilene	Sep 2006	12	15
Exodus	Jan 2002	201	434
Sprint	Jan 2002	604	2268
Level3	Jan 2002	624	5298
Verio	Jan 2002	960	2821
Mesh	N/A	100	4950

Table 3: Network topologies used in the evaluation (including real topologies, Rocketfuel topologies, and a synthetic full mesh).

Algorithm	Family
Random	Randomized Coloring
Static Degree	Importance Coloring
Static Load	Importance Coloring
Dynamic Degree	Importance Coloring
Dynamic Load	Importance Coloring
Partition	Region Coloring
Cluster	Region Coloring
Redundancy-Next	Redundancy Coloring
Redundancy-Random	Redundancy Coloring
Role	Role Coloring

Table 4: Summary of proposed coloring algorithms and their corresponding family classification. The ten algorithms can be grouped into five families.

rithms would perform in a mesh-like or full-mesh topology, as a base case where disconnecting a node has little or no effect on the rest of the graph.

Implementation details: We implement the proposed metrics and coloring algorithms using the JUNG graph library [13]. We use the graph clustering algorithm from [36] since it scales linearly with the topology size (i.e. $O(n + m)$) and allows to predefine the number of clusters. We use the graph partition algorithms from [16], available at [22], which employ multilevel recursive-bisection and scale well while providing good flexibility. Any other clustering or partitioning algorithms could be used, though as we will show, algorithms that generate balanced and contiguous partitions are preferred.

Methodology: We evaluate the performance of the coloring algorithms shown in Table 4 using the network topologies listed in Table 3 as follows. We vary the size of the color set from 2 to 20. For each size of the color set, we 1) color the graph using the desired coloring algorithm; 2) for each color in the color set, disconnect all nodes with that color and compute the robustness metrics on the color-removed subgraph; and 3) compute the robustness metric for the colored graph from the robustness metrics values obtained in Step (2). We say a coloring algorithm is *good* if it yields colored graphs with high minimum and average robustness.

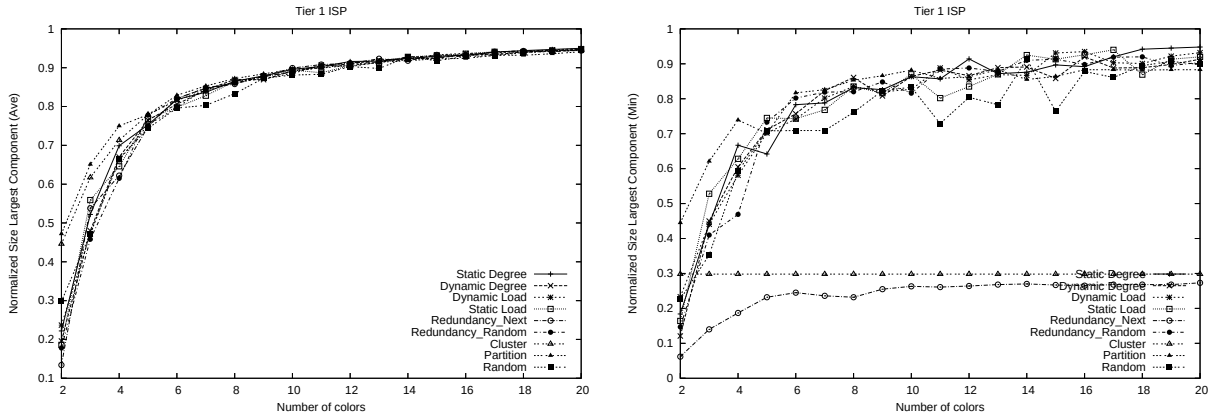
5.2 Coloring Algorithms

In this section, we compare the robustness achieved by the coloring algorithms from the Randomized, Importance, Region and Redundancy families, which assume no distinction between backbone and router nodes. This is usual in medium size networks such as Abilene or Cenic, but might not hold for large Tier-1 ISPs. We present results for the Role family in Section 5.5.

Figure 3 shows the robustness achieved on the Tier-1 topology by the different algorithms measured using the normalized size of the largest component (NSLC). Similarly, Figure 4 shows the result for the pair connectivity (PC) on the Sprint topology. We observe from Figures 3(a) and 4(a) that algorithms from the Region family, Partition and Cluster, outperform the others with respect to the average robustness. We also observe that all other families of algorithms perform similarly and somewhat surprisingly very close to the Random algorithm.

These results indicate that coloring nodes at random locations in the graph may not be a good strategy, since the Region family which uses the opposite policy outperforms the other families. However, the Redundancy and Importance families seem to be doing exactly that, i.e., coloring nodes differently at random spots in the graph, depending on the node importance or its backup policy. Recall that disconnecting all nodes from one color creates two different impacts: the disconnection of the nodes themselves, which cannot be avoided, and the disconnection of any other node who relied on those nodes to connect to the rest of the graph. The second factor can be reduced and sometimes eliminated with a good coloring. This is where the Region family seems to excel, i.e., the simultaneous removal of nodes that are located nearby, in the same region of the graph, has little impact on other nodes located further away in different regions.

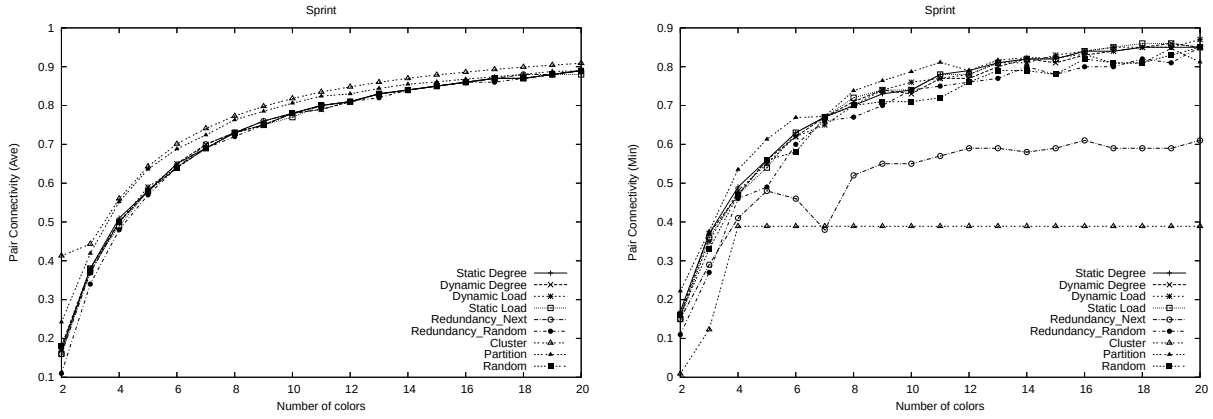
Unbalanced partitions: Figure 4(a) shows that the Cluster algorithm outperforms the Partition algorithm on the Sprint topology in terms of average robustness. This is due to the unbalanced coloring performed by the Cluster algorithm. The Cluster algorithm uses 2 colors on the Sprint topology, assigning one color to 58 nodes and the other color to the remaining 546 nodes. Removing the first color yields a PC of 0.817 and removing the second one yields a PC of 0.009. Thus, the average PC is



(a) Average Normalized Size Largest Component

(b) Minimum Normalized Size Largest Component

Figure 3: Normalized Size of the Largest Component of the coloring obtained using the different coloring algorithms on the Tier-1 ISP topology



(a) Average Pair Connectivity

(b) Minimum Pair Connectivity

Figure 4: Pair Connectivity of the coloring obtained using the different coloring algorithms on the Sprint topology

0.413. On the other hand, the Partition algorithm assigns the first color on 309 nodes and the other color on 295 nodes. This is expected since a graph partitioning algorithm tries to balance the number of nodes in each partition. Thus, in both cases the PC is close to 0.25. As a result, the minimum PC yielded by the Cluster algorithm is 0.009, while the corresponding figure for the Partition algorithm is 0.25. Note that the above example highlights some interesting observation on the pair connectivity. If the colored graph has $\frac{n}{2}$ nodes using each color (most balanced coloring with 2 colors), the maximum value for PC is close to 0.25, while that for NSLC is 0.5. It might surprise the reader that in the above example the average robustness is 0.413, larger than 0.25. Since the PC metric is quadratic in the number of nodes, a case with very good connectivity overcompensates a case with small connectivity.

Figures 3(b) and 4(b) show that the Partition algorithm regularly outperforms all others in terms of the minimum robustness. We also observe that the minimum robustness for the Cluster algorithm tends to stay flat. This is because the Cluster algorithm we used is deterministic, i.e., it always finds the same partitions on different runs over the same graph. Once a bad partition is found, e.g., an unbalanced or discontiguous partition, it is likely to be kept and adding more colors will not improve the minimum robustness. Also, we found that the Redundancy-Next algorithm yields bad minimum robustness. Again, this is due to unbalanced partitions.

Since we do not have prior knowledge on which color

might fail, we need to optimize for both average and minimum robustness. Otherwise, a failure of the color with the largest number of nodes would have a major impact on the network. As we have seen, unbalanced partitions can generate good average robustness, but they will introduce bad minimum robustness which is not desired. Thus, we prefer the Partition algorithm over the Cluster algorithm, since the Partition algorithm tends to generate balanced partitions.

Non-contiguous partitions: In Figure 3(b), the minimum robustness takes a dip at $k = 5$. This is also observed on other topologies. We found that one of the partitions in the split is non-contiguous, that is, the same partition contains two groups of nodes that have intra-group connectivity but no inter-group connectivity. The main problem with non-contiguous partitions is that they might disconnect other partitions that depend on the non-contiguous partition for connectivity to the rest of the graph.

Good graph partitioning algorithms tend to avoid non-contiguous partitions but this is not always easy given that the number of nodes in each partition needs to be balanced. Some algorithms such as the one in [16] have a parameter to control how unbalanced the partitions can be. Our solution for non-contiguous partitions is to re-run the partitioning algorithm with an increased value of this parameter if any of the partitions is found to be non-contiguous. This increases the final robustness but increases the runtime. Since the Partition algorithm is fast, we can generate and color multiple partitions and then select the best coloring, without significantly impacting the runtime. Appendix C shows the runtime for the different algorithms.

In summary, we observe that the best approach to color the network is to use the same color for each region of the graph, since the failure of one node impacts closer nodes more than nodes located further away (i.e. fate-sharing). Among the two algorithms that use this approach, Cluster and Partition, the Partition algorithm works best because it produces a more balanced coloring. In addition a good partitioning algorithm should provide contiguous partitions as these improve the overall robustness of the network.

Note that for a coloring algorithm to provide a robust coloring, it is not enough to produce balanced colorings. All four algorithms from the Importance family produce very balanced colorings, but do so by coloring nodes at random locations. These algorithms are regularly outperformed by the algorithms from the Region family.

5.3 Network Topologies

Next, we compare the robustness of the various topologies for a given coloring algorithm. We use the Partition algorithm which outperformed the others in Section 5.2. As a baseline we use the synthetic Mesh topology since it is the most resilient topology. That is, after removing a color, the remaining nodes are still fully connected; there is no additional impact from disconnecting a node.

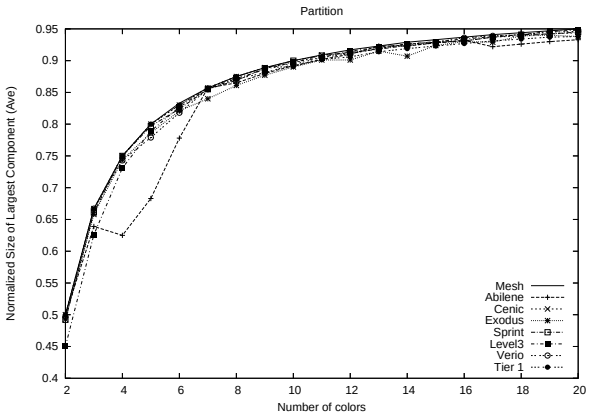
Figure 5 shows the robustness of different topologies using the Partition algorithm. For this experiment, we first force the graph partitioning algorithm to try to find very balanced partitions. If any of the generated partitions is non-contiguous, we relax the balancing constraint. We find that we rarely have to relax the balancing constraint more than once for any color set size. We observe that, using the Partition algorithm, most topologies achieve average and minimum robustness close to the Mesh topology. Thus, when any router can run any implementation, we can achieve robustness close to the optimal obtained with a full mesh, by coloring the graph using the Partition algorithm.

It is not surprising to find that the Abilene topology often performs worse than others. This can be explained by its simple topology with few nodes and minimal link redundancy between nodes. Being a non-commercial network, it is unlikely that robustness was one of Abilene's design goals. However, we also find that the Exodus topology also performs worse than the other commercial network topologies, specially in the worst case. This indicates that the Exodus topology has less geographical redundancy.

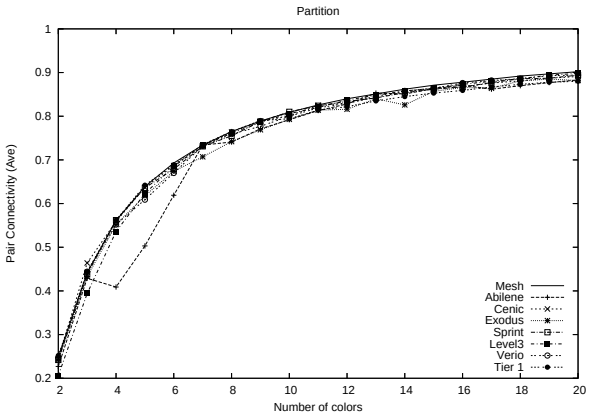
5.4 Node Importance

Nodes in a topology may not be equally important. This can be modeled by assigning a weight to each node. Some partitioning algorithms such as the one we use [22] allow node weights to be specified and compute the partitions to achieve similar total weight in every partition, rather than a similar number of nodes in each partition. Thus, in this case a balanced partition, has approximately the same sum of node weights in each partition.

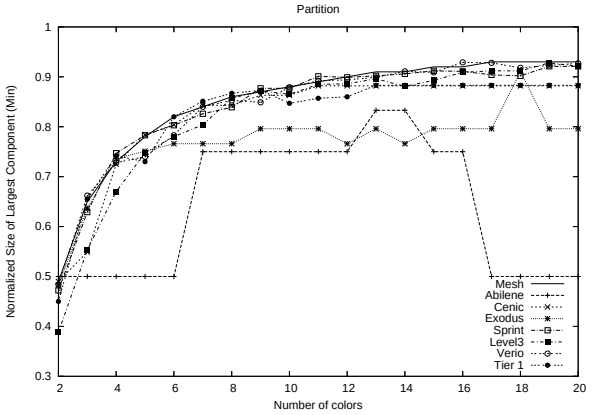
As an example, Figure 6 shows the robustness of the weighted Cenic topology using the Partition algorithm, where each node is assigned a weight equal to the number of customers that connect to that node. We select the Cenic topology because we have such information available. Note that the significance of the weights can be arbitrary and we use this specific weight just as an illustrative example. The dashed line shows the result when we run the Partition algorithm forcing balanced partitions,



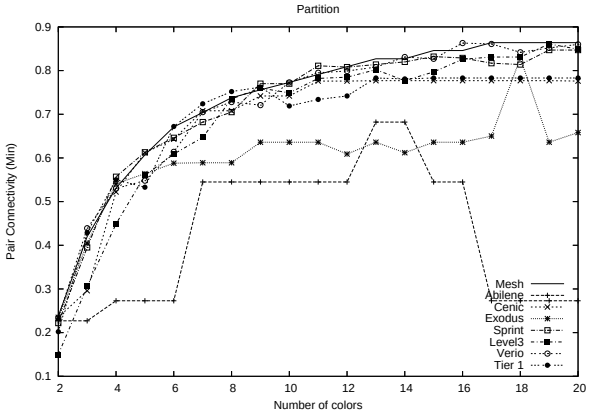
(a) Average Normalized Size Largest Component



(b) Average Pair Connectivity

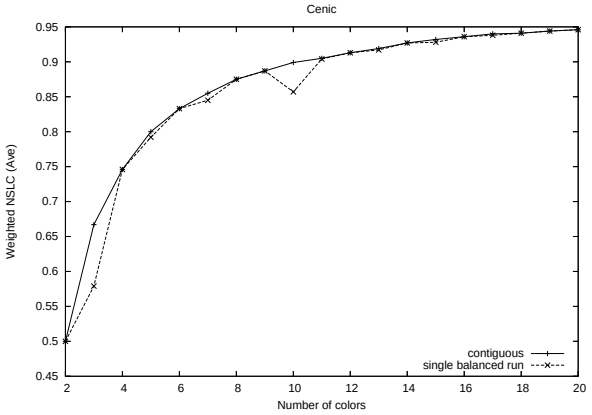


(c) Minimum Normalized Size Largest Component

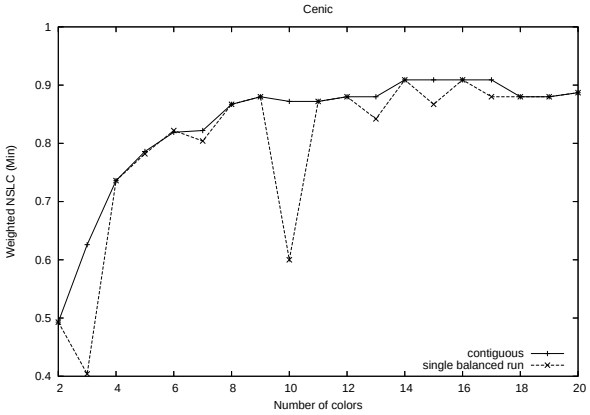


(d) Minimum Pair Connectivity

Figure 5: Robustness of the different topologies when colored using the Partition algorithm.



(a) Average Weighted Normalized Size Largest Component



(b) Minimum Weighted Normalized Size Largest Component

Figure 6: Robustness of the Cenic topology, with node weights assigned based on the number of customers connected to the router, and selecting the split that gives no discontiguous partitions.

while the solid line shows the final result where if one of the partitions was non-contiguous, we relax the balancing constraint. Compared with Figure 5, we observe that similar robustness can be achieved in the weighted topology as that in the unweighted topology. There are two points (i.e. $k = 3$ and $k = 10$) where the partitioning algorithm is not able to find contiguous partitions if constrained to very balanced partitions. When we relax this constraint, we obtain partitions that are more unbalanced, but contiguous, which improves the final result specially for minimum robustness.

5.5 Optimizing the Use of Existing Diversity

In this section we first present the diversity that currently exists in the Tier-1 topology, which we call *original coloring*. Then, we compare the robustness of the original coloring to the robustness in the colored graphs obtained by the Partition and Role algorithms.

The Tier-1 topology uses 8 implementations with two of the implementations being used by over 90% of the nodes. The fact that there are two dominating implementations supports the hypothesis that current networks are highly homogeneous. We can now raise an interesting question: given the amount of existing diversity in the network, can we perform better using our coloring algorithms? Table 5 compares the robustness of the Tier-1 ISP using the original coloring, with the robustness achieved using the Partition and Role algorithms. Since there are eight colors in the original coloring, we set the number of available colors for the Partition algorithm to eight and allow any router to select any of the colors. For the Role algorithm we need two sets of colors: one for the backbone nodes and one for the access nodes. We observe that, among the eight implementations in the original coloring, only two are used for backbone routers. Therefore, we allow the Role algorithm to use two colors for backbone routers and six colors for access routers.

We observe that the original coloring has bad minimum robustness when the color failure affects one of the colors used by the backbone routers. The Partition and Role algorithms achieve improved average and minimum robustness by distributing the available diversity better throughout the network. This is specially significant for minimum robustness, achieving values much closer to the average robustness. The Role algorithm shows worse minimum robustness than the Partition algorithm, since it has an additional constraint that routers can only use implementations adapted to their role. But this constraint is more realistic for Tier-1 ISP networks, where the role separation of backbone and access routers becomes evident. In smaller topologies, such as Cenic or Abilene, where the role separation is not so clear, the Partition algorithm remains the best choice.

Figure 7 shows the degree of diversity needed to achieve a certain level of robustness in the Tier-1 topology using the Role algorithm. From the total number of colors shown in the x-axis, two colors are always used for the backbone routers and the rest used for the access routers. The results show that two colors are enough to guarantee the robustness of the backbone, given the existing amount of redundancy. This is shown as a good average robustness with any number of access colors. It also shows that we start to achieve good overall robustness (e.g., minimum robustness ≥ 0.5 and average robustness ≥ 0.75) when we use at least 3 colors for the access routers, that is with a total of 5 colors.

To conclude, we observe that the original Tier-1 network has a significant amount of diversity, but the diversity is not best used to maximize robustness against color failures. Using a realistic coloring algorithm such as Role coloring, we can achieve a significant increase in the robustness of the topology against color failures, without increasing the amount of implementations used in the network.

6 Related Work

Zhang et al. [38] first proposed the use of diversity to increase the survivability of a network, inserting diversity at each level of the networking stack. More recently, O'Donnell et al. [25] studied the problem of how to use diversity to limit the spread of malware on a network topology, assuming that the spread could only happen between adjacent nodes in the topology. There are significant differences between [25] and our work. First, our goal is to improve the robustness of a topology under a color failure. Second, in [25] the authors assume that each node runs multiple implementations and takes an online decision, in a distributed fashion, of which implementation to use at any given time. In our case, nodes run a single implementation and network operators select in a centralized and offline manner, using our coloring algorithms, which implementation each node should run from the set of available implementations. Finally, we do not assume that faults can only spread between adjacent nodes, in fact they affect all nodes with the same color. Stamp also argued in favor of software diversity in [32].

Previous work on network robustness has focused on the Internet topology. Albert et al. [28] showed that scale-free networks are resilient to random faults but vulnerable to attacks that target the most connected nodes. Park et al. [27] studied the susceptibility of the Internet to a mix-

Metric	Original coloring	Partition coloring	Role coloring
Average robustness (NSLC)	0.713	0.875	0.855
Minimum robustness (NSLC)	0.055	0.867	0.760
Average robustness (PC)	0.647	0.765	0.739
Minimum robustness (PC)	0.016	0.752	0.578

Table 5: Comparison of the robustness of the Tier-1 topology using the original coloring and the coloring obtained by the Partition and Role algorithms. All cases use 8 colors.

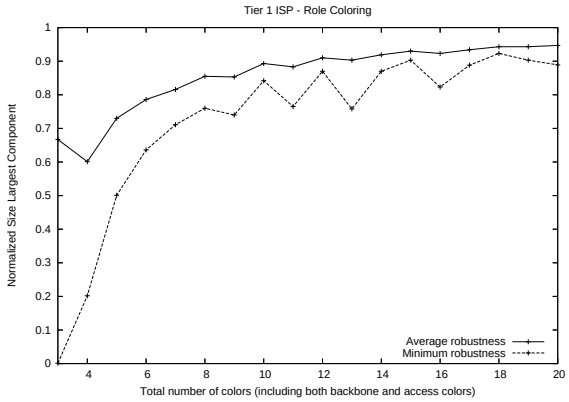


Figure 7: Tier-1 robustness using Role coloring. Two colors are always used for backbone routers.

ture of random faults and attacks. Magoni [19] studied the Internet robustness to attacks and found that the removal of a few nodes could significantly damage the Internet's connectivity but that such an attack would be costly to mount. Li et al. [17] raised concerns about the application of power laws to the router-level Internet topology. Markopoulou et al. [21] classified the different network faults from a tier-1 ISP network and found that 12% of all faults were router-related problems.

There has been extensive work on the Internet connectivity and generally in identifying graph connectivity metrics [26, 27, 28, 33, 37]. Several node importance metrics have also been proposed [8, 17, 23, 34]. Finally, there is also research work on graph partitioning. In our paper, we use [16, 36] to compare the differences between partitioning algorithms. Wu et al. in [36] uses a voltage ranker to find clusters while Karypis et al. [16] uses a k-way multilevel approach to partition hypergraphs.

7 Discussion

In this section we present extensions to our work that we are currently testing, other foreseeable applications for our approach and some future work.

External connections: ISPs need to route their customer's traffic to the rest of the Internet. For all destinations not directly connected to the ISP network, this means forwarding the traffic to another ISP, either an upstream provider or a peer. In order for the network to remain functioning upon a color failure, we need to ensure that there is at least one external connection in each partition when a color fails. With the Partition algorithm, we can apriori fix nodes in different regions, and the partitioning will try to keep those nodes in the specified regions. We use this property to fix border routers in different partitions. When using the Role algorithm we can split the backbone nodes into two different roles: border and internal nodes, and apply a similar procedure.

Small networks: One could argue that enterprise networks and small ISPs would be concerned about connections coming into specific nodes (e.g. data/hosting center), and thus metrics that assume that all nodes need to connect to all other nodes are not best. We argue this is not quite so, since when faced with a color failure what we need is the maximum number of customers being left in the same component where the data center is, and thus able to connect to it. Since the number of customers connecting through a node can be encoded in the node's weight and our results show that for small networks the Partition algorithm works best, then the approach to maximize client access to these data centers during a color failure is identical to the external connections case mentioned above. That is, geographically replicate the data center so that one replica is placed in each colored partition.

Robustness beyond connectivity: In this paper we have limited our analysis to the connectivity of the network, because connectivity is the fundamental property which needs to be present for the network to be functional. Once we have addressed connectivity, we can study the impact of color failures on higher layers. Clearly, the disconnection of a large number of routers impacts the routing and the end-to-end quality of service and might introduce higher CPU load on the remaining routers, packet loss increments due to congestion on the remaining links, or an increase in end-to-end delay due to longer paths. The study of these effects on the higher layers is part of our future work.

Robustness beyond color failures: There is a need to develop a general framework to measure network robustness, that encompasses any type of failure and any kind of impact. Such a framework would allow us to predict the impact of failures and to understand the effect of a proposed local change (e.g., add or remove a router or a link) on the global network robustness. This problem is very challenging because it involves multiple elements that can fail, varied causes for such failures, and different manifestations of their impact. Developing such a framework is our long-term goal. In this paper we have presented a first step towards that goal by studying the network robustness under router software failures. We consider this a worst case scenario because it involves simultaneous and geographically disperse node failures. This kind of failures are infrequent but have great impact. In our future work we plan to extend our approach to handle more frequent failures with more limited impact.

8 Conclusion

In this paper, we have used a graph theoretic approach to study if the introduction of diversity would really increase the robustness of the routing infrastructure. We have answered three fundamental questions about the introduction of diversity. First, we have showed how to measure the robustness of the network when faced with a color failure, using robustness metrics that represent the average and worst cases and are independent of the connectivity metric used. Second, we have proposed five different families of coloring algorithms, each using a different approach, and have found two approaches that best utilize the available diversity to increase the robustness of the network. If any router can run any implementation, then the best approach is to divide the network into geographically contiguous partitions. With such partition, failures are isolated to a region of the network and the impact on the rest of the graph is minimized. On the other hand, if routers have roles that constrain the implementations that they can use, then the best approach is to divide nodes according to their roles and color them independently. Finally, our experiments have shown what degree of diversity is needed to achieve a certain robustness level. For example, in a Tier 1 ISP topology using the same number of implementations currently available we can increase the number of nodes in the largest component from 5% to at least 76%.

References

[1] Abilene. http://abilene.internet2.edu/.

[2] Avici. http://www.avici.com/.

[3] B. Bollobás and O. Riordan. Slow emergence of the giant component in the growing m-out graph. Published as *Random Struct. Algorithms*, **27**(1):1–24.

[4] Cenic. http://www.cenic.org.

[5] Cert Vulnerability Note VU#205225 (July 2006). http://www.kb.cert.org/vuls/id/205225.

[6] Cisco Software Bug Toolkit. http://www.cisco.com/cgi-bin/Support/Bugtool/home.pl.

[7] M. Faloutsos, P. Faloutsos, and C. Faloutsos. On power-law relationships of the Internet topology. *Proceedings of SIGCOMM 1999*.

[8] L. C. Freeman. A set of measures of centrality based on betweenness. *Sociometry*, **40**:35 – 41.

[9] Cert Cisco Information for VU#139491 (April 2002). http://www.kb.cert.org/vuls/id/JPLA-53D2H9.

[10] Cert Vulnerability Note VU#583638 (Jan 2005). http://www.kb.cert.org/vuls/id/583638.

[11] Cert Vulnerability Note VU#959203 (Aug 2005). http://www.kb.cert.org/vuls/id/959203.

[12] T. R. Jensen and B. Toft. *Graph Coloring Problems*. Wiley-Interscience.

[13] JUNG: Java Universal Network/Graph Framework. http://jung.sourceforge.net/.

[14] Juniper. http://www.juniper.net.

[15] Cert Vulnerability Note VU#409555 (May 2006). http://www.kb.cert.org/vuls/id/409555.

[16] G. Karypis and V. Kumar. Multilevel k-way Hypergraph Partitioning. *VLSI Design*, **11**(3):285 – 300.

[17] L. Li, D. Alderson, W. Willinger, and J. Doyle. A First Principles Approach to Understanding the Internet's Router Technology. *Proceedings of ACM SIGCOMM 2004*.

[18] F. Lorrain and H. C. White. Structural equivalence of individuals in social networks. *Journal of Mathematical Sociology*, 1:49 – 80.

[19] D. Magoni. Tearing down the Internet. *IEEE Journal on Selected Areas in Communications*, **21**(6):949 – 960.

[20] Router maket share. http://www.infonetics.com/resources/resource.html.

[21] A. Markopoulou, G. Iannaccone, S. Bhattacharyya, C. Chuah, and C. Diot. Characterization of Failures in an IP Backbone. *Proceedings of the 23rd Conference on Computer Communications (INFOCOM 04)*.

[22] METIS: A family of Multilevel Partitioning Algorithms. http://glaros.dtc.umn.edu/gkhome/views/metis.

[23] M. E. J. Newman. A set of measures of centrality based on betweenness. *Physics Review*, **67**.

[24] Software Errors Cost U.S. Economy $59.5 Billion Annually. http://www.nist.gov/public_affairs/releases/n02-10.htm.

[25] A. J. O'Donnell and H. Sethu. On achieving software diversity for improved network security using distributed coloring algorithms. *Proceedings of the 11th ACM conference on Computer and Communications Security (CCS04)*.

[26] C. Palmer, G. Siganos, M. Faloutsos, C. Faloutsos, and P. B. Gibbons. The connectivity and Fault-Tolerance of the Internet Topology. *Workshop on Network Related Data Management (NRDM01)*.

[27] S. Park, A. Khrabrov, D. M. Pennock, S. Lawrence, C. L. Giles, and L. H. Ungar. Static and dynamic analysis of the Internet's susceptibility to faults and attacks. *Proceedings of the 22nd Conference on Computer Communications (INFOCOM 03)*.

[28] A. L. Barabási R. Albert, H. Jeong. Error and attack tolerance in complex networks. *Nature*, **406**:378 – 382.

[29] Rocketfuel topologies. http://www.cs.washington.edu/research/networking/rocketfuel/.

[30] Cert Advisory CA-2002-03 (Feb 2002). http://www.cert.org/advisories/CA-2002-03.html.

[31] N. Spring, R. Mahajan, D. Wetherall, and T. Anderson. Measuring ISP Topologies with Rocketfuel. *Proceedings of ACM SIGCOMM 2002*.

[32] M. Stamp. Risks of monoculture. *Commun. ACM*, **47**(3):120. ACM Press.

[33] H. Tangmunarunkit, R. Govindan, S. Jamin, S. Shenker, and W. Willinger. Network Topology Generators: Degree-Based vs Structural. *Proceedings of ACM SIGCOMM 2002*.

[34] S. L. Tauro, C. Palmer, G. Siganos, and M. Faloutsos. A simple conceptual model for the Internet topology. *Global Telecommunications Conference 2001 (GLOBECOM01)*.

[35] Wired News. The Backhoe: A Real Cyberthreat. http://www.wired.com/news/technology/1,70040-0.html.

[36] F. Wu and B. A. Huberman. Finding communities in linear time: a physics approach. *European Physical Journal B*, **38**:331-338.

[37] E. W. Zegura, K. L. Calvert, and M. J. Donahoo. A quantitative comparison of graph-based models for Internet topology. *IEEE/ ACM Transactions on Networking*, **5**(6):770–783.

[38] Y. Zhang, S. K. Dao, H. Vin, L. Alvisi, and W. Lee. Heterogeneous Networking: A New Survivability Paradigm. *Workshop on New Security Paradigms 2001 (NSPW01)*.

A Redundancy Algorithm Pseudocode

```
/* Redundancy Coloring Algorithms */
/* Randomize a starting color */
currColor = randomizeColor();

Order all nodes in graph by decreasing
  importance (e.g. using degree);

For (each node v) {
  If (isColored(v))
    next;
  Else {
    getRedundantNeighbors(v);
    red = most redundant neighbor;
    If (!isColored(red)) {
      v.color = currColor;
      /* Color red with a random
       color != v.color */
      do
        randomColor = randomizeColor();
      while (randomColor == v.color);
      red.color = randomColor;
    }
    Else {
      /* This is where both redundancy
       algorithms differ */
      If (algorithm == Redundancy-Next){
        do
          nr = next most redundant node;
        while (isColored(nr));
      }
      If (algorithm == Redundancy-Random){
        red.color = randomColor;
        /* Color red with a random
       color != red.color */
        do
          randomColor = randomizeColor();
        while (randomColor == red.color);
        v.color = randomColor;
      }
    }
    currColor++;
  }
}
```

B Randomized Coloring Theoretical Analysis

Here, we conduct theoretical analysis for the effectiveness of the randomized coloring on large power-law graphs, such as the Internet AS-level topology [7]. Note that the power-law model does not apply to router-level topologies [17] and here we deal with the AS-level Internet topology where each AS randomly selects a router implementation to use on all its routers.

Let $H_m^{(n)}$ represent a random graph consisting of n nodes formed in the following way (a.k.a. preferential attachment): nodes are added one at a time, by joining each new node to an independently chosen set of m earlier nodes where the probability for choosing a given node is proportional to its degree. We then construct a subgraph $H_m^n(p)$ by deleting each node independently with probability $1-p$. Bollobás and Riordan [3] have shown that there is a "phase transition": there is a certain critical value p_c such that a component of order $\Theta(n)$ remains with high probability as $n \to \infty$ if and only if $p > p_c$.

Let $C_1(G)$ represent the size of the largest component of G, $C_2(G)$ represent the size of the second largest component of G. More formally, we have the following theorem [3]:

Theorem 1. *Let $m \geq 2$ and $0 < p < 1$ be fixed, and set*

$$p_c = \frac{1}{2}\left(1 - \sqrt{\frac{m-1}{m}}\right) \tag{1}$$

If $p \leq p_c$, then

$$C_1(H_m^{(n)}(p)) = o(n)$$

holds with high probability as $n \to \infty$. If $p = p_c + \epsilon$, $\epsilon > 0$, then

$$C_1(H_m^{(n)}(p)) = f_m(\epsilon)n + o(n) \tag{2}$$

and

$$C_2(H_m^{(n)}(p)) = o(n)$$

holds with high probability as $n \to \infty$, where

$$f_m(\epsilon) = exp(-\Theta(1/\sqrt{\epsilon})).$$

As m gets higher the critical value p_c gets lower. Thus, with $m = 2$, which is the lowest possible value of m, we have $p_c = 0.146$. Thus, for every graph with $m \geq 2$, if we retain edges with probability slightly higher than 0.146, then we expect to observe a very large component, linear to n.

Note that Theorem 1 states the necessary condition in order to retain a giant component, which is a component that has size linear to the size of the initial graph, when deleting nodes at random in a preferential-attachment random graph. With randomized coloring using k colors, when we remove all nodes of the same color, we can easily see that it is essentially equivalent to deleting nodes at random in the same graph with probability $1/k$.

Algorithm	Run Time (19 runs)	Run Time (1 run average)
Dynamic Load	711.8	37.5
Cluster	100.1	5.3
Static Load	42.8	2.2
Dynamic Degree	31.6	1.7
Redundancy-Next	29.9	1.6
Static Degree	29.6	1.6
Random	29.5	1.6
Redundancy-Random	29.3	1.5
Partition	2.2	0.1

Table 6: Algorithm runtime (in minutes) on the Verio topology. The middle column shows the total runtime for coloring the graph 19 times, from 2 to 20 colors. The rightmost column shows the average time per color set

Therefore, we can conclude that for a large graph following the preferential attachment model (where $n \rightarrow \infty$), even the minimum number of two colors is sufficient in order to have a giant component, because in this case we are retaining nodes with probability $p = 0.5 > p_c$. When we have 3 colors we are retaining nodes with probability $p = 0.666$. Indeed, even though in our graphs the number of nodes does not really approach infinity, as required by the above theorem, we have performed experiments that show that when using 3 colors we achieve a high robustness.

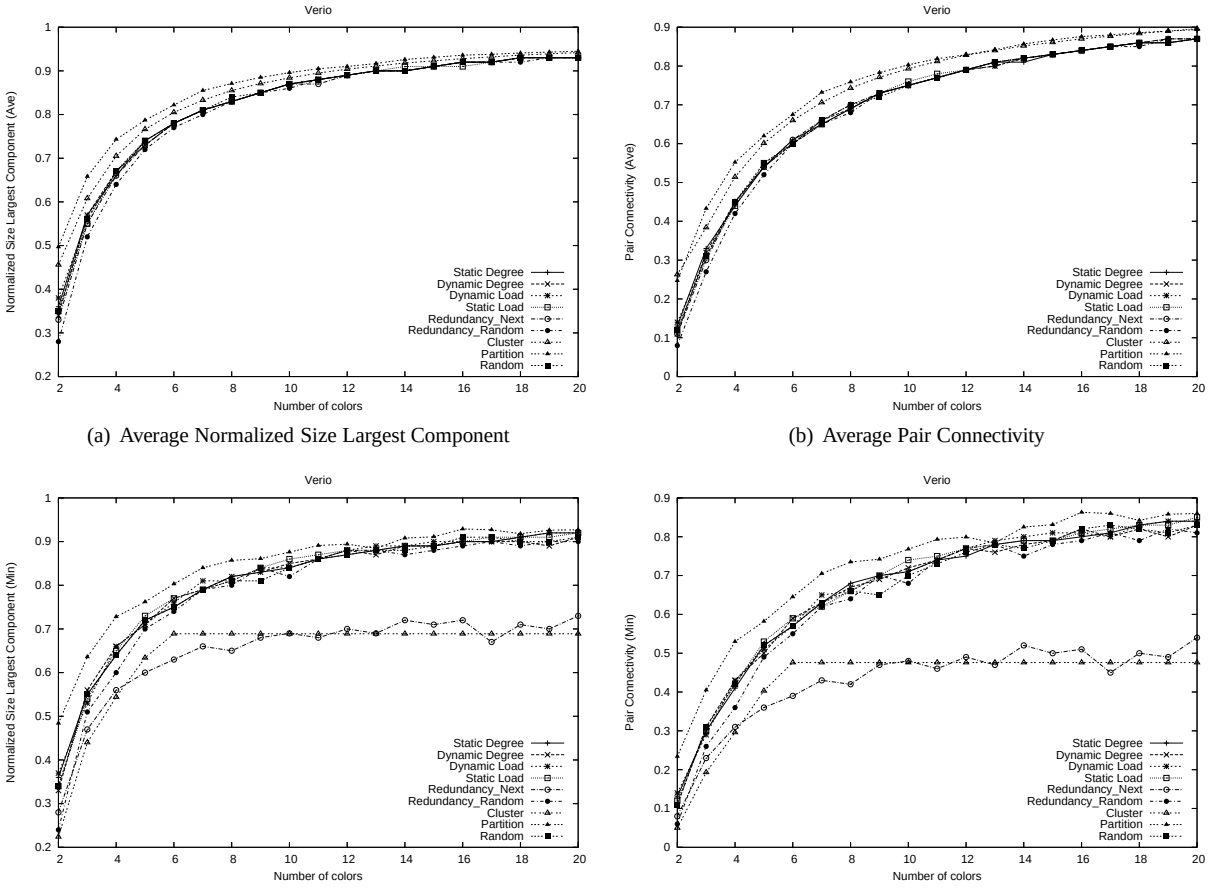
C Algorithm Runtime

In this section we study the runtime of proposed coloring algorithms. Table 6 shows the algorithms runtime on the Verio topology, which has the largest number of nodes. The algorithms are run on a Pentium IV desktop computer with 1.8 GHz CPU and 1 GB of RAM. The runtime includes the time to color the graph and the time to compute the robustness of the colored graph. The middle column shows the total runtime for 19 different color set sizes, varying from 2 to 20. This is useful to identify how many colors need to be used to achieve a certain level of robustness. The rightmost column shows the average time to color the graph and compute the robustness for a single color set size. Results show that most algorithms take a few minutes per color set and thus can be efficiently run on the topologies of large Tier-1 ISPs.

The Dynamic Load algorithm is the most expensive, taking 37 minutes per color set. On the other hand, the Partition algorithm takes only a few seconds per color set, which includes both generating the partitions and coloring them. Thus, it is quite feasible to run the Partition algorithm multiple times, e.g., by varying the degree of unbalance in the graph, and selecting the partition with highest robustness. This gives extra protection against non-contiguous partitions that are occasionally found by graph partitioning algorithms.

D Additional Results

We provide additional results on the robustness of the coloring obtained using the different coloring algorithms on the Verio and Cenic topologies in Figures 8 and 9. The general observations are similar to what we have presented in Section 5.



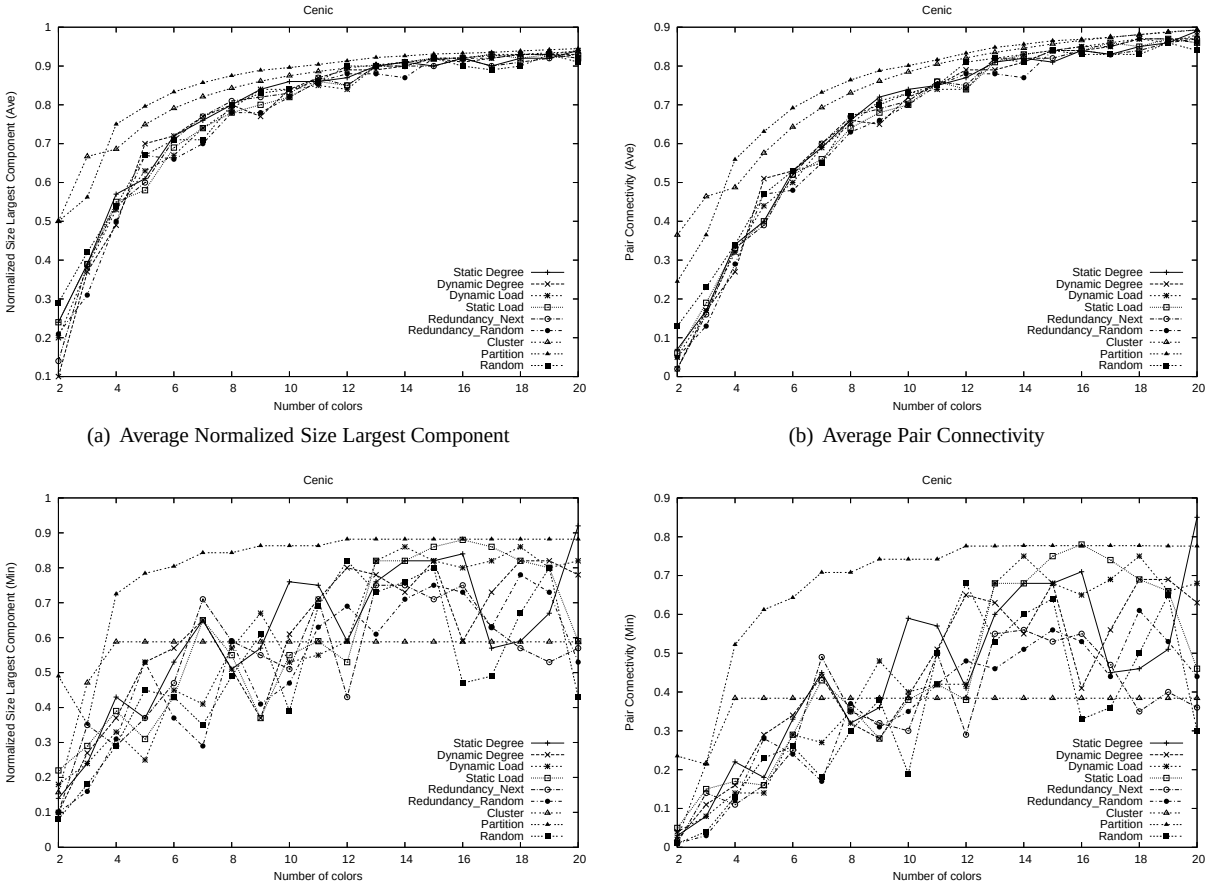
(a) Average Normalized Size Largest Component

(b) Average Pair Connectivity

(c) Minimum Normalized Size Largest Component

(d) Minimum Pair Connectivity

Figure 8: Robustness of the coloring obtained using the different coloring algorithms on the Verio topology



(a) Average Normalized Size Largest Component

(b) Average Pair Connectivity

(c) Minimum Normalized Size Largest Component

(d) Minimum Pair Connectivity

Figure 9: Robustness of the coloring obtained using the different coloring algorithms on the Cenic topology