An evolutionary controllers' placement algorithm for reliable SDN networks

Jean-Michel Sanner, Yassine Hadjadj-Aoul, Meryem Ouzzif, Gerardo Rubino

An evolutionary controllers' placement algorithm for reliable SDN networks

Jean-Michel Sanner[§], Yassine Hadjadj-Aoul[§], Meryem Ouzzif[*], Gerardo Rubino[‡],
[*]Orange Labs, Cesson Sévigné, France
[‡]INRIA, Rennes, France
[§]B-COM, Rennes, France

***Abstract*—SDN controllers placement in TelCo networks are generally multi-objective and multi-constrained problems. The solutions proposed in the literature usually model the placement problem by providing a mixed integer linear program (MILP). Their performances are, however, quickly limited for large sized networks, due to the significant increase in the computational delays. In order to avoid the inherent complexity of optimal approaches and the lack of flexibility of heuristics, we propose in this paper a genetic algorithm designed from the NSGA II framework that aims to deal with the controller placement problem. Genetic algorithms can, indeed, be both multi-objective, multi-constraints and can be designed to be computed in parallel. They constitute a real opportunity to find good solutions to this category of problems. Furthermore, the proposed algorithm can be easily adapted to manage dynamic placements scenarios. The goal chosen, in this work, is to maximize the clusters average connectivity and to balance the control's load between clusters, in a way to improve the networks' reliability. The evaluation results on a set of network topologies demonstrated very good performances, which achieve optimal results for small networks.**

***Index Terms*—Placement, SDN, NFV, Genetic algorithm, Reliability**

I. Introduction

Mastering the increasing complexity of current and future networks, while reducing the operational and investments costs, is one of the major challenges faced by network operators (NO). This explains in large part the recent enthusiasm of NOs towards Software Defined Networking (SDN) and Network Function Virtualization (NFV). Indeed, on the one hand, the SDN makes it possible to get rid of the control plane distribution complexity, by centralizing it logically, while allowing its programmability. On the other hand, the NFV allows virtualizing the network functions, which considerably facilitates the deployment and the orchestration of the network resources.

Providing a carrier grade network involves, however, several requirements such as providing a robust network meeting the constraints of the supported services [1]. In order to achieve this objective, it is clearly necessary to scale up the number of controllers, while placing them strategically in a way to guarantee the system's responsiveness.

In this paper, our main concern is to design a controllers' placement strategy maximizing the constituted clusters average connectivity while balancing the controllers' load. Another important concern consists in finding the suitable number of controllers, which is considered as a fixed parameter in most of the existing works. In order to avoid the inherent complexity of optimal approaches and the lack of flexibility of heuristics to be extended with new metrics or constraints, we propose an evolutionary algorithm-based technique to explore efficiently the very large space of solutions.

The remainder of this paper is organized as follows. Section III provides a detailed description of the system and the considered optimization problem. Section IV describes the proposed genetic algorithm-based placement strategy. Section V portrays the simulation setup and discusses the obtained results. Finally, the paper concludes in Section VI.

II. Related works

The problem of the placement of virtual network functions, and more specifically network controllers, has been the subject of many studies in the literature.

In [1], the authors analyzed the impact of the latency - minimal and average - and the number of controllers on network performance. Guaranteeing a minimal latency between a node and its corresponding controller is certainly an important metric to be considered. Contrariwise, while sufficient in most use cases, the average latency hides worst cases, which may lead to unsatisfied applications' requirements [2]. The authors, in [3] and [4], proposed a heuristic to balance the controllers' loads by creating clusters of nodes with similar sizes.

It should be noted, however, that the controllers' placement is influenced by network operators, who wants to get a return on their investments, and the services' providers who wants to have guarantees in the service delivery. This makes more complex the decision-making process as it should realize a trade-off between conflicting objectives.

In [2], the authors proposed a greedy algorithm, which automatically determines the number of controllers, while considering, in addition to the load, the latency between the nodes and their respective controller in a way to be compliant with the requirements of the supported services. In [5], the authors proposed a placement strategy maximizing the reliability of the network, which is a key feature to provide a carrier grade network. Similarly, the authors, in [6], suggested the use of the expected percentage of control path loss as a reliability metric. The obtained results clearly show an improved reliability.

The space's exploration, while computationally efficient with these heuristics, can ends up generally at a local minimum and it could not be easily generalized to other metrics. In

this way, similarly to the MOCO approach [7], the authors in [8], proposed to use a standard NSGA II genetic algorithm to deal with a three objective placement strategy, minimizing the latency between nodes and their respective controllers, minimizing the inter-controllers latency and minimizing the unbalance between clusters. However, the number of controllers is considered as a fixed parameter.

To go further, in this paper we propose a new approach based on evolutionary algorithms, which address the network reliability and load balancing of the controllers. The proposed solution is also able to determine the appropriate number of controllers, in opposition to most of the existing work. Besides, the algorithm can be easily extended to include other metrics and can eventually be integrated in a dynamic placement policy.

III. Overview of the system

A. System description

In this paper, we consider the architecture of a Telco NO, in which network elements are driven by few distributed SDN controllers [9]. The latter are responsible of the control plane, which drives the execution of network services, while the network's elements (i.e., switches) are in charge of the forwarding plane functions.

Without loss of generality we consider in the following, an architecture where each controller can be located on any node of the network. Besides, our objective when deploying services is to choose conveniently the number of controller while locating them optimally in the network in a way to respect the services constraints.

Several metrics can be considered here, however, our focus mainly concerns achieving:

- **High average connectivity:** The average connectivity gives a better picture of the graph reliability, since it provides a measure of the global "amount" of connectivity of a graph [10]. Thus, by maximizing the average connectivity of the clusters, we expect to maximize the number of disjoint paths between the nodes of a cluster and their corresponding controller [11] and, therefore, the global network reliability.
- **Good balance between the controllers' load:** By minimizing the imbalance of control traffic between clusters, we expect to reduce the latency between the controllers and the switches in a cluster. This, also, prevents controllers' overload.
- **Reduced number of controllers:** By minimizing the number of controllers we expect reducing operational costs.

B. Model description

The goal of our model is to provide the theoretical framework to solve a cluster configuration of nodes which maximizes the sum of the average edge connectivity of clusters while minimizing the imbalance between clusters. The model is based on linear programming, and uses the maximum flow, minimum cut theorem [12].

Let $G(V, E)$ an undirected graph, where V is the set of nodes in the network with $|V| = m$ and E is the set of edges with $|E| = n$. Let defined $c(u_i) \in \{-1, 0, 1\}$ as the capacity function of edges (except for the virtual return edge u_r, $c(u_i) \leq \infty$).

The following linear program expresses the maximum flow between a source node s and a destination node d and therefore the edge connectivity between them in graph G [12].

$$\begin{aligned} \underset{u_r}{\text{maximize}} \quad & f(u_r) \\ \text{subject to} \quad & A.f = 0 \end{aligned} \tag{1}$$

where A is the incidence matrix of G, u_r is the virtual return edge between s and d, which is used for the maximum flow calculation, and f is a vector in $\mathbb{R}^n$ representing the flow with $f(u_i) \leq c(u_i)$ for all u_i.

The maximum flow equals the minimum cut between s and d, and, therefore, it also represents the connectivity K between s and d in G:

$$k_G(s, d) = \arg \max_f f(u_r) \tag{2}$$

Using an enumeration of all the pairs of nodes of the graph, we can express the average edge connectivity of the graph $k(G)$ as the average connectivity of each pair of nodes of the graph:

$$\bar{k}(G) = \sum_{(i,j),\ i<j} \frac{k_G(i, j)}{\binom{m}{2}} \tag{3}$$

Let define $f_{ij} \in \mathbb{R}^n$ as the flow related to x_i and x_j, including the virtual return edge between them. Using equations (1) and (3) we obtain:

$$\begin{aligned} \underset{f_{ij}}{\text{maximize}} \quad & \frac{\sum_{(i,j),\ i<j} f_{ij}(x_i, x_j)}{\binom{m}{2}} \\ \text{subject to} \quad & \\ & \forall (i, j),\ i < j, A.f_{ij} = 0 \end{aligned} \tag{4}$$

The following intermediate linear problem gives the sub-graph (i.e. the cluster) $G_k(V_k, E_k)$ of number of nodes p (*i.e.* $|V_k| = p$) which has the highest average connectivity.

$$\begin{aligned} \underset{A_k}{\text{maximize}} \quad & \frac{\max \sum_{(i,j) i<j} f^k_{ij}(x_i, x_j)}{\binom{p}{2}} \\ \text{subject to} \quad & \\ & \forall k, |V_k| = p \\ & \forall k, (i, j), i < j, A_k.f^k_{ij} = 0 \end{aligned} \tag{5}$$

where A_k is the incidence matrix of sub-graph G_k, f^k_{ij} is its maximum flow, and $\binom{p}{2}$ is the number of pairs of nodes in the sub-graph.

We have then our final model which deals with several clusters with a variable number of nodes and the load induced by nodes to controllers. Our goal is to select the optimal clusters to maximize the average connectivity and to balance

the load between them. We express, then, the additional objective of balancing the load between clusters.

We have the additional following parameters:

- a number of clusters $k_m \in [k_{min}, k_{max}]$ to deal with the load induced by all the switches in the network and to find a configuration with the best balance of load between clusters.
- each G_{k_m} denotes a cluster with an arbitrary number of nodes
- p_{k_m} is the number of nodes of G_{k_m}
- $l(G_{k_m})$ is the load of sub-graph G_{k_m} associated to the cluster with $l(G_{k_m}) = \sum_{i \in V_{k_m}} l_i$, where l_i is the load of the node i.

The final bi-objective linear programming model we want to solve is as follows:

$$\begin{aligned} &\underset{\cup G_{k_m}}{\text{maximize}} && \frac{\min_{k_m \in [1,k_m]} l(G_{k_m})}{\max_{k_n \in [1,k_m]} l(G_{k_n})} \\ &\underset{\cup G_{k_m}}{\text{maximize}} && \sum_{k=1}^{k_m} \frac{\sum_{(i,j) i<j} f^k_{m_{ij}}(x_i, x_j)}{\binom{p_{k_m}}{2}} \\ &\text{subject to} && \\ & && \sum_{i=1}^{k_m} |G_{k_i}| = m \\ & && \forall k_m, (i,j), i<j, A_{k_m}.f^{k_m}_{ij} = 0 \end{aligned} \tag{6}$$

C. Complexity analysis

Let S be a set of K clusters *i.e.* a subset families of V with $|S| = K$. The number of existing partitions, *i.e.* the size of the research space is $\left\{ {N \atop K} \right\}$. This number is known as the stirling number of the second kind and grows exponentially with N for a fixed K. It is worth noting that this number is much greater than the number of combinations of possible K controllers usually given in papers on controllers placement. For instance, with $K = 3$ and N = 30: we have $\binom{N}{K} = 4060$ and $\left\{ {N \atop K} \right\} = 34314651811530$.

On the other hand, our problem is a dual problem of a special case of the minimum set covering problem [13] where the weight of a partition of the reference set is the average connectivity of the corresponding sub-graph. Therefore our reference problem is proved to be NP-hard.

IV. Proposal

A. The algorithm

NSGA-II [14] multi-objective evolutionary algorithm demonstrates very good performances particularly in the case where objectives are conflicting each other. As the maximum average connectivity and the load imbalance between clusters are uncorrelated objectives, we decide to use NSGA-II as a basic framework to solve our problem. As the framework NSGA-II has a very generic structure, it is useful to notice that it can also be extended easily to a third objective (or more) aiming to minimize for example the number of controllers. The structure of the algorithm is pictured with a pseudo code in Algorithm 1. The input and output parameters are:

- *nbGen*, the number of generations, *popSize*, the population size,
- *netGraph*, the data structure which represents the network graph,
- *MUTPB*, the mutation rate, *CXPB*, the crossover rate,
- *kMin*, *kMax*, the acceptable minimum and maximum number of clusters,
- *pf*, is the output *i.e.* the set of solutions constituting the Pareto front at the end of algorithm,

The first step of the algorithm, in lines 3 to 6, consists in creating randomly a population of individuals. In our case, individuals are a list of clusters of nodes *i.e.* a list of sub-graphs which cover the entire network graph. The number of clusters of each individual: *k*, can vary between *kMin* and *kMax*. These two thresholds are estimated using the cumulative control plane load induced by all the nodes in the network. *kMin* must be large enough to provide a sufficient number of controllers to accept the control plane load induced by all the switches. *kMax* must be large enough to prevent from controllers overhead.

1) The evaluation operator: The evaluation operator, in lines 8 to 11, computes a fitness, corresponding to each objective, of each individual. In our case, the first one, is the average of the average connectivity of each cluster *i.e.* sub-graphs of the individual. The average connectivity of a sub-graph is computed using a classical maximum flow based routine between each pair of nodes and averaged with the number of pairs of nodes in the sub-graph. The average connectivity of a sub-graph is set to 0 if the sub-graph has several connected components *i.e.* is disconnected. This allows us to accelerate strongly the convergence of the algorithm. The second objective is the load imbalance between the heaviest cluster and the lightest cluster of an individual. The load of a cluster is the sum of the control plane load induced by each node in the cluster. The imbalance of an individual is the ratio of the lightest cluster to the heaviest cluster. If the value is 1, the load is perfectly balanced between the different clusters of an individual.

2) The mutation operator: The mechanism of the mutation operator, in lines 12 to 14, consists in exchanging some nodes between clusters. Based on the mutation rate, a number of nodes are randomly extracted from the network and reallocated randomly using a uniform law in the clusters of the current individual. The proportion of nodes randomly extracted is defined by the *MUTPB* parameter.

3) The cross-over operator: The cross-over operator, in line 15 to 19, is optional. It is not used in this version of the algorithm and is the subject of further studies.

After the mutation and the optional crossover step, a new offspring population is produced, in lines 20 to 22. The offspring population is merged with the current population, and a new population is set up using the NSGAII selection process with the fast dominated sorting and diversity preservation strategies which are the essential features of NSGAII [14]. As a result, the pareto-front data is updated.

Algorithm 1 MAIN: Evolutionary algorithm

INPUT: *nbGen*, *popSize*, *netGraph*, *MUTPB*, *CXPB* *kMin*, *kMax*
OUTPUT: *pf*

for *index* $= 1$ **to** *popSize* **do**
  $k \Leftarrow$ `random`(*kMin*..*kMax*)
  population $\Leftarrow$ `randomClustering`(k, *netGraph*)
end for

pf $= \emptyset$

for *index* $= 1$ **to** *nbGen* **do**
  for *individual* $\in$ *population* **do**
    `Evaluate`(*individual*)
  end for

  for *individual* $\in$ *population* **do**
    `Mutate`(*MUTPB*, *individual*)
  end for

  for *index* $= 1$ **to** *popSize* **do**
    parents $\Leftarrow$ `binarySelection`(*population*)
    child $=$ `cross-over`(*CXPB*, *parents*)
    offSpring $\Leftarrow$ *child*
  end for

  population $\Leftarrow$ `SelectNSGA2`(*population* $\cup$ *offSpring*)
  pf $\Leftarrow$ `UpdateParetofront`(*pf*, *population*)
end for
return *pf*

B. Computational complexity

The computational complexity of the NSGA II algorithm is driven by the Pareto front classification process which is in $O(M \cdot popSize^2)$ where M is the number of objectives.
To express the overall computational complexity of our algorithm, we have also to take in account the number of generations $nbGen$ and the computational complexity of additional operations inside each generation run: the evaluation process, the mutation process and the crossover process and finally the selection process. The complexity of the individual evaluation process is bounded by the complexity of computing the connectivity of a pair of nodes which is in $O(|E|^2 \cdot |V|)$ plus computing the global load of the graph in $O(|V|)$. The complexity of the mutation operator is bounded by the number of nodes $O(|V|)$. The complexity of the cross-over process depends of the cross-over strategy chosen. To summarize, in our case without a cross-over process the computational complexity is bounded roughly in
$O(nbGen \cdot (M \cdot popSize^2 + popSize \cdot |E|^2 \cdot |V|^3))$.
It is acceptable even if the network contains some hundreds of nodes.

V. Performance evaluation

Having described the detail of our solution, we direct now our focus on its evaluation. We use for our evaluation network instances extracted from the zoo-topology database [15], and completed in a way to have connected graphs. A control plane load was generated randomly and attached to the nodes of the network's instances.

A. Reference solution

In order to have a reference solution, we compare the pure mutation approach with the optimal results obtained using the Geocode solver and the minizinc modelling language [16] to express our ILP model described in section IV. We compare the solution provided by the solver and the average connectivity value of some solutions chosen in the Pareto front. The prohibitive execution time needed to perform this optimization problem for the solver limits us to networks with a small number of nodes and edges, roughly around 20 nodes or edges. The obtained results are displayed in diagram 1. The results obtained with the solver are identical to those provided by the algorithm. There is one mismatch (Fatman) due to the limited time provided to solve the problem and to find the optimal solution (*i.e.* several hours) because the number of clusters to solve is greater than 3 in this case.

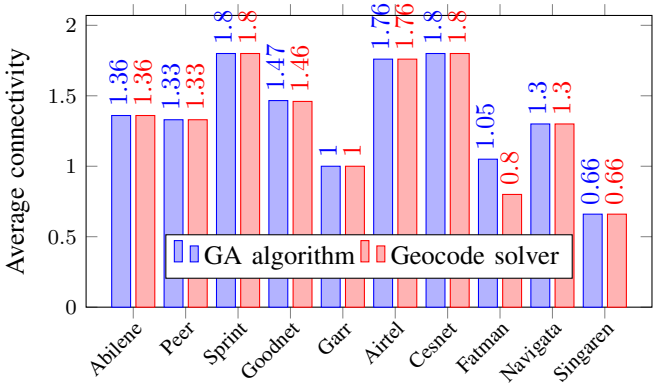

Fig. 1: Comparison GA/Optimal

B. Qualitative analysis of the algorithm

We also evaluate the algorithm on some small relevant networks configurations using the pure mutation version.

In figure 2 we have several clusters with different average connectivity values. The mutation version of the algorithm is used with a low number of generations –20– due to the small size of the network instance. When selecting a reasonable number of clusters, varying from 2 to 6, the best solution is obtained for three clusters: $\{1, 2, 3, 4\}\{5, 6, 7, 8\}\{9, 10, 11, 12\}$, where the average connectivity is: 3 for the first cluster, 2.16 for the second cluster and 2 for the third one. The average connectivity of the three clusters put together is 2.38. Theses results are consistent with those provided by an implementation on our linear model using the Minizinc solver.

C. Comparison with Kmean++

To complete the evaluation of the algorithm ability , we compare its performances on some medium size networks (*i.e.*using a Kmean++ [17] clustering algorithm implementation and a restricted scenario, where we fix the number of

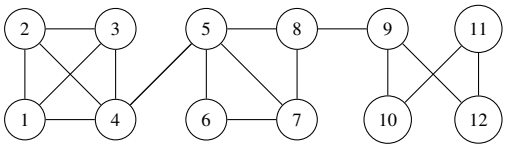

Fig. 2: Small network example

clusters with an arbitrary value, ($k = 3$) as it is required by Kmean. Moreover, the algorithm uses as an input the connectivity matrix of the network graph to measure the similarity between nodes. In addition to that, the control load induced by each node is fixed at the same value. Then we run the GA on 100 generations, which is a reasonable number to achieve good results with the selected network instances. The difference in results for the average connectivity and the load are displayed in Figure 3. It shows that the GA provides better results even in this simplistic scenario. The average connectivity points and their imbalance values obtained with the Genetic algorithm are always better. Even if sometimes the average connectivity is the same, the imbalance objective is always worse with Kmean. The number of nodes and edges of each network varies between 20 and 70 in this experiment and are displayed in the table II.

Networks features: nodes and edges		
Network	Nb of nodes	nb of Edges
Garr200212	28	27
Arnes	34	45
PionnierL3	38	45
Geant2012	40	60
SwitchL3	42	62
Renater2010	43	55
Garr201004	54	68
Forthnet	62	62
AsnetAM	65	77
Telcove	72	73
Latnet	74	69

TABLE I: Networks features: nodes and edges

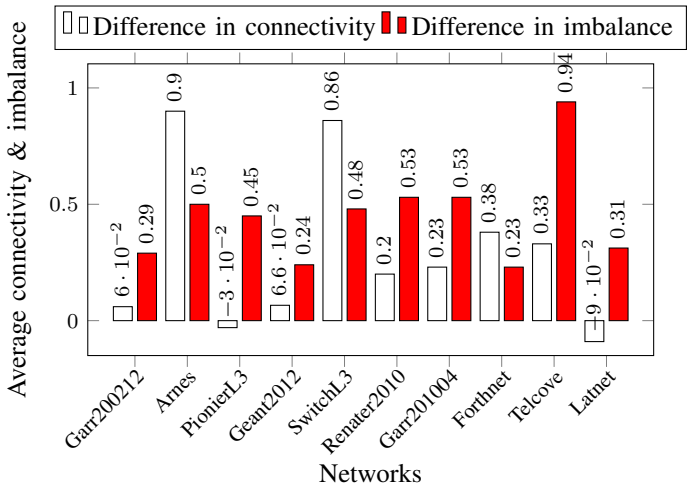

Fig. 3: Comparison Kmean/GA

D. Algorithm convergence

Figure 4 illustrates an example of the Pareto front obtained on a typical medium size network with 40 nodes and 60 edges extracted from the zoo topology database [15]. The curve in green shows a Pareto front obtained after a 50 generations run with the pure mutation version of the algorithm and a number of clusters varying from 2 to 6. The mutation rate MUTPB is fixed to 0.05 and the crossover rate CXPB is fixed to 0.9, the size of the population is 200 individuals. The optimal value for the load balancing is 1, the average connectivity depends on networks topologies. For low average connectivity values, the clusters are perfectly balanced while for higher connectivity values the imbalance increases. Each point of the Pareto front is annotated with the corresponding number of controllers of the solution which is always 3 in this run. The red curve shows the Pareto front obtained with a run along 100 generations. We can see the curve delimiting the Pareto front moving on the right towards better average connectivity solutions and that the number of clusters goes to the minimum. After 150 generations, there are no significant improvements of the solution and we are probably close to the optimal.

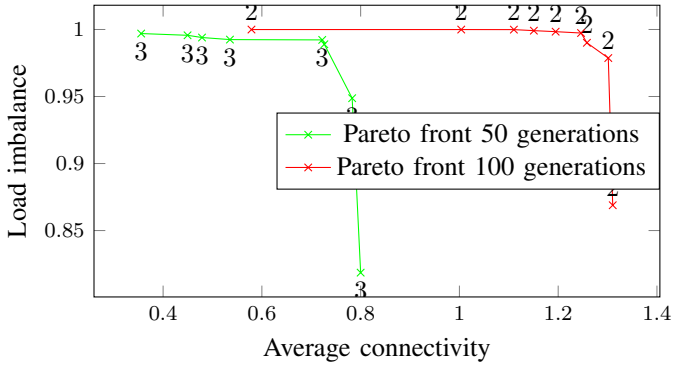

Fig. 4: Middle size network: 40 nodes, 60 edges

Figure 5 displays with smooth lines the convergence along the generations of the algorithm running on the middle size network already tested. We iterate 50 times the algorithm to provide a relevant average value of each point along the generations. There are three criteria to evaluate the convergence of the algorithm: the maximum average connectivity value found in the Pareto front, the attached imbalance value of the selected point, the hyper-volume indicator [18] of the Pareto front.

The hyper-volume indicator is calculated relatively to the reference point with the best imbalance value *i.e.* 1.0 and with the highest rounded average connectivity value among all the points of the different Pareto fronts of each generation. The lower is the hyper-volume indicator, the better are the solutions provided by the Pareto front. There are no significant improvements of the three criteria beyond 150 generations.

E. Computation time

A comparison of the computation time of the algorithm and of the solver is not really relevant because the solver computation time become quickly prohibitive even with small

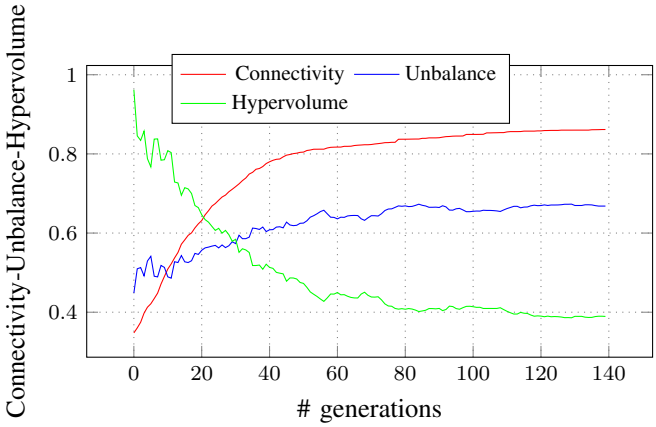

Fig. 5: Evolution along 150 generations - Mutation only GA

networks instances. For example with the Fatman network having 17 nodes and 21 edges, it takes half of an hour to the solver locked on three clusters to provide the optimal solution when running on a $IntelXeon(R)CPU_E5-1620v2\ 3.7Ghz$. Otherwise, the algorithm coded in interpreted Python language provides solutions in $64s$. The evolutionary algorithm uses the following parameters:$popSize = 200$, $nbGen = 200$, $kmin = 2$, $kmax = 6$.

The table II illustrates how the algorithm computation time can vary with the networks number of nodes and edges. The computation time is strongly dependant of the network topology and connectivity because it depends on the probability that the clusters selected by the algorithm are connected or not.

Network	Nb of nodes	nb of Edges	Time
Garr200212	28	27	73 s
Arnes	34	45	276 s
PionnierL3	38	45	369 s
Geant2012	40	60	306 s
SwitchL3	42	62	304 s
Renater2010	43	55	467 s
Garr201004	54	68	721 s
Forthnet	62	62	391 s
AsnetAm	65	77	78 s
Telcove	72	73	318 s
Latnet	74	69	54 s
Pern	129	127	3324 s

TABLE II: Computation time on some networks

VI. Conclusion

We propose in this paper a controllers' placement strategy, which allows obtaining optimal solutions, when having small network instances, as we validated by comparison with the solutions provided by an ILP solver. For medium network instances, it is able to provide good solutions in a small number of generations, around 150.

Thanks to the structure of the algorithm, it can be easily extended to a large set of placement problems, such as virtual machines placements in data centers, or network virtual functions placements or even chains of virtual network functions. The number of objectives can easily be extended thanks to the NSGA II algorithm structure. Moreover, the optional cross-over step can be, also, considered to improve the algorithm in a way to reduce the number of generations. However, a cross-over strategy need to be defined and deeply analyzed. This will be considered in our future work.

References

[1] B. Heller, R. Sherwood, and N. McKeown, "The controller placement problem," in *Proc. of the First Workshop on Hot Topics in Software Defined Networks*, ser. HotSDN '12, 2012, pp. 7–12.

[2] J. M. Sanner, Y. Hadjadj-Aoul, M. Ouzzif, and G. Rubino, "Hierarchical clustering for an efficient controllers' placement in software defined networks," in *2016 Global Information Infrastructure and Networking Symposium (GIIS)*, Oct 2016, pp. 1–7.

[3] G. Yao, J. Bi, Y. Li, and L. Guo, "On the capacitated controller placement problem in software defined networks," *IEEE Communications Letters*, vol. 18, no. 8, pp. 1339–1342, Aug 2014.

[4] J. Liao, H. Sun, J. Wang, Q. Qi, K. Li, and T. Li, "Density cluster based approach for controller placement problem in large-scale software defined networkings," *Computer Networks*, vol. 112, pp. 24 – 35, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1389128616303620

[5] F. J. Ros and P. M. Ruiz, "On reliable controller placements in software-defined networks," *Comput. Commun.*, vol. 77, no. C, pp. 41–51, Mar. 2016. [Online]. Available: http://dx.doi.org/10.1016/j.comcom.2015.09.008

[6] Y. Hu, W. Wang, X. Gong, X. Que, and S. Cheng, "On reliability-optimized controller placement for software-defined networks," *China Communications*, vol. 11, no. 2, pp. 38–54, Feb 2014.

[7] S. Lange, S. Gebert, T. Zinner, P. Tran-Gia, D. Hock, M. Jarschel, and M. Hoffmann, "Heuristic approaches to the controller placement problem in large scale sdn networks," *IEEE Transactions on Network and Service Management*, vol. 12, no. 1, pp. 4–17, 2015.

[8] A. Jalili, V. Ahmadi, M. Keshtgari, and M. Kazemi, "Controller placement in software-defined wan using multi objective genetic algorithm," in *2015 2nd International Conference on Knowledge-Based Engineering and Innovation (KBEI)*, Nov 2015, pp. 656–662.

[9] J. M. Sanner, M. Ouzzif, and Y. Hadjadj-Aoul, "Dices: A dynamic adaptive service-driven sdn architecture," in *Proceedings of the 2015 1st IEEE Conference on Network Softwarization (NetSoft)*, April 2015, pp. 1–5.

[10] L. W. Beineke, O. R. Oellermann, and R. E. Pippert, "The average connectivity of a graph," *Discrete Mathematics*, vol. 252, no. 1, pp. 31 – 45, 2002. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0012365X01001807

[11] J. Bar-Ilan and D. Peleg, *Approximation algorithms for selecting network centers*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1991, pp. 343–354. [Online]. Available: http://dx.doi.org/10.1007/BFb0028274

[12] M. Sakarovitch, *Optimisation combinatoire: Programmation discrète*, ser. Collection Enseignement des sciences. Hermann, 1984. [Online]. Available: https://books.google.fr/books?id=uUrvAAAAMAAJ

[13] R. Karp, "Reducibility among combinatorial problems," in *Complexity of Computer Computations*, R. Miller and J. Thatcher, Eds. Plenum Press, 1972, pp. 85–103.

[14] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: Nsga-ii," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, Apr 2002.

[15] S. Knight, H. Nguyen, N. Falkner, R. Bowden, and M. Roughan, "The internet topology zoo," *Selected Areas in Communications, IEEE Journal on*, vol. 29, no. 9, pp. 1765 –1775, october 2011.

[16] N. Nethercote, P. J. Stuckey, R. Becket, S. Brand, G. J. Duck, and G. Tack, "Minizinc: Towards a standard cp modelling language," in *Proceedings of the 13th International Conference on Principles and Practice of Constraint Programming*, ser. CP'07, 2007, pp. 529–543.

[17] D. Arthur and S. Vassilvitskii, "K-means++: The advantages of careful seeding," in *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, ser. SODA '07, 2007, pp. 1027–1035.

[18] A. Auger, J. Bader, D. Brockhoff, and E. Zitzler, "Hypervolume-based multiobjective optimization: Theoretical foundations and practical implications." *Theor. Comput. Sci.*, vol. 425, pp. 75–103, 2012.