# Reimplementation of Research: Slice as an Evolutionary Service: Genetic Optimization for Inter-Slice Resource Management in 5G Networks

İdil Zeynep Alemdar
*Department of Computer Engineering*
*Middle East Technical University*
Ankara, Turkey
idil.alemdar@metu.edu.tr

***Abstract*—Network slicing is a recently emerging networking paradigm aiming to reduce hardware dependencies in networking by establishing virtualized network slices over a single physical infrastructure. As a result, it will be possible to launch several differently configured networks serving different purposes, as well as to have these networks dynamic and flexible. In this work, I tried to reimplement a research conducted by Han et al. [1] which used evolutionary computation techniques to handle incoming slice generation requests while aiming to optimize overall network utilization. The results I obtained showed that increasing population also increased both the fitness of the best solution as well as the fitness of the overall population.**

***Index Terms*—network slicing, 5G, evolutionary computation, optimization, resource management**

## I. Introduction

Network slicing is a promising new concept for both wireless networks and telecommunications networks. Especially in the new generation telecommunications technology 5G, it is expected to be one of the most prominent technologies combined with Software Defined Networking (SDN) and Network Function Virtualization (NFV). With the help of network slicing, it will be possible to deploy multiple virtual networks over a single physical network infrastructure, each of them configured differently for meeting different Quality of Service (QoS) requirements. For instance same physical network can simultaneously serve an IoT network with many interconnected devices, an emergency network where the bandwidth is maximized, and an ultra reliable network established for remote surgery or robot control where very little delay is a must. Not only network slicing enables establishing differently configurable networks, but also it enables dynamic allocation, deallocation and scaling of these networks. The mobile network operator (MNO) can thus adopt a policy to optimally utilize its capacity. In order to establish this dynamic environment, various autonomous optimization techniques on network slices have been researched. Even though the state-of-the art in this area uses rather deep reinforcement learning methods [2], [3], [4], evolutionary computing can also yield interesting results.

In this research, the researchers tried to simulate a simplified network environment where the network receives requests for generating various slice types and decides whether or not to accept those requests based on its decision function. The genetic algorithm operates on a population of various strategies. According to the network utility obtained at each generation, the genetic algorithm evolves the population of those strategies to find a better one.

Details of the model and how my model reinterpreted that model will be explained in more detail in *Methodology* section. How I conducted the experiments and the results I obtained from them is explained in *Experiment Results* section. Final comments and what can be done to improve the model or to explore new things will be explained in *Conclusion and Further Research* section.

## II. METHODOLOGY

### A. Space of Resource Feasibility and Space of Free Decision

The researcher started modeling the environment by defining the concepts of *Space of Resource Feasibility* and *Space of Free Decision* [1]. There will be *M* different network resource types denoted by $r = [r_1, r_2, ...r_M]^T$ and *N* different slices types denoted by $n \in \{1, 2, ...N\}$. Each slice type is defined through their *cost vector* $c_n = [c_{1,n}, c_{2,n}, ...c_{M,n}]^T$, that is, how much resource it requires from each resource type. Combining all cost vectors, we obtain a cost matrix of dimensions $MxN$, $C = [c_1, c_2, ...c_N]$. The allocated slices at a given time instance denote the *active slice set* $s = [s_1, s_2, ...s_N]^T$ where each $s_n$ denote the number of active slices of type $n$. The multiplication of $Cxs = a$ then denotes the resource assignment in the network at that moment. Since we cannot allocate more resources than we have in our resource pool, the space of resource feasibility is defined through the expression:

$$S = s : r_m - a_m \geq 0, \forall 1 \leq m \leq M \tag{1}$$

that is, the set of all possible active slice sets which does not exceed the existing resource limits.

For the sake of simplicity, the number of different resources and slices are kept minimal in both my work and the research of subject. The resource is a single normalized resource $r = [r_1] = [1]$. The number of different slices $N$ is 2, whose cost vectors are $c_1 = c_2 = [0.3]$. In that case, the space of resource feasibility is as follows: $S = \{[0,0],[0,1],[0,2],[0,3],[1,0],[1,1],[1,2],[2,0],[2,1],[3,0]\}$.

The network cannot accept any slice request if that would couse to exceed the resource limitations. However, it may or may not accept any other request. The domain in which the network may or may not accept slice requests form the *space of free decision*, denoted formally in Equation 2. The policy whether or not to accept slice requests who are in the space of free decision is subject to the optimization process conducted by the genetic algorithm.

$$D = \{(s,n) : \forall s \in S, \forall 1 \leq n \leq N, d(s,n) \in S\} \quad (2)$$

that is, the active slice set formed by accepting a request must also be in the space of resource feasibility. In our case, the space of free decision is as follows:
$D = \{[0,0,1],[0,0,2],[0,1,1],[0,1,2],[0,2,1],[0,2,2],$
$[1,0,1],[1,0,2],[1,1,1],[1,1,2],[1,2,1],[1,2,2],[2,0,1],$
$[2,0,2],[2,1,1],[2,1,2]\}$ where the first two element in each vector denote the current active slice set and the third element in each vector is the type of incoming slice request.

### B. Generating and Handling Slice Generation Requests

Slice generation requests arrive to the network at the beginning of each *operations period*. An operations period is one cycle of loop where the network receives slice generation requests, decides whether or not to allocate each of them according to its decision policy and assign resources to those which it decides to allocate. Slice requests consist of information on the requested slice type, cost of this slice, utility of this slice, which is an integer value, and lifetime of the slice in terms of operations period. Utilities used in the research and in my experiments are 2 and 1 respectively. At each operations period, the number of slice requests made for each type of slice is determined with Poisson distribution. Besides, each slice request comes with the lifetime information, that is, how long (i.e. how many operations period) will this slice exists, if accepted by the network. Lifetime of each slice request is also determined randomly with exponential distribution. After each operations period, the network calculates the overall utility generated by the active slice set. This cycle goes on until one *evolution term* T (T ¿ 1 operations period) is terminated. Then, normalized average of the utilities generated during one evolution term is calculated, which is treated as the fitness value for the genetic algorithm.

### C. Genetic Algorithm

*1) Encoding:* The individuals of the population are binary encoded (True/False) decision vectors of length $|D|$. That means for each scenario in the space of free decision, the encoding tells the network whether (True) or not (False) to accept the slice generation requests. The fitness of the population, as already mentioned before, is the normalized average of utilities generated by applying the selected strategy.

*2) Reproduction:* Roulette wheel selection with respect to the fitness values is performed, then the new population is shuffled for crossover.

*3) Crossover:* Single point crossover with uniform randomly generated crossover point is performed.

*4) Mutation:* Point mutation is performed with a given mutation probability for each bit of gene of the individual.

## III. Experiment Results

TABLE I
Model Parameters for request arrival and lifetime

| Scenario | $\lambda_1$ | $\lambda_2$ | $\mu_1$ | $\mu_2$ |
|---|---|---|---|---|
| 1 | 0.5 | 2 | 2 | 10 |
| 2 | 0.3 | 1 | 2 | 3 |
| 3 | 1 | 0 | 2 | 5 |

In my experiments, I used full crossover rate $P_c = 1.0$ and ten percent mutation rate for each bit $P_m = 0.1$. The population evolved for 20 generations in each experiment. For each of the scenarios given in Table 1, which denote the Poisson and exponential distribution parameters for determining number of slice request and lifetime of them, I conducted 2 sets of experiments for each scenario, one with a population size of 10 and the other with 50. I repeated 500 times each of those 6 experiment sets. I inspected the results with respect to:

- Average normalized population fitness
- Average normalized best fitness observed during the evolution.

As a general inspection, one can observe from Figures 1 thru 6 that for both the average fitness population and best fitness obtained during the evolution, more crowded populations yielded better results for each configuration of distribution parameters. Even though average fitness of larger population may at some points go below of the smaller population in Scenario 2 and 3, in general larger populations perform better compared to their smaller counterparts. Besides, each of them ended up better at the end of 20 generations.

Figure 1 denotes the average population fitness changing with respect to the generation, using parameters from Scenario 1. The experiment with population size 10 starts with an average fitness just below 0.23 and improves itself up until just 0.26. The larger population, on the other hand, start with an average population fitness of close to 0.24 an goes up until $\approx 0.27$. Considering these values as percentages, in either case the algorithm didn't perform well enough to fully utilize the network. Nevertheless, larger population yielded a slightly better result due to the fact that crossover and mutation created more variability in the population.

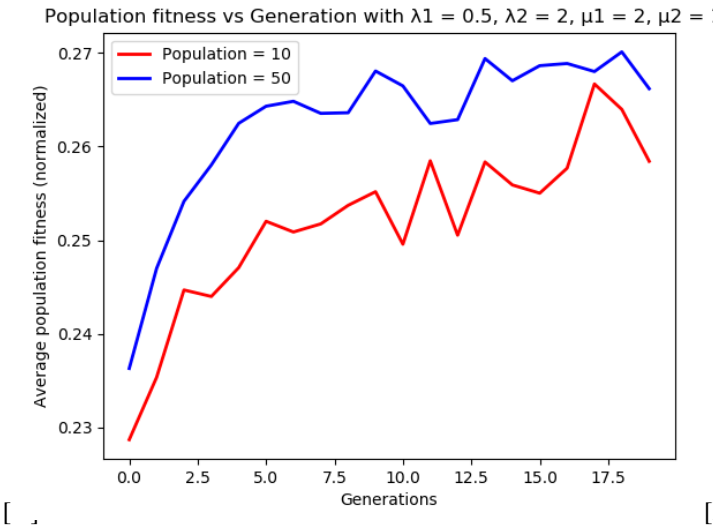



Fig. 1. Population fitness vs Generation (Scenario 1)

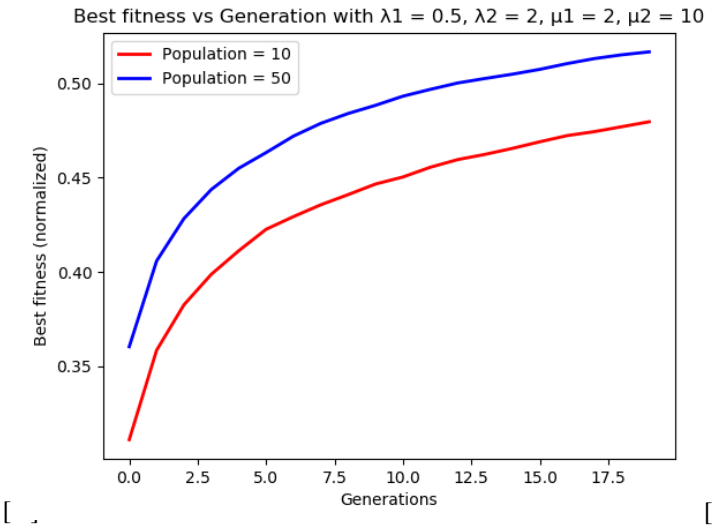



Fig. 2. Best fitness vs Generation (Scenario 1)

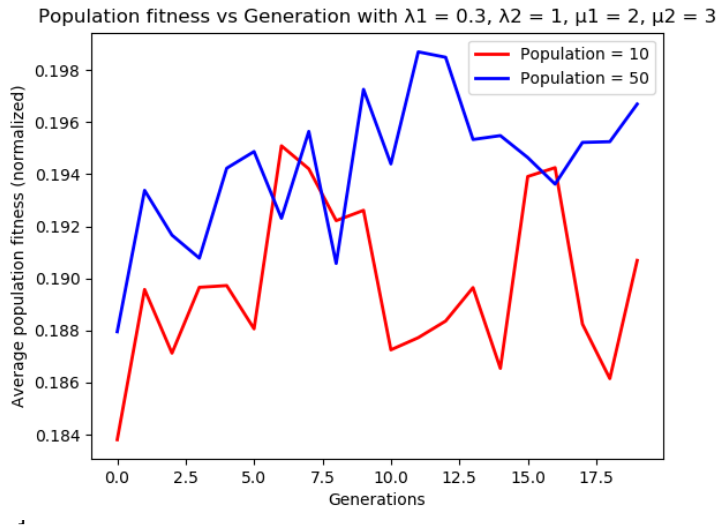



Fig. 3. Population fitness vs Generation (Scenario 2)

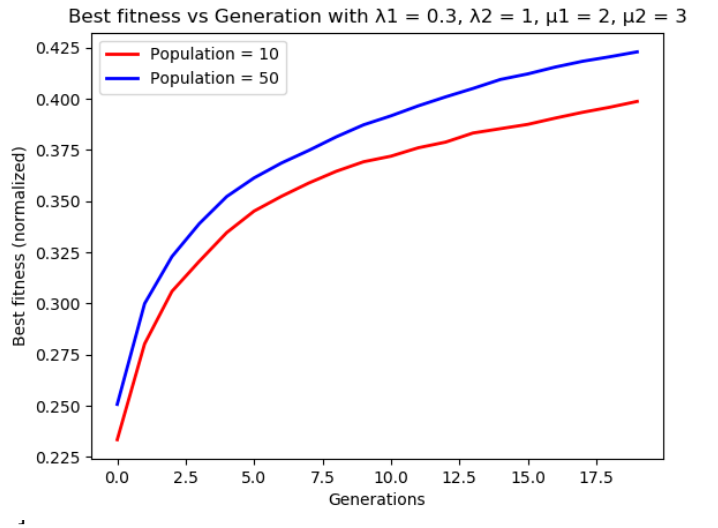



Fig. 4. Best fitness vs Generation (Scenario 2)

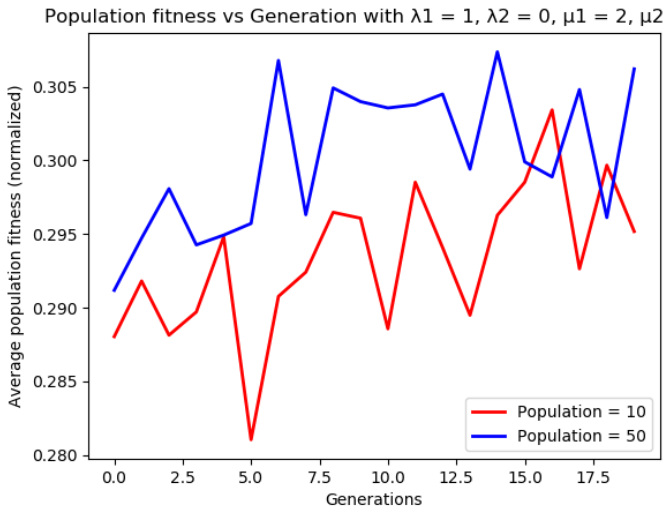
Figure 2 shows the best fitness obtained in the same case. Even though the average fitness values of populations are relatively low, the best individual at the end of 20 generations of evolution has a fitness of almost 0.55 with the larger population and 0.47 with the smaller population. This means that the network can be utilized up to its half capacity with the best strategy.

Figure 3-4 and Figure 5-6 show the same inspections as Figure 1-2 with Scenario 2 and 3 respectively. THhe worst performance is clearly obtained in Scenario 2. In neither size of the populations, the average population fitness could not reach 0.2, as well as the best individual's fitness can only increase up until 0.425 and 0.4 respectively. The best of them, on the other hand, is Scenario 3, where the average fitness of the population is 0.305 and 0.295 for the larger and smaller populations respectively. Best fitness values obtained in Scenario 3 experiments are also better, for large and small populations $\approx 0.6$ and 0.55 respectively.

The difference in performances in different scenarios is mainly caused by the request frequency of different slice types. Since the cost of both types of slices are same, allocating more type 1 slices (utility = 2) than type 2 slices (utility = 1) would yield more network utilization. As we inspect Table 1, we can see that type 1 slice requests arrive the most frequently at Scenario 3 (greatest Poisson constant) and the lest frequently at Scenario 2 (smallest Poisson constant).

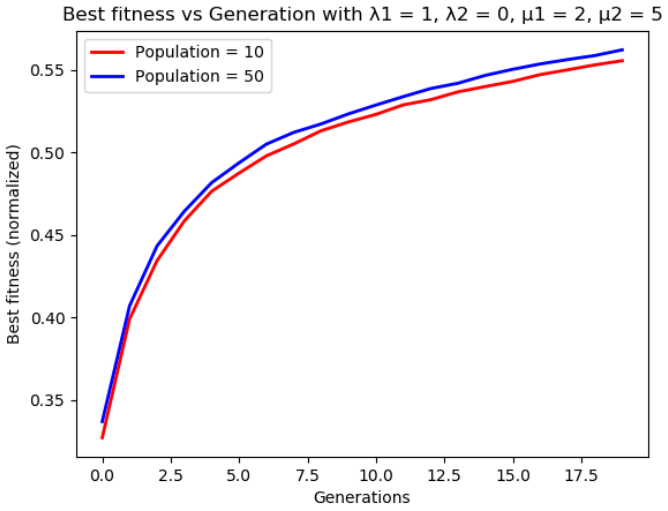
## IV. Conclusion and Further Research

In conclusion, in either configuration of parameters, I was able to observe evolution for the given problem. Both the




[ ]

Fig. 5. Population fitness vs Generation (Scenario 3)




[ ]

Fig. 6. Best fitness vs Generation (Scenario 3)

average fitness of the overall population and the best fitness in the population increased throughout the evolution process. Even though there are some ups and downs in the average fitness, eventually it ended up relatively better than the initial fitness. To improve the results, different crossover and mutation rates can be tried to better approximate the global minimum. Besides, I did not implement a queuing or wait-and-retry mechanism for the rejected slice requests. If there was such a mechanism, the rejected requests could eventually be accepted and allocated, which can also increase the network utility as well as tenant satisfaction in a real world case. Furthermore, experimenting with the same setup using different cost and utilities can also yield interesting results. For even better and more realistic models, resource and slice types can be increased and the experiments can be done on actual network simulators.

## ACKNOWLEDGMENT

This project is conducted for the CENG713 Evolutionary Computation course given by Onur Tolga Şehitoğlu at METU.

## REFERENCES

[1] B. Han, J. Lianghai, H. Schotten, Slice as an Evolutionary Service: Genetic Optimization for Inter-Slice Resource Management in 5G Networks, 6th ed., : IEEE Access, 2018, p. 33137 - 33147.

[2] R. Li, Z. Zhao, Q. Sun, C.-L. I, C. Yang, X. Chen, M. Zhao, and H. Zhang, "Deep Reinforcement Learning for Resource Management in Network Slicing," IEEE Access, vol. 6, pp. 74429–74441, 2018.

[3] S. D. Bast, R. Torrea-Duran, A. Chiumento, S. Pollin, and H. Gacanin, "Deep Reinforcement Learning for Dynamic Network Slicing in IEEE 802.11 Networks," IEEE INFOCOM 2019 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), 2019.

[4] Q. Liu and T. Han, "When Network Slicing meets Deep Reinforcement Learning," Proceedings of the 15th International Conference on emerging Networking EXperiments and Technologies, Sep. 2019.