Service Chain Composition with Failures in NFV Systems: A Game-Theoretic Perspective

Simeng Bian, Xi Huang, Xin Gao, Ziyu Shao, Yang Yang
School of Information Science and Technology, ShanghaiTech University
Email: {biansm, huangxi, gaoxin, shaozy, yangyang}@shanghaitech.edu.cn

***Abstract*—Network function virtualization (NFV) initiates a revolution of network service (NS) delivery by forming each NS as a chain of virtual network functions across commodity servers. However, it still remains a key challenge in NFV to decide the chains that induce short latency and low congestion, *a.k.a.* service chain composition problem. Existing works mainly resort to centralized solutions that require full knowledge of the network state to coordinate different users' traffic and NSs, overlooking privacy issues and the non-cooperative interactions among users. Moreover, handling the possible failures due to user/resource unavailability makes the problem even more challenging. By modeling the service chain composition with respect to both user and resource failures as a *non-cooperative game*, we formulate the problem as searching the Nash Equilibrium (NE) with the optimal system performance. By exploiting the unique problem structure, we show that the game is a *weighted potential game* and propose DISCCA, a low-complexity distributed algorithm that guides the system towards the NE with short latency and low congestion, through decision making by individual users with local information. Results from extensive simulations show that DISCCA effectively achieves near-optimal system performance within mild-value of iterations, even in the presence of failures.**

I. Introduction

Motivated by the ever-growing variety in the quality and demands of network services (NSs), traditional NS provisioning based on dedicated hardware has come to its end because of its poor flexibility and scalability in network management. Recently, network function virtualization (NFV) revolutionizes the way of NS delivery by virtualizing and deploying network functions as applications on commodity servers, facilitating better flexibility in NS provisioning, manageability, reliability, and performance guarantee.

In a typical NFV system, each virtual network function (VNF) has multiple instances distributed on different servers out of scalability and fault tolerance. A NS is constructed by chaining instances of different VNFs successively, *a.k.a.* service chain composition [5]. Once deployed, users who subscribe the NS generate request traffic that flows through the instances and acquires treatment in a pipeline fashion. The key challenge is to decide the service chaining decisions that induce short latency and low congestion. On the one hand, if the chained instances in a NS are distributed on servers with poor network conditions, users will experience long latency, leading to SLA violation and reduction in operator revenues. On the other hand, NSs may share the same VNF instance on the same server, leading to resource contention and service quality degradation.

The service chain composition problem is in general $\mathcal{NP}$-hard [6]. Existing works mainly resort to designing heuristics or approximation algorithms to solve the problem. Bari *et al*. [1] formulate the problem as an ILP problem, and develop a heuristic solution based on Viterbi algorithm. Beck *et al*. [2] propose a heuristic method that coordinate the composition of VNF chains and their embedding into the substrate network, scaling well with hundreds of nodes. Zhang *et al.* [3] leverage the open Jackson queue network model and propose heuristic algorithms for joint service chaining and request scheduling problems. All above solutions investigate the problem in a centralized way, coordinating NSs to reach the optimal system performance with fully knowledge of system state. These solutions can be impractical due to privacy issues and non-cooperative interaction among users in practice.

Then D'Oro *et al.* [4] model the chain composition as a *congestion game*, and propose a distributed algorithm with provable convergence performance to a Nash Equilibrium (NE) state of the game in polynomial time. However, the resultant equilibrium state is not necessarily the one with optimal system performance. In fact, finding the optimal NE is in general $\mathcal{NP}$-hard. On the other hand, though being distributed, their solution still requires considerable overhead induced by network state broadcasting. Moreover, their model does not consider failures due to user/resource unavailability, which are inevitable in practice and can have a significant impact on the system performance [7]–[11].

In this paper, we model the service chain composition with both user and resource failures in NFV systems from a game-theoretic perspective. Our main results and contributions are summarized as follows.

Modeling and Formulation: We are the first to study the service chaining composition problem with the possibility of both user and resource failures. By exploiting the problem structure, we show the game is a weighted potential game.

Algorithm Design: By adopting Markov approximation techniques [12], we propose DISCCA, a distributed and low-complexity algorithm that decides the chain compositions with performance guarantee, effectively shortening latency and reducing congestion. DISCCA allows decision making of individual users with only local information and minimum interactions with the others.

Experimental Verification and Analysis: Simulation results show that DISCCA achieves near-optimal system performance, in terms of both short latency and low congestion, requiring only mild-value of iterations. Besides, simulations

TABLE I
COMPARISON BETWEEN EXISTING SOLUTIONS AND OURS.

	[1]	[2]	[3]	[4]	Ours
Performance Guarantee	No	No	Yes	Yes	Yes
Distributed	No	No	No	Yes	Yes
Game-theoretic Perspective	No	No	No	Yes	Yes
Failure-aware	No	No	No	No	Yes
Overhead	Medium	Medium	Medium	High	Low

reveal the rapid adaptivity of DISCCA to significant change in network dynamics, even in the presence of failures.

Table I summarizes the comparison between existing solutions and our solution. The rest of the paper is organized as follows. Section II presents the game-theoretic problem formulation and proves the game is a weighted potential game. In Section III we propose DISCCA, a low-complexity distributed algorithm to decide chaining that achieves near-optimal latency and effectively mitigates congestion. Next, Section IV shows our simulation results and analysis, while Section V concludes this paper. Due to page limits, all proofs are relegated to our technical report [13].

II. PROBLEM FORMULATION

A. Basic Settings

We consider a NFV system owned by a Telco Operator that provides network services to its users [4]. Figure 1 shows such a sample NFV system. Inside the system are a cluster of servers $\mathcal{V}$, with a size of $V \triangleq |\mathcal{V}|$. Servers are able to host virtual network function (VNF) instances on virtual machines (VMs). Each server $v \in \mathcal{V}$ is equipped with a fixed set of VNFs, denoted by $\mathcal{F}_v$. Each VNF $f \in \mathcal{F}_v$ has only one instance on server v, which is deployed in a standalone VM. We use pair (v, f) to denote the instance of VNF f on server v. Following this notation, we use $\mathcal{R}$ to denote the set of all VNF instances (VMs)[1] in this system. For each $r = (v, f) \in \mathcal{R}$, we denote v by r_1 and f by r_2. Note that our model can be directly extended to the case where each VNF may have more than one instance on a server.

In practice, a virtual machine may break down unexpectedly due to various reasons [**?**]. Thus for VM $r \in \mathcal{R}$, we define its failure probability as

$$\eta_r \triangleq \Pr\{\text{VM } r \text{ survives}\}, \bar{\eta}_r \triangleq \Pr\{r \text{ fails}\} = 1 - \eta_r, \quad (1)$$

which can be estimated by analyzing historical data or real-time monitoring in practice.

Besides servers, the system also includes a set of gateway routers $\mathcal{S}$ that manage the communication with external networks. For any two nodes $u_1, u_2 \in \mathcal{S} \cup \mathcal{R}$, l_{u_1,u_2} denotes the latency between them.[2] Outside the system, there is a group of users $\mathcal{N}$ that subscribe different NSs, with a total size of $N \triangleq |\mathcal{N}|$. Each user $i \in \mathcal{N}$ generates a request traffic with a rate of λ_i. The traffic flows into the system through some gateway router $s_i \in \mathcal{S}$, and leaves from another router $t_i \in \mathcal{S}$. Due to uncertain network conditions, a user may lose its connection [7]. For user $i \in \mathcal{N}$, we define its probability of unavailability by

$$\theta_i \triangleq \Pr\{\text{User } i \text{ survives}\}, \ \bar{\theta}_i \triangleq \Pr\{i \text{ fails}\} = 1 - \theta_i. \quad (2)$$

Each user i subscribes a specific network service (NS) $\mathcal{F}_i$, denoted by an ordered VNF chain $\{f_{i,1}, f_{i,2}, \cdots, f_{i,|\mathcal{F}_i|}\}$ of length $F_i \triangleq |\mathcal{F}_i|$. All required VNFs are assumed available in the system, *i.e.*, $\bigcup_{i\in\mathcal{N}} \mathcal{F}_i \subseteq \bigcup_{v\in\mathcal{V}} \mathcal{F}_v$. For each user's request $\mathcal{F}_i$, we use $\mathbf{w}_i$ to denote the service chaining decision. Note that $\mathrm{w}_{i,j} = r$ indicates that user i's jth VNF $f_{i,j}$ will be assigned onto VM r to execute. Note that a VNF f can only be assigned to those VMs that run its instances, *i.e.*, $\mathrm{w}_{i,j} \in \{r \in \mathcal{R} \mid r_2 = f_{i,j}\}$. For simplicity, we use $\mathbf{w}_{-i}$ to denote the service chaining decisions of all others except user i, *i.e.*, $\mathbf{w}_{-i} \triangleq \{\mathbf{w}_k \mid k \in \mathcal{N}/\{i\}\}$, and denote $\mathbf{w} \triangleq (\mathbf{w}_i, \mathbf{w}_{-i})$.

In this paper, we assume all VMs (users) have the same survival probability. Specifically, we set

$$\eta_r = \eta, \ \forall r \in \mathcal{R}; \quad \theta_i = \theta, \ \forall i \in \mathcal{N}. \quad (3)$$

Nonetheless, our model can be a basis for system designers to consider more complex scenarios with heterogeneous survival probabilities [11]. Table II summarizes the main notations.

B. User Costs

We consider the costs incurred by individual users in terms of failures, latency, and the congestion they experience. In particular, for each user i,

- ◇ If user i fails (with probability $\bar{\theta}$), it will receive an incompletion cost (penalty), denoted by Ω_i.
- ◇ If user i survives, but at least one of its selected VMs fails, it will also receive an incompletion cost Ω_i. Such a case occurs with probability $\theta \cdot (1 - \eta^{F_i})$.
- ◇ Given that both user i and all the chosen VMs survive, which occurs with probability $\theta \cdot \eta^{F_i}$, the cost function consists of two parts.

  Latency: Given decision $\mathbf{w}_i$, we define the total latency experienced by user i as

$$c_i^{(\mathrm{L})}(\mathbf{w}_i) \triangleq l_{s_i,\mathrm{w}_{i,1}} + \sum_{j=2}^{F_i} l_{\mathrm{w}_{i,j-1},\mathrm{w}_{i,j}} + l_{\mathrm{w}_{i,F_i},t_i}. \quad (4)$$

  Congestion: Since traffic flows of different users may share the same instance on the same server, the congestion due to their resource contention becomes inevitable. Given all users' service chaining decisions $\mathbf{w}$, we define the set of users that choose r as

$$\mathcal{M}_r(\mathbf{w}) \triangleq \{k | k \in \mathcal{N}, r \in \mathbf{w}_k\}. \quad (5)$$

[1] In this paper, we use the terms VNF instance and VM interchangeably.

[2] The latency can be approximated by the number of hops, round-trip time, *etc*.

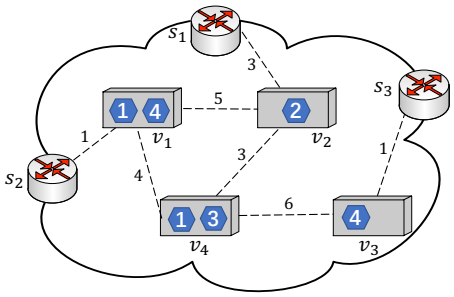


Fig. 1. A sample NFV system with four servers $\mathcal{V} = \{v_1, v_2, v_3, v_4\}$ and three gateway routers $\mathcal{S} = \{s_1, s_2, s_3\}$. The blue hexagons are VNF instances (VMs) running on servers, *e. g.* $\mathcal{F}_{v_1} = \{1, 4\}$.

Given that user i is available, we define the workload on VM r chosen by i as

$$\delta_{i,r}(\mathbf{w}) \triangleq \lambda_i \ + \sum_{k \in \mathcal{M}_r(\mathbf{w})/\{i\}} \theta\lambda_k. \tag{6}$$

Then the congestion cost of user i is

$$c_i^{(\mathrm{C})}(\mathbf{w}) \triangleq \sum_{r \in \mathbf{w}_i} \delta_{i,r}(\mathbf{w}). \tag{7}$$

Combining the three cases above, the expected cost of user i is defined as

$$\begin{aligned} c_i(\mathbf{w}_i, \mathbf{w}_{-i}) \triangleq \bar{\theta} \cdot \Omega_i + \theta \cdot \left(1 - \eta^{F_i}\right) \cdot \Omega_i \\ + \theta \cdot \eta^{F_i} \cdot \left(\alpha c_i^{(\mathrm{L})}(\mathbf{w}_i) + c_i^{(\mathrm{C})}(\mathbf{w}_i, \mathbf{w}_{-i})\right). \end{aligned} \tag{8}$$

where α is a positive weighting parameter.

C. Game-Theoretic Formulation

In this section, we turn to reformulating the chain composition problem from a game-theoretic perspective. We consider a non-cooperative game among a set of players $\mathcal{N}$.[3] The VNF instances $\mathcal{R}$ are actually the resources shared and competed among players. Each player i picks an action, *i.e.* a chaining decision $\mathbf{w}_i$, from a finite action space $\mathcal{W}_i$, including all possible service chaining decisions of player i. In general, we use $\mathcal{W} \triangleq \bigcup_{i \in \mathcal{N}} \mathcal{W}_i$ to denote the joint action space of all players. The cost function of player i is defined as in (8). We define the corresponding game $\mathcal{G}$ by the following triple:

$$\mathcal{G} \triangleq (\mathcal{N}, \mathcal{W}, (c_i)_{i \in \mathcal{N}}). \tag{9}$$

By exploiting the problem's unique structure, we have the following theorem.

Theorem 1: The game $\mathcal{G}$ is a weighted potential game with potential function

$$\Phi(\mathbf{w}) = 2\alpha\theta \sum_{k \in \mathcal{N}} \lambda_k \cdot c_k^{(\mathrm{L})}(\mathbf{w}_k) + \sum_{r \in \mathcal{R}} \left(\sum_{k \in \mathcal{M}_r(\mathbf{w})} \theta \cdot \lambda_k \right)^2, \tag{10}$$

such that for any state transition $\mathbf{w} = (\mathbf{w}_i, \mathbf{w}_{-i})$ and $\mathbf{w}' = (\mathbf{w}_i', \mathbf{w}_{-i})$ that involves only player i decision change,

$$\Phi(\mathbf{w}) - \Phi(\mathbf{w}') = \frac{2\lambda_i}{\eta^{F_i}} \left[c_i(\mathbf{w}) - c_i(\mathbf{w}')\right]. \tag{11}$$

[3]We view each user as a player in the game-theoretic model.

Proof 1: From the definition of Φ in (10), we derive that for any two states $\mathbf{w}$ and $\mathbf{w}'$ that only differs in player i's action:

$$\begin{aligned} \Phi(\mathbf{w}) - \Phi(\mathbf{w}') &= 2\theta\lambda_i \left[\alpha c_i^{(\mathrm{L})}(\mathbf{w}_i) - \alpha c_i^{(\mathrm{L})}(\mathbf{w}_i')\right] \\ &+ \sum_{r \in \mathcal{R}} \left[\left(\sum_{k \in \mathcal{M}_r(\mathbf{w})} \theta\lambda_k \right)^2 - \left(\sum_{k \in \mathcal{M}_r(\mathbf{w}')} \theta\lambda_k \right)^2 \right]. \end{aligned} \tag{12}$$

Let's focus on the second summation term on the right-hand side. We divide the set of VMs $\mathcal{R}$ into three subsets.

- $\mathcal{R}_1$: VMs adopted by player i in $\mathbf{w}_i$ but not in $\mathbf{w}_i'$;
- $\mathcal{R}_2$: VMs adopted in decision $\mathbf{w}_i'$ but not in $\mathbf{w}_i$;
- $\mathcal{R}_3$: VMs that are both adopted in $\mathbf{w}_i$ and $\mathbf{w}_i'$ or neither.

Note that $|\mathcal{R}_1| = |\mathcal{R}_2|$ because for any VM r such that $r \in \mathbf{w}_i, r \notin \mathbf{w}_i'$, there must exist another VM r' such that $r' \in \mathbf{w}_i', r' \notin \mathbf{w}_i$. Then,

$$\begin{aligned} &\sum_{r \in \mathcal{R}} \left[\left(\sum_{k \in \mathcal{M}_r(\mathbf{w})} \theta\lambda_k \right)^2 - \left(\sum_{k \in \mathcal{M}_r(\mathbf{w}')} \theta\lambda_k \right)^2 \right] \\ &= \sum_{r \in \mathcal{R}_1} \left[\left(\sum_{k \in \mathcal{M}_r(\mathbf{w})} \theta\lambda_k \right)^2 - \left(\sum_{k \in \mathcal{M}_r(\mathbf{w})} \theta\lambda_k - \theta\lambda_i \right)^2 \right] \\ &+ \sum_{r \in \mathcal{R}_2} \left[\left(\sum_{k \in \mathcal{M}_r(\mathbf{w}')} \theta\lambda_k - \theta\lambda_i \right)^2 - \left(\sum_{k \in \mathcal{M}_r(\mathbf{w}')} \theta\lambda_k \right)^2 \right] \\ &= 2\theta\lambda_i \left[\sum_{r \in \mathcal{R}_1} \left(\sum_{k \in \mathcal{M}_r(\mathbf{w})} \theta\lambda_k \right) - \sum_{r \in \mathcal{R}_2} \left(\sum_{k \in \mathcal{M}_r(\mathbf{w}')} \theta\lambda_k \right) \right] \\ &= 2\theta\lambda_i \left[\sum_{r \in \mathcal{R}_1} \delta_{i,r}(\mathbf{w}) - \sum_{r \in \mathcal{R}_2} \delta_{i,r}(\mathbf{w}) \right] \\ &= 2\theta\lambda_i \left(c_i^{(\mathrm{C})}(\mathbf{w}) - c_i^{(\mathrm{C})}(\mathbf{w}') \right). \end{aligned} \tag{13}$$

Recalling (8), we have

$$\begin{aligned} &c_i(\mathbf{w}) - c_i(\mathbf{w}') \\ &= \theta\eta^{F_i} \cdot \left[\alpha c_i^{(\mathrm{L})}(\mathbf{w}_i) - \alpha c_i^{(\mathrm{L})}(\mathbf{w}_i') + c_i^{(\mathrm{C})}(\mathbf{w}) - c_i^{(\mathrm{C})}(\mathbf{w}')\right]. \end{aligned} \tag{14}$$

Combining (12), (13) and (14), we obtain

$$\Phi(\mathbf{w}) - \Phi(\mathbf{w}') = \frac{2\lambda_i}{\eta^{F_i}} \left(c_i(\mathbf{w}) - c_i(\mathbf{w}')\right). \tag{15}$$

Theorem 1 is proved. ■

Zooming into the potential function (10), we see that the first term is the summation of the product of each player's traffic rate and the latency it experiences, which actually represents the overall traffic load in the system. The second term is the square-sum of the workloads of all VMs, which reflects the congestion level of the system. Therefore, we can view the weighted sum of overall traffic load and congestion load, *i.e.* the potential function, as one measurement of the

TABLE II
MAIN NOTATIONS

Notation	Description
$\mathcal{V}$	Set of commodity servers
V	Number of servers, $V \triangleq \lvert\mathcal{V}\rvert$
$\mathcal{F}_v$	Set of VNFs provided by server v
$\mathcal{R}$	Set of VMs (VNF instances) in the system
η_r	Survival probability of VM r
$\bar{\eta}_r$	Failure probability of VM r, $\bar{\eta}_r \triangleq 1-\eta_r$
$\mathcal{S}$	Set of gateway routers
l_{u_1,u_2}	Latency between node u_1 and u_2
$\mathcal{N}$	Set of users
N	Number of users, $N \triangleq \lvert\mathcal{N}\rvert$
θ_i	Survival probability of user i
$\bar{\theta}_i$	Failure probability of user i, $\bar{\theta}_i \triangleq 1-\theta_i$
λ_i	Traffic rate of user i
s_i, t_i	Ingress and egress gateway routers of user i
$\mathcal{F}_i$	Network service request of user i
F_i	Length of user i's request, $F_i \triangleq \lvert\mathcal{F}_i\rvert$
$f_{i,j}$	The *jth* VNF in user i's NS request
Ω_i	Incompletion cost of user i
$\mathbf{w}_i$	Service chaining decision of user i
$\mathrm{w}_{i,j}$	The VM chosen for the *jth* VNF of user i's request
$\mathbf{w}_{-i}$	Service chaining decisions of other users excluding i
$\mathbf{w}$	Combination of all users' decisions

overall system performance, referred as the *social cost*. To attain the best possible service, each player aims to search such an action that reduces its own cost. On the other hand, the Telco operator's goal is to minimize the overall social cost, *i.e.* search the *social optimum*. The weighted potential property of the problem greatly bridges the objectives of individual users and the NFV operator together. To find the social optimum, we formulate the following problem

$$\begin{aligned} \underset{\mathbf{w}}{\text{Minimize}} \quad & \Phi(\mathbf{w}) \\ \text{Subject to} \quad & \mathbf{w} \in \mathcal{W}. \end{aligned} \tag{16}$$

Denote $\mathbf{w}^*$ as the solution of problem (16). We state that $\mathbf{w}^*$ is a Nash Equilibrium (NE) of game $\mathcal{G}$. The reasoning is as follows. Assume that player i intends to change its action from $\mathbf{w}_i^*$ to $\mathbf{w}_i$. Since

$$\Phi(\mathbf{w}_i, \mathbf{w}_{-i}^*) - \Phi(\mathbf{w}_i^*, \mathbf{w}_{-i}^*) \geq 0. \tag{17}$$

according to Theorem 1, we have

$$c_i(\mathbf{w}_i, \mathbf{w}_{-i}^*) - c_i(\mathbf{w}_i^*, \mathbf{w}_{-i}^*) \geq 0, \tag{18}$$

which indicates that player i will not attain any benefits by changing its action from $\mathbf{w}_i^*$ to any other. Therefore, $\mathbf{w}^*$ is a NE because no player has the incentive through its unilateral action change. In the following, we refer the social optimum $\mathbf{w}^*$ as the optimal NE of game $\mathcal{G}$.

III. ALGORITHM DESIGN

A. Problem Transformation

Problem (16) is essentially a combinatorial problem which is in general $\mathcal{NP}$-hard. By applying *log-sum-exponential* approximation [12], we transform problem (16) into the following form:

$$\begin{aligned} \underset{\pi \succeq 0}{\text{Minimize}} \quad & \sum_{\mathbf{w}\in\mathcal{W}} \pi_{\mathbf{w}} \Phi(\mathbf{w}) + \frac{1}{\beta} \sum_{\mathbf{w}\in\mathcal{W}} \pi_{\mathbf{w}} \log \pi_{\mathbf{w}} \\ \text{Subject to} \quad & \sum_{\mathbf{w}\in\mathcal{W}} \pi_{\mathbf{w}} = 1. \end{aligned} \tag{19}$$

where β is a positive tunable parameter.

Theorem 2: The optimality gap incurred by the approximation is upper bounded by $\frac{1}{\beta} N \log V^F$ where $F \triangleq \max_i F_i$ is the length of the longest SC request among all players.

Proof 2: To be done... ■

Problem (19) can also be viewed as the problem of deciding the time proportion $\pi_{\mathbf{w}}$ that the system spends on each state $\mathbf{w}$, in a bid to minimize the time-average overall cost Φ over a period of time with an entropy term. By solving the KKT conditions of problem (19), we obtain the optimal probability distribution as

$$\pi_{\mathbf{w}}^* = \frac{\exp[-\beta\Phi(\mathbf{w})]}{\sum_{\mathbf{w}'\in\mathcal{W}} \exp[-\beta\Phi(\mathbf{w}')]}, \ \forall \ \mathbf{w} \in \mathcal{W}. \tag{20}$$

B. Markov Chain Design

It is impractical to solve the optimal solution (20) directly since that requires the full knowledge of states in the enormous search space $\mathcal{W}$. Instead, we resort to designing a discrete-time Markov chain while adopting random sampling techniques to compute $\pi_{\mathbf{w}}^*$ in an asymptotic manner. In general, we have two degrees of freedom in designing the Markov chain.

1) Topology: We construct a Markov chain with its state space as $\mathcal{W}$. Each state in the chain corresponds to one possible combination of all players' actions, *i.e.*, $\mathbf{w} \in \mathcal{W}$. We define $H_{\mathbf{w},\mathbf{w}'}$ as the Hamming distance between the two states $\mathbf{w}$ and $\mathbf{w}'$.

$$H_{\mathbf{w},\mathbf{w}'} \triangleq \sum_{i\in\mathcal{N}} \mathbf{1}_{\mathbf{w}_i \neq \mathbf{w}_i'}, \tag{21}$$

where $\mathbf{1}_A$ is an indicator function such that $\mathbf{1}_A = 1$ if event A occurs and 0 otherwise. In the topology, two states are connected if their Hamming distance $H_{\mathbf{w},\mathbf{w}'} \leq 1$, *i.e.*, at most one player changes its action.

To design a time-reversible Markov chain, we should first ensure the connectivity of the topology. Actually, the Hamming distance $H_{\mathbf{w},\mathbf{w}'}$ equals the number of steps needed to reach state $\mathbf{w}'$ from state $\mathbf{w}$. From the definition of Hamming distance in (21), we conclude that $H_{\mathbf{w},\mathbf{w}'} \leq N$, because there are at most N differences in action change. Therefore, any state pairs in the topology are reachable within at most N transitions, the topology is connected.

2) Transition probability: We define the self transition probability with respect to state $\mathbf{w}$ as:

$$p_{\mathbf{w},\mathbf{w}} \triangleq \frac{1}{N} \sum_{i\in\mathcal{N}} \frac{\exp[-\beta\Phi(\mathbf{w})]}{\sum_{\bar{\mathbf{w}}\in\mathcal{A}_i} \exp[-\beta\Phi(\bar{\mathbf{w}})]}, \tag{22}$$

where $\mathcal{A}_i$ is a subset of the state space that only player i's action is changed while others' remain unchanged, *i.e.* $\mathcal{A}_i = \{(\bar{\mathbf{w}}_i, \mathbf{w}_{-i}) \mid \bar{\mathbf{w}}_i \in \mathcal{W}_i\}$.

Next, we consider two states $\mathbf{w}$ and $\mathbf{w}'$ with $H_{\mathbf{w},\mathbf{w}'}=1$. For simplicity, we use i to denote the only player that changes its action. Then, the transition probability is defined as

$$p_{\mathbf{w},\mathbf{w}'} \triangleq \frac{1}{N}\frac{\exp[-\beta\Phi(\mathbf{w}')]}{\sum_{\bar{\mathbf{w}}\in\mathcal{A}_i}\exp[-\beta\Phi(\bar{\mathbf{w}})]}. \tag{23}$$

The transition probability is zero for any other state pair. With the above design, we have the following theorem.

Theorem 3: The resultant Markov chain is time-reversible with its stationary distribution as (20).

Proof 3: We have shown that the topology is connected in the above, which means the Markov chain is irreducible. Next we prove the following two things.

- ◇ Transition probabilities out of one state sum to 1.
  For any state $\mathbf{w}\in\mathcal{W}$,

$$\begin{aligned}
&\sum_{\mathbf{w}'\in\mathcal{W}} p_{\mathbf{w},\mathbf{w}'} = \sum_{\substack{\mathbf{w}'\in\mathcal{W}\\ H_{\mathbf{w},\mathbf{w}'}=0}} p_{\mathbf{w},\mathbf{w}'} + \sum_{\substack{\mathbf{w}'\in\mathcal{W}\\ H_{\mathbf{w},\mathbf{w}'}=1}} p_{\mathbf{w},\mathbf{w}'}\\
&= p_{\mathbf{w},\mathbf{w}} + \sum_{i\in\mathcal{N}}\sum_{\substack{\hat{\mathbf{w}}\in\mathcal{A}_i\\ \hat{\mathbf{w}}\neq\mathbf{w}}} p_{\mathbf{w},\hat{\mathbf{w}}}\\
&= \frac{1}{N}\sum_{i\in\mathcal{N}}\frac{\exp[-\beta\Phi(\mathbf{w})]}{\sum_{\bar{\mathbf{w}}\in\mathcal{A}_i}\exp[-\beta\Phi(\bar{\mathbf{w}})]}\\
&\quad + \sum_{i\in\mathcal{N}}\sum_{\substack{\hat{\mathbf{w}}\in\mathcal{A}_i\\ \hat{\mathbf{w}}\neq\mathbf{w}}}\frac{1}{N}\frac{\exp[-\beta\Phi(\hat{\mathbf{w}})]}{\sum_{\bar{\mathbf{w}}\in\mathcal{A}_i}\exp[-\beta\Phi(\bar{\mathbf{w}})]}\\
&= \frac{1}{N}\sum_{i\in\mathcal{N}}\frac{\sum_{\hat{\mathbf{w}}\in\mathcal{A}_i}\exp[-\beta\Phi(\hat{\mathbf{w}})]}{\sum_{\bar{\mathbf{w}}\in\mathcal{A}_i}\exp[-\beta\Phi(\bar{\mathbf{w}})]} = 1.
\end{aligned} \tag{24}$$

- ◇ Transition probabilities between any two states should satisfy the detailed balance equation.
  The detailed balance equation is naturally satisfied for state self transition. For any two different connected states $\mathbf{w}$ and $\mathbf{w}'$, recalling (20) and (23), we have

$$\begin{aligned}
&\pi^*_{\mathbf{w}}p_{\mathbf{w},\mathbf{w}'}\\
=&\frac{\exp[-\beta\Phi(\mathbf{w})]}{\sum_{\bar{\mathbf{w}}\in\mathcal{W}}\exp[-\beta\Phi(\bar{\mathbf{w}})]}\cdot\frac{1}{N}\frac{\exp[-\beta\Phi(\mathbf{w}')]}{\sum_{\bar{\mathbf{w}}\in\mathcal{A}_i}\exp[-\beta\Phi(\bar{\mathbf{w}})]}\\
=&\frac{\exp[-\beta\Phi(\mathbf{w}')]}{\sum_{\bar{\mathbf{w}}\in\mathcal{W}}\exp[-\beta\Phi(\bar{\mathbf{w}})]}\cdot\frac{1}{N}\frac{\exp[-\beta\Phi(\mathbf{w})]}{\sum_{\bar{\mathbf{w}}\in\mathcal{A}_i}\exp[-\beta\Phi(\bar{\mathbf{w}})]}\\
=&\pi^*_{\mathbf{w}'}p_{\mathbf{w}',\mathbf{w}}.
\end{aligned} \tag{25}$$

For now we have proven that the Markov chain is irreducible with finite states ($|\mathcal{W}|$), it is also aperiodic because it adopts a self transition. Further we have also proven the detailed balance equation. Therefore, we claim that the resultant Markov chain is time-reversible with its stationary distribution as (20) ■

C. Algorithm Design

We implement the discrete-time Markov chain in the following way. During each iteration, a player is chosen uniformly randomly among N players, with the chance to update its action. Once player i is chosen, the in-step transition probability from state $\mathbf{w}=(\mathbf{w}_i,\mathbf{w}_{-i})$ to $\mathbf{w}'=(\mathbf{w}'_i,\mathbf{w}_{-i})$ is defined as $p^{(i)}_{\mathbf{w},\mathbf{w}'}$. Note that $\mathbf{w}'_i$ may equal $\mathbf{w}_i$, indicating a self transition.

$$p^{(i)}_{\mathbf{w},\mathbf{w}'} \triangleq \frac{\exp[-\beta\Phi(\mathbf{w}')]}{\sum_{\bar{\mathbf{w}}\in\mathcal{A}_i}\exp[-\beta\Phi(\bar{\mathbf{w}})]}. \tag{26}$$

The transition probabilities for other state pairs are all zeros.

To implement the player selection in a distributed way, we equip each player with a Poisson countdown clock with rate μ independently. A player is chosen once its clock expires before others' do, while the other players are notified to suspend its clock current iteration. In this way, we have:

- ◇ The probability that any one of the N players is chosen within an iteration is $1/N$.
- ◇ The probability that more than one players' clocks expire at the same time is 0.
- ◇ By setting $\mu\gg 1$, only a small fraction of time in each iteration is used to decide the selected player.

On the other hand, it is impractical to calculate the transition probabilities in (26), because often times the potential values are hard to obtain in real NFV systems. To address the challenge, we bridge the potential and cost function by defining a function $B_i:\mathcal{W}_{-i}\to\mathbb{R}$ as

$$\begin{aligned}
B_i(\mathbf{w}_{-i}) = 2\alpha\theta\sum_{k\in\mathcal{N}/\{i\}}\lambda_k c_k^{(\mathrm{L})}(\mathbf{w}_k) - F_i\theta^2\lambda_i^2\\
+\sum_{r\in\mathcal{R}}\left(\sum_{k\in\mathcal{M}_r(\mathbf{w})/\{i\}}\theta\cdot\lambda_k\right)^2.
\end{aligned} \tag{27}$$

which is unrelated to $\mathbf{w}_i$, the action of player i. Then we can rewrite the potential function in (10) as

$$\Phi(\mathbf{w}) = 2\theta\lambda_i c_i(\mathbf{w}) + B_i(\mathbf{w}_{-i}), \tag{28}$$

Substituting (28) into (26) and applying the definition of $\mathcal{A}_i$, the transition probability can be reformulated as

$$p^{(i)}_{(\mathbf{w}_i,\mathbf{w}_{-i}),(\mathbf{w}'_i,\mathbf{w}_{-i})} = \frac{\exp[-2\beta\theta\lambda_i c_i(\mathbf{w}'_i,\mathbf{w}_{-i})]}{\sum_{\bar{\mathbf{w}}_i\in\mathcal{W}_i}\exp[-2\beta\theta\lambda_i c_i(\bar{\mathbf{w}}_i,\mathbf{w}_{-i})]}. \tag{29}$$

Defined in such a way, the transition probability only depends on each player's cost function, which requires only local information from individual player.

Based on the above design, we propose an algorithm named DIstribute Service Chain Composition Algorithm (DISCCA). DISCCA solves the problem (19) and hence solves the initial problem (16) in an approximate way. We show the pseudo-code of DISCCA in **Algorithm 1**.

Remarks:

- ◇ DISCCA requires only local information for the decision making by individual players; therefore, it can run in a distributed manner.
- ◇ If we use one bit 0 to represent the *SUSPEND* signal and 1 to represent the *RESET* signal, then only two bits are broadcast within each iteration.
- ◇ By setting the survival probability η and θ as 1, the algorithm degrades to the scenario without failures.

Algorithm 1 DIstributed Service Chain Composition Algorithm (DISCCA) of player i

Input: At the beginning of each iteration, a *RESET* signal is broadcast to all players.

Repeat do
  if Player i receives a signal *SUSPEND* **then**
    Suspend the clock
  end if
  if Player i receives a signal *RESET* **then**
    Reset the clock with rate μ
    Begin count down again
  end if
  if Player i's clock expires **then**
    Broadcast a signal *SUSPEND*
    Perform action $\mathbf{w}_i'$ chosen from action space $\mathcal{W}_i$ with probability $p^{(i)}_{(\mathbf{w}_i,\mathbf{w}_{-i}),(\mathbf{w}_i',\mathbf{w}_{-i})}$ in (29)
  end if

- ◇ According to theorem 2, we can reduce the approximation gap by increasing β; however, that also leads to a longer convergence time to the optimal distribution [12]. In practice, the tradeoff is often left to the system designers to make.

IV. Numerical Results

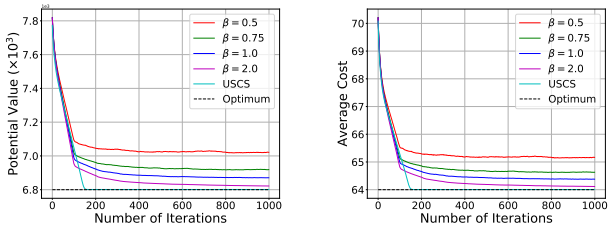
In this part, we evaluate DISCCA under various simulation scenarios. We assume there are $N = 100$ players, with an average traffic rate of 1Mbps. Besides, there is a cluster of 5 servers, each equipped with three VNFs, *e.g.* Firewall (FW), Load Balancer (LB), and Intrusion Detection System (IDS). The survival probabilities η and θ are set as 0.9 by default. The latency between any gateway router $s \in \mathcal{S}$ and VM $r \in \mathcal{R}$, and the latency between two VMs within the same server are set as 1. Other latency values are sampled randomly and uniformly from the interval $[2, 6]$. We assume all players require the same network service, *i.e.*, FW $\rightarrow$ LB $\rightarrow$ IDS. The parameters α and β are set as 1 by default. In our simulation, the clock rate μ is set as 10, and the length of an iteration is 1s. All the results in the following are obtained by averaging over 100 runs.

A. Performance

To evaluate DISCCA, we first investigate how the potential value and the average cost per player change over iterations. We define the average cost per player (hereinafter shorten as average cost) as

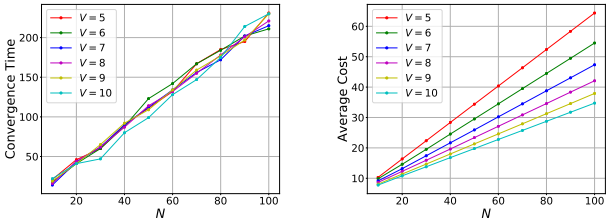
$$\bar{C} = \frac{1}{N}\sum_{i\in\mathcal{N}} c_i(\mathbf{w}_i, \mathbf{w}_{-i}). \tag{30}$$

In addition to DISCCA, we also implement another existing scheme as the baseline, *i.e.*, the Unilateral Service Chain Selection (USCS) algorithm [4]. Note that USCS is also a distributed algorithm but requires large amounts of overheads due to network state broadcasting per iteration. Figure 2 shows the performance of DISCCA in terms of (a) potential value and (b) average cost with different values of parameter



(a) Potential value. (b) Per-player average cost.

Fig. 2. Comparison of USCS and DISCCA under various values of β in terms of (a) potential value and (b) per-player average cost.



(a) Convergence time. (b) Per-player average cost.

Fig. 3. Performance of DISCCA under different settings of N and V in terms of (a) convergence time and (b) per-player average cost.

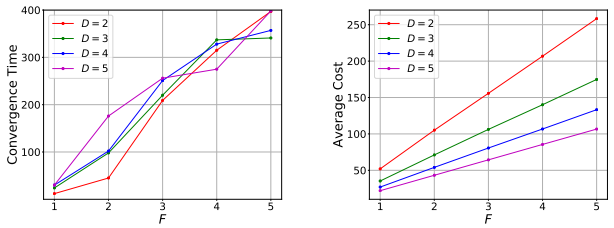
β as well as the corresponding results of USCS. The black dotted lines present the corresponding optimal values (6800 and 64). We make the following observations:

i) USCS achieves a near-optimal cost, at the notable cost of broadcasting large amount of data per iteration.
ii) Both potential value and average cost decreases gradually and levels off as the number of iterations increases, indicating the convergence of DISCCA.
iii) The values after convergence of both potential value and average cost decrease as β increases from 0.5 to 2. This is consistent with our previous analysis: larger β makes the Markov chain stays at low-potential (low-cost) states with higher probabilities, hence incurs lower potential value (average cost).
iv) With $\beta = 2$, the curve approximates the optimal value closely within small number of iterations. For example, in Figure (b), the average cost at iteration 400 is 64.22, which is only 0.34% higher than the optimal value 64.

B. Scalability

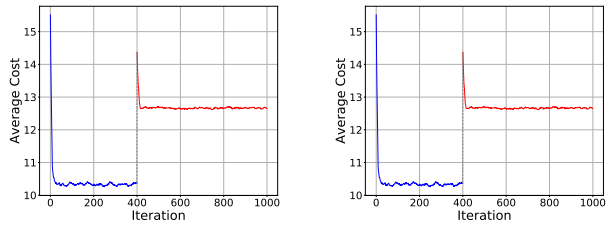
An algorithm should be scalable to be deployed in practical, large-scale systems. In this part, we investigate how the following factors affect the scalability of DISCCA: the number of players N, the number of servers V, the length of service chain, and the number of instances for each VNF. In this simulation, we assume all players have the same length of service chain, denoted by F. Each VNF is assumed to have the same number of instances, denoted by D.

Figure 3(a) shows how parameters N and V affect the convergence time of DISCCA. We say the process converges if the sum of n successive potential differences is less than a small value ϵ. In fact, we set $n = 5$ and $\epsilon = 1$. From Figure 3(a), we make the following two observations:



(a) Convergence time. (b) Average cost.

Fig. 4. Performance of DISCCA under different settings of F and D in terms of (a) convergence time and (b) per-player average cost.



(a) Player dynamics. (b) VM dynamics.

Fig. 5. Average cost v.s. iteration with (a) player and (b) VM dynamics.

i) The increase in convergence time is almost linearly proportional to the number of players N.
ii) Increasing the number of servers nearly has no effect on the convergence of DISCCA.

Figure 3(b) presents how parameters N and V affect the average cost. We have the following observations:

i) The average cost increases as N increases from 10 to 100. This is because with a larger N, each individual will compete with more players, thereby suffering a higher congestion cost.
ii) The average cost decreases as V grows from 5 to 10. In fact, with more servers, players are able to be distributed more separately, therefore the congestion cost is reduced.

Figure 4(a) shows how F and D affect the convergence time. We observe that the convergence time increases as F almost linearly, while parameter D has little effect. Figure 4(b) presents the average cost under different values of F and D. The average cost goes up as the service chain goes longer. The reason is that with a longer SC, the player experiences larger latency cost as well as higher congestion level. We also observe that with the same F, increasing the number of VNF replications D helps reduce the average cost. The reason is similar as increasing V, both cases bring more resources and lower the congestion cost.

C. Adaptivity

In practice, users often join and leave NFV systems dynamically [14]. Moreover, a VM may be offline due some various reasons and restart again after a while. Therefore, we should investigate the adaptivity of DISCCA in response to such network dynamics.

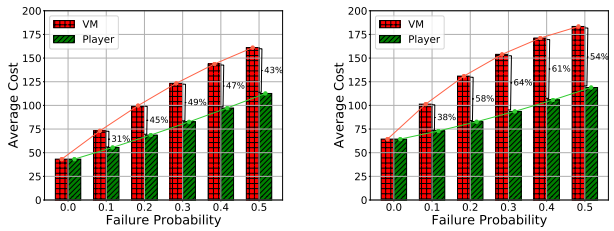
We run the simulation within 1000 iterations, and a new player with rate 5Mbps will join at the 400th iteration. Once the player joins, she picks an action randomly and starts an exponential count-down clock of rate μ. Figure 5(a) presents the performance of DISCCA in such case with $N = 10$. We could observe that when the new player joins, the per-player average cost increases immediately and sharply. This is because the arrival of the new player increases the congestion costs of some players. But within a short period (less than ten iterations) the cost decreases and converges again. Results from Figure 5(a) shows the rapid adaptivity of DISCCA to player dynamics.

We also consider potential VM failures in the simulation. If one VM breaks down, all the tasks will be migrated to another randomly selected VM of the same type. During each simulation, we set the Firewall instance on the first server to terminate at 400th iteration. Figure 5(b) shows that when the VM is down, the cost rises up immediately because the random migration may cause imbalanced player distribution and incur high congestion costs. From the figure, we notice that after the sharp increment, the cost converges within ten iterations again, verifying the rapid adaptivity of DISCCA to VM dynamics.

Combining Figure 5(a) and 5(b) together, we conclude that DISCCA algorithm reacts to both player and VM dynamics in an adaptive manner.

TABLE III
AVERAGE COST UNDER DIFFERENT $\bar{\eta}$ AND $\bar{\theta}$ WITH $F = 2$.

$\bar{\eta}$ \ $\bar{\theta}$	0	0.1	0.2	0.3	0.4	0.5
0	43.2	73.1	99.8	123.4	143.8	161.1
0.1	55.6	83.1	107.7	129.5	148.3	164.1
0.2	68.7	93.7	116.1	135.9	153	167.4
0.3	82.6	105	125	142.7	157.9	170.8
0.4	97.2	116.8	134.4	149.8	163.2	174.4
0.5	112.6	129.3	144.2	157.3	168.6	178.2



(a) $F = 2$. (b) $F = 3$.

Fig. 6. Average cost v.s. probability of player (green stripped bar) and VM (red meshed bar) unavailability under different settings of SC length F.

D. DISCCA with Failures

As mentioned before, players and VMs may fail with some positive probability. In this part we investigate how the failure probability affect the performance of DISCCA. Table III shows that the average cost under different settings of $\bar{\eta}$ and $\bar{\theta}$ while the length of SC is set as 2. From this table, we observe that the average cost increases from left to right, top to bottom. In summary, larger failure probability (whether player or VM) incurs a larger average cost.

In Figure 6(a), we present the first row and first column of Table III. The red meshed bars present the case when $\bar{\theta}$

increases from 0 to 0.5 and $\bar{\eta}$ is default as 0 (the first row). Similarly the green stripped bars present the case when $\bar{\eta}$ ranges from 0 to 0.5 and $\bar{\theta} = 0$ (the first column). Figure 6(b) presents similar results but under the setting with $F = 3$. We make the following observations:

i) Higher failure probability leads to higher average costs.
ii) The red meshed bars are always higher than the green stripped bars when failure probability ranges from 0.1 to 0.5. In other words, with the same probability value, VM failure has a greater impact on the average cost than player failure. The reason is that multiple VMs are employed to process a player's request, magnifying the impact of VM failures.
iii) With a fixed failure probability (*e.g.*, 0.2), the difference between the two bars in Figure 6(b) (58%) is larger than that in Figure 6(a) (45%). The reason is that the impact of VM failures expands further as the service chain length grows.

V. Conclusion

In this paper, we studied the service chain composition in NFV systems. By considering both user and resource failures in our model, we formulated the chain composition process as a non-cooperative game. We showed the game is a weighted potential game. By leveraging Markov approximation framework, we proposed DISCCA, a distributed and low-complexity algorithm with performance guarantee that guides the system towards the optimal NE of the game through distributed decision making by individual users with only local information. Extensive simulation results showed that DISCCA achieves the social optimum within mild-value of iterations, in terms of both short latency and low congestion, while adapting to network dynamics rapidly, even in the presence of failures.

References

[1] M. F. Bari, S. R. Chowdhury, R. Ahmed, and R. Boutaba, "On orchestrating virtual network functions," in *International Conference on Network and Service Management (CNSM)*. IEEE, 2015, pp. 50–56.

[2] M. T. Beck and J. F. Botero, "Scalable and coordinated allocation of service function chains," *Computer Communications*, vol. 102, pp. 78–88, 2017.

[3] Q. Zhang, Y. Xiao, F. Liu, J. C. Lui, J. Guo, and T. Wang, "Joint optimization of chain placement and request scheduling for network function virtualization," in *International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2017, pp. 731–741.

[4] S. DOro, L. Galluccio, S. Palazzo, and G. Schembra, "Exploiting congestion games to achieve distributed service chaining in nfv networks," *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 2, pp. 407–420, 2017.

[5] P. Quinn and T. Nadeau, "Problem statement for service function chaining," Tech. Rep., 2015.

[6] R. Cohen, L. Lewin-Eytan, J. S. Naor, and D. Raz, "Near optimal placement of virtual network functions," in *International Conference on Computer Communications (INFOCOM)*. IEEE, 2015, pp. 1346–1354.

[7] P. Gill, N. Jain, and N. Nagappan, "Understanding network failures in data centers: measurement, analysis, and implications," in *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 4. ACM, 2011, pp. 350–361.

[8] D. Cotroneo, L. De Simone, A. K. Iannillo, A. Lanzaro, R. Natella, J. Fan, and W. Ping, "Network function virtualization: Challenges and directions for reliability assurance," in *International Symposium on Software Reliability Engineering Workshops (ISSREW)*. IEEE, 2014, pp. 37–42.

[9] M. Penn, M. Polukarov, and M. Tennenholtz, "Congestion games with failures," in *Proceedings of the 6th ACM Conference on Electronic Commerce*. ACM, 2005, pp. 259–268.

[10] R. Meir, M. Tennenholtz, Y. Bachrach, and P. Key, "Congestion games with agent failures." in *AAAI*, vol. 12, 2012, pp. 1401–1407.

[11] Y. Li, Y. Jia, H. Tan, R. Wang, Z. Han, and F. C. M. Lau, "Congestion game with agent and resource failures," *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 3, pp. 764–778, 2017.

[12] M. Chen, S. C. Liew, Z. Shao, and C. Kai, "Markov approximation for combinatorial network optimization," *IEEE Transactions on Information Theory*, vol. 59, no. 10, pp. 6301–6327, 2013.

[13] S. Bian, X. Huang, Z. Shao, and Y. Yang, "Reliable service chain composition in nfv systems: A game perspective," Tech. Rep. [Online]. Available: https://github.com/SimengBian/ICC19TechReport/blob/master/submission.pdf

[14] D. Shah and J. Shin, "Dynamics in congestion games," *ACM SIGMETRICS Performance Evaluation Review*, vol. 38, no. 1, pp. 107–118, 2010.