Optimized Service Chain Placement Using Genetic Algorithm

Mohammad Ali Khoshkholghi, Javid Taheri, Deval bhamare, Andreas Kassler
Department of Mathematics and Computer Science, Karlstad University, Karlstad, Sweden
{ali.khosh-kholghi, javid.taheri, deval.bhamare, Andreas.kassler}@kau.se

***Abstract*— Network Function Virtualization (NFV) is an emerging technology to consolidate network functions onto high volume storages, servers and switches located anywhere in the network. Virtual Network Functions (VNFs) are chained together to provide a specific network service. Therefore, an effective service chain placement strategy is required to optimize the resource allocation and consequently to reduce the operating cost of the substrate network. To this end, we propose four genetic-based algorithms using roulette wheel and tournament selection techniques in order to place service chains considering two different placement strategies. Since mapping of service chains sequentially (One-at-a-time strategy) may lead to suboptimal placement, we also propose Simultaneous strategy that places all service chains at the same time to improve performance. Our goal in this work is to reduce deployment cost of VNFs while satisfying constraints. We consider Geant network as the substrate network along with its characteristics extracted from SndLib. The proposed algorithms are able to place service chains with any type of service graph. The performance benefits of the proposed algorithms are highlighted through extensive simulations.**

***Index Terms*—Network Function Virtualization, Optimization, Genetic Algorithm, Service Chain Placement.**

I. INTRODUCTION

One the most interesting features of 5G networks is network slicing which is enabled by Software Defined Networking (SDN) and NFV [1]. Having this feature, 5G verticals can provide their own services with the specific requirements they want, such as minimum throughput and maximum latency. NFV enables running of such services in the form of VNF chains on containers or virtual machines (VMs). In addition, NFV is able to significantly decrease the capital and operational cost and outperform resource allocation more efficient and flexible. The VNF chains need to be operated in a predefined order and connected via virtual links to provide a service in response to a service demand. Undoubtedly, the placement of VNFs over containers or VMs can significantly influence on an efficient deployment of service chains over the substrate network [2].

As finding an optimum service chain placement is a NP-hard problem [3], exact solutions (e.g. Mixed liner Programming) needs huge amount of time and computational resources, and so they are impractical for the real-world networks. Heuristic algorithms have been proposed to improve the scalability of solution and to handle the large infrastructures. These approaches have to make a trade-off between optimality of the solution, complexity and execution time. For the purpose of cost reduction, we proposed four genetic-based optimization algorithms using roulette wheel and tournament selection techniques in order to place service chains to the available physical hosts in cloud environments.

978-1-5386-9376-6/19/$31.00 ©2019 IEEE

Although much research attention is given to VNF placement, but the previous studies aim to place the service chains one by one. However, this strategy can lead to suboptimal placement rather than considering all the service chains to be placed at the same time. Contrary to the literature, we consider both simultaneous and one by one placement strategies. Our proposed algorithms find an overall optimization by considering all VNFs simultaneously and also provide a one by one solution by considering each service chain individually. In addition, many previous heuristics are limited on the type of resources, most of the time CPU, but we consider the optimization problem as a four-dimension problem corresponding to CPU, memory, disk and network. We compare the results obtained by both placement strategies with each other and also with several greedy algorithms [4].

The rest of the paper is organized as follows. The related work is discussed in Section II. In Section III, we formulate the placement problem and present our optimization model to reduce the deployment cost. In Section IV, we propose the GA-based placement algorithms in details. Section V provides the performance evaluation of the proposed algorithms compared with the benchmark algorithms. Finally, we conclude the paper in section VI.

II. RELATED WORKS

VNF placement has recently obtained much attention in the literature. The related works can be categorized regarding various aspects such as system models, objectives and proposed solutions. A number of studies have formulated the placement problem as an integer programming problem, and solved it using optimization solvers; or applied heuristics or greedy approaches to place VNFs. Authors in [5] proposed a Mixed Integer Linear Programming (MILP) model to optimize utilization and to reduce network operational costs, and solved it using CPLEX. Luizelli et al. [6] proposed an ILP formulation to decrease the number of deployed instances.

In [7] [8], the authors proposed a MILP and Mixed Integer Quadratically Constrained Program (MIQCP) for VNF placement in data centers, respectively. These approaches are effective to find the exact or near-optimal solutions, but only for small size infrastructures and are not suitable for the networks including large set of physical nodes. A polynomial complexity heuristic has been proposed for VNF placement in [9]. The placement problem is modeled as a Multi-Stage directed graph with associated costs and then, VNFs are placed leveraging a Viterbi algorithm [6]. Agrawal et al. [10] introduced a queuing-based system model along with a heuristic to minimize the network latency, but they considered only CPU in their work and neglected other resources.

In respect of objective, some other works address the VNF placement regarding load balancing, service resilience and energy efficiency [11-14]. Authors in [11] presented a solution using game theory to minimize energy consumption and processing delay. In [15], also the authors presented an energy-aware placement for NFV environments using a game-theory approach. However, they proposed the solution for small infrastructures including only a few physical nodes. One of the main objectives to solve the problem of VNF placement is reducing the deployment cost [16] [17] and delays [24]. Deployment cost is defined as the cost of the computing and communication resources which are needed to execute service chains. The main issue of the problem is how to place the VNFs onto physical nodes in a way that QoS is satisfied and the cost of service provider is reduced. Xia et al. [18] study the on-demand VNF placement to minimize the cost by placing the VNFs onto fewer physical nodes. Cohen et al. [19] address the VNF placement for the purpose of minimizing the deployment cost. They showed that an optimal placement can improve network reliability, performance of the network and also operating cost. However, the precedence constraint for VNFs in the service chains is neglected in this work. The authors in [20] address the problem of VNFs mapping and scheduling. They have proposed three greedy algorithms and a Tabu search algorithm to minimize the cost and increase the revenue. As a drawback of this work, authors consider only VNFs as nodes without any link between them and consequently they did not consider service chain placement in their work.

In addition, several studies have focused on the edge server placement in Mobile Edge Computing (MEC) environments. An online application placement for MEC has been proposed by Wang et al. [21]. The authors modeled the placement problem using Markov Decision Processes (MDP) and tried to reduce the state space of the problem. They derived a new MDP model in which only distance between servers and users define the states and then, they proposed a greedy algorithm to place the applications. Qiang Fen et al. [22] introduce an edge server placement approach to minimize the energy efficiency in MEC. They have proposed a particle swarm optimization to find the optimum solution.

All the previous works only considered one by one strategy to place service chains onto physical nodes. On the contrary, we propose both Simultaneous and One-at-a-time placement strategies in order to minimize the deployment cost in large scale environments considering CPU, memory, disk and network bandwidth.

III. Problem Formulation

In this section, the service chain placement is formulated as an optimization problem. The goal of optimization model is to place the service chains onto physical nodes so that the deployment monetary cost is minimized satisfying placement constraints. Table 1 shows the description of parameters and variables. The substrate network is defined as an undirected physical graph, denoted by $G = (V, E)$ where V is a set of physical nodes denoted by $V = \{v_1, v_2, \dots, v_n\}$ and E is a set of physical links between nodes denoted by $E = \{e_1, e_2, \dots, e_k\}$. Each physical node $v \in V$ is characterized by its processing capacity, memory capacity, disk capacity, and resource costs, and is able to host one or several network functions. Binary variable X_{fv} indicates if VNF f is placed on physical node v ($X_{fv} = 1$) or not ($X_{fv} = 0$).

The set of all service chains is indicated as S. Let $s \in S$ denotes a service chain consists of several VNFs with strict precedence connections, and is modeled as a directed graph denoted by $s = (F, L)$ where F is the set of virtual functions and L is the set of virtual links. Each virtual function $f \in F$ is described by its processing demand, memory demand, disk demand, and each virtual link $l \in L$ is characterized by its bandwidth demand. We consider different virtual topologies consist of VNFs and virtual links in response to any type of requested service chains. Each f can be placed to a physical node $v \in V$ which meets required physical resources, and each $l \in L$ can be mapped to a physical path P that has enough bandwidth.

The best method to find the optimal placement of service chains is to investigate all the physical nodes and links available in the substrate network, but as this method requires huge amount of time and computational resources, we proposed a genetic-based heuristic to find a near-optimal placement. Taking all the issues discussed above together, we present an optimization model for placement problem as:

Minimize:

$$\sum_{v\in V}\sum_{s\in S}\sum_{f\in F}(cost_{sf}^{v}) = \sum_{v\in V}\sum_{s\in S}\sum_{f\in F}(cost_{CPU_{sf}^{v}} + cost_{Mem_{sf}^{v}} + cost_{St_{sf}^{v}} + cost_{BW_{sf_if_j}^{v_iv_j}}) \quad (1)$$

Subject to:

$$\sum_{s\in S}\sum_{f\in F}(d_{CPU}^{sf}X_{fv}) \le C_{CPU}^{v}, \forall\, v \in V \quad (2)$$

$$\sum_{s\in S}\sum_{f\in F}(d_{mem}^{sf}X_{fv}) \le C_{mem}^{v}, \forall\, v \in V \quad (3)$$

$$\sum_{s\in S}\sum_{f\in F}(d_{st}^{sf}X_{fv}) \le C_{st}^{v}, \forall\, v \in V \quad (4)$$

$$\sum_{s\in S}\sum_{f\in F}(d_{BW}^{sf_if_j}X_{f_iv_i}X_{f_jv_j}) \le C_{BW}^{v_iv_j}, \forall\, v_iv_j \in V \quad (5)$$

$$\bigcup_{v_i\in V} f_{v_i} = F \quad (6)$$

$$f_{v_i} \bigcap_{v_i\neq v_j} f_{v_j} = \emptyset \quad (7)$$

Constraints (2) - (5) ensure that for all set of VNFs mapped to the same physical node, the demanded CPU, memory, disk and bandwidth will not exceed the total amount of resource capacity. Constraints (6) and (7) demonstrates that each VNF will be assigned to one and only one physical node.

IV. GA-based Service Chain Placement Algorithms

There are a variety of optimization solvers such as CPLEX and Gurobi which can provide the exact solution for the

TABLE I. DESCRIPTIONS OF PARAMETERS AND VARIABLES

Parameter	Description
V, S, F	set of physical nodes, service chains, and VNFs in each service chain
f_{v_i}	set of VNFs mapped to physical node v_i
X_{fv}	binary variable. $X_{fv} = 1$, if VNF f is placed on physical node v, otherwise $X_{fv} = 0$
C^v_{CPU}, C^v_{mem} , C^v_{st}	CPU, memory and disk capacities of physical node v
$C^{v_i v_j}_{BW}$	bandwidth capacity of physical link e between physical node v_i and v_j
d^{sf}_{CPU}, d^{sf}_{mem}, d^{sf}_{st}	CPU, memory and disk demands by VNF f from service chain s
$d^{sf_i f_j}_{BW}$	bandwidth demand by the virtual link between VNF f_i and f_j from service chain s
$cost_{CPU^v_{sf}}$, $cost_{Mem^v_{sf}}$, $cost_{St^v_{sf}}$	costs of CPU, memory and disk used by VNF f from service chain s mapped to physical node v
$cost_{BW^{v_i v_j}_{sf_i f_j}}$	cost of bandwidth used by virtual link l between VNF f_i and VNF f_j mapped to physical path between physical node v_i and physical node v_j

aforementioned placement problem, although along with high time and computational complexity. In the purpose of decreasing complexity, we propose a cost-aware heuristic based on genetic algorithm using two approaches as Roulette wheel and Tournament. As an evolutionary algorithm, GA imitates the natural evolution so that the solution proceeds towards a more optimal solution after each generation. We consider two placement strategies as Simultaneous strategy for the placement of current service chains and One-at-a-time strategy for the placement of new service chains arriving to the system.

A. Encoding Scheme

In the proposed GA algorithms, each potential placement solution is defined as a chromosome (individual). Every individual consists of several genes and each gene corresponds to a physical node where a VNF is mapped to. The gene's value is represented by an integer [1, *N*] interval, where *N* is the number of physical nodes. Individuals are created randomly and must satisfy the constraints of the optimization model. Fig. 1 shows an example of the encoding scheme for both types of placement strategies including two sample service chains. In Simultaneous strategy, the number of genes in each individual is equal to the total number of VNFs in all service chains as we aim to find the solution for all service chains at the same time. However, in One-at-a-time strategy by which the service chains are placed one by one, each individual is composed of *m* genes where *m* is equal to the number of VNFs in the given service chain. The reason is that Simultaneous strategy is proposed in a case that all service chains are available and an overall solution is desired. However, the second strategy is used to obtain an optimum placement for each service chain arrived to the system in different time frames. The service chains are composed in different topologies with strict precedence between VNFs which are connected by directed virtual links. A fixed number of chromosomes compose the population as an initial set of solutions.

B. GA Operators

Given the initial population, GA intends to generate new population consists of individuals with lower cost. Fitness value is defined as follows:

$$\sum_{i=1}^{g} cost(cpu_i) + cost(mem_i) + cost(st_i) + \sum_{j=1}^{vl}\sum_{k=1}^{p} cost(link_{jk}) \quad (8)$$

Where g indicates the number of genes, vl indicates number of virtual links and p depicts the physical path for each virtual link mapped on the substrate network.

For this reason, the proposed algorithms aim to merge the individuals to create offspring using crossover operator and then mutate them using mutation operator. Each offspring inherits its genes from two parents selected for crossover. We use two individual selection techniques in our work and based on that we propose two GA algorithms for the placement problem.

1. *Tournament-based Genetic Algorithm (TGA):* TGA aims to select the parents for crossover using a tournament technique. First, all the individuals in the population are sorted in an increasing order regarding their fitness value. Next, a number of individuals from the top of the list with lowest fitness values are selected as elites based on the elitism rate, and they will be directly added to the new population. Then, for generating the rest of individuals in the new population, each time five individuals are selected randomly from the current population and the fittest individual (lowest cost) are selected as the first parent. This procedure is repeated for the second parent as well. Given the following condition, the offspring are created from the parents:

$$\begin{cases} G_i^{off} \leftarrow G_i^{p_1}, & if\ R \geq crossover\ rate \\ G_i^{off} \leftarrow G_i^{p_2}, & else \end{cases} \quad (9)$$

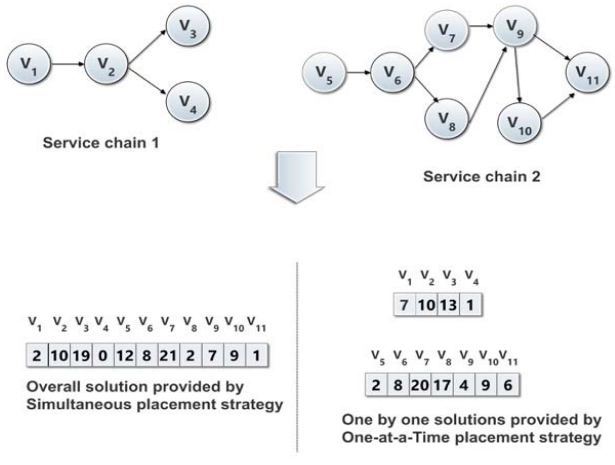


Fig. 1. Encoding scheme provided by both placement strategies for two sample service chains

Where R is a random real number in the range 0 to 1, G_i^{off} indicates the i^{th} gene of the offspring, $G_i^{p_1}$ and $G_i^{p_2}$ indicate i^{th} gene of the first and second parent, respectively. The proposed GA is a self-healing algorithm. After generating each offspring, all the constraints will be checked and must be satisfied, otherwise for the same parents, crossover operator has to generate another offspring which satisfies the constraints. To avoid getting stuck in an infinite loop and consequently increasing computational time, we set a condition using a fixed integer ($\propto$) as follows:

$$\begin{cases} P_i^{new} \leftarrow I_i^{off}, & if\ I_i^{off}\ satisfies\ the\ constraints \\ repeat\ crossover, & if\ I_i^{off}\ does\ not\ satisfy\ the\ constraints,\ and\ t < \propto \\ P_i^{new} \leftarrow I_i^{p1}, & if\ I_i^{off}\ does\ not\ satisfy\ the\ constraints,\ and\ t \geq \propto \end{cases} \quad (10)$$

Where P_i^{new} is the i^{th} place in the new population, I_i^{off} indicates the i^{th} offspring and t is an iterator. Crossover operation will be repeated until all offspring individuals have been generated and added to the new population. Having the new population, mutation as the second GA operator aims to mutate each individual from top of the list using the below condition:

$$\begin{cases} G_i^{new} \leftarrow G_i^m & if\ R \leq mutation\ rate \\ G_i^{new} \leftarrow G_i & else \end{cases} \quad (11)$$

Where R is a random real number in the range [0, 1], G_i^{new} is the i^{th} gene in the individual after mutation, G_i^m indicates the mutated gene and G_i indicates the same gene as before mutation. The same as the crossover operator, in order to achieve self-healing, each new individual is checked with the constraints. If the individual satisfies the constraints, it will be added to the new population, and if not so, mutation operator generates another individual regarding the following condition:

$$\begin{cases} P_i^{new} \leftarrow I_i^m, & if\ I_i^m\ satisfies\ the\ constraints \\ repeat\ mutation, & if\ I_i^m\ does\ not\ satisfy\ the\ constraints,\ and\ t < \propto \\ P_i^{new} \leftarrow I_i, & if\ I_i^m\ does\ not\ satisfy\ the\ constraints,\ and\ t \geq \propto \end{cases} \quad (12)$$

Where P_i^{new} is the i^{th} place in the new population, I_i^m indicates the i^{th} individual after mutation, I_i indicates i^{th} individual before mutation, and t is an iterator. Given the new population generated after mutation, the fittest individual will be selected as the solution for that specific generation. The GA algorithm continues to generate new populations until converging to a final solution, which is defined as obtaining the same solution for the last k generations.

2. *Roulette-wheel-based Genetic Algorithm (RGA):* RGA leverages the same procedure as TGA except the individual selection technique for crossover operator. RGA selects the parents using the roulette wheel technique. All the individuals are sorted in an increasing order. Unlike the original roulette wheel method that associates more probability of selection with bigger numbers, as in our work the fitness values aim to be lower amounts, we modify the probability assignment. First, fitness value of each individual is mapped to a real number in the range [1, 10], as follows:

$$fit_i = (1 - 10)\left(\frac{fit_i - fit_0}{fit_n - fit_0}\right) + 10 \quad (13)$$

Where n is the number of individuals in the population and fit_i indicates the fitness value of individual i. Then, a probability of selection is assigned to each individual as follows:

$$pr_i = \frac{fit_i}{\sum_{j=1}^{n} fit_j} \quad (14)$$

RGA selects individual i as the first parent in a way that $pr_i \geq R > pr_{i+1}$, where R is a random real number between 0 to 10. This procedure is repeated for the second parent as well.

Algorithm 1 shows the GA-based service chain placement algorithm leveraging both roulette wheel and tournament selection techniques, which can be used for both placement strategies. In the case of One-at-a-time strategy, this algorithm will be executed for each single service chain, but for Simultaneous strategy the final solution will be provided by one-time execution of the algorithm for all the available service chains.

Algorithm 1: GA-based service chain placement algorithm

Input:
substrate Network's XML file; service chains' XML file; population size; crossover rate; mutation rate; elitism rate
Output:
service chains placement

read network graph and generate matrix G (V, E)
read service graph and generate matrix s (F, L)
// create population randomly
foreach individual p in population P
  foreach VNF f in p
  {
    pick a physical node $v \in V$ randomly
    if the constraints (2) - (7) are satisfied
    allocate f to v
  }
foreach $p \in$ P
  calculate the fitness value using equation 8
sort P in an increasing order
add elites *el* to the new population $\acute{P}$
// crossover operator
foreach ($p \in P$)
{
  //select parents p_1 and p_2 using tournament method
  select five p randomly
  select the fittest p as p_1
  select five p randomly
  select the fittest p as p_2
  //or select parents p_1 and p_2 using roulette wheel method
  foreach $p \in P$
    map the fitness value to a real number using (13)
  foreach $p \in P$
    allocate a probability selection value to each p using (14)
  generate a random number R and select p_1 so that $pr_i \geq R > pr_{i+1}$
  generate R and select p_2 so that $pr_i \geq R > pr_{i+1}$
  generate offspring I using (9)
  check the condition (10) and fill the rest of $\acute{P}$
}
// mutation operation
foreach ($p \in P$)
{
  generate R and check the condition (11)
  check the condition (12) and update $\acute{P}$
}
select the fittest p as the optimal solution for the current generation
continue creating generations
if the same solution is generated for the last k iterations
  select it as the final placement

V. PERFORMANCE EVALUATION

In this section, we evaluate and compare the performance of the proposed algorithms with each other and also with three benchmark algorithms. Given two types of placement strategies as Simultaneous and One-at-a-time, and two proposed genetic algorithms for each placement strategy, we evaluate the performance of four proposed algorithms as: Simultaneous Tournament-based Genetic Algorithm (STGA), Simultaneous Roulette-wheel-based Genetic Algorithm (SRGA), One-at-a-time Tournament-based Genetic Algorithm (OTGA), One-at-a-time Roulette-wheel-based Genetic Algorithm (ORGA). The proposed algorithms are compared with three greedy placement algorithms as First Fit, Best Fit and Worst Fit [3].

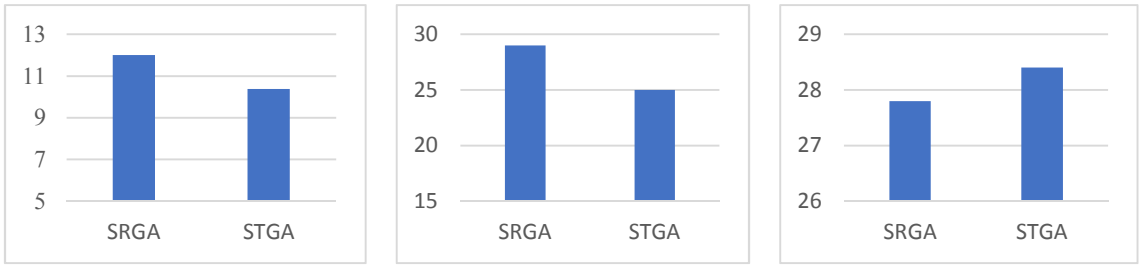
The proposed genetic algorithms and greedy algorithms have been implemented in Java and run on an Intel core i5 1.8 GHz machine equipped with 8 GB of RAM. In the experiments, the population size is set to 100 individuals. Crossover rate, mutation rate, elitism rate and $\propto$ are set to 0.5, 0.015, 0.1 and 5, respectively. The number of service chains is 19 and they totally include 110 VNFs. Each service chain comprises a random number of VNFs in a [4, 10] interval.

We considered Geant network as the substrate network along with its characteristics extracted from SndLib [23] including 22 nodes and 36 links. Topology and requirements of service chains, and topology and capacities of substrate network are imported as two XML files to the system. The CPU capacity of physical nodes were randomly generated with a value in [3720, 5320] Mips and each node hosts maximum 5 VNFs. The memory, storage and bandwidth capacities were generated in [4, 8] Gbps, [500, 1000] Gbps and [500-1000] Mbps, respectively. The CPU, memory and storage requirements of VNFs were randomly picked up from [100, 2700] Mips, [100, 3000] Mbps, [50, 100] Gbps, respectively; and bandwidth was set to 40 Mbps.

Fig. 2 depicts the evaluation results of two GA algorithms proposed for Simultaneous strategy regarding time in minutes, number of generation and cost in dollars. We can see that although STGA is faster and can converge to the final solution by a smaller number of generations, but SRGA obtains a better result based on cost which is the objective of this study. It because roulette wheel selection technique is more successful to generate individuals in a way that minimizes total placement cost for all VNFs; however, the difference between both cost values is not too big.

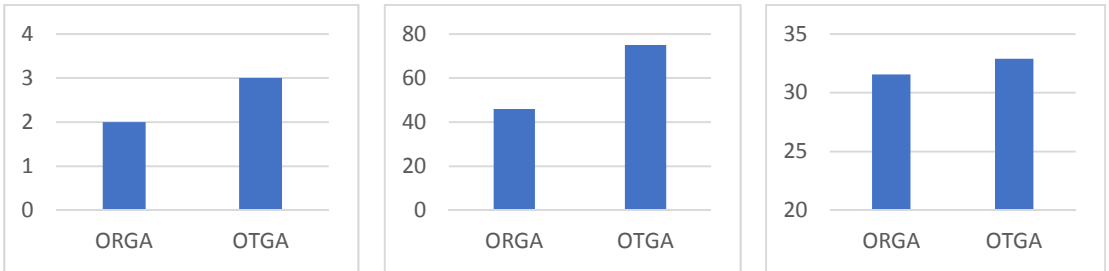
The evaluation results of two GA algorithms proposed for One-at-a-time placement strategy is shown in Fig. 3 based on time in minutes, number of generation and cost in dollars. Unlike the results achieved by Simultaneous strategy, ORGA obtained better results not only based on cost but also time and number of generations. Given the results for both placement strategies, we can conclude that the GA algorithm using roulette wheel selection technique minimizes the cost in both strategies (although the cost values are close together), also it is faster and converges the final solution with smaller number of generations in One-at-a-time strategy; while the algorithm using tournament technique is faster and provides the final solution with smaller number of generations in Simultaneous strategy.

Fig. 4 illustrates the evaluation results of four GA algorithms against benchmark algorithms regarding time and cost. As shown in Fig. 4a, all proposed algorithms significantly outperform the benchmark algorithms based on cost. Considering four GA algorithms, SRGA and STGA reduce the amount of cost rather than ORGA and OTGA. It shows that Simultaneous placement is more successful to decrease the cost in compassion with One-at-a-time placement. The reason is that Simultaneous GA algorithms aim to find a solution for all VNFs at the same time and for that, they regenerate solutions for all service chains until reaching an optimal placement, and then VNFs will be placed. However, One-at-a-time GA algorithms consider a service chain, find an optimal solution only for that specific service chain, place it and then start the second service chain



(a) Time (b) Number of generation (c) Cost

Fig.2 Comparing SRGA and STGA based on time, number of generation and cost



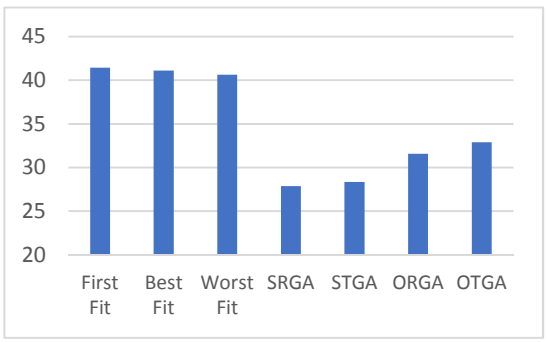
(a) Time (b) Number of generation (c) Cost

Fig. 3. Comparing ORGA and OTGA based on time, number of generation and cost

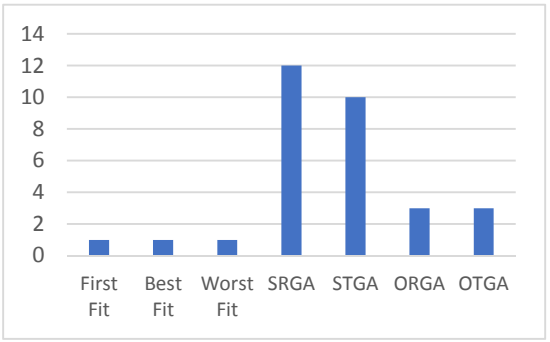
placement. So, we expect that the general placement will have some deviation in comparison with Simultaneous strategy which is proved by the results.

As discussed earlier, roulette wheel algorithm obtains better results over cost rather than tournament; so SRGA minimizes cost compared to STGA and ORGA reduces cost more than OTGA. SRGA as the best algorithm outperform first fit algorithm by 33%, best fit algorithm by 32%, worst fit by 31%, STGA by 2%, ORGA by 12% and OTGA by 15%. As shown in Fig. 4b, the greedy algorithms are very fast and provide the solution less than a minute since bin packing algorithms are not evolutionary, unlike GA algorithm. SRGA and STGA spend more time to converge to the final solution, but ORGA and OTGA are quite fast in providing the optimal solution. From the results we can conclude that ORGA and OTGA provide an effective trade-off between reducing cost and the time spent for converging to the final solution, while SRGA and STGA provide the least amount of cost with more processing time. Fig. 4c indicates the cost obtained by all algorithms in the first two minutes of the evaluation. As it can be observed, all four GA algorithms improve the cost rather than greedy algorithms. However, unlike Fig.4a, ORGA and OTGA achieved the best results compared to the Simultaneous algorithms. The reason is these two algorithms are faster and can converge to final solution earlier than SRGA and STGA. As shown in Fig. 4d., STGA and SRGA algorithms (as the best algorithms in terms of cost) are compared per different time slots from 2 to 12 minutes. The proposed algorithms generate a desirable downward sloping curve. Using STGA, cost rapidly decreases until 4th minutes, and further decreases with a gentle slope until 10 minutes. At that point, the value does not change anymore and becomes the final solution. Using SRGA, also cost decreases rapidly until 4th minutes, and then further decreases slowly until 12th minutes.

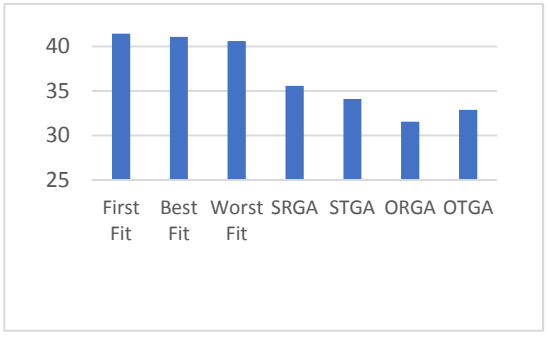
Fig. 5 compares the proposed GA algorithms against greedy algorithms in terms of resource utilization and number of used physical nodes. Average resource utilization for all physical nodes is shown in Fig. 5a. SRGA and STGA utilize resources slightly less than benchmark algorithms, but ORGA and OTGA increase the average value rather than other algorithms. SRGA obtained the least percentage of resource utilization. Fig. 5b shows the average resource utilization for used physical nodes. Unlike the prior metric, we observe that all GA algorithms utilize smaller percentage of resources than First Fit and Best Fit but bigger than Worst Fit. The reason to obtain least percentage by Worst Fit is that all algorithms except Worst Fit includes at least one physical node with 100% of resource utilization (as shown in Fig. 5c) and also, they have at least one unused physical node (as shown in Fig. 5d), but Worst Fit distributes service chains into all 22 available physical nodes. Fig. 5d indicates the host with minimum resource utilization which is zero for all algorithms except Worst Fit. Finally, Fig. 5e shows the number of used physical nodes for the algorithms. Both SRGA and STGA have used 19 nodes out of 22, OTGA uses 20 and ORGA uses 21 physical nodes. Best Fit with only 15 used nodes aims to place all VNFs into smaller number of physical nodes.



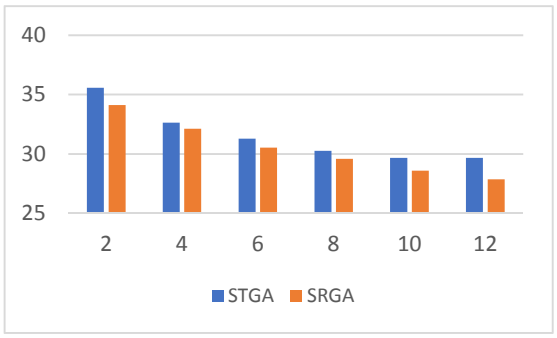
(a) Cost



(b) Time

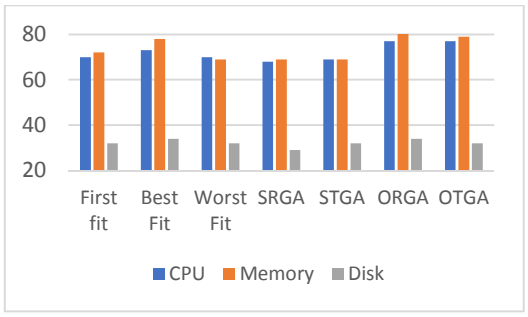


(c) Cost in 2 minutes

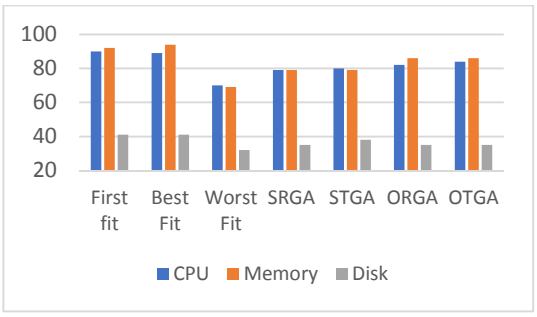


(d) Cost per different time slots

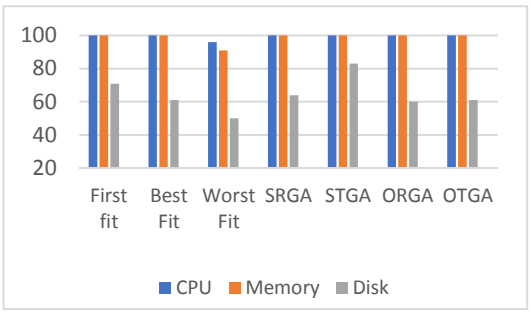
Fig. 4 Comparing the proposed GA algorithms and benchmark algorithms based on time and cost



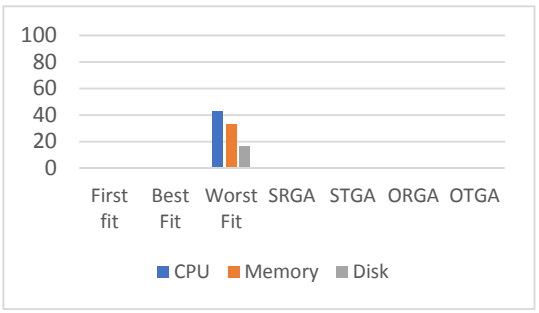
(a) Average resource utilization for all physical nodes



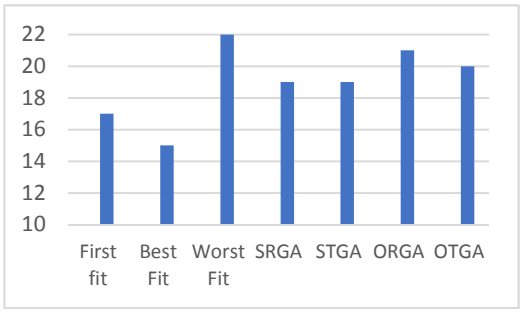
(b) Average reosurce utilization for used physical nodes



(c) Maximum resource utilization



(d) Minimum resource utilization



(e) Number of used physical nodes

Fig. 5. Comparing the proposed GA algorithms against benchmark algorithms in terms of resource utilization and number of used physical nodes

VI. Conclusion

In this paper, we studied the important problem of service chain placement in order to minimize the VNFs deployment cost for multiple resources as CPU, memory, storage and network link. We proposed two GA algorithms using Roulette wheel selection and tournament selection techniques. Then, combining with two types of placement strategies as Simultaneous and One-at-a-time, we proposed four GA-based placement algorithms to find optimum solution. We evaluated the proposed algorithms by conducting extensive simulations and compared them against each other and also with three greedy algorithms as First Fit, Best Fit and Worst Fit. The results showed that proposed GA algorithms significantly minimize the cost when compared to the greedy algorithms. In addition, we observed that Simultaneous algorithms are more successful in reducing cost compared with One-at-a-time algorithms. However, One-at-a-time algorithms obtain an effective tradeoff between the processing time and cost reduction. Our future work includes considering latency constraint as another important factor in order for placing service chains.

VII. Acknowledgment

The authors received partial funding from the Knowledge Foundation of Sweden through the Profile HITS (Grant Number 20140037).

References

[1] Ahmed, E., & Rehmani, M. H. (2017). Mobile edge computing: opportunities, solutions, and challenges.

[2] Y. Li , M. Chen , Software-defined network function virtualization: a survey, in: Access, 3, IEEE, 2015, pp. 2542–2553.

[3] Khebbache, S., Hadji, M., & Zeghlache, D. (2017). Scalable and cost-efficient algorithms for VNF chaining and placement problem. In 2017 20th Conference on Innovations in Clouds, Internet and Networks. (ICIN), pp. 92-99.

[4] EG Cofman Jr, E. C., Garey, M. R., & Johnson, D. S. (1996). Approximation algorithms for bin packing: A survey. Approximation Algorithms for NP-Hard Problems, 46-93.

[5] M. F. Bari, S. R. Chowdhury, R. Ahmed, and R. Boutaba, “On orchestrating virtual network functions,” in Proc. 11th Int. Conf. Netw. Service Manage. (CNSM), 2015, pp. 50–56.

[6] M. C. Luizelli, L. R. Bays, L. S. Buriol, M. P. Barcellos, and L. P. Gaspary, “Piecing together the NFV provisioning puzzle: Efficient

placement and chaining of virtual network functions," in Proc. IFIP/IEEE Int. Symp. Integr. Netw. Manage. (IM), May 2015, pp. 98–106.

[7] E. Amaldi, S. Coniglio, A. M. Koster, and M. Tieves, "On the computational complexity of the virtual network embedding problem," Electron. Notes Discrete Math., vol. 52, pp. 213–220, 2016.

[8] T.-W. Kuo, B.-H. Liou, K. C.-J. Lin, and M.-J. Tsai, "Deploying chains of virtual network functions: On the relation between link and server usage," in Proc. IEEE INFOCOM, 35th Annu. IEEE Int. Conf. Comput. Commun., Apr. 2016, pp. 1–9.

[9] R. Mijumbi, J. Serrat, J.-L. Gorricho, N. Bouten, F. De Turck, and S. Davy, "Design and evaluation of algorithms for mapping and scheduling of virtual network functions," in Proc. 1st IEEE Conf. Netw. Softwarization (NetSoft), Apr. 2015, pp. 1–9.

[10] Agarwal, S., Malandrino, F., Chiasserini, C. F., & De, S. (2018, April). Joint VNF Placement and CPU Allocation in 5G. In *IEEE INFOCOM 2018-IEEE Conference on Computer Communications* (pp. 1943-1951).

[11] R. Bruschi, A. Carrega, and F. Davoli, "A game for energy-aware allocation of virtualized network functions," J. Elect. Comput. Eng., vol. 2016, no. 2, Jan. 2016, Art. no. 4067186.

[12] T. Taleb, A. Ksentini, and B. Sericola, "On service resilience in cloudnative 5G mobile systems," IEEE J. Sel. Areas Commun., vol. 34, no. 3, pp. 483–496, Mar. 2016.

[13] M. T. Beck, J. F. Botero, and K. Samelin, "Resilient allocation of service function chains," in Proc. IEEE Conf. Netw. Function Virtualization Softw. Defined Netw. (NFV-SDN), Nov. 2016, pp. 128–133.

[14] Carpio, F., Dhahri, S., & Jukan, A. (2017, May). VNF placement with replication for Loac balancing in NFV networks. In *Communications (ICC), 2017 IEEE International Conference on* (pp. 1-6).

[15] S. Mehraghdam, M. Keller, and H. Karl, "Specifying and placing chains of virtual network functions," in Proc. IEEE 3rd Int. Conf. Cloud Netw. (CloudNet), Oct. 2014, pp. 7–13.

[16] B. Addis, D. Belabed, M. Bouet, and S. Secci, "Virtual Network Functions Placement and Routing Optimization," IEEE CloudNet, pp.171-177, Oct. 2015.

[17] W. Rankothge, F. Le, A. Russo, and J. Lobo, "Optimizing Resource Allocation for Virtualized Network Functions in a Cloud Center Using Genetic Algorithms," IEEE Trans. on Network and Service Management (TNSM), vol. 14, no. 2, pp. 343-356, Jun. 2017.

[18] Xia, M., Shirazipour, M., Zhang, Y., Green, H., & Takacs, A. (2015). Network function placement for NFV chaining in packet/optical datacenters. *Journal of Lightwave Technology*, *33*(8), 1565-1570.

[19] Cohen, R., Lewin-Eytan, L., Naor, J. S., & Raz, D. (2015, April). Near optimal placement of virtual network functions. In *Computer Communications (INFOCOM), 2015 IEEE Conference on* (pp. 1346-1354).

[20] R. Mijumbi, J. Serrat, J. Gorricho, N. Bouten, F. D. Turck, and S. Davy, "Design and evaluation of algorithms for mapping and scheduling of virtual network functions," in Proceedings of the 1st IEEE Conference on Network Softwarization, NetSoft 2015, London, United Kingdom, April 13-17, 2015, 2015, pp. 1–9.

[21] S. Wang, R. Urgaonkar, M. Zafer, T. He, K. Chan, and K. K. Leung. Dynamic service migration in mobile edge-clouds. In IFIP Networking Conference, pages 1–9, 2015.

[22] Q. Fan and N. Ansari, "Cost Aware cloudlet Placement for big data processing at the edge," in Proceedings of the IEEE International Conference on Communications (ICC 2017), 2017, pp. 1–6.

[23] "SNDlib." [Online]. Available: http://sndlib.zib.de.

[24] D. Bhamare, R. Jain, M. Samaka, A. Erbad, L. Gupta, H. A. Chan, "Optimal Virtual Network Function Placement and Resource Allocation in Multi-Cloud Service Function Chaining Architecture," Computer Communications, 2017, pp. 1–16.