A Genetic Algorithm for Dynamic Controller Placement in Software Defined Networking

Workshop Paper

Samuel Champagne
Dalhousie University
Computer Science
Halifax, Nova Scotia
sam.champagne@dal.ca

Tokunbo Makanju
New York Institute of Technology
Vancouver, British Colombia
madetoku@nyit.edu

Chengchao Yao
Dalhousie University
Computer Science
Halifax, Nova Scotia
ch250407@dal.ca

Nur Zincir-Heywood
Dalhousie University
Computer Science
Halifax, Nova Scotia
zincir@cs.dal.ca

Malcolm Heywood
Dalhousie University
Computer Science
Halifax, Nova Scotia
mheywood@cs.dal.ca

ABSTRACT

The Software Defined Networking paradigm has enabled dynamic configuration and control of large networks. Although the division of the control and data planes on networks has lead to dynamic reconfigurability of large networks, finding the minimal and optimal set of controllers that can adapt to the changes in the network has proven to be a challenging problem. Recent research tends to favor small solution sets with a focus on either propagation latency or controller load distribution, and struggles to find large balanced solution sets. In this paper, we propose a multi-objective genetic algorithm based approach to the controller placement problem that minimizes inter-controller latency, load distribution and the number of controllers with fitness sharing. We demonstrate that the proposed approach provides diverse and adaptive solutions to real network architectures such as the United States backbone and Japanese backbone networks. We further discuss the relevance and application of a diversity focused genetic algorithm for a moving target defense security model.

CCS CONCEPTS

• **Networks** → **Network properties**; **Network security**; **Network structure**; **Network reliability**; • **Computing methodologies** → **Machine learning**; **Unsupervised learning**; **Genetic algorithms**;

KEYWORDS

Routing and Layout, Genetic Algorithms, Fitness Evaluation, Multiple Solutions/Niching, Software Defined Networking, Moving Target Defense

ACM Reference Format:
Samuel Champagne, Tokunbo Makanju, Chengchao Yao, Nur Zincir-Heywood, and Malcolm Heywood. 2018. A Genetic Algorithm for Dynamic Controller Placement in Software Defined Networking: Workshop Paper. In *GECCO '18 Companion: Genetic and Evolutionary Computation Conference Companion, July 15–19, 2018, Kyoto, Japan.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3205651.3208244

GECCO '18 Companion, July 15–19, 2018, Kyoto, Japan

ACM ISBN 978-1-4503-5764-7/18/07…$15.00
https://doi.org/10.1145/3205651.3208244

1 INTRODUCTION

Software Defined Networking (SDN) is a relatively new technology that decouples the forwarding plane and control plane on a network. Decoupling the forwarding plane and control plane introduces a layered model which facilitates network programmability. SDN technology is being used for increasingly large networks [3, 4], such as backbone networks, that demand stability, security and redundancy over time in variable load requirements. Given longevity and reliability requirements, it is important to correctly establish the number of controllers and domain subgraph partition pairs within a large scale SDN implementation. As introduced by Heller et al. [4], the controller placement problem has been studied by several researchers from different perspectives.

The controller placement problem, similar to the facility location problem, is concerned with the discovery of an optimal number of controllers and their placement in the topology. Ideally, the number of controllers should be minimized for cost, while also minimizing communication latency. Reducing to a known problem, the controller placement problem is NP hard [4]. The controller placement problem is increasingly challenging in precarious networks where fault tolerance is desired and static optimal solutions are not sufficient.

Servicing backbone networks requires resilient network design which is adaptable to changing demand from domain nodes. To answer the need for security, redundancy and reliability, it is necessary to develop a solution set with inherent variability that provides

efficient load distribution and minimal latency. In this paper, our goal is to provide a large set of unique near optimal solutions to the controller placement problem for network redundancy and network flexibility. We think that these near optimal solutions could then be used to achieve an applicable security model for moving target defense while improving network redundancy and reliability.

Given that finding an optimal solution to the controller placement problem is computationally NP-hard [4] and offers no redundancy in case of link failures, the Genetic Algorithm (GA) approach could offer reasonable solutions to this problem. In this case, the GA approach lends itself to varying and adaptive network topologies. Moreover, by assuming a multi-objective approach, we are able to simultaneously minimize the number of controllers, inter-controller latency and load distribution of domain subgraphs while improving evolutionary population diversity. Specifically, we aim to show the utility of such a GA by applying it to simulated backbone networks and discussing the relevance of these results for a moving target defense security model.

Moving target defense is more easily implemented with SDNs because of their large search space and the large search space of SDNs makes them a great candidate for Evolutionary Computation [9]. Our GA provides better diversity for better placement choices which can be used in the moving target defense model to avoid easy surveillance from adversaries. Given these advantages, we later propose a moving target defense model which utilizes a diversity focused GA.

We first provide a summary of related research in this field in Section 2. We then describe the GA model, required data structures and algorithm process in Section 3. Finally, we test the proposed multi-objective GA against a small test network, the United States backbone and Japan backbone and analyze the results in Section 4. We later propose the relevance of this work to a moving target defense model in Section 5. Section 6 concludes the paper and discusses future works.

2 RELATED WORK

The controller placement problem has been shown to be NP-hard [4], and there is a large amount of ongoing research on the subject. There has been recent work on the application of machine learning and specifically evolutionary computation to the problem which warrants a closer look.

Wang et al. introduce an approximate algorithm, the Greedy Sub-Graph Cover Problem (GSGCP) algorithm, to find the smallest subset of nodes from a graph such that the subsets cover all nodes in the graph [12]. In doing so, they provide a flexible approximate solution that finds the least number of controllers required for a given graph. However, the algorithm does not consider communication latency and is not appropriate for dynamic networks. We plan to improve on the GSGCP algorithm by implementing a GA which initializes a population using a modified GSGCP algorithm and then improves diversity and fitness over time.

Bo et al. use a two stage algorithm to find an optimal number of controllers for a given topology [3]. In the first stage, a multi-objective GA is used to find a connection relationship of controllers and switches that share load equally across domains. The fitness check is defined as load diversity and is used for load balancing. In the second stage, they use a minimal delay algorithm to find an optimal location that minimizes latency between clusters found in the first stage. Their limited use of mutation makes it less flexible to large networks and dynamic changes in network weights.

Sanner et al. provide a framework to solve the cluster configuration of nodes using linear programming and a GA [11]. They implement two fitness checks to maximizing the sum of the average edge connectivity of clusters and minimizing the imbalance between clusters. Their mutation factor exchanges nodes between clusters efficiently, but did not support reproduction as a variation operator. On the other hand, we suggest a GA with a multi-objective fitness function that utilizes reproduction.

Sabeeh et al. use heuristic optimization paired with neuro-evolution to minimize the load on SDN switches by making them change adaptively [10]. Inputs are defined as flows of flow tables and objectives are defined in terms of throughput and delay. They apply Particle Swarm Optimization (PSO) and a GA to a Neural Network (NN) for optimal performance output.

Makanju et al. suggest a Moving Target Defense (MTD) paradigm, i.e. a continuous adaptation of a network environment to evade attacks, would be best implemented using Evolutionary Computation techniques in Software Defined Networking [9]. Our implementation is well suited for MTD because the multi-objective GA provides a large solution space which can be referenced for network configuration alternatives without needing reiteration overhead.

We believe that there is room for improvement in addressing the controller placement problem using GA techniques to find a larger set of solution vectors. To the best of our knowledge, previous research has not specifically focused on diversity of solutions and we believe that our research provides evidence of usefulness. Given a large solution set, a large network could improve its reliability by enabling quick link changes without computational overhead thanks to the larger solution space provided by a multi-objective GA focusing on diversity.

3 GENETIC ALGORITHM FRAMEWORK

The multi-objective GA that is employed in this work uses three fitness functions with variable reproduction and mutation operations. The following introduces the framework in detail.

3.1 Formal Definitions

The controller placement problem aims to find a minimum set of controllers and their respective locations. Such a set cover problem focuses on finding the minimum sub-collection of sets required to represent a given universe. In our implementation, we state that the universe of a network is the set of all nodes and all links. Grouping domains and links into subgraphs or sub-collections, the minimum set cover of a network is the set of subgraphs which cover the network while retaining fast connectivity and a minimized number of links. These subgraphs can be associated to controllers and used to then find a minimum set of controllers and their location respective to the subgraphs they control. After generating domain subgraphs and finding a minimum set cover, the multi-objective GA aims to solve the controller placement problem by minimizing the number of controllers needed and swapping or generating subgraphs according to special reproduction and mutation operations.

3.2 System description

In the proposed model, we assume a multi-domain network and implement variable set covers for the network, defined as subgraphs of the network, in a usable SDN configuration. To simulate real networks, we define that given X controllers and Y domains, links may exist between domains, but not for every possible pair of domains. For simplicity of SDN implementation, we further state that links exist between every two controllers. Based on these assumptions, we establish flexible relationships between controllers and domains. One domain can be managed by multiple controllers, but a subgraph can not be managed by more than one controller.

We define a Chromosome, or an individual, as a set of controllers such that every controller is a Gene and every subgraph is an Allele. We have established three fitness metrics to evaluate individuals based on the three goals of improving connectivity between controllers, improving load balancing across controllers and reducing the number of controllers.

- **Inter-Controller Latency.** Increasing connectivity is expected to improve the reliability of a given implementation.
- **Load Distribution.** Improving load distribution will have the expected result of balancing the total number of subgraphs evenly across all controllers. This should have the further effect of reducing latency across controllers.
- **Number of Controllers.** Reducing the total number of controllers will reduce implementation overhead and cost. We are not necessarily interested in finding a dissimilarity threshold and are fine with having increased individuality within a given optima. As such, we implement niching through fitness sharing to retain diversity within an optima.

3.3 Model Description

The GA begins by initializing the population with domain subgraphs, set covers, from the network graph using the GSGCP algorithm [12]. The resulting subgraphs are each assigned to a random controller and then evaluated based on three fitness functions.

1) *Fitness evaluation:* The most suitable individuals are selected based on the ability to *minimize* the value of all three fitness functions: inter-controller latency (1), load distribution (2) or (3) and number of controllers (4) or (5). Both the latency and load distribution metric involve a summation term. An additional fitness sharing niche term will be introduced (6) and hereafter referred to as 'niching'.

For the first fitness function, inter-controller latency, we let $L_{i,j}$ be the inter-controller latency between two controllers i, j used in chromosome $\mathcal{I}_k$ such that the we can define its fitness, in relation to inter-controller latency. The total overall latency of chromosome k is given by Eq. (1):

$$\forall i, j \in \mathcal{I}_k : \mathcal{L}_k = \sum_{i,j} t_i \tag{1}$$

For the second fitness function, load distribution, we first define D_k as the total number of domains in chromosome k, C_k as the total number controllers in chromosome k and D_j as the total number of domains controlled by controller j in chromosome k. Then, for chromosome k, we define its fitness, in relation to load distribution, using either Eq. (2) or Eq. (3). Our implementation in section 4 uses Eq. (3) as it normalizes the fitness in relation to D_K.

$$\sum_{j=1}^{C} \left| D_j - \frac{D_k}{C_k} \right| \tag{2}$$

$$\sum_{j=1}^{C} \frac{\left| D_j - \frac{D_k}{C_k} \right|}{D_k} \tag{3}$$

For the number of controllers, we first define C as the total number of controllers in a chromosome and c as the number of active controllers in a chromosome. We then define $C_{\min}$ as the minimum number of controllers used in any chromosome and $C_{\max}$ as the maximum number of controllers used in any chromosome. We can then define fitness of a chromosome, in relation to the number of controllers utilized, using either Eq. (4) or Eq. (5). We use Eq. (5) in section 4 because it provides a chromosome specific evaluation whereas Eq. (4) provides a fitness evaluation that correlates with the maximum and minimum number of controllers used in any chromosome within the population.

$$1 - \frac{c - C_{\min}}{C_{\max} - C_{\min}} \tag{4}$$

$$1 - \frac{c - 1}{C - 1} \tag{5}$$

From the above fitness functions, we observe that the number of controllers is the first to be minimized as it is the easiest conformity adjustment for any individual. Given the tendency of the genetic algorithm to drift into an optimum with a single active controller, even with Pareto ranking, we decided to implement niching in the form of fitness sharing. Given $m(k)$ as the niche count equaling the index of chromosome k in a specific fitness layer, F as the set of all fitness values in chromosome $\mathcal{I}_k$, then fitness sharing can be implemented as Eq. (6) to scale the fitness of chromosome k based on its niche count.

$$\forall f \in F, f = f \times m(k)^2 \tag{6}$$

In summary, the three objectives will be minimized using Pareto ranking in which the non-dominated cases are rewarded for increasing the number of solutions they dominate. Niching is applied in combination with Pareto ranking as defined by Kumar and Rockett [7] by ranking the fitness summation of the population. We focus on fitness sharing to improve diversity over time.

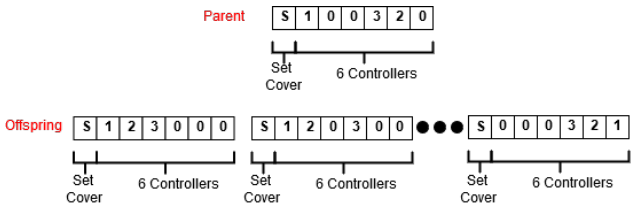

Figure 1: Reproduction A

2) *Reproduction operation:* The reproduction operation selects a random number of individuals in the population, 2 to 4 in our implementation, and performs a reproduction operation to generate children to be added into the population. Asexual reproduction is used to prevent the production of nonviable offspring that would

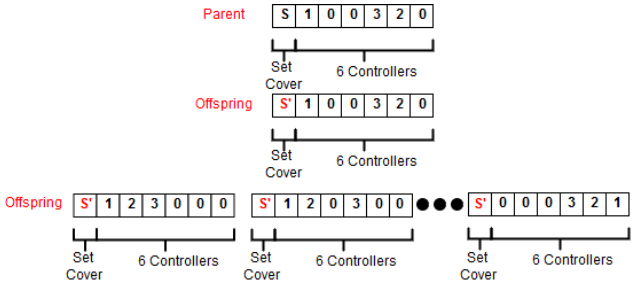

Figure 2: Reproduction B

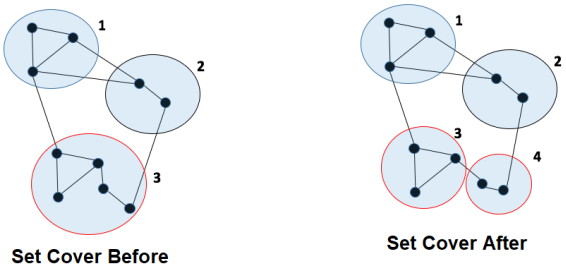

Figure 3: Set Cover Mutation

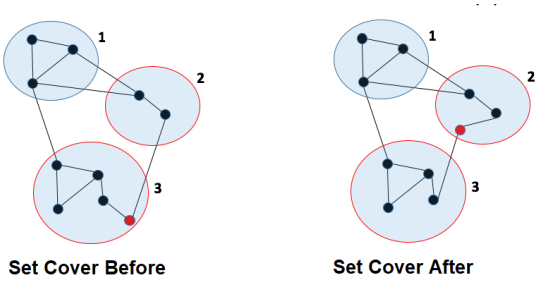

Figure 4: Domain Relocation Mutation

be generated from multi-individual reproductions. Should two different individuals be used in the reproduction operation, it would become increasingly difficult to enforce controller subgraph uniqueness and it could potentially lead to disconnected domains from differing mutations of two individuals. The first reproduction operation, Figure 1, uses the subgraph of a selected chromosome and generates a new chromosome with randomly assigned controllers. The second reproduction operation, Figure 2, selects a chromosome and mutates its set cover S using either of the mutation operations defined below and then generates a new chromosome with randomly assigned controllers from the set cover S'. The reproduction operation to be used is selected randomly every iteration according to some probability $P(a)$ and $P(b)$ where $P(a)$ is the probability of inducing reproduction A and $P(b)$ is the probability of inducing reproduction B. In our implementation $P(a) = P(b) = 0.5$. We have decided on equal probabilities in order to reduce result bias generated from favoritism to either reproduction operations.

3) *Mutation operation:* Mutation consists of four possible operations and is only applied to children generated through the reproduction operation. The first mutation operation randomly selects a utilized controller A and an un-utilized controller B and moves the subgraphs of A to B, deactivating A. The second mutation operation randomly selects two utilized controllers A and B and switches their subgraph. The third mutation operation, Figure 3, performs subgraph mutation by first selecting a random subgraph from the network and repeating the set cover generation process on the network graph, it then updates the chromosome representation by assigning new subgraphs to the controller. The fourth mutation operation, Figure 4, performs subgraph mutation by selecting two subgraphs A and B that have common links and moves a domain from A to B if the movement does not create disconnectedness for A. The mutation operation used is selected randomly with equal probability of 1/4 for all four mutation operations.

The population is trained in real time, according to Figure 5, by initializing the population using the GSGCP algorithm [6] and then evaluating the fitness of all individuals according to inter-controller latency, load distribution and number of controllers before being added to the population and then being ranked by the sum of the three fitness evaluations scaled by an individual's position in their fitness niche layer. Once the individuals have been added to the population and ranked, The multi-objective GA then generates a subpopulation of offspring using the reproduction operation followed by mutation and then restarts the fitness process until a termination generation count is reached. Generally speaking, 1000 generations should be sufficient to discover a relatively large solution set with an average fitness that improves upon the initial population.

4 EVALUATION

To properly evaluate our proposed approach, we chose to test it by running simulations on the graph implementations of the United States backbone (Section 4.1) and the Japanese backbone (Section 4.2). The United States backbone network is obtained from the Internet2 Abilene backbone [1], and the Japanese backbone is obtained from the Japanese NNTNET topology [8]. Although the number of available controllers is flexible and can be changed as required, both simulations were done using 10 controllers to clearly demonstrate the difference between a medium size graph/network and a large size graph/network given the same parameterization.

The initialization of the population is done with a modified GSGCP algorithm [12] which can generate different populations over different iterations. To avoid aberrant results due to the initial variability, and to reduce time complexity, all simulations were run a hundred times over 1000 generations before aggregation of the data. Other than maintaining the default amount of 10 controllers, we also used a mutation probability of 20% and a reproduction probability of 80% for both simulations. Fine tuning these parameters for specific networks could yield better results in all cases, but these were omitted for comparative purposes.

To demonstrate the applicability of the proposed approach, we chose to first use an 11 node United States backbone network, Figure 6. Figure 6 overlays an example of one of the best individuals of the proposed multi-objective GA. All gray links are considered inactive

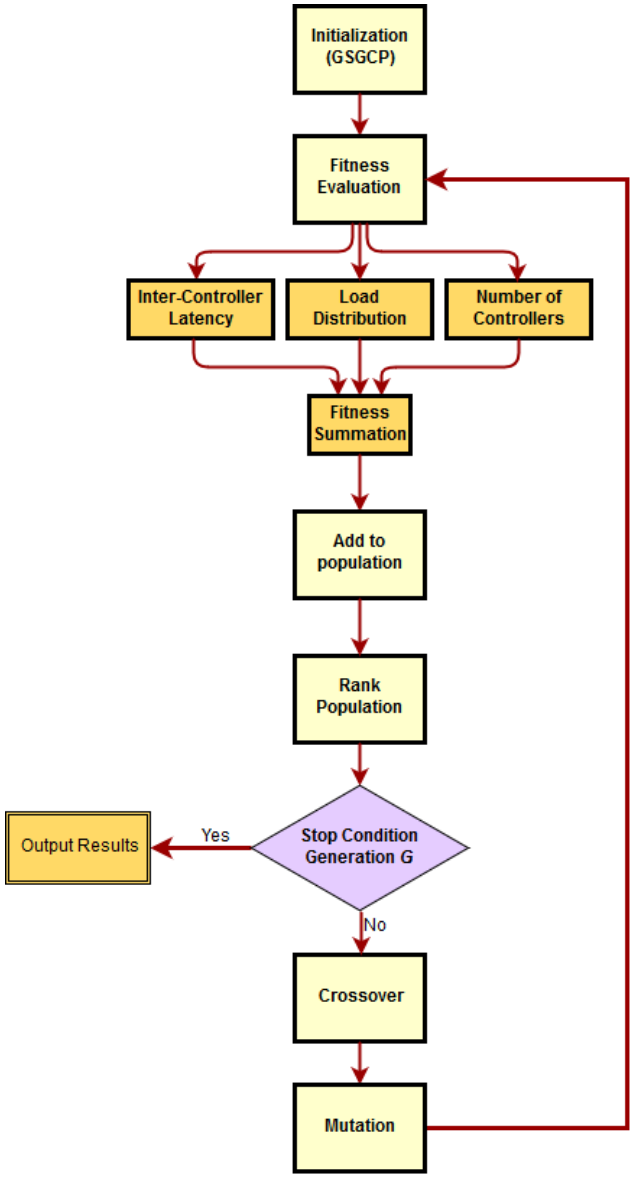

Figure 5: Genetic algorithm flowchart

and controllers (shown as red nodes on the figure), *Ci*, manage one or more colored links. This specific individual that is overlaid on this map has four out of 10 controllers active and a total fitness value of 8.1. This individual was one of 27 unique individuals out of 100 total individuals. Through simulations on the United States backbone, we noticed an increase in diversity and improvement in fitness over time.

4.1 United States Backbone

We compared diversity over time using fitness sharing and Pareto ranking and observed an overall increase in diversity with the use of fitness sharing. In Figure 7, we see that after 1000 generations the proposed multi-objective GA discovered 15 unique individuals

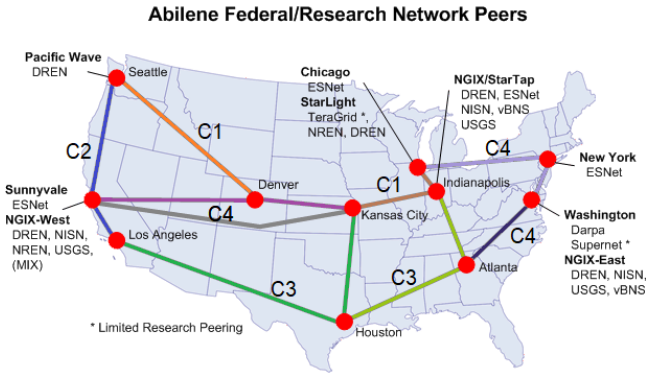

Figure 6: United States Backbone, Internet2 [1]

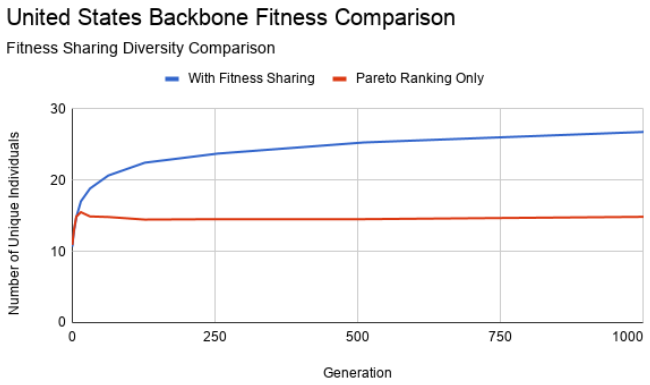

Figure 7: Comparison with and without fitness sharing.

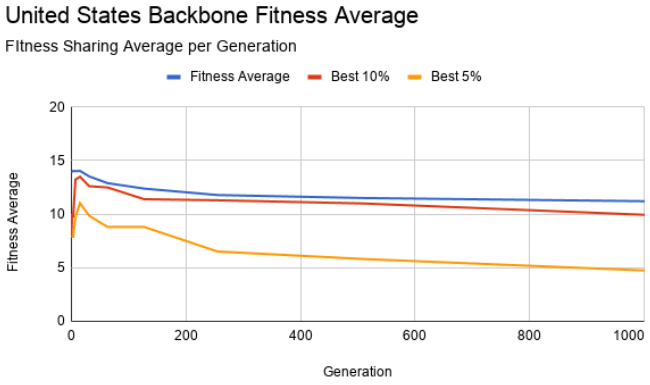

Figure 8: Count of unique individuals.

(out of 100 total individuals) without fitness sharing, as opposed to 27 unique individuals (out of 100 total individuals) with fitness sharing. Without fitness sharing, the Pareto ranking solutions with slightly less than optimal fitness were discarded too quickly.

Furthermore, although the fitness average did increase using fitness sharing, the top 5% of individuals remained unchanged, thus providing good evidence that the fitness sharing approach found the same optimal individuals while retaining boundary individuals

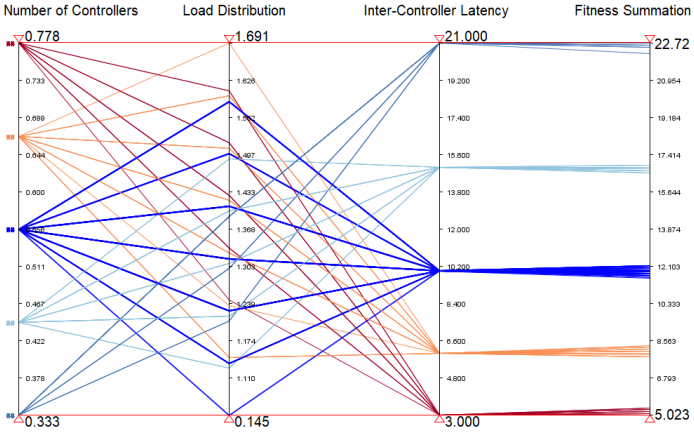

Figure 9: Fitness distribution of unique individuals

over time. In Figure 8, we notice that after 1000 generations our GA had an average fitness of 11.2, where the top 5% of individuals had a fitness of 4.7 and the top 10% had a fitness of 9.9.

Observing the fitness distribution of unique individuals after 1000 generations, Figure 9, we can visually identify a negative correlation between the fitness of controller counts and fitness of inter-controller latency. This should not be surprising as an increase in total active controllers, lowering fitness for the number of controllers, will increase the communication costs between controllers. Similarly, load distribution is loosely correlated to the number of active controllers as increasing the number of active controllers reduces the number of domains available per controller. The multi-objective GA attempts to minimize fitness and as such we notice that inter-controller latency has the largest contribution to fitness summation. Given the negative correlation between controller count fitness and inter-controller fitness, it is understandable why the proposed GA approach attempts to minimize the number of controllers in order to minimize inter-controller latency.

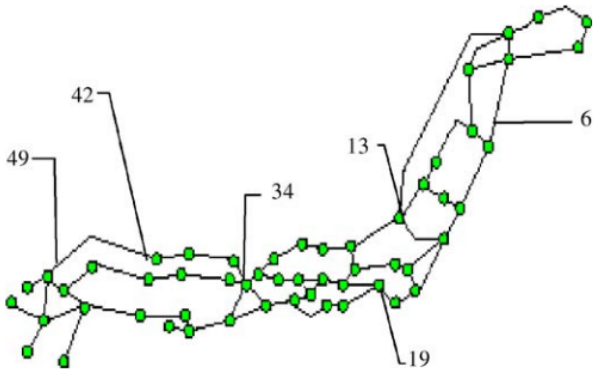

Figure 10: Japanese NNTNET topology, S. Liang et al. [8]

4.2 Japanese backbone

We also simulated the proposed GA approach on a 55 node Japanese Backbone, Figure 10, to test the GA against a larger real world network and evaluate its efficiency under pressure. That is to say, the Japanese Backbone has a long and thin topology resulting in several critical nodes that are potentially 'bottleneck' locations. Conversely, the US Backbone has fewer nodes that are more evenly distributed.

Comparing diversity over time of fitness with and without fitness sharing, we observed a similar trend as with the United States backbone. In Figure 11, we see that after 1000 generations the multi-objective GA discovered 34 unique individuals (out of 100 total individuals) without fitness sharing, as opposed to 61 unique individuals (out of 100 total individuals) with fitness sharing. Considering previous results from the United States backbone simulations, we notice an increase of nearly twice the amount of unique individuals with fitness sharing versus Pareto ranking. Similarly, the fitness sharing approach found the same optimal individuals while retaining boundary individuals over time. In Figure 12, we notice that after 1000 generations, the GA had an average fitness of 40.2 where the top 5% of individuals had a fitness of 33.9 and the top 10% had a fitness of 38.1.

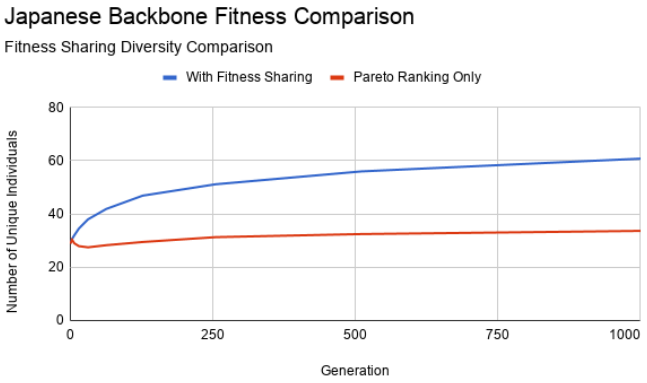

Figure 11: Comparison with and without fitness sharing.

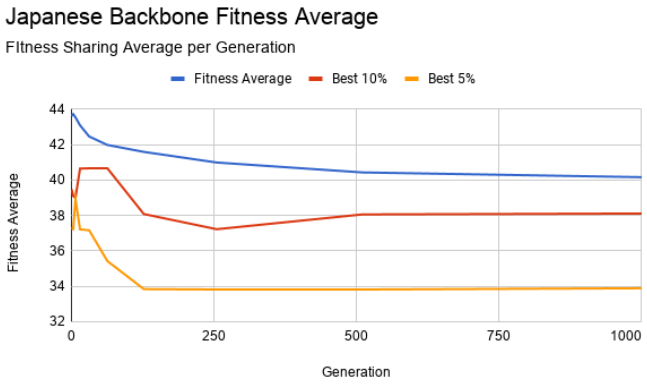

Figure 12: Count of unique individuals.

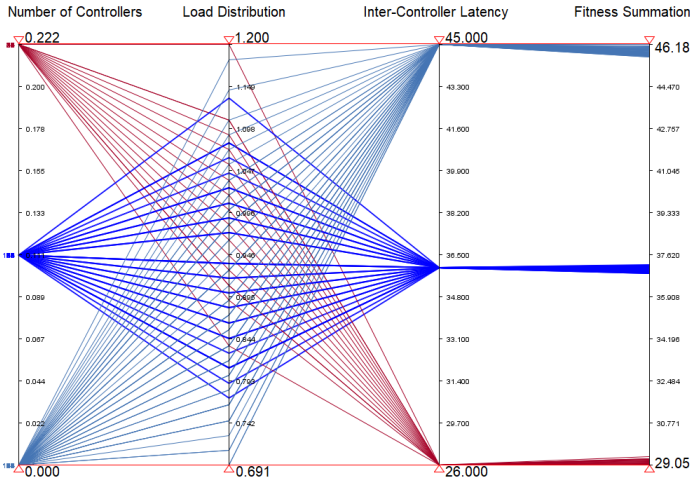

Figure 13: Fitness distribution of unique individuals

From the fitness distribution of unique individuals in the Japanese backbone, Figure 13, we observe the same correlations as in the United-States backbone. Recalling that the GA minimizes fitness, the Japanese backbone has a larger diversity of load distribution fitness frequencies caused by the increase of domains per controller and the corresponding controller sub-graph associations. As this simulation was kept at 10 controllers, it is easy to understand why their is a lower amount of diversity in the number of controllers and inter-controller latency, because it would not make sense to distribute 55 domains across fewer than 8 controllers. We can achieve a higher diversity count on these factors with an increase of the total amount of available controllers. However, because of the increased density caused by the lower amount of total controllers available, we have an increased cluster diversity as compared to the United States backbone example.

In all cases, the algorithm proved to be time efficient on a linear growth scale. Completing 1000 generations in 5.3 seconds with the 11 node United States backbone and 8.5 seconds with the 55 node Japanese backbone. These results demonstrate that the algorithm could be used in real time if diversity is a requirement as the increased cluster diversity provides longevity to any iteration of the multi-objective GA.

5 DISCUSSION

From both the US backbone and Japan backbone, we notice clear clustering patterns in population diversity upon a closer observation of Figures 13 and 9. Clusters can be labeled as individuals sharing the same number of controllers and inter-controller latency. We maintain that diversity is observed as the total number of individuals within a cluster as well as the total number of clusters.

Diversity within a cluster groups together unique individuals that are relatively close to each other in fitness. A cluster can therefore be seen as its own solution space. Individuals within the same cluster provide similar network alternatives without any significant difference in fitness. However, diversity among clusters provides a larger amount of solution spaces which can provide more solutions that are farther apart from each other in fitness. This diversity is critical because it provides a greater variety of choices for dynamic network reconfiguration.

Given the high diversity among clusters of any given fitness summation group, seen in Figures 9 and 13, we believe that our results show usefulness in the application of moving target defense in SDN networks using rapid network reconfiguration for attack mitigation. As such, we provide a security model using moving target defense in Figure 14 that could be implemented in an Intrusion Detection System (IDS) to reconfigure a network configuration on the fly in order to provide improved security.

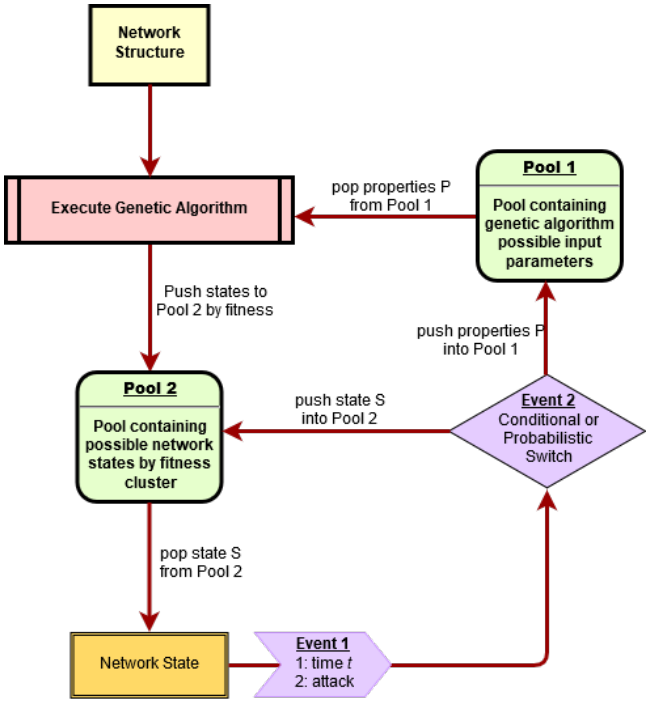

Figure 14: Moving target defense flowchart

Given a network structure and an optional set of parameters as input, the proposed GA could provide clusters arranged by fitness which could then be used to change a network's configured state when a given timer has elapsed or an attack is detected by an IDS.

The optional first pool includes sets of viable input parameters for the GA given that multiple parameters can be changed at the start of the algorithm to produce different cluster sets, we can choose different input sets from the first pool to produce increased variability between model cycles. Parameters such as the number of reproduction offspring, reproduction mutation rates, the population size and the total number of available controllers would all provide differing output sets which could gradually increase or replace the set of viable configurations within the second pool.

The second pool would consist of viable network configurations that are pushed from the GA's output, arranged into fitness clusters, from the lowest to the highest. Based on these a network configuration state would be popped from the second pool and used as a network state.

The network state could then be changed according to a timer event or an incoming attack to mitigate the value of reconnaissance and the effect of an attack on the network. Given that the network state could be changed without reiteration of the multi-objective GA, this rotation could occur without delay if desired.

If a time delay is acceptable, then a second switch event can be used to increase the size of the second pool by rerunning the GA with different parameters, chosen from the first pool. This second event could be conditional, if any state S has been used more than some desired amount, or probabilistic.

We believe that a similar model to the one shown in Figure 14 could be used for reliability assurance by establishing different conditions on the execution of the first event. The diversity of and within fitness clusters provides similar network configurations that could change the network configuration should a different subgraph distribution be required.

6 CONCLUSION

Looking at the results, we notice that fitness sharing improves diversity over time without removing the top individuals. Adding on the initial variability from the modified GSGCP algorithm [12] to the variability provided by fitness sharing, the proposed multi-objective GA approach is able to discover a large solution space which could be used for network reliability or in moving target defense. The varying fitness clusters demonstrate a large set of alternative network configurations with similar controller layouts and inter-controller latencies which provide varying controller subgraph distributions with unique load distributions. This variance in load distributions offers similar controller subgraph distributions for reliability by providing different domain subgraphs with similar inter-controller latencies.

Furthermore, we suggest that the wide solution space could be used to implement moving target defense in SDNs of varying sizes which retain near optimal controller placement. We believe that the implementation of this approach for moving target defense could improve network security by increasing active information gathering complexity and mitigating denial of service attacks through dynamic controller subgraph and domain reallocation. Future work will include testing of the suggested MTD security model in Figure 14 using the proposed multi-objective GA based approach. Moreover, we will explore parameter optimization of the GA for the general and specific network cases. Our results seem to indicate that more parameter testing may lead to substantial improvements in diversity and fitness in all network topologies.

7 ACKNOWLEDGEMENT

This research was partially supported by Natural Science and Engineering Research Council of Canada (NSERC). It is conducted as part of the Dalhousie University Network Information Management and Security (NIMS) Lab at: http://projects.cs.dal.ca/projectx/.

REFERENCES

[1] 2017. Abilene Backbone Network. Internet2. (2017). https://www.internet2.edu
[2] Md. Faizul Bari, Arup Raton Roy, Shihabur Rahman Chowdhury, Qi Zhang, Mohamed Faten Zhani, Reaz Ahmed, and Raouf Boutaba. 2013. Dynamic Controller Provisioning in Software Defined Networks. In *International Conference on Network and Service Management*. 18–25.
[3] H. Bo, W. Chuan'an, and W. Ying. 2016. The controller placement problem for software-defined networks. In *IEEE International Conference on Computer and Communications*. 2435–2439.
[4] Brandon Heller, Rob Sherwood, and Nick McKeown. 2012. The controller placement problem. *Computer Communication Review* 42, 4 (2012), 473–478.
[5] David Hock, Steffen Gebert, Matthias Hartmann, Thomas Zinner, and Phuoc Tran-Gia. 2014. POCO-framework for Pareto-optimal resilient controller placement in SDN-based core networks. In *IEEE Network Operations and Management Symposium*. 1–2.
[6] Ahmad Jalili, Vahid Ahmadi, Manijeh Keshtgari, and Morteza Kazemi. 2015. Controller Placement in Software-Defined WAN Using Multi Objective Genetic Algorithm. In *International Conference on Knowledge-based Engineering and Innovation*. 656–662.
[7] Rajeev Kumar and Peter Rockett. 2002. Improved Sampling of the Pareto-Front in Multiobjective Genetic Optimizations by Steady-State Evolution: A Pareto Converging Genetic Algorithm. *Evolutionary Computation* 10, 3 (2002), 283–314.
[8] S. Liang, A.N. Zincir-Heywood, and M.I. Heywood. 2006. Adding more intelligence to the network routing problem: AntNet and Ga-agents. *Applied Soft Computing* 6, 3 (2006), 244–257.
[9] Adetokunbo Makanju, A. Nur Zincir-Heywood, and Shinsaku Kiyomoto. 2017. On evolutionary computation for moving target defense in software defined networks. In *ACM Genetic and Evolutionary Computation Conference*. 287–288.
[10] A. Sabeegh, Y. Al-Dunainawi, M. F. Abbod, and H. S. Al-Raweshidy. 2016. A hybrid intelligent approach for optimising software-refined networks performance. In *International Conference on Information Communication and Management*. 47–51.
[11] Jean-Michel Sanner, Yassine Hadjadj Aoul, Meryem Ouzzif, and Gerardo Rubino. 2017. An evolutionary controllers' placement algorithm for reliable SDN networks. In *International Conference on Network and Service Management*. 1–6.
[12] G. Wang, Z. Zhao, J. Peng, R. Li, and H. Zhang. 2014. An approximate algorithm of controller configuration in multi-domain SDN architecture. In *International Conference on Communications and Networking in China*. 601–605.