Energy Efficient NFV Resource Allocation in Edge Computing Environment

Xiao Chen
Department of Computer Science, Texas State University, San Marcos, TX 78666
Email: xc10@txstate.edu

***Abstract*—With the development of IoT and 5G communication, a recent trend is to shift the Network Function Virtualization (NFV) from the centralized cloud computing to edge computing. In this paper, we study the energy efficient NFV-Resource Allocation problem in the edge computing environment. We define two problems. In the first problem, we assume that the physical resources (PRs) on the edge do not have energy constraint. Our objective is to find an optimal deployment so that the maximum energy consumption on the PRs is minimized. In the second problem, we assume that the PRs have energy constraint and aim to find an optimal deployment to reduce the number of PRs. We prove both problems NP-complete and propose heuristic algorithms to solve them. We also design baseline algorithms using genetic programming to find approximate optimal solutions to these problems. We conduct simulations to evaluate the performance of our proposed algorithms. Simulation results show that our algorithms produce results very close to those of the baseline algorithms in a much shorter time.**

***Index Terms*—edge computing, NFV, physical resource, resource allocation, VNFR**

I. Introduction

With the development of IoT and 5G communication, a recent trend is to shift the *Network Function Virtualization* (NFV) [6] from the centralized cloud computing to *edge computing* [9]. Edge computing pushes mobile computing, network control and storage to distributed devices at the network edge to provide server resources, data analysis and artificial intelligence closer to data collection sources. It offers faster data processing, generates less network traffic, and costs less than cloud computing. Different from the cloud environment, the *physical resources* (PRs) at the edge are not abundant.

The main challenge for the deployment of NFV at the edge is the NFV-Resource Allocation ($NFV - RA$) problem, i.e., how to allocate PRs to satisfy a set of *virtual network function requests (VNFRs)* [7]. The current NFV-RA solutions seek to minimize the total placement cost [4], [5], achieve load balancing [2], [3], [8], and support QoS [10]. NFV is still in its early stage. There are still important aspects that should be investigated to efficiently manage and allocate resources in NFV-based edge computing. One aspect that was not discussed much is the provisioning of energy-aware strategies to find efficient NFV-RA solutions within reasonable time [7].

In this paper, we will study energy efficient strategies for the NFV-RA problem at edge. Here we assume that VNFRs are independent and their energy consumptions can be measured and estimated before the placement. We define two problems in NFV-RA. In the first problem, we assume that the PRs at the edge do not have energy allowance constraint, i.e., the PRs can accept any number of VNFRs running on them without worrying about the energy consumption of these requests. Also we want the energy consumption on these PRs to be balanced. So our objective is to find an optimal deployment so that the maximum energy consumption on the PRs is minimized. In the second problem, we assume that the PRs have energy constraint $maxE$. That is, the total amount of energy consumed by the VNFRs deployed on a PR cannot exceed $maxE$. In this problem, we aim to find an optimal deployment to minimize the number of PRs. We show that both problems are NP-complete and propose greedy heuristic algorithms to solve them. Furthermore, we design baseline algorithms using genetic programming to find approximate optimal solutions to these problems. We conduct simulations to evaluate the performance of our proposed algorithms by comparing them with the baseline algorithms. Simulations show that the results generated by our heuristic algorithms are very close to those of the baseline algorithms but are produced in much less time.

The rest of the paper is organized as follows: Section II cites the related work; Sections III and IV describe the problems and solutions; Section V presents the simulations; and conclusion is in Section VI.

II. Related Work

The current work in NFV mainly focuses on how to deploy network functions on commodity servers and minimize the total placement cost [4], [5], achieve load balancing [2], [3], [8], and support QoS [10]. Abu-Lebdeh et al. in [4] studied the VNF placement problem and aimed at minimizing the operational cost without violating the performance requirements. Cohen et al. in [5] addressed the actual placement of the virtual functions within the network with the goal to minimize the total system cost which comprises the sum of the setup costs of the functions and the sum of the distances between the clients and the nodes from which they get service. The authors provided an integer linear programming formulation and proposed a tabu search algorithm to solve large instances of the problem. Carpio et al. in [3] investigated the problem of VNF placement with replications, especially the potential of VNF replications to help load balance the network. Ma et al. in [2], [8] explored the optimal placement of NFV middleboxes by considering different middlebox traffic changing effects and dependency relations and showed how to achieve load balancing using a Software Defined Networking approach. Vizarreta et al. in [10] focused on the placement of virtualized network functions to support service differentiation between users while minimizing the associated service deployment cost for the operator. The energy efficient strategies for NFV resource allocation have not been mentioned much and will be our task here.

III. No Constraint on PR Energy Consumption

A. Problem Formulation

In this problem, we assume that there are m independent VNFRs $\{r_1, r_2, \cdots, r_m\}$ to be allocated to n PRs $\{p_1, p_2, \cdots, p_n\}$. In this paper, any VNFR can be assigned to any PR and a VNFR cannot be stopped before it is done. Each VNFR cannot be divided into smaller VNFRs. The energy consumption of each VNFR is given by e_i $(1 \leq i \leq m)$. We introduce a variable $x_i \in \{0, 1\}$ to indicate whether a VNFR is assigned to a PR p_j or not. After all the VNFRs are allocated, we denote the total energy consumption of the VNFRs allocated to p_j as E_{p_j}, which is equal to $\sum_i e_i x_i$. In this problem, we assume there is no energy constraint on the PRs, i.e., any number of VNFRs can be allocated to a PR without being worried about the energy limit of the PR. We set minimizing E_{p_j} as our optimization goal. Furthermore, in order to balance the energy consumption of the PRs, we aim to achieve $\min(\max(E_{p_j}))$. We call this problem the *No Energy Constraint* (NEC) problem. We have proved that the problem is NP-complete. The proof is skipped due to space limitation.

B. Our Solution

Since the NEC problem is NP-complete, we put forward a greedy heuristic algorithm called *Request Allocation with No Energy Constraint* (RANEC) in Fig. 1 to solve it.

In RANEC, we first sort VNFRs by their energy consumptions in a non-decreasing order and then assign the first n VNFRs to the n PRs one by one. Then for each VNFR from $n + 1$ to m, we assign it to the PR with the minimal energy consumption so far. After calculation, the total time complexity of the algorithm is no more than $O(m \log_2 n)$.

Let us look at an example. Suppose there are 3 PRs and 9 VNFRs. The energy consumptions of these VNFRs are $80, 42, 24, 3, 64, 97, 51, 70,$ and 14. The result using RANEC is in Fig. 2. The algorithm tries to balance the energy consumption on the PRs and the minimum of the maximum energy consumption is 155 which occurs on p_2.

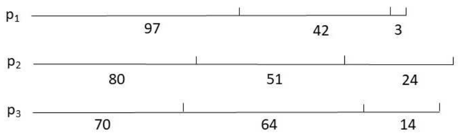


Fig. 2. An example using algorithm RANEC

In real situations, the NEC problem has to be tackled as an online problem. That is, VNFRs arrive at the system dynamically and stay in the network for an arbitrary amount of time. Algorithm RANEC is an online algorithm that can allocate the VNFRs to PRs one after another.

Algorithm RANEC: Request Allocation with No Energy Constraint

```
Require: Input: Energy consumption of $m$ VNFRs $e_1, e_2, \cdots, e_m$; The number of PRs $n$
    Output: VNFR allocation to each PR $S_{p_1}, S_{p_2}, \cdots, S_{p_n}$ and $\min(\max(E_{p_j}))$
1: sort VNFRs in a non-increasing order based on their energy consumption $e_i$
2: Initialize each $S_{p_i} = \emptyset$
3: /* Allocate a VNFR to a PR */
4: for $i = 1$ to $n$ do
5:     $S_{p_i} = S_{p_i} \cup \{r_i\}$;
6:     $E_{p_i} = e_i$;
7:     record $\min(\max(E_{p_i}))$
8: end for
9: for $k = n + 1$ to $m$ do
10:    /* Find the PR that has consumed the least energy so far */
11:    $E_{p_j} = \min_{1 \leq i \leq n}\{E_{p_i}\}$;
12:    /* Allocate the $k$-th VNFR to $p_j$ and update $p_j$'s energy consumption */
13:    $S_{p_j} = S_{p_j} \cup \{r_k\}$;
14:    $E_{p_j} = E_{p_j} + e_k$;
15:    record $\min(\max(E_{p_j}))$
16: end for
```

Fig. 1. Request Allocation with No Energy Constraint

Baseline: Genetic Algorithm 1 (GA1)

```
Require: Input: Energy consumption of $m$ VNFRs $e_1, e_2, \cdots, e_m$; The number of PRs $n$
    Output: VNFR allocation to each PR $S_{p_1}, S_{p_2}, \cdots, S_{p_n}$ and $\min(\max(E_{p_j}))$
1: Generate a random chromosome population set $POP$ of size $|POP|$
2: for $i = 1$ to $LOOPS$ do
3:     Generate $CO\_rate(\%) * |POP|$ of crossovers by picking any two chromosomes in $POP$ each time
4:     Generate $MT\_rate(\%) * |POP|$ of mutations by using any one chromosome in $POP$ each time
5:     Generate $RD\_rate(\%) * |POP|$ new random chromosomes, $RD\_rate = 1 - CO\_rate - MT\_rate$
6:     Select the best $|POP|$ chromosomes based on the fitness function from $2 * |POP|$ chromosomes
7: end for
8: Output the best chromosome as the allocation and $\min(\max(E_{p_j}))$
```

Fig. 3. Genetic algorithm 1 to find the optimal solution to the NEC problem

C. Baseline: Genetic Algorithm 1 (GA1)

In order to demonstrate the effectiveness and efficiency of our greedy algorithm, we come up with *Genetic Algorithm* 1 (GA1) in Fig. 3. Genetic algorithm [1] is a meta-heuristic method targeted at providing an approximate optimal solution for general optimization problems inspired by natural selection.

Here, the assignment of VNFRs to the PRs is the chromosome. For example, if there are three VNFRs r_1, r_2, r_3 assigned to PRs p_2, p_5, p_3, respectively, the chromosome is [2 5 3]. The fitness function here is the calculation of $\min(\max(E_{p_j}))$ in the allocation. In GA1, we first randomly generate a population of chromosomes POP whose size is denoted by $|POP|$. Then we go into the loop. In each loop, based on the current population of the chromosomes, we generate a certain percentage (CO_rate) of crossovers, a certain percentage (MT_rate) of mutations, and a certain percentage ($RD\_rate = 1 - CO\_rate - MT\_rate$) of random chromosomes. The crossover is generated by two randomly chosen chromosomes and the mutation is from one chromosome. We set $CO\_rate > MT\_rate >> RD\_rate$ to pass on good genes to the next generation to avoid local optima. In GA1, the major time complexity comes from the number of LOOPS to find an approximate optimal solution. As we will see later in the simulations, this time is substantially larger than that of the heuristic algorithm.

For the same numerical example we presented in Fig. 2, the best solution found by GA1 is to allocate VNFRs with energy consumptions $80, 42, 24$, 3 on p_1, 97, 51 on p_2, and $64, 70$, 14 on p_3, resulting in a min-max energy consumption of 149, which is better than 155 in Algorithm RANEC.

IV. Constraint on PR Energy Consumption

A. Problem Formulation

In this problem, we assume that there is a constraint on energy consumption in the PRs. Suppose the maximum energy consumption allowed in each PR should not exceed E_{max}. We still assume that there are m independent VNFRs $\{r_1, r_2, \cdots, r_m\}$ that will be allocated to n PRs $\{p_1, p_2, \cdots, p_n\}$. The energy consumption of each VNFR is given by e_i ($1 \leq i \leq m$). A variable $x_i \in \{0, 1\}$ indicates whether a VNFR is assigned to a PR p_j or not. After allocation, the total energy consumed on each PR $E_{p_j} = \sum_i e_i x_i \leq E_{max}$. In order to finish all the VNFRs, we may need to involve more PRs in the network. The objective of this problem is to minimize the number of PRs needed to finish all the VNFRs. We denote this problem as the *Energy Constraint* (EC) problem. We have proved this problem to be NP-complete.

B. Our Solutions

We propose four heuristic *request allocation algorithms with energy constraint* (RAECA) to solve the problem.

1) Algorithm RAECA1: The main idea of RAECA1 is as follows: we first allocate r_1 to p_1. We define $rea(j)$ (≥ 0) as the *remaining energy allowed* on p_j after E_{max} subtracts the current energy consumption of the VNFRs allocated on it. If $rea(p_1) \geq e_{r_2}$, we allocate r_2 to p_1. Otherwise we need to add p_2 and put r_2 there. Generally speaking, if we have already put $r_1, r_2, \cdots, r_k$ on $p_1, p_2, \cdots, p_r$, and if $e_{r_{k+1}}$ is less than the remaining energy allowed in some of the PRs, we will allocate r_{k+1} to the PR with the smallest label. Otherwise, we will add a new PR p_{r+1} and run r_{k+1} there. We repeat this process until all the VNFRs are allocated.

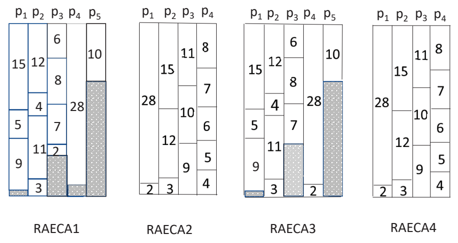


Fig. 4. An example using RAECA1, RAECA2, RAECA3, and RAECA4

2) Algorithm RAECA2: In RAECA2, we first sort VNFRs in a non-increasing order according to their energy consumptions and then call algorithm RAECA1.

3) Algorithm RAECA3: RAECA3 follows the idea of RAECA1 except that when there are multiple PRs whose remaining energy allowed is large enough for r_{k+1}, we put r_{k+1} on the PR with the smallest remaining energy allowed.

4) Algorithm RAECA4: We first sort VNFRs in a non-increasing order according to their energy consumptions and then call RAECA3.

Let us use an example to explain the four heuristic algorithms. Suppose the energy consumptions of 13 VNFRs are $15, 5, 12, 9, 4, 11, 3, 6, 8, 7, 28, 2$, and 10, respectively. The maximum allowed energy consumption on each PR $E_{max} = 30$. We obtain the VNFR allocations from the four algorithms in Fig. 4. In this example, RAECA2 and RAECA4 have the same allocation and use the least number of PRs, which is 4.

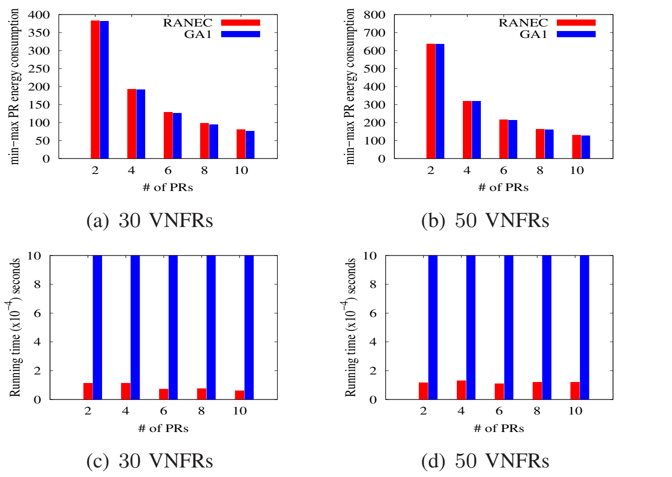
Algorithms RAECA1 and RAECA3 are online algorithms in that they can be used when VNFRs keep coming in. But Algorithms RAECA2 and RAECA4 are offline algorithms because they need to sort the VNFRs first.

C. Baseline: Genetic Algorithm 2 (GA2)

We follow the same idea of GA1 to write *Genetic Algorithm* 2 (GA2) for the EC problem. Each chromosome is still made up of the PR numbers assigned to the VNFRs. We first generate $|POP|$ chromosomes and then generate a certain number of crossovers, mutations, and random chromosomes. Then we select the best $|POP|$ chromosomes according to the fitness function for the next generation. This process repeats many times. One difference here is that the fitness function is the minimum number of PRs used in each chromosome. Another difference is that we need to make sure that the crossovers, mutations, and random chromosomes produced are valid. That is, the total energy consumption of the VNFRs placed on a PR cannot exceed $maxE$.

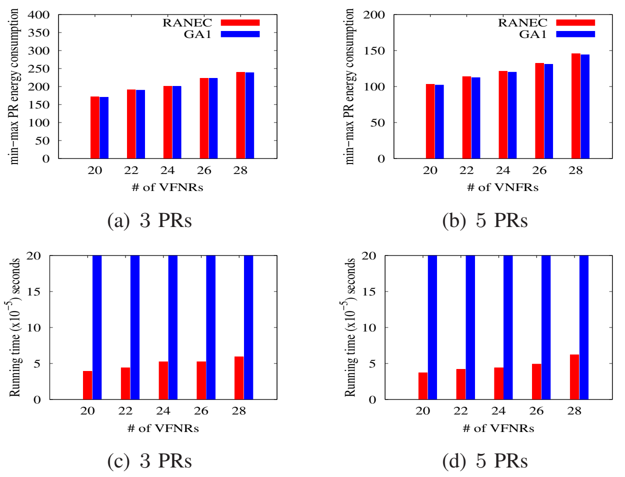
V. Simulations

In this section, we evaluate the performance of our proposed algorithms by comparing them with the baseline genetic algorithms using a simulator written in Matlab. We first compare the algorithms for the NEC (no constraint) problem and then the algorithms for the EC (constraint) problem.



(a) 30 VNFRs (b) 50 VNFRs (c) 30 VNFRs (d) 50 VNFRs

Fig. 5. Comparison of RANEC and GA1 algorithms fixing # of VNFRs



(a) 3 PRs (b) 5 PRs (c) 3 PRs (d) 5 PRs

Fig. 6. Comparison of RANEC and GA1 algorithms fixing # of PRs

A. No Constraint on PR Energy Consumption

In this experiment, we compare our proposed RANEC algorithm with GA1. We use two metrics: the minimum of maximum PR (min-max PR) energy consumptions produced by the two algorithms and the corresponding running times producing the results.

We set the number of VNFRs to be 30 and 50, respectively, and withdrew the energy consumption of each VNFR randomly from [1, 50]. The number of PRs we tried was from 2 to 10, with a step of 2. In GA1, we set the population of POP to be 50, the number of loops to be $10k$, the chromosome crossover rate CO_rate, the mutation rate MT_rate, and the random rate RD_rate to be 65%, 30%, and 5%, respectively. We ran each setting 1000 times and averaged the results. Figs. 5(a) and (b) show the average min-max PR energy consumptions produced by the two algorithms when the number of VNFRs is 30 and 50, respectively. The corresponding running times are separately presented in Figs. 5 (c) and (d).

From the simulation results, we can see that in both 30-VNFR and 50-VNFR cases, with the increase of the number of PRs from 2 to 10, the min-max PR energy consumption is reduced. The 50-VNFR case has more min-max PR energy consumption than the 30-VNFR case due to larger numbers of VNFRs. In both cases, the proposed RANEC algorithm produces results very close to the baseline genetic algorithm. In terms of the running time, if we just look at the RANEC algorithm, with the rise of the number of PRs, the running time of the algorithm does not change much due to the fact that the number of VNFRs is fixed. If we compare RANEC with GA1, the running time of GA1 is substantially higher than that of the RANEC algorithm - It takes GA1 more than 10 hours rather than several 10^{-4} seconds in the RANEC algorithm to produce a slightly better result. Because of this big contrast, we draw the running time of the genetic algorithm to the full range of the vertical axis in Figs. 5(c) and (d) and present the scale here. In the rest of the paper, we do the same thing to all the running times of the genetic algorithms. From this experiment, we can conclude that the RANEC algorithm is both effective and efficient.

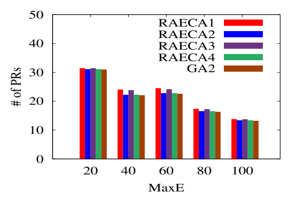
Next, we fixed the number of PRs to be 3 and 5, respectively and varied the number of VNFRs from 20 to 28, with a step of 2. All the other parameters were kept the same as the previous experiment. Figs. 6(a) and (b) show the average min-max PR energy consumptions produced by the two algorithms when the number of PRs is 3 and 5, respectively. The corresponding running times are separately presented in Figs. 6 (c) and (d).

In both 3-PR and 5-PR cases, with the increase of the number of VNFRs, more energy consumption is required. Comparing the 3-PR case with the 5-PR case, the 5-PR case produces lower min-max PR energy consumption due to more PRs. In both cases, the min-max PR energy consumption of the RANEC algorithm is very close to that of GA1. In terms of the running time, in both cases, when the number of VNFRs goes up, the running time of the RANEC algorithm also goes up due to the increase of VNFRs. Comparing with RANEC, GA1 uses more than 10 hours rather than several 10^{-5} seconds in RANEC to achieve a little better result. This again confirms the effectiveness and efficiency of the proposed algorithm.

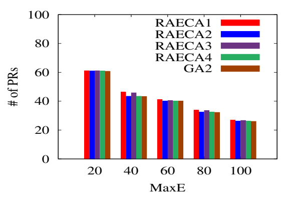
B. Constraint on PR Energy Consumption

In this experiment, we compare our proposed four RAECA algorithms with GA2. We use two metrics: the number of PRs produced by the algorithms and the corresponding running time producing the results.

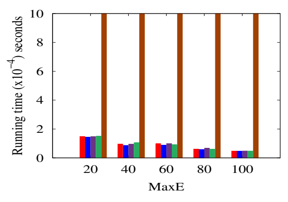
We set the number of VNFRs to be 50 and 100, respectively and drew the energy consumption of each VNFR randomly from [1, 50]. The maximum allowed energy consumption $maxE$ in each PR was changed from 20 to 100, with a step of 20. In the genetic algorithm, we set the population of POP to be 50, the number of loops to be $10k$, the chromosome crossover rate CO_rate, the mutation rate MT_rate, and the random rate RD_rate to be 65%, 30%, and 5%, respectively. We ran each simulation 1000 times and averaged the results. Figs. 7(a) and (b) show the average number of PRs produced by the algorithms when the number of VNFRs is 50 and 100, respectively. And the corresponding running times are separately presented in Figs. 7 (c) and (d).



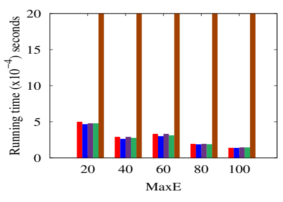
(a) 50 VNFRs



(b) 100 VNFRs

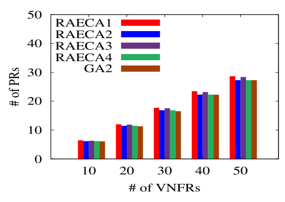


(c) 50 VNFRs

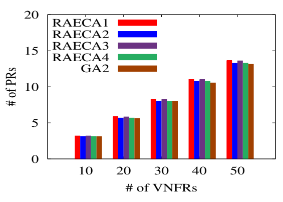


(d) 100 VNFRs

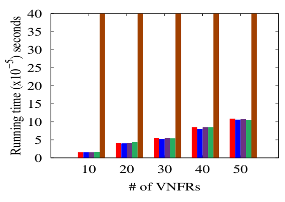
Fig. 7. Comparison of RAECA and GA2 algorithms fixing # of VNFRs



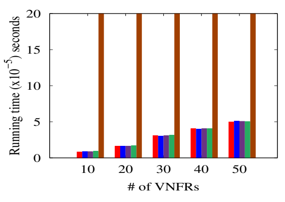
(a) MaxE=50



(b) MaxE=100



(c) MaxE=50



(d) MaxE=100

Fig. 8. Comparison of RAECA and GA2 algorithms fixing $MaxE$

From the simulation results, we can see that in both the 50- and 100-VNFR cases, with the increase of $MaxE$ from 20 to 100, the number of PRs allocated is reduced. This is straightforward because if the PRs allow more energy consumption, fewer resources are needed. Also, the PR numbers generated by the four proposed algorithms are very close to those of the genetic algorithm GA2. Among the four proposed algorithms, RAECA2 and RAECA4 use fewer PRs because they are offline algorithms. But the more practical online algorithms RAECA1 and RAECA3 can produce very similar results. In terms of the running time, there is not much difference among the four heuristic algorithms. At some data points, RAECA2 and RAECA4 use less time than RAECA1 and RAECA3. This indicates that sorting the VNFRs first can make the algorithms allocate resources more efficiently. In contrast to the several 10^{-4} second running times of the heuristic algorithms, it takes GA2 at least 10 hours to reach a slightly better result. This experiment confirms that our proposed heuristic algorithms can produce good results in a much shorter time.

In the next experiment, we fixed $maxE$ to be 50 and 100, respectively. We increased the number of VNFRs from 10 to 50, with a step of 10. We kept the rest of the parameters the same as the previous experiment. Figs. 8(a) and (b) show the average number of PRs produced by the algorithms when $maxE$ is 50 and 100, respectively. The related running times are separately presented in Figs. 8(c) and (d).

Comparing the results in Fig. 8(a) with those in Fig. 8 (b), it is obvious that for the same number of VNFRs, increasing $maxE$ can reduce the number of PRs. In each figure, it is evident that with the increase of the number of VNFRs, the number of PRs is also increased. The PR numbers produced by the four proposed algorithms are very close to those of GA2. Among the four proposed algorithms, the PR numbers produced by the RAECA2 and RAECA4 algorithms are a little lower than those of the online algorithms RAECA1 and RAECA3 due to their offline nature. For the running time, there is not much difference among the four heuristic algorithms. We again observe that RAECA2 and RAECA4 use less time than RAECA1 and RAECA3 at some data points in our experiment. This indicates that sorting VNFRs first can make the algorithms more efficient. Comparing with the several 10^{-5} second running times of the heuristic algorithms, it takes GA2 at least 10 hours to come up with a little better result. Once more, we can assert that our proposed heuristic algorithms are effective and much more efficient.

VI. Conclusion

In this paper, we have worked on the energy efficient strategies for the NFV-RA problem in the edge computing environment. We have discussed two NP-complete problems and proposed heuristic algorithms to solve them. We have also put forward baseline algorithms using genetic programming to find approximate optimal solutions to the problems. We have conducted simulations to evaluate the performance of our proposed algorithms by comparing them with the baseline algorithms. Simulation results have proved that our proposed algorithms are both effective and efficient. In the future, we hope to measure VNFR energy consumption using some tools.

References

[1] Genetic Algorithm. https://en.wikipedia.org/wiki/Genetic_algorithm.
[2] *Traffic Aware Placement of Interdependent NFV Middleboxes*, 2017.
[3] *VNF Placement with Replication for Load Balancing in NFV Networks*, 2017.
[4] M. Abu-Lebdeh, D. Naboulsi, R. Glitho, and C. W. Tchouati. On the Placement of VNF Managers in Large-Scale and Distributed NFV Systems. *IEEE Transactions on Network and Service Management*, 14(4):875–889, 2017.
[5] R. Cohen, L. Lewin-Eytan, J. Naor, and D. Rax. Near Optimal Placement of Virtual Network Functions. In *IEEE INFOCOM*, 2015.
[6] B. Han, V. Gopalakrishnan, L. Ji, and S. Lee. Network function virtualization: Challenges and opportunities for innovations. *IEEE Communications Magazine*, 53(2):90–97, 2015.
[7] J. G. Herrera and J. F. Botero. Resource Allocation in NFV: A Comprehensive Survey. *IEEE Transactions on Network and Service Management*, 13(3):518–532, 2016.
[8] W. Ma, J. Beltran, Z. Pan, D. Pan, and N. Pissinou. SDN-based Traffic Aware Placement of NFV Middleboxes. *IEEE Transactions on Network and Service Management*, 14:528–542, 2017.
[9] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu. Edge computing: Vision and challenges. *IEEE Internet of Things Journal*, 3(5):637–646, 2016.
[10] P. Vizarreta, M. Condoluci, C. M. Machuca, T. Mahoodi, and W. Kellerer. QoS-driven Function Placement Reducing Expenditures in NFV Deployments. In *IEEE ICC*, 2017.