# Virtual Network Function Scheduling: A Matching Game Approach

Chuan Pham, Nguyen H. Tran, and Choong Seon Hong

***Abstract*—Network function virtualization is a promising technique for telecom providers to efficiently manage network services at low cost. However, existing works mainly focus on resource allocation and thus leave behind an important issue: the virtual network function (VNF) scheduling. Current approaches, e.g., round-robin scheduling or heuristic algorithms, still expose some unsolved issues, such as high computational cost and inability to perform online scheduling. In this letter, we propose a matching-based algorithm to solve the NP-hard VNF scheduling problem. This approach can guarantee a stable scheduling, in which all network services are satisfied with the assignment. Finally, the effectiveness of our method is verified through numerical evaluation, showing that our approach can increase the number of completed VNFs by 36.8% compared with the current round-robin method.**

***Index Terms*—Network function virtualization, service chain.**

## I. Introduction

NETWORK function virtualization (NFV) [1] potentially provides flexible and efficient network services to network operators based on robust virtualization technologies. Network functions are mainly built on hardware appliances (e.g., firewall, load balancers). NFV enables software-oriented network functions, where virtual network functions (VNFs) can run as softwares on commodity servers [1], [2]. Hence, VNFs can be easily and dynamically deployed over the infrastructure to manage network services at low cost (both CAPEX and OPEX)[1] by telecom operators. However, to enable NFV implementation, the network operator has to solve two problems: a) VNF resource allocation with service chains (SCs) and b) VNF scheduling.

Currently, most works mainly concentrate on the first issue, often called the VNF placement. Given the limited network resources (CPU, memory, network bandwidth), a VNF placement solution needs to efficiently allocate VNFs on physical nodes and to map virtual links onto physical links. In this letter, we focus on the second issue, *VNF scheduling, which finds an execution scheme for VNFs depending on their SC*, given the computational resources and VNF placement results.

Due to the diversity of network services, the VNF scheduling problem is not solvable in polynomial-time [3]. The authors in [3] formulated the VNF scheduling as the job-shop scheduling problem, which is NP-hard. Similarly, [4] also formulated the problem as mixed integer linear program (MILP), and proposed heuristic algorithms to solve. Even though these methods can obtain acceptable results in analysis, they are impractical because of several reasons. First, their approaches can only solve static SCs, in which during the scheduling time the number of VNFs does not change (i.e., offline VNF scheduling). Thus, when a new VNF request arrives, it has to wait until the current schedule is complete. In practice, an online VNF scheduling, suitable for unknown future network services arrival rate, is often considered owing to dynamic VNF arrivals. Second, the complexity is shown to increase exponentially when the number of VNFs is large [3], [4]. Therefore, a practical solution needs both efficient implementation and low computational complexity.

We address these challenges by advocating matching game theory, a promising technique for resource allocation in networks that provides mathematically tractable solutions for the combinatorial optimization problem [5]. Unlike existing algorithms in [3] and [4], the proposed matching-based algorithm achieves efficient computational cost and adapts to online VNF scheduling. Our main contributions are summarized as follows:

- The VNF scheduling problem is first posed as a combinatorial optimization problem, which cannot find the optimal solution in polynomial time. To solve this problem, we then propose a matching-based algorithm to match VNFs to resource nodes at each time slot. This approach does not guarantee the optimal solution; however, it can achieve a stable scheduling that can satisfy all network constraint, and all physical nodes and SCs are "happy" with that schedule. Furthermore, this approach can extend from offline VNF scheduling to online VNF scheduling where the controller does not need to know the future network service arrival rate.
- Numerical results show that the proposed matching-based approach can reach 87.2% of the optimal result. Furthermore, the comparison result shows that our approach can increase the number of completed VNFs by 36.8% compared to round-robin approach.

## II. System Model and Problem Formulation

We consider a set $\mathcal{S}$ of network SCs, and for each network service chain $i \in \mathcal{S}$, there is a set $\mathcal{F}_i$ of VNFs. All VNFs in a service chain have to be executed in a corresponding order (e.g., to run the web application, the firewall function has to verify the request packets before serving at the web service function). Hence, we let $f_{i,j}$, $(i \in \mathcal{S}, j \in \mathcal{F}_i)$ to represent the $j^{\text{th}}$ VNF of the service chain $i$.

In the resource allocation of NFV, VNF scheduling is the second phase that the scheduling calculation is based on the placement result of the first phase [4]. A different placement scheme will lead to a different solution for VNF scheduling. Given the computational resource constraints and the execution order of VNFs in each SC (which is the output of the

Manuscript received June 29, 2017; accepted August 17, 2017. Date of publication August 31, 2017; date of current version January 8, 2018. This work was supported by Institute for Information & communications Technology Promotion (IITP) grant funded by the Korea government (MSIT) (R0126-16-1009, Development of Smart Mediator for Mashup Service and Information Sharing among ICBMS Platform). The associate editor coordinating the review of this letter and approving it for publication was D. Calin. *(Corresponding author: Choong Seon Hong.)*

The authors are with the Department of Computer Science and Engineering, Kyung Hee University, Seoul, South Korea (e-mail: pchuan@khu.ac.kr).

Digital Object Identifier 10.1109/LCOMM.2017.2747509

[1]CAPital EXpenditures and OPerational EXpenditures.

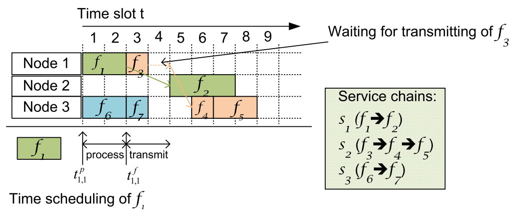

Time slot t
1 2 3 4 5 6 7 8 9
Waiting for transmitting of $f_3$
Node 1
Node 2
Node 3
Service chains:
$s_1 (f_1 \rightarrow f_2)$
$s_2 (f_3 \rightarrow f_4 \rightarrow f_5)$
$s_3 (f_6 \rightarrow f_7)$
process transmit
Time scheduling of $f_1$

Fig. 1. Example of VNF scheduling with 3 SCs that are executed on 3 physical nodes (node 1 is hosting $f_1$ and $f_3$, node 2 is hosting $f_2$, and node 3 is hosting $f_4$, $f_5$, $f_6$, $f_7$).

first phase), the controller needs to find the shortest feasible schedule to execute VNFs of these SCs at the corresponding physical nodes. Following [4], we assume that the execution time slot can be divided into a small duration time $\Delta t$ (e.g., 10 ms).

We consider a virtual network consisting of a set $\mathcal{N}$of physical nodes hosting network services. We use a binary assignment matrix $\boldsymbol{x}^p$ where each element $x^p_{i,j,n} = 1$ indicates the $j^{\text{th}}$ VNF of the SC $i$ located on node $n \in \mathcal{N}$, and $x^p_{i,j,n} = 0$ otherwise. We denote the set of virtual links of SC $i$ by $\mathcal{V}_i$ in which the $j^{\text{th}}$ virtual link connects the $j^{\text{th}}$ VNF and the $(j+1)^{\text{th}}$ VNF in SC $i$. We also define a binary mapping matrix $\boldsymbol{x}^f$, where the matrix element $x^f_{i,j,k} = 1$ represents the $j^{\text{th}}$ virtual link of SC $i$ mapped into the physical link $k$ belonging to the set $\mathcal{K}$ of physical links, and $x^f_{i,j,k} = 0$ otherwise. These matrices in our model are given since they are the result of mapping processing of the first phase. Each physical link $k \in \mathcal{K}$ with a link capacity $c_k$ connects two physical nodes $n, n' \in \mathcal{N}$. We assume that before scheduling the controller obtains the exact traffic flow of SC $i$ with a certain requirement $r_i$ and the physical link transfers only one embedded virtual link until it finishes. Consequently, the transmission time of the $j^{\text{th}}$ virtual link of SC $i$ on the physical link $k$ is $r_i/c_k$ if $x^f_{i,j,k} = 1$.

*Processing and Transmission Delay Model:* We assume that a VNF can start its service if and only if its preceding VNF (following the order of its SC) finished its service and completely forwarded all of the flow's packets [3]. We denote by $p_{i,j}$ the processing time of the $j^{\text{th}}$ VNF of the SC $i$. We define the variable $t^p_{i,j}$ to describe the starting time to process the $j^{\text{th}}$ VNF of SC $i$. For scheduling, we define a binary variable $\tau^p_{i,j,n,t}$ to describe the assignment of the $j^{\text{th}}$ VNF of SC $i$ on node $n$ to be processed at time slot $t$ in the scheduling duration $T$. Considering the last VNF in the schedule that has the latest completion time, we then define $\sigma$ as the completion time of that final VNF as follows: $\sigma = \max_{i \in \mathcal{S}, j \in \mathcal{F}_i, n \in \mathcal{N}} \{t^p_{i,j} + p_{i,j}\}$. Furthermore, the constraints to ensure the order of *processing* can be formulated as follows:

$$t^p_{i,j} + p_{i,j} \le t^p_{i,j+1}, \tag{1}$$
$$\sum_{t=1}^{T} x^p_{i,j,n} \tau^p_{i,j,n,t} \Delta t \ge x^p_{i,j,n} p_{i,j}, \tag{2}$$
$$\sum_{i \in \mathcal{S}, j \in \mathcal{F}_i} x^p_{i,j,n} \tau^p_{i,j,n,t} \le 1, \tag{3}$$
$$t^p_{i,j} + \sum_{t=1}^{T} x^p_{i,j,n} \tau^p_{i,j,n,t} \Delta t \le t^p_{i,j'}, \tag{4}$$
$$t^p_{i,j} + \sum_{t=1}^{T} \tau^p_{i,j,n,t} x^p_{i,j,n} \Delta t \le \bar{t}_{i,j},$$
$$\forall i \in \mathcal{S}, \quad \forall j \in \mathcal{F}_i, \ \forall n \in \mathcal{N}, \ \forall t = 1, \dots, T,$$

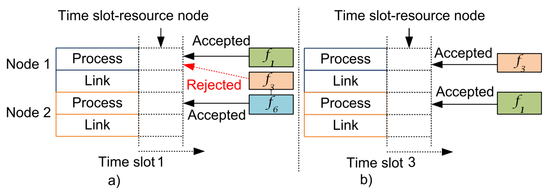


Fig. 2. Example of a matching-based approach for VNF scheduling at timeslots 1 and 3.

$$\text{and } \forall j' \notin \mathcal{F}_i | x^p_{i,j',n} = x^p_{i,j,n} = 1, \quad t^p_{i,j} < t^p_{i,j'}. \tag{5}$$

Constraint (1) ensures the order of the starting time to process the $j^{\text{th}}$ and $(j+1)^{\text{th}}$ VNFs in SC $i$. (2) guarantees that the total allocated time slots for the $j^{\text{th}}$ VNF must be sufficient to complete the process of the $j^{\text{th}}$ VNF. (3) ensures that at each time slot node $n$ can only process for one VNF. (4) guarantees non-preemptive scheduling. This implies that if any VNF $j'$ hosted on node $n$ starts to process later than $j$ (i.e., $t^p_{i,j} < t^p_{i,j'}$), it cannot start during processing time of $j$. (5) ensures the hard deadline $\bar{t}_{i,j}$ of processing the $j^{\text{th}}$ VNF in SC $i$.

Similarly, we assume that a physical link can only forward traffic from only one virtual link at the same time until the transmission of all of the flow's packets from the current network service is completed. We define $t^f_{i,j}$ to describe the starting time to forward the flow's packets of the $j^{\text{th}}$ link of SC $i$. Let $t^f_{i,j,k,t}$ be the assignment variable to indicate the $j^{th}$ virtual link of the SC $i$ transmitted on the physical link $k$ at time slot $t$. The constraints that guarantee the order of *process and transmit* in a service chain can be formulated as follows:

$$t^p_{i,j} + \sum_{t=1}^{T} \tau^p_{i,j,n,t} x^p_{i,j,n} \Delta t \le t^f_{i,j}, \tag{6}$$
$$t^f_{i,j} + \sum_{t=1}^{T} \tau^f_{i,j,k,t} x^f_{i,j,k} \Delta t \le t^p_{i,j+1}, \tag{7}$$
$$\sum_{t=1}^{T} x^f_{i,j,k,t} \tau^f_{i,j,k,t} \Delta t \ge r_i/c_k, \tag{8}$$
$$\sum_{i \in \mathcal{S}, v \in \mathcal{V}_i} x^f_{i,j,k} \tau^f_{i,j,k,t} \le 1,$$
$$\forall i \in \mathcal{S}, \quad \forall j \in \mathcal{F}_i, \ \forall n \in \mathcal{N}, \ \forall k \in \mathcal{K}, \ \forall t = 1, \dots, T. \tag{9}$$

Constraint (6) indicates that the traffic of the $j^{\text{th}}$ VNF is only forwarded after being processed completely. Similarly, (7) ensures that the next VNF is processed only when the preceding VNF is forwarded completely. (8) ensures that the total allocated time slot of transmission has to satisfy the requirement. (9) guarantees the unique forwarding link at one time slot. In condensed form, the VNF scheduling problem can be formulated as follows

$$\mathbf{P_{VNS}} : \min \sigma \quad \text{subject to constraints (1)} - \text{(9)}.$$

$\mathbf{P_{VNS}}$ is a combinatorial optimization problem that assigns VNFs to node resources at each time slot (considering two types of resources: processing resources and link resources).

Since $\mathrm{P_{VNS}}$is NP-hard, it cannot be found in the polynomial time. We use matching-based approach as a solution to $\mathbf{P_{VNS}}$ in each time slot. As proposed in [5] and [6], the network resource assignment problem can be solved with an effective solution using the framework of matching theory. Furthermore, the matching approach can be applied to online scheduling, where newly arrived VNFs can join to the scheduling process without waiting time as current works.

## III. Matching Approach for VNF Scheduling

The problem $\mathbf{P_{VNS}}$ is formulated as an *offline VNF scheduling*, where the number of VNFs is given and the duration for scheduling is calculated based on $\max_{i\in\mathcal{S},j\in\mathcal{F}}\{\bar{t}_{ij}\}$.

*Offline VNF Scheduling:* The assignment of VNFs to resource nodes at each time slot (named resource node-slot) can be considered as an outcome of the one-to-one matching game. Considering the example shown in Fig. 2, at time slot 1, we have six resource node-slots corresponding to the processing resources and physical links of three physical nodes (with the same placement scheme of Fig. 1). On the other side, VNFs play the role as the "men" in the stable marriage problem [5], who "propose" to the resource node-slots. For example, in case a) of Fig. 2, VNF $f_1$ and VNF $f_3$ propose to the processing slot of Node 1; however, only $f_1$ is accepted while $f_3$ is rejected. This procedure is similar to the one-to-one matching game. For notation at simplicity, we denote $\mathcal{T}$ as the set of resource node-slots ($|\mathcal{T}| = \max_{i\in\mathcal{S},j\in\mathcal{F}}\{\bar{t}_{ij}\}$) and $\mathcal{F}$ as the set of VNFs. We also define $r^m_{n,t} \in \mathcal{T}$ as the resource node-slot of type $m$ (including process and link resources) on node $n \in \mathcal{N}$ at time slot $t$. Before presenting our matching algorithm, we define basic concepts as follows.

*Definition 1:* The outcome of a VNF scheduling in problem $\mathbf{P_{VNS}}$ is a matching function $\mu : \mathcal{F}\cup\mathcal{T} \to 2^{\mathcal{F}\cup\mathcal{T}}$ such that:

- $\mu(r^m_{n,t}) = j$ if and only if $\mu(j) = r^m_{n,t}$, in which resource node-slot $r^m_{n,t}$ is matched to VNF $j$.
- If $\mu(r^m_{n,t})$ is not in $\mathcal{F}$, then $r^m_{n,t}$ is unmatched.
- If $\mu(j)$ is not in $\mathcal{T}$, then $j$ is unmatched.

To execute VNF scheduling as a matching game, we need to define the preference lists for both sides as follows.

### *A. Resource Node-Slot's Preference List*

At each time slot, each node exists as both a) a processing resource slot and b) physical link resource slot, which can be processed in parallel. A resource node-slot prefers executing/forwarding a VNF that has a smaller processing/transmission time (This intuition guarantees a 2−approximation algorithm for scheduling, proven in [7]). Hence, the resource node-slot's preference list $P_{r^m_{n,t}}$ is made based on the duration of processing/transmission time of VNFs. However, to avoid the worse case that a VNF has to wait forever (because the VNF has the longest processing/transmission time), we define the priority $y_j$ for VNF $j$, which is calculated based on the following rules: a) if VNF $j$ is processing/forwarding, it has the highest priority; b) otherwise, if the hard deadline $\bar{t}_{i,j}$ is shorter than $\bar{t}_{i,j'}$, $j$ has higher priority than $j'$. Then, $r^m_{n,t}$ ranks a VNF based on the following preference function

$$\Theta_{r^m_{n,t}}(j) = y_j. \tag{10}$$

### *B. VNF's Preference List*

Similar to the procedure in the one-to-one matching game, a VNF "proposes" to unassigned resource node-slots depending on "his" preference list. A VNF prefers to be processed/forwarded as soon as possible. Hence, the preference list $P_j$ of a VNF follows the time index of $r^m_{n,t}$ as follows:

$$\Theta_j(r^m_{n,t}) = t. \tag{11}$$

*Definition 2:* A VNF scheduling is stable if and only if there is no blocking pair and all VNFs are assigned.

Traditionally, a stable matching in the deferred-acceptance algorithm is a state that no player wants to change their partners. This implies that the matching is stable, if there is no blocking pair, which is defined as follows:

*Definition 3:* A matching $\mu$ is blocked by a pair of agents $(j, r^m_{n,t})$ if there exists a pair $(j, r^m_{n,t})$ with $j \notin \mu(r^m_{n,t})$ and $r^m_{n,t} \notin \mu(j)$ such that $j \succ_{r^m_{n,t}} \mu(r^m_{n,t})$ and $r^m_{n,t} \succ_j \mu(j)$. Such a pair is called a *blocking pair*.

The existing of a blocking pair in the VNF scheduling implies that the agents prefer to change their partners to complete processing/forwarding tasks earlier.

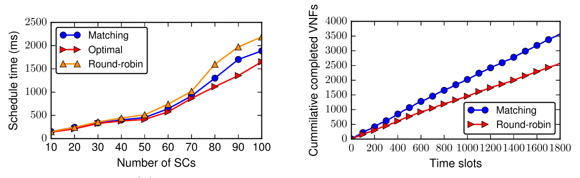
**Algorithm 1** Matching-Based VNF Scheduling

```
Input: $\mathcal{F}$
1  Initialization: $t = 1$ ;
2  Calculate priority $y_j$;
3  Calculate $P_{r^m_{n,t}}$, $P_j$ by (10) and (11), respectively;
4  repeat
5      forall the unassigned/uncompleted VNF $j \in \mathcal{F}$ do
6          $j$ proposes to its most preferred in $P_j$;
7      end
8      forall the unassigned $r^m_{n,t}$ do
9          if $r^m_{n,t}$ receives the proposed VNF $j \in P_{r^m_{n,t}}$ then
10             if $j \succ_{r^m_{n,t}} \mu(r^m_{n,t})$ then
11                 Remove $\mu(r^m_{n,t})$ and all $j'$ from $P_{r^m_{n,t}}$, where
                   $j \succ_{r^m_{n,t}} j'$;
12                 Remove $r^m_{n,t}$ from $P_{j'}$;
13                 Set $\mu(r^m_{n,t}) = j$;
14                 Set executing status for $j$;
15                 Set $y_j = 1$, when remaining time $j$ to
                   process/forward packet is positive;
16             else
17                 Remove $j$ from $P_{r^m_{n,t}}$ and $r^m_{n,t}$ from $P_j$;
18             end
19         end
20     end
21     $t \leftarrow t + 1$;
22 until All VNFs are assigned and completed.;
```

### *C. Proposed Algorithm*

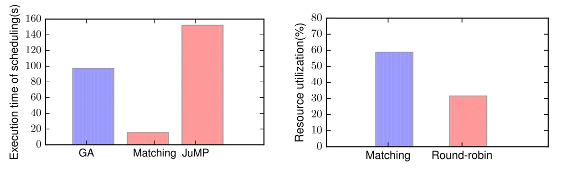
As a solution of the one-to-one matching game, we propose a novel VNF scheduling to produce a stable matching in Alg. 1. However, to guarantee the order of the SC, we create a rule for the proposing step for VNFs, where a VNF $j'$ cannot propose to a process before $j$, if $j < j'$ and $x^p_{i,j,n} = x^p_{i,j',n} = 1$. Furthermore, VNF $j$ cannot propose to the resource node-slot $n'$ if VNF $j$ is hosted on node $n$, (i.e., $\Theta_j(r^m_{n',t}) = \infty, x^p_{i,j,n} = 1$). This rule does not affect the results of the one-to-one matching game [5].

Similar to the deferred-acceptance algorithm, in Alg. 1, VNFs propose to available resource node-slots (lines 6-7). For each $r^m_{n,t}$, "she" receives proposals from unassigned/uncompleted process VNF $j$ that ranks $r^m_{n,t}$ as the highest rank in $j$'s preference list (line 10). Following the preference



(a) (b)

Fig. 3. Comparison with baseline and optimal results.



(a) (b)

Fig. 4. Evaluation of scheduling performance.

list of $r^m_{n,t}$, VNF $j$ is accepted if $j$ has the highest ranking (i.e., $j$ has the shortest deadline, or is processing/forwarding). If $r^m_{n,t}$ is assigned, $j$ removes $r^m_{n,t}$ out of "his" preference list. If $r^m_{n,t}$ is accepted to process/forward packets of $j$, the next time slot $r^m_{n,t+1}$ will update "her" preference list with the highest ranking for $j$, when $j$ is processing/transmitting (lines 11-19). Alg. 1 will stop when all VNFs are assigned.

*Theorem 1: Alg. 1 converges to a stable assignment.*

*Proof:* We adapt the proof from [6] to prove this theorem by contradiction. Suppose that Alg. 1 produces a matching $\mu$ with a blocking pair $(r^m_{n,t}, j)$ by Def. 3. Since $j \succ_{r^m_{n,t}} \mu(r^m_{n,t})$, $j$ must have proposed to $r^m_{n,t}$ and has been rejected due to lower priority on ranking of $r^m_{n,t}$ than $\mu(r^m_{n,t})$. When $j$ was rejected, then $j' = \mu(r^m_{n,t})$ was either rejected with $j$, or was unable to propose because $r^m_{n,t}$ is removed from the preference list of $j'$. Thus, it means that $j' \notin \mu(r^m_{n,t})$, which contradicts with the previous mention, $j' = \mu(r^m_{n,t})$. $\square$

We next consider the case of unknown the number of VNFs arrival (*i.e., online VNF scheduling*).

*Online VNF Scheduling:* With unknown workload, Alg. 1 is still adaptable. Specifically, in each time $t$, the resource node-slots need to update their preference list to re-rank new VNFs (line 3), while new VNFs calculate their preference lists before proposing (line 3). Alg. 1 iteratively executes until there is no unassigned VNF. The newly arrived VNFs at each iteration are considered as unassigned VNFs. The steps for VNFs and resource node-slots (lines 5-20) are repeated in every time slot similar to the offline scheduling, where unassigned VNFs propose to resource-node-slots to process/forward packets. The approval process of resource-node-slots is executed similar to the offline scheduling (lines 8-20). Therefore, new VNFs, arriving to the system at time slot $t$, can automatically join to be scheduled with the current VNFs without waiting, as in the offline scheduling of [3] and [4].

## IV. Numerical Results

In this section, we evaluate the performance of the matching-based VNF scheduling approach in both the offline and online cases. The physical network is created with the set of 50 nodes that are fully connected. The total input/output bandwidth of each node ranges from 2 to 10 Mbps and the traffic requirement between VNFs is set in the range of 60 to 100 kbps. Furthermore, we set the execution time of VNFs randomly from [10, 30] milliseconds. We create SCs randomly based on [2], which exists from 3 to 5 VNFs per each SC. These SCs are placed randomly on the given physical network, where each node has a capacity to host from 1 to 10 VNFs.

We compare the performance of our approach (denoted as "matching") with the optimal solution solved by JuMP [8] (denoted as "optimal") and the current round-robin approach [3] (denoted as "round-robin"). Fig. 3a depicts the scheduling time of three approaches corresponding to the increasing number of SCs from 10 to 100. By using the matching approach in offline scheduling, our method can attain 87.2% of the optimal result and outperforms the round-robin approach. For the online scheduling, the matching algorithm can increase by 36.8% the number of completed VNFs observed during 1800 time slots as shown in Fig. 3b.

Furthermore, we compare the execution time by GA [4] and JuMP solver. The matching approach outperforms others with only taking 15.3s to make a schedule for 100 SCs and 50 nodes (Fig. 4a). Moreover, compared to the round-robin approach, Fig. 4b shows that the resource utilization in the matching approach increases by 27.3% during observing time.

## V. Conclusion

We consider the VNF scheduling problem, which is not solvable in polynomial time. To address this challenge, we proposed a matching game approach for scheduling VNFs in which VNFs and resource slots are cast as players in a one-to-one matching game. The comparison with current methods shows that our approach can obtain good results, which reaches 87.2% of the optimal result. Furthermore, our proposed method can be applied easily in online VNF scheduling, which also outperforms the existing round-robin approach.

## References

[1] *NFV*. [Online]. Available: https://portal.etsi.org/nfv/nfv_white_paper.pdf

[2] C. Pham, N. H. Tran, S. Ren, W. Saad, and C. S. Hong, "Traffic-aware and energy-efficient vnf placement for service chaining: Joint sampling and matching approach," *IEEE Trans. Services Comput.*, to be published.

[3] J. F. Riera, E. Escalona, J. Batalle, E. Grasa, and J. A. Garcia-Espin, "Virtual network function scheduling: Concept and challenges," in *Proc. Int. Conf. Smart Commun. Netw. Technol. (SaCoNeT)*, Jun. 2014, pp. 1–5.

[4] L. Qu, C. Assi, and K. Shaban, "Delay-aware scheduling and resource optimization with network function virtualization," *IEEE Trans. Commun.*, vol. 64, no. 9, pp. 3746–3758, Sep. 2016.

[5] Y. Gu, W. Saad, M. Bennis, M. Debbah, and Z. Han, "Matching theory for future wireless networks: Fundamentals and applications," *IEEE Commun. Mag.*, vol. 53, no. 5, pp. 1–52–59, May 2015.

[6] A. E. Roth and M. Sotomayor, "Two-sided matching," *Handbook Game Theory With Economic Application*, vol. 1. Amsterdam, The Netherlands: Elsevier, 1992, pp. 485–541.

[7] D. P. Williamson and D. B. Shmoys, *The Design of Approximation Algorithms*. Cambridge, U.K.: Cambridge Univ. Press, 2011.

[8] *JuMP*. [Online]. Available: http://jump.readthedocs.org/en/latest/index.html