DPoS: Decentralized, Privacy-Preserving, and Low-Complexity Online Slicing for Multi-Tenant Networks

Hailiang Zhao, Shuiguang Deng, *Senior Member, IEEE*, Zijie Liu, Zhengzhe Xiang, *Member, IEEE*, Jianwei Yin, Schahram Dustdar, *Fellow, IEEE*, and Albert Y. Zomaya, *Fellow, IEEE*

Abstract—Network slicing is the key to enable virtualized resource sharing among vertical industries in the era of 5G communication. Efficient resource allocation is of vital importance to realize network slicing in real-world business scenarios. To deal with the high algorithm complexity, privacy leakage, and unrealistic offline setting of current network slicing algorithms, in this paper we propose a fully decentralized and low-complexity online algorithm, DPoS, for multi-resource slicing. We first formulate the problem as a global social welfare maximization problem. Next, we design the online algorithm DPoS based on the primal-dual approach and posted price mechanism. In DPoS, each tenant is incentivized to make its own decision based on its true preferences without disclosing any private information to the mobile virtual network operator and other tenants. We provide a rigorous theoretical analysis to show that DPoS has the optimal competitive ratio when the cost function of each resource is linear. Extensive simulation experiments are conducted to evaluate the performance of DPoS. The results show that DPoS can not only achieve close-to-offline-optimal performance, but also have low algorithmic overheads.

Index Terms—Network slicing, decentralized algorithm, posted price mechanism, privacy preserving, multi-tenant networks

1 Introduction

5G creates tremendous opportunities for social digitalization and industrial interconnection. On top of the physical infrastructure, diversified service requirements (eMBB, mMTC, and uRLLC) can be met in the service-oriented end-to-end network slicing (E2E-NS) architecture. The E2E-NS architecture supports both the co-existent accesses of multiple standards (5G, LTE, and Wi-Fi), and the coordination between different site types (macro cell, micro cell, and pico cell base stations), which is mainly attributed to the flexible orchestration and on-demand deployment of virtualized network functions (VNFs) [1], [2], [3].

The substantive characteristics of the E2E-NS architecture is *cloudification*. It involves the transformation from traditional hardbox network functions to the all-on-cloud management & control planes [4]. In this architecture, network slicing is the key to enabling networking capabilities for vertical industries. Many business players, such as infrastructure providers (InPs), mobile network operators (MNOs), cloud providers (one kind of InPs actually), edge & cloud service providers (a.k.a. *tenants*), service subscribers (i.e., *users*), service brokers and mobile virtual network operators (MVNOs) are involved [5], [6]. For the scenario considered in this paper, the InP offers the physical network infrastructure to the MVNO by leasing or selling and is responsible for hardware upgrades and maintenance. After having control of the physical networks, the MVNO virtualizes the network resources, divides each kind of resource into slices, and rents them to the tenants according to tenants' demand. Therewith, each tenant creates service instances based on its slices, and provides services to its subscribers. Normally, the level of services are stated in Service Level Agreements (SLAs). SLAs define the metrics to measure and show if the expected *quality of service* (QoS) is achieved or not. The process[1] is illustrated in Fig. 1.

The key problem underlying network slicing is *efficient resource allocation* for VNFs [7], [8], which is algorithmicaly NP-hard [9]. There has been lots of research done so far for different scenarios, including slicing the radio access networks (RANs) [10], [11], [12], [13], the core networks (5GCs) [14], [15], [16], and the federated edge [17], [18], [19], etc. In these cases, survivability constraints, heterogeneous QoS requirements, geographical limitations, and other *scenario-specific* constraints are taken into considerations to formulate complicated combinatorial non-convex problems. To solve them, the most typical and general class of works are based on fine-tuned heuristics [20] or deep machine learning

- *Hailiang Zhao, Shuiguang Deng, Zijie Liu, and Jianwei Yin are with the College of Computer Science and Technology, Zhejiang University, Hangzhou, Zhejiang 310058, China. E-mail: {hliangzhao, dengsg, liuzijie, zjuyjw}@zju.edu.cn.*
- *Zhengzhe Xiang is with the Zhejiang University City College, Hangzhou, Zhejiang 310015, China. E-mail: xiangzhengzhe@zju.edu.cn.*
- *Schahram Dustdar is with the Distributed Systems Group, TU Wien, 1040, Vienna, Austria. E-mail: dustdar@dsg.tuwien.ac.at.*
- *Albert Y. Zomaya is with the School of Computer Science, University of Sydney, Sydney, NSW 2006, Australia. E-mail: albert.zomaya@sydney.edu.au.*

Manuscript received 3 Jan. 2021; revised 8 Apr. 2021; accepted 18 Apr. 2021. Date of publication 22 Apr. 2021; date of current version 3 Nov. 2022. (corresponding author: Shuiguang Deng.)
Digital Object Identifier no. 10.1109/TMC.2021.3074934

1. To be clear, this is not the only business model in network slicing. In some scenarios, the MNOs are the owners and maintainers of physical resources. They create slices on top of the resources, which will be offered to the MVNOs to perform services to subscribers. For all this, the network slicing model provided in this paper is applicative.

Fig. 1. Business players involved in network slicing and how does the process works.

models such as deep Q-network (DQN) [18], [21]. These algorithms can achieve (approximately) optimal solutions and make the communication systems smart and intelligent [22], [23]. However, they are usually complex and do not scale with the types of resources and the number of tenants. Take Deep Q-Network (DQN) as an example, it could take days even weeks for obtaining *not-particularly-good* actions even though the state and action spaces have been discretized. Although several reinforcement learning methods can avoid privacy leakage, such as [24] and [25], the centralized algorithms are generally built on the complete knowledge regarding all preferences of involved business players, including the monetary budget of tenants, the number and purchasing-power of service subscribers, etc. The formulation of the centralized optimization problem itself is a detriment on privacy and trade secrets.

To avoid insufferable complexity and privacy leakage, in recent years, many researchers establish network slicing models based on standard economic frameworks, such as *Fisher markets* [26], [27], and different auction-based mechanisms, such as the VCG-Kelly mechanisms [28], [29]. In these works, all tenants get together and bid for maximizing their profits. For instance, Wang *et al.* studied the relationship between resource efficiency and profit maximization and developed an optimization framework to maximize net social welfare [30]. Similarly, Jiang *et al.* addressed a joint resource and revenue optimization problem and solved it with the auction mechanisms [31]. Furthermore, some works resort to game theory to model tenants' and MVNOs' strategic (or non-strategic) behaviors, and take the *price of anarchy* (PoA) to analyze the efficiency of potentially existent Nash equilibrium (NE) [32]. For instance, Caballero *et al.* studied the resource allocation mechanism by formulating a network slicing game [33]. They proved that when the game associated with strategic behavior of tenants, i.e., adjusting their preferences depending on perceived resource contention, convergence to a Nash equilibrium (*under some specific conditions*) can be achieved. Luu *et al.* also study a network slicing game, but under specific constraints of RAN [10]. Generally, auction mechanisms are efficient and scalable to diversified service requirements. However, most of these auction-based works are designed under an *offline* setting, i.e., the MVNO knows the willingness to bid and many other private information of all tenants during each bidding round. Besides, a tenant's partial private information might be disclosed to all the remaining tenants. Nevertheless, this may not possible in many real-world business transactions because it is rare that all the tenants negotiate the rental business details simultaneously. The MVNO should not know anything about the arrival sequence of tenants, much less the private information of the served users of each tenant. It should only have the knowledge on the resource surplus and the attributes saved in the generic network slice templates (GSTs) [34]. In addition, a tenant private information should not be available to the other tenants.

The above analysis shows that auction mechanisms may not be ideal for online network slicing problems. In addition to the above reasons, auctions take time and require multiple communication rounds between the MVNO and the tenants [35]. They may have poor performances when the distribution of bidders' arrival instance is unknown [36]. By contrast, take-it-or-leave-it, i.e., *posted price*, is a more practical option for online settings. Therefore, in this paper, we design an online slicing algorithm based on the posted price mechanism. A decentralized, low-complexity, and privacy-preserving algorithm, DPoS, mainly based on previous theoretical works on the online primal-dual algorithms [37], [38], and [39], is proposed. Specifically, we extend the basic model proposed in [38] into multi-resource scenarios. DPoS is consists of two parts, $\mathsf{DPoS-MVNO}$ (agent for the MVNO) and $\mathsf{DPoS}-TNT_n$ (agent for the nth tenant), with a complexity of $O(NC)$ and $O(C)$, respectively. Here N is the number of tenants, and C is the number of type of resources. DPoS runs in a fully decentralized way. Each time a new tenant n arrives, $\mathsf{DPoS-TNT}_n$ decides to rent the demand resources or not according to the rental prices of each type of resource, published by $\mathsf{DPoS-MVNO}$ beforehand. Therewith, $\mathsf{DPoS}-TNT_n$ sends the decision and payment (if tenant n has the willingness to pay) to $\mathsf{DPoS-MVNO}$. Then, $\mathsf{DPoS-MVNO}$ checks whether the resource surplus can satisfy tenant n and inform $\mathsf{DPoS-TNT}_n$ the transaction is succeeded or failed. Note that each tenant may experience different prices on the same kind of resource, which depends on the pricing mechanism the MVNO adopts. In the above procedure, only a small flow of privacy-irrelevant information are transferred between the MVNO and each tenant. No information are transferred among tenants. Trade secrets, especially the information of service subscribers and the pricing policies of tenants, will not be disclosed.

Our main contributions are summarized as follows.

- We design a decentralized, privacy-preserving online network slicing algorithm, DPoS. This algorithm enjoys low complexity, and it is practicable under diversified multi-resource requirements. Trade secrets and related private information can be fully preserved.
- We find that, when the cost function of each resource is linear, DPoS achieves the *optimal* competitive ratio over all the online algorithms for the maximization of social welfare.
- We verify the superiority of DPoS from multiple angles, including social welfare achieved, cross-agent communication data size, algorithm execution time, etc. The experimental results show that DPoS not only achieves close-to-offline-optimal performance, but also has low algorithmic overheads.

The remainder of the paper is organized as follows. Section 2 presents the system model and formulates the global offline problem. Section 3 demonstrates the design details

TABLE 1
Summary of Key Notations

Notation	Description
$\mathcal{C}$	The set of network resources
$\mathcal{N}$	The set of tenants
$\mathcal{S}_n$	The set of users of tenant $n \in \mathcal{N}$
$\{d_n^c\}_{\forall c \in \mathcal{C}}$	Resource demands of tenant n
v_n	The revenue in estimation of tenant n
e_n^c	The valuation density of tenant n
$\underline{p_c}$ and $\overline{p_c}$	The lower (upper) bound of earning density
$x_n \in \{0,1\}$	The decision variable of tenant n
π_n	The payment made by tenant n
$y_c \in [0,1]$	The resource rent out of type c
$\{f_c\}_{\forall c \in \mathcal{C}}$	Non-decreasing zero-startup cost functions
$\underline{\varsigma_c}$ and $\overline{\varsigma_c}$	The derivative of $f_c(\cdot)$ at point 0 and 1
$\{\tilde{f}_c\}_{\forall c \in \mathcal{C}}$	The extended cost functions
$\{F_{p_c}\}_{\forall c \in \mathcal{C}}$	The profit functions
$\{h_c\}_{\forall c \in \mathcal{C}}$	The maximum profit functions
ψ_n and p_c	The dual variables corresponding to x_n and y_c
$\{\phi_c\}_{\forall c \in \mathcal{C}}$	The pricing functions
α	Competitive ratio of online algorithms

of the algorithm DPoS. Theoretical analysis on the competitive ratio is provided in Section 4. The experiment results are demonstrated in Section 5. Section 6 reviews related works and Section 7 concludes this paper.

2 Problem Formulation

To simplify the notations without damaging its economic structure, our model concerns one InP, one MVNO, several tenants and each tenant's served users. Our model and algorithm can be directly adapted to multi-MVNO multi-InP scenarios. We consider the scenario where multiple network slices are built upon an SDN/NFV-enabled 5G network infrastructure, which is rented from the InP by the MVNO. Roughly, physical resources in this infrastructure can be divided into computation, storage, and forwarding/bandwidth. At great length, physical resources are usually organized as a *weighted directed graph* [30], [40], where the *node* can be base station at the access, forwarding router in the bearer, and physical machine or virtual machine in the regional datacenters, and the *edge* is a directed link with certain propagation speed. Each node is the carrier of VNFs with different capabilities while link has unique bandwidth and data transfer rate. To ensure the universality of the model, we do not add any specific limitation and simply use $\mathcal{C} \triangleq \{1, \ldots, C\}$ to denote the set of resources. Without loss of generality, the capacity limit of each resource is normalized to be 1. The key notations used in this paper are summarized in Table 1.

Let us use $\mathcal{N} \triangleq \{1, .., N\}$ to denote the set of tenants. In our model, each tenant requests one (and only one) slice from the MVNO.[2] Generally, a slice is a collection of different types of resources, the topology of which can be mapped onto the substrate network as a connected subgraph.[3] We use $\{d_n^c\}_{\forall c \in \mathcal{C}}$ to denote the requirements of the nth tenant (slice), where d_n^c is the demand of resource of type $c \in \mathcal{C}_n \subseteq \mathcal{C}$, and

$$d_n^c \begin{cases} > 0 & \text{if } c \in \mathcal{C}_n \\ = 0 & \text{otherwise.} \end{cases} \tag{1}$$

The traffic demand on the nth slice is denoted by $\{f_s(\gamma, \tau)\}_{\forall s \in \mathcal{S}_n}$, where s is a service subscriber from tenant n's served users $\mathcal{S}_n$, and $f_s(\gamma, \tau)$ is a data flow with promissory data rate γ and latency constraint τ from some source node to some destination node. A slice's consumption on resources is embodied in the execution of VNFs and the occupation of bandwidth. For tenant n, we define a function $\sigma_n : \{f_s(\gamma, \tau)\}_{\forall s \in \mathcal{S}_n} \to \mathbb{R}$ to calculate the payment of user s for enjoying the data flow $f_s(\gamma, \tau)$. We regard σ_n as *private* because it involves business secrets of the tenant. The *estimated* revenue of each tenant n is from the payment of its service subscribers, which is defined as follows:

$$v_n \triangleq \sum_{s \in \mathcal{S}_n} \varrho_s \cdot \sigma_n\Big(f_s(\gamma, \tau)\Big). \tag{2}$$

In (2), ϱ_s is the level of QoS for subscriber $s \in \mathcal{S}_n$, which is decided based on the commitment on delay torelant, reliability, isolation level, etc [41]. A full list of attributes can be found at [34]. Under normal circumstances, the higher the level of QoS, the faster data rate and tighter latency constraints on the data flow, which in turn leads to more resource consumption. Note that all the $\{d_n^c\}_{\forall c \in \mathcal{C}}$ are used to support the data flows $\{f_s(\gamma, \tau)\}_{\forall s \in \mathcal{S}_n}$. If we assume that the mapping function from data flow $f_s(\gamma, \tau)$ to the cth resource occupation is $g_c : \{f_s(\gamma, \tau)\}_{\forall s \in \mathcal{S}_n} \to [0, d_n^c]$, we have the following identity:

$$d_n^c = \sum_{s \in \mathcal{S}_n} g_c\Big(f_s(\gamma, \tau)\Big), \forall c \in \mathcal{C}_n.$$

In practice, v_n can be interpreted as the *willingness-to-pay* of tenant n for renting the required resources [38]. $\forall c \in \mathcal{C}, \forall n \in \mathcal{N}$, we define the *earning density* e_n^c as v_n / d_n^c. e_n^c can be interpreted as the estimated revenue per unit of resource c to the tenant n. Following [38], [42], we define $\underline{p_c}$ and $\overline{p_c}$ as follows.

$$\forall c \in \mathcal{C} : \begin{cases} \underline{p_c} \le \min_{\forall n \in \mathcal{N} : d_n^c \neq 0} e_n^c \\ \overline{p_c} \ge \max_{\forall n \in \mathcal{N} : d_n^c \neq 0} e_n^c. \end{cases} \tag{3}$$

The lower bound means that the MVNO will reject the tenant n directly if $\exists c \in \mathcal{C}, e_n^c$ is lower than $\underline{p_c}$. The role of the lower bound is to avoid the tenants deliberately overstating their resource demands to get a discount. In other words, the tenants are forced to engage the transactions with their true preferences and no resource will be wasted. The upper bound in (3) is to eliminate irrational tenants or mock auctions, which exists naturally in a wholesome market.

For each tenant $n \in \mathcal{N}$, we use $x_n \in \{0, 1\}$ to indicate whether the deal is successful. The utility of tenant n is defined as $U_n \triangleq (v_n - \pi_n) \cdot x_n$, where π_n is the payment. The utility of the MVNO is defined as $U_o \triangleq \sum_{n \in \mathcal{N}} \pi_n \cdot x_n - \sum_{c \in \mathcal{C}} f_c\Big(\sum_{n \in \mathcal{N}} d_n^c x_n\Big)$, where $\forall c \in \mathcal{C}, f_c : [0, 1] \to \mathbb{R}$ is a non-

2. In the following, we may interpret n as the nth tenant or the nth slice. It depends on the content.

3. There have been lots of works on the VNF placement and mapping [7], [15], [17], [33]. But this is not the subject of this paper.

decreasing zero-startup cost function of resource c. We set f_c as a non-decreasing function because more resources virtualized and sliced, more operating and maintenance costs on those rent-out slices.

In the above formulation, we take σ_n and the mapping $\{f_s(\gamma,\tau)\}_{\forall s\in\mathcal{S}_n}\rightarrow\mathcal{S}_n$ as private information of tenant n which should not be accessible to the MVNO and the other tenants. The former comes down to the payment models and pricing strategies adopted by tenant n and reveals the purchasing power of it served users. The latter establishes the relationship between data flows and their initiators. In online settings, the deals between the MVNO and the tenants are made one-by-one according to the arrival sequence of tenants. Our goal is to (approximately) maximize the social welfare of this ecosystem, i.e., the sum of the MVNO's utility and all the tenants', in an *online* and *decentralized* setting. Before introducing the online algorithm proposed in this paper, we first formulate the global *offline* social welfare maximization problem as follows.

$$\mathcal{P}_1:\max_{\{x_n\}_{\forall n\in\mathcal{N}}}\sum_{n\in\mathcal{N}}v_nx_n-\sum_{c\in\mathcal{C}}f_c\Big(\sum_{n\in\mathcal{N}}d_n^cx_n\Big)$$

$$s.t.\quad\sum_{n\in\mathcal{N}}d_n^cx_n\le\ 1,\ \forall c\in\mathcal{C},\tag{4a}$$

$$x_n\in\{0,1\},\ \forall n\ \in\mathcal{N}.\tag{4b}$$

In $\mathcal{P}_1$, v_n is obtained through (2). Even though the problem is hard to solve, it is formulated based on the complete knowledge of the ecosystem. In other words, the formulation of $\mathcal{P}_1$ itself is a detriment on privacy. In an online setting, the MVNO should only know the setup information $\{f_c,\underline{p_c},\overline{p_c}\}_{\forall c\in\mathcal{C}}$ and the attributes defined in the GSTs $\{\varrho_s\}_{\forall s\in\mathcal{S}_n,\forall n\in\mathcal{N}}$ handed in by tenants as a priori. It should not know anything about the private tuple

$$\boldsymbol{\theta}\triangleq\Big(\{\sigma_n\}_{\forall n\in\mathcal{N}},\big\{\{f_s(\gamma,\tau)\}_{\forall s\in\mathcal{S}_n}\rightarrow\mathcal{S}_n\big\}_{\forall n\in\mathcal{N}}\Big),$$

and the arrival sequence of tenants. In addition, each tenant should know nothing about the other tenants at all. As a result, to solve the problem in a privacy-preserving decentralized setting, we need to ensure that the deal is made with only a small flow of information transferred between the MVNO and each tenant without revealing any sensitive information. In the proposed algorithm DPoS, which will be introduced in the following, each time when a new tenant n arrives, the tenant makes the decision x_n *by itself* according to the disclosed information such as current rental price of each kind resource. If x_n is set as 1, then tenant n sends $(1,\pi_n,\{d_n^c\}_{\forall c\in\mathcal{C}})$ to the MVNO. Otherwise $(0,0,0)$ is sent. The MVNO can only access the data transferred to it.

3 Algorithm Design

To maximize the social welfare in an online and decentralized setting, we first introduce some notations, then demonstrate the designing of the **D**istributed **P**rivacy-preserving **o**nline **S**licing algorithm, DPoS.

3.1 The Primal-Dual Approach

$\forall c\in\mathcal{C}$, we introduce the *extended cost function* $\tilde{f}_c$ as follows.

$$\tilde{f}_c(y)\triangleq\begin{cases}f_c(y) & \text{if } y\in[0,1]\\ +\infty & \text{if } y\in(1,+\infty),\end{cases}\tag{5}$$

$\tilde{f}_c$ extends the domain of f_c to $[0,+\infty)$. Correspondingly, we define the *profit function* F_{p_c} of resource c for the MVNO as follows:

$$F_{p_c}(y_c)\ \triangleq\ p_cy_c-\tilde{f}_c(y_c),\forall y_c\in[0,+\infty).\tag{6}$$

Regarding y_c as the total resource rented out of type c and p_c as the rental price of resource c, $F_{p_c}(y_c)$ is the revenue obtained by renting out y_c unit of resource c minus the maintainers cost of it. Based on (6), we denote the *maximum profith*$_c$ of resource c when the rental price is p_c by

$$h_c(p_c)\triangleq\max_{y_c\ge0}F_{p_c}(y_c).\tag{7}$$

Following the primal-dual approach [38], [39], we introduce the *Relaxed Primal Problem* $\mathcal{P}_2$.

$$\mathcal{P}_2:\max_{\boldsymbol{x},\boldsymbol{y}}\sum_{n\in\mathcal{N}}\sum_{s\in\mathcal{S}_n}\varrho_s\cdot\sigma_n\Big(f_s(\gamma,\tau)\Big)x_n-\sum_{c\in\mathcal{C}}\tilde{f}_c(y_c)$$

$$s.t.\quad\sum_{n\in\mathcal{N}}d_n^cx_n\le y_c,\forall c\in\mathcal{C},\tag{8a}$$

$$\boldsymbol{x}\le\mathbf{1},\boldsymbol{x}\ge\mathbf{0},\boldsymbol{y}\ge\mathbf{0},\tag{8b}$$

where $\boldsymbol{x}=[x_n]_{n\in\mathcal{N}}\in[0,1]^N$, and $\boldsymbol{y}=[y_c]_{y\in\mathcal{C}}\in\mathbb{R}^C$. In terms of the relation between $\mathcal{P}_1$ and $\mathcal{P}_2$, we have the following proposition.

Proposition 1. *$\mathcal{P}_2$ is equivalent to $\mathcal{P}_1$ except the relaxation of $\{x_n\}_{\forall n\in\mathcal{N}}$.*

Proof. To maximize the objective of $\mathcal{P}_1$, the optimal $\boldsymbol{y}^\star$ must be located in $[0,1]^N$. Because f_c is non-decreasing for all kinds of resource $c\in\mathcal{C}$, the optimal $y_c^\star$ must be the minimum allowed, i.e., $\sum_{n\in\mathcal{N}}d_n^cx_n$. Thus, except relaxing $\{x_n\}_{\forall n\in\mathcal{N}}$ to the continuous interval $[0,1]^N$, $\mathcal{P}_2$ is the same as $\mathcal{P}_1$. □

Take $\mathcal{P}_2$ as the primal problem, the following proposition gives the dual problem $\mathcal{P}_3$.

Proposition 2. *The dual problem of $\mathcal{P}_2$ is:*

$$\mathcal{P}_3:\min_{\boldsymbol{p},\boldsymbol{\psi}}\sum_{n\in\mathcal{N}}\psi_n+\sum_{c\in\mathcal{C}}h_c(p_c)\tag{9a}$$

$$s.t.\quad\psi_n\ge v_n-\sum_{c\in\mathcal{C}}p_cd_n^c,\forall n\in\mathcal{N},\tag{9b}$$

$$\boldsymbol{\psi}\ge\mathbf{0},\boldsymbol{p}\ge\mathbf{0},\tag{9c}$$

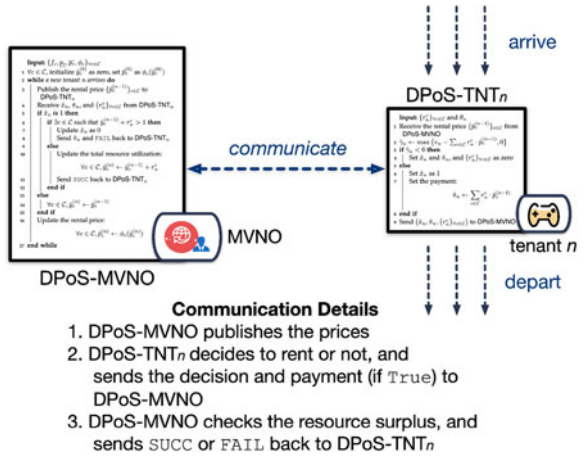


Fig. 2. How DPoS works. Each time a new tenant n arrives, only a small flow of privacy-irrelevant data are transferred between DPoS − MVNO and DPoS − TNT$_n$.

where $\boldsymbol{\psi} = [\psi_n]_{n\in\mathcal{N}} \in \mathbb{R}^N$ *and* $\boldsymbol{p} = [p_c]_{c\in\mathcal{C}} \in \mathbb{R}^C$ *are the dual variables corresponding to* $\boldsymbol{x}$ *and* $\boldsymbol{y}$*, respectively.*

Proof. By introducing the Lagrangian multipliers $\{p_c\}_{\forall c\in\mathcal{C}}$ and $\{\psi_n\}_{\forall n\in\mathcal{N}}$ for (8a) and the first inequality of (8b), respectively, the Lagrangian of $\mathcal{P}_2$ is

$$\Lambda(\boldsymbol{x},\boldsymbol{y},\boldsymbol{\psi},\boldsymbol{p}) = \sum_{c\in\mathcal{C}}\Big(p_c y_c - \tilde{f}_c(y_c)\Big) + \sum_{n\in\mathcal{N}}\psi_n + \sum_{n\in\mathcal{N}} x_n\left(\sum_{s\in\mathcal{S}_n}\varrho_s\cdot\sigma_n\Big(f_s(\gamma,\tau)\Big) - \sum_{c\in\mathcal{C}} p_c d_n^c - \psi_n\right).$$

Thus, we have

$$\min_{\boldsymbol{\psi},\boldsymbol{p}}\max_{\boldsymbol{x},\boldsymbol{y}}\Lambda = \min_{\boldsymbol{\psi},\boldsymbol{p}}\left(\max_{\boldsymbol{y}}\sum_{c\in\mathcal{C}}\Big(p_c y_c - \tilde{f}_c(y_c)\Big) + \sum_{n\in\mathcal{N}}\psi_n\right),$$

when $\forall n\in\mathcal{N},\ \psi_n \geq v_n - \sum_{c\in\mathcal{C}} p_c d_n^c$. Therein, $\sum_{s\in\mathcal{S}_n}\varrho_s\cdot\sigma_n\Big(f_s(\gamma,\tau)\Big)$ is replaced by v_n through (2). The result is immediate with (7). □

Regarding ψ_n as the utility of tenant n. The objective of $\mathcal{P}_3$ is the aggregate utilities of all tenants plus the optimal utility of the MVNO. Both the objective of $\mathcal{P}_1$ and $\mathcal{P}_3$ indicate the social welfare of the ecosystem.

3.2 The DPoS Algorithm

Note that the rental price p_c of resource c is a global variable known to all tenants. Thus, if the final optimal price $\boldsymbol{p}$ is known to the MVNO, each time a tenant n arrives, then this tenant can make the rent decision x_n without worrying about whether the optimal social welfare is achieved or not. However, it is impossible to know the exact value of $\boldsymbol{p}$ in advance without the arrival sequence and $\boldsymbol{\theta}$. To tackle with this problem, inspired by [37], [38] and [39], we design the DPoS algorithm based on the alternating update of primal & dual variables (of $\mathcal{P}_2$ and $\mathcal{P}_3$) and the *predict-and-update* of $\boldsymbol{p}$. In the following, we place a hat on top of variables that denote the decisions made online.

Algorithm 1. DPoS − MVNO

Input: $\{f_c, \underline{p_c}, \overline{p_c}, \phi_c\}_{\forall c\in\mathcal{C}}$
$\forall c\in\mathcal{C}$, initialize $\hat{y}_c^{(0)}$ as zero, set $\hat{p}_c^{(0)}$ as $\phi_c(\hat{y}_c^{(0)})$
while *a new tenant* n *arrives* **do**
  Publish the rental price $\{\hat{p}_c^{(n-1)}\}_{c\in\mathcal{C}}$ to DPoS − TNT$_n$
  Receive $\hat{x}_n$, $\hat{\pi}_n$, and $\{d_n^c\}_{\forall c\in\mathcal{C}}$ from DPoS − TNT$_n$
  if $\hat{x}_n$ **is** *1* **then**
    if $\exists c\in\mathcal{C}$ *such that* $\hat{y}_c^{(n-1)} + d_n^c > 1$ **then**
      Update $\hat{x}_n$ as 0
      Send $\hat{\pi}_n$ and FAIL back to DPoS − TNT$_n$
    else
      Update the total resource utilization:

$$\forall c\in\mathcal{C}, \hat{y}_c^{(n)} \leftarrow \hat{y}_c^{(n-1)} + d_n^c$$

      Send SUCC back to DPoS − TNT$_n$
    end if
  else
    $\forall c\in\mathcal{C}, \hat{y}_c^{(n)} \leftarrow \hat{y}_c^{(n-1)}$
  end if
  Update the rental price:

$$\forall c\in\mathcal{C}, \hat{p}_c^{(n)} \leftarrow \phi_c(\hat{y}_c^{(n)})$$

end while

DPoS consists of two parts, DPoS − MVNO and DPoS− TNT_n (each for a tenant). Before a new tenant n arrives, DPoS − MVNO prices for each resource c with a function ϕ_c:

$$\hat{p}_c^{(n-1)} = \phi_c(\hat{y}_c^{(n-1)}),\ \forall c\in\mathcal{C}. \tag{10}$$

The pricing functions $\{\phi_c\}_{c\in\mathcal{C}}$ are closely associated to the properties of the cost functions $\{f_c\}_{c\in\mathcal{C}}$. We will provide the analytic forms of them in the follwing subsection.

DPoS − MVNO discloses the rental prices $\{\hat{p}_c^{(n-1)}\}_{c\in\mathcal{C}}$ to tenant n. Then, tenant n judges whether it has *positive* utility if it decides to rent $\{d_n^c\}_{\forall c\in\mathcal{C}}$ ($\hat{x}_n \leftarrow 1$). If yes, DPoS − TNT$_n$ sets the payment $\hat{\pi}_n$ as $\sum_{c\in\mathcal{C}} d_n^c\cdot\hat{p}_c^{(n-1)}$. Otherwise, both $\hat{x}_n$ and $\hat{\pi}_n$ are set as zero. In the end, DPoS − TNT$_n$ sends $(\hat{x}_n, \hat{\pi}_n, \{d_n^c\}_{\forall c\in\mathcal{C}})$ to DPoS − MVNO.

When DPoS − MVNO receives the message from DPoS− TNT_n, it checks whether the resource surplus can satisfy tenant n. If yes, DPoS − MVNO sends the indicator SUCC to DPoS − TNT$_n$ to inform the success of this transaction. Otherwise, it sends FAIL and returns the rent $\hat{\pi}_n$. If succeed, tenant n hands in the GST and other matters that need to be provided.

The procedure is visualized in Fig. 2. Note that the data transfer between DPoS − MVNO and DPoS − TNT$_n$ is *stop-and-wait*, i.e., a new arrival tenant will not be handed by the MVNO until the transaction between the MVNO and the previous tenant is done. In DPoS, only a small flow of privacy-irrelevent data $(\hat{x}_n, \hat{\pi}_n, \{d_n^c\}_{\forall c\in\mathcal{C}})$ are transferred between DPoS − TNT$_n$ and DPoS − MVNO. The MVNO cannot collect any information from $\boldsymbol{\theta}$. In addition, each tenant knows nothing about the other tenants. DPoS is implemented in a posted price manner [43], [44], where the rent decision made by each tenant is only *take-it-or-leave-it*. A tenant cannot get

any discount even if it rents relatively large amounts of resources, which leads to the fact that *how much to use, how much to rent*. No resource will be wasted.

It is easy to verify that the complexity of DPoS is linear with the scale of tenants and type of resources. In DPoS−*MVNO*, the while-loop terminates after all the $|\mathcal{N}|$ tenants finish their transactions in turn. During the loop, the most time-consuming operation lies in step 6 and step 10, where the MVNO needs check whether each type of resource c is enough to support a transaction and take d_n^c off if permitted. In worst case, the number of operations is $2|\mathcal{C}|$. Considering that all the left steps can be executed in $O(1)$-complexity, the worst-case complexity of $\mathsf{DPoS}-\mathsf{MVNO}$ is $O(|\mathcal{N}|\cdot|\mathcal{C}|)$. As for $\mathsf{DPoS}-\mathsf{TNT}_n$, time-consumption operations lie in step 2, step 4, and step 7, all of which are $O(|\mathcal{C}|)$-complexity. Therefore, $\mathsf{DPoS}-\mathsf{TNT}_n$ is of $O(|\mathcal{C}|)$-complexity in worst case.

3.3 The Dynamic Pricing Functions

In DPoS, the only difficulty lies in that how the pricing functions $\{\phi_c\}_{c\in\mathcal{C}}$ are designed. As mentioned before, the analytic forms of $\{\phi_c\}_{c\in\mathcal{C}}$ strongly rely on the properties of cost functions $\{f_c\}_{c\in\mathcal{C}}$. Even so, we claim that *in* DPoS, $\{\phi_c\}_{\forall c\in\mathcal{C}}$ *are monotonically non-decreasing positive functions*. We set ϕ_c as a non-decreasing function because it profoundly reflects the underlying economic phenomenon, i.e., *a thing is valued in proportion to its rarity*. The later the tenant comes to renting the remaining resources, the higher cost it has to pay.

Algorithm 2. $\mathsf{DPoS}-\mathsf{TNT}_n$

Input: $\{d_n^c\}_{\forall c\in\mathcal{C}}$ and θ_n
1 Receive the rental price $\{\hat{p}_c^{(n-1)}\}_{c\in\mathcal{C}}$ from $\mathsf{DPoS}-\mathsf{MVNO}$
2 $\hat{\psi}_n \leftarrow \max\{v_n - \sum_{c\in\mathcal{C}} d_n^c\cdot\hat{p}_c^{(n-1)}, 0\}$
3 **if** $\hat{\psi}_n < 0$ **then**
4 Set $\hat{x}_n$ and $\hat{\pi}_n$, and $\{d_n^c\}_{\forall c\in\mathcal{C}}$ as zero
5 **else**
6 Set $\hat{x}_n$ as 1
7 Set the payment:

$$\hat{\pi}_n \leftarrow \sum_{c\in\mathcal{C}} d_n^c\cdot\hat{p}_c^{(n-1)}$$

8 **end if**
9 Send $(\hat{x}_n, \hat{\pi}_n, \{d_n^c\}_{\forall c\in\mathcal{C}})$ to $\mathsf{DPoS}-\mathsf{MVNO}$

Now, we demonstrate the forms of $\{\phi_c\}_{\forall c\in\mathcal{C}}$ when the costs are linear. Concretely, if $\forall c\in\mathcal{C}$, the cost function has the form

$$f_c(y) = q_c y, \tag{11}$$

where $0 < q_c < \underline{p_c}$. Then, in DPoS, the pricing function ϕ_c is set as follows:

$$\phi_c(y) = \begin{cases} \underline{p_c} & y\in[0, w_c) \\ q_c + (\underline{p_c} - q_c)\cdot e^{y/w_c - 1} & y\in[w_c, 1] \\ +\infty & y\in(1, +\infty), \end{cases} \tag{12}$$

where

$$w_c = \left(1 + \ln\frac{\sum_{c'\in\mathcal{C}}(\overline{p_{c'}} - q_{c'})}{\underline{p_c} - q_c}\right)^{-1}, \tag{13}$$

is a threshold. Tan *et al*. also discuss the construction of the pricing function (for single resource and multiple substitutable resources) when the resource's cost function is strictly-convex [38], which involves the solving of several first-order two-point boundary value problems (BVPs) [45]. In the next section, we will show that the competitive ratio of DPoS is the optimal one over all the online algorithms when $\{f_c\}_{c\in\mathcal{C}}$ are linear.

4 Theoretical Analysis

The commonly used measure for online algorithms is the standard competitive analysis framework [46]. The definition of *competitive ratio* for any online algorithm to $\mathcal{P}_1$ is given below.

Definition 1. *For any arrival instance $1, 2, \ldots, N$, denoted by $\mathcal{A}$, the competitive ratio for an online algorithm is defined as*

$$\alpha \triangleq \max_{\forall\mathcal{A}} \frac{\mathbf{\Theta}_{\text{off}}(\mathcal{A})}{\mathbf{\Theta}_{\text{on}}(\mathcal{A})}, \tag{14}$$

where $\mathbf{\Theta}_{\text{off}}(\mathcal{A})$ is the maximum objective value of $\mathcal{P}_1$, $\mathbf{\Theta}_{\text{on}}(\mathcal{A})$ is the objective function value of $\mathcal{P}_1$ obtained by this online algorithm.

Obviously, $\alpha \geq 1$ always holds. The smaller α is, the better the online algorithm. An online algorithm is *competitive* if its competitive ratio is upper bounded. Further, we can define the optimal competitive ratio as

$$\alpha^* \triangleq \inf \max_{\forall\mathcal{A}} \frac{\mathbf{\Theta}_{off}(\mathcal{A})}{\mathbf{\Theta}_{on}(\mathcal{A})}, \tag{15}$$

where the inf is taken w.r.t. all possible online algorithms. In the following, we drop the parenthesis and $\mathcal{A}$ for simplification. Note that whether optimal or not, competitive ratio only gives the *worst-case* guarantee.

To analyze the competitive ratio achieved by DPoS, we need to introduce several propositions and theorems beforehand. We will first verify that DPoS is α-competitive for some constant α, then prove that it is the optimal one over all online algorithms when $\{f_c\}_{\forall c\in\mathcal{C}}$ are linear. The first proposition introduced is related to the maximum utility h_c.

Proposition 3. *$\forall c\in\mathcal{C}$, the function h_c, defined in (7), can also be written as*

$$h_c(p_c) = \begin{cases} F_{p_c}\left(f_c'^{-1}(p_c)\right) & p_c\in[\underline{\varsigma_c}, \overline{\varsigma_c}] \\ F_{p_c}(1) & p_c\in(\overline{\varsigma_c}, +\infty), \end{cases} \tag{16}$$

where $\underline{\varsigma_c} \triangleq f_c'(0)$, $\overline{\varsigma_c} \triangleq f_c'(1)$, f_c' is the derivative of f_c, and $f_c'^{-1}$ is the inverse of f_c'.

Proof. $\forall c\in\mathcal{C}$, when $\underline{\varsigma_c} \leq p_c \leq \overline{\varsigma_c}$, regarding p_c as the derivative of the non-decreasing f_c, then we have $f_c'^{-1}(p_c)\in[0,1]$. Now we need to find the y_c^* which maximizes $F_{p_c}(y_c)$. By analyzing the property of $\partial F_{p_c}(y_c)/\partial y_c$, which is $p_c - f_c'(y_c)$, we can find that the exact y_c^* satisfies $p_c =$

$f'(y_c^*)$. Thus $h_c(p_c)$ is $F_{p_c}\big(f_c'^{-1}(p_c)\big)$ when $\underline{\varsigma_c} \le p_c \le \overline{\varsigma_c}$. The same applies to the second segment of (16). □

(16) is known as the *convex conjugate* of $\tilde{f}_c$ [47]. For a given online algorithm, denote the objective of $\mathcal{P}_2$ and $\mathcal{P}_3$ by $\mathbf{\Theta}^n_{\mathcal{P}_2}$ and $\mathbf{\Theta}^n_{\mathcal{P}_3}$ after processing tenant n, respectively. Also, we use $\mathcal{V}_{\mathcal{P}_2} \triangleq \big(\{\hat{x}_n\}_{\forall n\in\mathcal{N}}, \hat{y}_N\big)$ and $\mathcal{V}_{\mathcal{P}_3} \triangleq \big(\{\hat{\psi}_n\}_{\forall n\in\mathcal{N}}, \hat{p}_N\big)$ to denote the complete set of online primal and dual solutions, respectively. In the following, we demonstrate the sufficient conditions of designing an α-competitive online algorithm for $\mathcal{P}_1$, and then show that DPoS satisfies the conditions.

Proposition 4. *(Adapted from proposition 3.1 of [38]) When $\{f_c\}_\forall c \in \mathcal{C}$ are linear[4], an online algorithm is α-competitive if the following conditions are satisfied:*

- *All the online primal solutions in $\mathcal{V}_{\mathcal{P}_2}$ are feasible to $\mathcal{P}_1$;*
- *All the online dual solutions in $\mathcal{V}_{\mathcal{P}_3}$ are feasible to $\mathcal{P}_3$;*
- *There exists a tenant $k \in \mathcal{N}$ such that*

$$\mathbf{\Theta}^k_{\mathcal{P}_2} \ge \frac{1}{\alpha}\mathbf{\Theta}^k_{\mathcal{P}_3}, \tag{17}$$

and $\forall n \in \{k+1, \ldots, N\}$,

$$\mathbf{\Theta}^n_{\mathcal{P}_2} - \mathbf{\Theta}^{n-1}_{\mathcal{P}_2} \ge \frac{1}{\alpha}\big(\mathbf{\Theta}^n_{\mathcal{P}_3} - \mathbf{\Theta}^{n-1}_{\mathcal{P}_3}\big), \tag{18}$$

holds.

Proof. Let us denote the optimal objective of $\mathcal{P}_2$ and $\mathcal{P}_3$ as $\mathbf{\Theta}^*_{\mathcal{P}_2}$ and $\mathbf{\Theta}^*_{\mathcal{P}_3}$, respectively. Then,

$$\mathbf{\Theta}_{off} \le \mathbf{\Theta}^*_{\mathcal{P}_2} = \mathbf{\Theta}^*_{\mathcal{P}_3} \le \mathbf{\Theta}^N_{\mathcal{P}_3}. \tag{19}$$

The reason for the first inequality is that $\mathcal{P}_2$ is a relaxation of $\mathcal{P}_1$. The reason for the first equality is that when $\{f_c\}_{\forall c\in\mathcal{C}}$ are linear, strong duality holds between $\mathcal{P}_2$ and $\mathcal{P}_3$. Besides, $\mathbf{\Theta}_{on} = \mathbf{\Theta}^N_{\mathcal{P}_2}$. As a result, to make $\alpha \ge \mathbf{\Theta}_{off}/\mathbf{\Theta}_{on}$ always hods, we can try to ensure that $\mathbf{\Theta}^N_{\mathcal{P}_2} \ge \frac{1}{\alpha}\mathbf{\Theta}^N_{\mathcal{P}_3}$ holds.

According to (18), the following inequalities holds:

$$\begin{aligned}
&\sum_{n\in\mathcal{N}, n>k}\Big(\mathbf{\Theta}^n_{\mathcal{P}_2} - \mathbf{\Theta}^{n-1}_{\mathcal{P}_2}\Big) \ge \frac{1}{\alpha}\sum_{n\in\mathcal{N}, n>k}\Big(\mathbf{\Theta}^n_{\mathcal{P}_3} - \mathbf{\Theta}^{n-1}_{\mathcal{P}_3}\Big)\\
\iff\quad & \mathbf{\Theta}^N_{\mathcal{P}_2} - \mathbf{\Theta}^k_{\mathcal{P}_2} \ge \frac{1}{\alpha}\Big(\mathbf{\Theta}^N_{\mathcal{P}_3} - \mathbf{\Theta}^k_{\mathcal{P}_3}\Big)\\
\iff\quad & \mathbf{\Theta}^N_{\mathcal{P}_2} - \frac{1}{\alpha}\mathbf{\Theta}^N_{\mathcal{P}_3} \ge \mathbf{\Theta}^k_{\mathcal{P}_2} - \frac{1}{\alpha}\mathbf{\Theta}^k_{\mathcal{P}_3}\\
\iff\quad & \mathbf{\Theta}^N_{\mathcal{P}_2} \ge \frac{1}{\alpha}\mathbf{\Theta}^N_{\mathcal{P}_3}. \qquad \triangleright(17)
\end{aligned}$$

We thus complete the proof. □

Proposition 4 gives three conditions for designing an α-competitive online algorithm when $\{f_c\}_\forall c \in \mathcal{C}$ are linear. If we can prove that these conditions hold for DPoS, then we prove that DPoS is at least α-competitive for some constant α. In the following, we prove that the first and the second condition hold.

4. Proposition 1 of [38] also requires that $\{\underline{\varsigma_c} < \underline{p_c}\}_{\forall c\in\mathcal{C}}$ holds, which is not required in this proposition.

- It is obvious that $\mathcal{V}_{\mathcal{P}_2}$ obtained by DPoS is feasible to $\mathcal{P}_2$ because the "**if statement**" in step 6 of $\mathsf{DPoS-MVNO}$ and step 4 & 6 of $\mathsf{DPoS-TNT}_n$ ensure that (4a) and (4b) can never be violated.
- From step 2 of $\mathsf{DPoS-TNT}_n$ we can find that $\forall c \in \mathcal{C}, \hat{\psi}_n \ge v_n - \sum_{c\in\mathcal{C}} d^c_n \cdot \hat{p}^{(n-1)}_c$. Because $\{\phi_c\}_{\forall c\in\mathcal{C}}$ defined in DPoS are non-decreasing positive functions, the following inequality

$$\hat{p}^{(N)}_c \ge \hat{p}^{(n)}_c \ge \hat{p}^{(n-1)}_c > 0,$$

holds. Thus $\forall n \in \mathcal{N}$, $\hat{\psi}_n \ge v_n - \sum_{c\in\mathcal{C}} d^c_n \hat{p}^{(N)}_c$ holds, where $\hat{p}^{(N)}_c$ is the final rental price of resource c, i.e., p_c in $\mathcal{P}_3$. Thus, (9b) not violated. Step 2 of $\mathsf{DPoS-TNT}_n$ ensures that $\hat{\boldsymbol{\psi}} \ge \mathbf{0}$ holds. Also note that in DPoS $\{\phi_c\}_{\forall c\in\mathcal{C}}$ are non-decreasing positive functions, which leads to $\hat{\boldsymbol{p}} \ge \mathbf{0}$ always holds. We thus prove that (9c) is not violated. Since both (9b) and (9c) are not violated, the second condition in proposition 4 holds for DPoS.

The proof of that the third condition holds is related to the design of the pricing functions $\{\phi_c\}_{\forall c\in\mathcal{C}}$. The following theorem shows that when $\{\phi_c\}_{\forall c\in\mathcal{C}}$ in DPoS are designed as (20) $\sim$ (23) indicate, the third condition in Proposition 4 holds.

Theorem 1. *(Adapted from Theorem 4.1 of [38]) When $\{f_c\}_{\forall c\in\mathcal{C}}$ are linear and $\{0 < \underline{\varsigma_c} < \underline{p_c}\}_{\forall c\in\mathcal{C}}$ holds, if $\forall c \in \mathcal{C}$, the pricing function ϕ_c in DPoS has the form:*

$$\phi_c(y) = \begin{cases} \underline{p_c} & y \in [0, w_c)\\ \varphi_c(y) & y \in [w_c, 1]\\ +\infty & y \in (1, +\infty), \end{cases} \tag{20}$$

where

$$w_c \in \Big[0, \arg\max_{y\ge 0} \underline{p_c} y - \tilde{f}_c(y)\Big], \tag{21}$$

is a threshold that satisfies

$$F_{\underline{p_c}}(w_c) \ge \frac{1}{\alpha_c} h_c(\underline{p_c}), \tag{22}$$

and $\varphi_c(y)$ is an increasing function that satisfies

$$\begin{cases} \varphi_c'(y) \le \alpha_c \cdot \frac{\varphi_c(y) - f_c'(y)}{h_c'(\varphi_c(y))}, \text{if } y \in (w_c, 1)\\ \varphi_c(w_c) = \underline{p_c}\\ \varphi_c(1) \ge \overline{p_c} + \sum_{c'\in\mathcal{C}\setminus\{c\}} h_{c'}(\overline{p_{c'}}), \end{cases} \tag{23}$$

then DPoS is $\max_{c\in\mathcal{C}}\alpha_c$-competitive.

Proof. Assume that $\forall c \in \mathcal{C}, w_c = \sum_{n=1}^k d^c_n$, which means that k is the number of tenants such that the total resource rented out of type c equals w_c. Substitute the definition of $F_{p_c}(\cdot)$ into (22), we have

$$\underline{p_c} \cdot \Big(\sum_{n=1}^k d^c_n\Big) - \tilde{f}_c\Big(\sum_{n=1}^k d^c_n\Big) \ge \frac{1}{\alpha_c} h_c(\underline{p_c}).$$

Because $\alpha_c \ge 1$ holds for each $c \in \mathcal{C}$ and $\hat{\boldsymbol{\psi}} \ge \mathbf{0}$, the above inequality leads to

$$\Big(1-\frac{1}{\alpha_c}\Big)\sum_{n=1}^{k}\hat{\psi}_n + \sum_{c\in\mathcal{C}}\Big(\underline{p_c}\cdot\Big(\sum_{n=1}^{k}d_n^c\Big)-\tilde{f}_c\Big(\sum_{n=1}^{k}d_n^c\Big)\Big) \geq \sum_{c\in\mathcal{C}}\frac{1}{\alpha_c}h_c(\underline{p_c}).$$

Further, we have

$$\sum_{n=1}^{k}\Big(\hat{\psi}_n+\sum_{c\in\mathcal{C}}\underline{p_c}\cdot d_n^c\Big)-\sum_{c\in\mathcal{C}}\tilde{f}_c\Big(\sum_{n=1}^{k}d_n^c\Big) \geq \min_{c'\in\mathcal{C}}\frac{1}{\alpha_{c'}}\Big(\sum_{n=1}^{k}\hat{\psi}_n+\sum_{c\in\mathcal{C}}h_c(\underline{p_c})\Big). \tag{24}$$

The pricing function in (20) indicates that the requirements of all tenants will be satisfied as long as each resource c's utilization is below w_c. Thus, we have $\hat{y}_c^{(k)}=\sum_{n=1}^{k}d_n^c=w_c$. Besides, the rental price of resource c these tenants experienced is the same, i.e., $\underline{p_c}$. Therefore, (24) indicates $\mathbf{\Theta}_{\mathcal{P}_2}^k\geq\min_{c\in\mathcal{C}}\frac{1}{\alpha_c}\mathbf{\Theta}_{\mathcal{P}_3}^k$. Meanwhile, it is obvious that w_c must be less than or equal to $\arg\max_{y\geq 0}\underline{p_c}y-\tilde{f}_c(y)$ because the rental price must be larger than or equal to the *marginal cost* $f_c'(w_c)$ (the result is immediate with (16)).

The above has proved that (17) holds. In the following, we prove (18) holds. The change in the objective of $\mathcal{P}_2$ when a new tenant n arrives is

$$\mathbf{\Theta}_{\mathcal{P}_2}^n-\mathbf{\Theta}_{\mathcal{P}_2}^{n-1}=\hat{\psi}_n+\sum_{c\in\mathcal{C}}\phi_c(\hat{y}_c^{(n-1)})\Big(\hat{y}_c^{(n)}-\hat{y}_c^{(n-1)}\Big)-\sum_{c\in\mathcal{C}}\Big(\tilde{f}_c(\hat{y}_c^{(n)})-\tilde{f}_c(\hat{y}_c^{(n-1)})\Big).$$

The change in the objective of $\mathcal{P}_3$ when a new tenant n arrives is

$$\mathbf{\Theta}_{\mathcal{P}_3}^n-\mathbf{\Theta}_{\mathcal{P}_3}^{n-1}=\hat{\psi}_n+\sum_{c\in\mathcal{C}}\Big(h_c(\hat{p}_c^{(n)})-h_c(\hat{p}_c^{(n-1)})\Big).$$

To guarantee (18) holds, it is *equivalent* to guarantee the following *per-resource* inequality

$$\phi_c(\hat{y}_c^{(n-1)})\Big(\hat{y}_c^{(n)}-\hat{y}_c^{(n-1)}\Big)-\Big(\tilde{f}_c(\hat{y}_c^{(n)})-\tilde{f}_c(\hat{y}_c^{(n-1)})\Big)\geq\frac{1}{\alpha_c}\Big(h_c(\hat{p}_c^{(n)})-h_c(\hat{p}_c^{(n-1)})\Big).$$

Divide both side of the above inequality by $\hat{y}_c^{(n)}-\hat{y}_c^{(n-1)}$, we get

$$\phi_c(y_c)-\tilde{f}_c'(y_c)\geq\frac{1}{\alpha_c}\cdot h_c'(\phi_c(y_c))\cdot\phi_c'(y_c) \tag{25}$$

when $y_c\in[w_c,1)$. This means that if $\forall y_c\in[w_c,1)$, (25) holds, the incremental inequality in (18) holds for all $y_c\in[w_c,1)$ for each type of resource. This result is exactly the first segment of (23). The second segment of (23) is to ensure the continuity of ϕ_c. The third segment of (23) is to make up the missing proof for (18) on the exact point $y_c=1$, which can be derived by the deformation of

$$\underline{p_c}w_c+\int_{w_c}^{1}\phi_c(y_c)dy_c-\tilde{f}_c(1)\geq\frac{1}{\alpha_c}\sum_{c\in\mathcal{C}}h_c(\overline{p_c}).$$

The above inequality is obtained by taking integration of both sides of (25).

So far, we have proved that when $\{\phi_c\}_{\forall c\in\mathcal{C}}$ in DPoS are designed as (20) $\sim$ (23) suggested, the thrid condition in Proposition 4, i.e., (17) and (18) hold. Thus, we have proved that DPoS is $\max_{c\in\mathcal{C}}\alpha_c$-competitive. □

In the following, we verify that the design of $\{\phi_c\}_{c\in\mathcal{C}}$ in DPoS when $\{f_c\}$ are linear, which is demonstrated in (12), satisfies the requirements defined in (20) $\sim$ (23). When $f_c(y)=q_cy$ and $q_c>0$, the conjugate $h_c(p_c)$ defined in (7) is given by

$$h_c(p_c)=\begin{cases}0 & p_c\in[0,q_c]\\ p_c-q_c & p_c\in(q_c,+\infty)\end{cases} \tag{26}$$

Note that $0<q_c<\underline{p_c}\leq\overline{p_c}$. In this case, (22) is equal to

$$\underline{p_c}w_c-f(w_c)\geq\frac{1}{\alpha_c}\Big(\underline{p_c}-f_c(1)\Big),$$

which indicates $w_c\geq\frac{1}{\alpha_c}$. (23) is thus equal to

$$\begin{cases}\varphi_c(y)-f_c'(y)\geq\frac{1}{\alpha_c}\cdot\varphi_c'(y)\cdot h_c'(\varphi_c(y)), w_c<y<1\\ \varphi_c(w_c)=\underline{p_c}\\ \varphi_c(1)=\sum_{c\in\mathcal{C}}\overline{p_c}-\sum_{c'\in\mathcal{C}\setminus\{c\}}q_{c'}.\end{cases}$$

To minimize α_c, it suffices to set w_c as $1/\alpha_c$ and thus the above BVP leads to (12) and (13).

The above analysis leads to the following theorem immediately.

Theorem 2. *When the cost functions $\{f_c\}_{\forall c\in\mathcal{C}}$ are linear and $\{0<\underline{\varsigma_c}<\underline{p_c}\}_{\forall c\in\mathcal{C}}$ holds, the competitive ratio α DPoS achieves is the optimal one over all possible online algorithms. Further, its value is*

$$\alpha=\max_{\forall c\in\mathcal{C}}\alpha_c=\max_{\forall c\in\mathcal{C}}\frac{1}{w_c}, \tag{27}$$

where w_c is defined in (13).

5 Experimental Results

In this section, we conduct extensive simulation experiments to evaluate the effectiveness and efficiency of DPoS. We first verify the performance of DPoS against several popular algorithms and handcrafted benchmarking policies on social welfare, efficiency, and competitive ratio. Then, we analyze the impact of several system parameters.

We summarize the key findings of our experiments as follows, and details can be found in Sections 5.2 and 5.3.

- DPoS not only achieves the highest social welfare among all the online algorithms compared, but also shows the *close-to-offline-optimal* performance, especially when the number of tenants is not more than 100 and the number of resource type is 1.
- In most cases, the ratio of the optimal social welfare to the social welfare achieved by DPoS (fluctuate between 1.00 and 2.57) is far less than the worst-case

TABLE 2
Default Parameter Settings

Parameter	Value	Parameter	Value
N	100	C	3
$\{d_n^c\}_{\forall c\in\mathcal{C}}$	$\sim N(\mu=\frac{1}{N},\sigma=\frac{1}{N^2})$	l_n^h	$\sim U(2,6)$
$\mathcal{S}_n$	$\sim N(\mu=10^6,\sigma=10^5)$	$\Pr(l_n^0)$	$\approx 40\%$
q_c	$\sim U(\frac{1}{6}\underline{p_c},\frac{5}{6}\underline{p_c})$	σ_n	identity

guarantee, i.e., the competitive ratio calculated by (13) and (27) (fluctuate between 5.82 and 8.54).

- DPoS is insensitive to environment parameters such as the distribution of $\{d_n^c\}_{\forall c\in\mathcal{C}}$ and the value of the coefficient of the linear cost, $\{q_c\}_{\forall c\in\mathcal{C}}$.
- DPoS achieves a satisfactory balance between the overheads (corss-agent communication data size, algorithm's running time, etc.) and the performance.

5.1 Experiment Setup

By default, we set the number of tenants N as 100. We also set the number of types of resources as 3 in default because the resources can be roughly divided into computation, storage, and forwarding/bandwidth. Note that 100 and 3 are only default settings. In Sections 5.2 and 5.3, we will analyze the scalability of DPoS extensively.

For each tenant n, $\{d_n^c\}_{\forall c\in\mathcal{C}}$ is uniformly sampled from the Gaussian distribution $N(\mu=\frac{1}{N},\sigma=\frac{1}{N^2})$. The pay level l_n is randomly sampled from $[2,6]$. The highest level of QoS, denoted by l_n^h, is randomly sampled from $U(2,6)$. The lowest level of QoS, denoted by l_n^0, is *free user level*. We set the percentage of free users near 40 percent for each tenant [48]. Moreover, the remaining users are randomly assigned to a QoS level according to the pyramid structure. The higher the QoS level, the fewer the users. The payment of user $s\in\mathcal{S}_n$ is proportional to his QoS level. By default, $\forall n\in\mathcal{N},\forall s\in\mathcal{S}_n$, we set σ_n as identity function. For each type of resource, we take linear cost defined in (11). By default, $\forall c\in\mathcal{C}$, q_c is randomly chosen from $[\frac{1}{6}\underline{p_c},\frac{5}{6}\underline{p_c}]$. Important parameter settings are summarized in Table 2.

DPoS is compared with the following algorithms. Thereinto, *CVX* and *Heuristic* are used to obtain the approximate optimal of the offline problem $\mathcal{P}_1$. *SCPA* [41] is a state-of-the-art auction-based algorithm. We also design online algorithms *Myopic Slicing* and *Random Slicing* as baselines.

- *CVX* (offline & centralized): This refers to the algorithm behinds CVXPY.[5] We use this as a professional solver to obtain the approximately optimal solution of the global offline problem $\mathcal{P}_1$.
- *Heuristic* (offline & centralized): We take Genetic Algorithm (GA) to obtain the approximate optimal solution of $\mathcal{P}_1$.
- *SCPA* (offline & decentralized) [41]: To adapt this algorithm to our model, we made some simple deformation. In this algorithm, all the tenants and the MVNO get together. The bids are the utilities. Specifically, in each bidding around, each tenant calculate its utility. If the utility is positive, it sends $x_n=1$ and $\{d_n^c\}_{\forall c\in\mathcal{C}}$ to the MVNO. The MVNO selects the exact tenant which can maximize the its own utility and accepts the transaction if resource surplus is satisfied. All the left tenants are rejected. The procedure ends when no tenant has the willingness to bid.
- *Myopic Slicing (MS)* (online & decentralized): This algorithm is almost the same with DPoS, expect the pricing functions. The pricing functions are designed as follows: $\forall c\in\mathcal{C},\phi_c'(y)\triangleq\frac{p_c+\overline{p_c}}{C}y$ when $y\le 1$, otherwise $+\infty$.
- *Random Slicing (RS)* (online): Each time when a new tenant arrives, randomly set x_n as 0 or 1. Note that if $x_n=1$, the resource surplus must be satisfied.

The following analyze is based on the average returns of 1000 trials.

5. https://www.cvxpy.org/

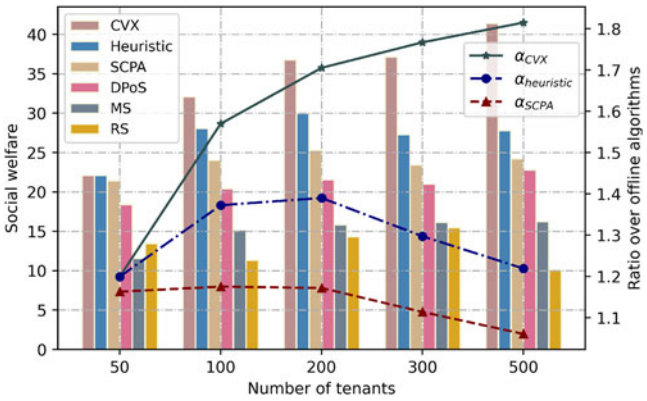


Fig. 3. The social welfare achieved by each algorithm and the ratio of social welfare achieved by each offline algorithm to DPoS, under different number of tenants.

5.2 Performace Verification

We first analyze the performance under different scales of tenants. As shown in Fig. 3, all the offline algorithms outperform the online algorithms. Therewith, CVX achieves the highest social welfare whatever the number of tenants. In the following, we will simply take CVX as the optimal solution. It is interesting to find that both Heuristic and SCPA show a trend of performance decline as the number of tenants increase. For Heuristic, as the solution space grows exponentially with the increase of tenant size, it becomes more difficult to find the approximate optimal solution under the constraints of iteration times and population size. When the scale of tenants grows, the performance of all the online algorithms present a rising trend. This is becasue the transaction success rate increases (although not by as much) with scale under the well-designed pricing functions. Further, we can find that DPoS not only achieves the highest social welfare among all the online algorithms, but also shows the *close-to-offline-optimal* performance. Specifically, we define the indicator α_{CVX}, $\alpha_{heuristic}$, and α_{SCPA}, where each is the ratio of the social welfare achieved by CVX, Heuristic, and SCPA to DPoS, respectively. From Fig. 3 we find that even in the worst case ($N=500$), the gap between CVX and DPoS is only $0.815\times$. This ratio is much better (lower) compared with previous work [49].

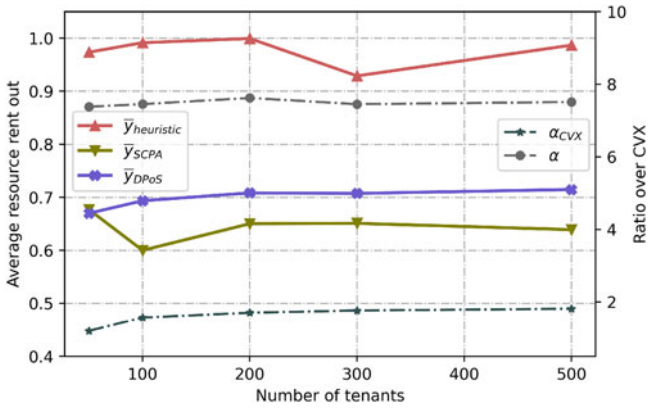


Fig. 4. Left y-axis: The average rental rate over 3 kinds of resources of Heuristic, SCPA, and DPoS. We do not draw the rental rate of CVX because the value is close to 1 under any circumstances. Right y-axis: the comparison of α_{CVX} and the theoretical worst-case competitive ratio α.

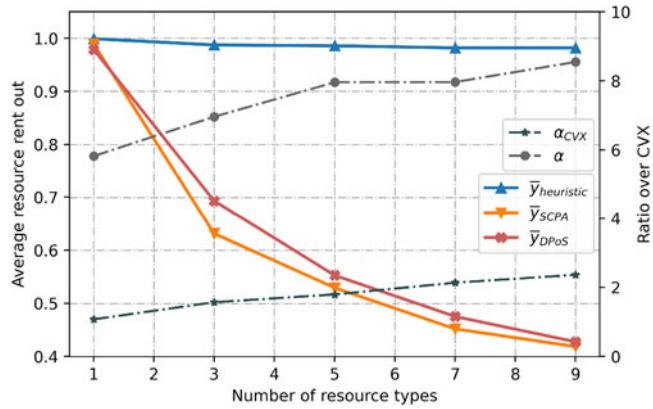


Fig. 6. Left y-axis: The average rental rate over 3 kinds of resources of Heuristic, SCPA, and DPoS. Right y-axis: The comparison of α_{CVX} and the theoretical worst-case competitive ratio α.

Compared with the popular offline Heuristic (GA), the gap is $0.390\times$ at the peak ($N = 200$). Compared with the state-of-the-art offline auciton-based algorithm SCPA [41], the gap is $0.175\times$ at the peak ($N = 100$). Becasue of the performance downgrade of Heuristic and SCPA, the ratio $\alpha_{heuristic}$ and α_{SCPA} shows a tendency to increase first and then decrease.

Fig. 4 demonstrates that Heuristic has a near-to-1 rental rate whatever the number of tenants but SCPA's and DPoS's rental rate are much lower (64.37 and 69.89 percent in average, respectively). However, from Fig. 3 we have concluded that the performance of Heuristic is much inferior to the optimal especially when N is 500. Thus, we can conclude that there is no *linear* relationship between the sum of net profits and the transaction success rate. In fact, this conclusion can also be draw by observing the analytic form of social welfare defined in $\mathcal{P}_1$. Besides, the scale of tenants has no significant impact on the rental rate, whether it is an offline algorithm, or DPoS. Another interesting point is that under normal circumstances, the worst-case theoretical guarantee, i.e., the competitive ratio calculated according to (13) and (27), is far from need.

In the following we analyze the performance of DPoS under different scale of resource types C. From Fig. 5, first, we find that DPoS is still the best online algorithm and has a close performance to Heuristic and SCPA. When $C = 1$, DPoS can achieve near-to-offline-optimal performance! Second, all the algorithms show a downward trend when the number of resource types increase, except CVX. This is becasue each tenant has requirements on all the resource types, and the increase in resource types significantly reduces the probability of requirements being satisfied. Ulteriorly, the transaction success rate reduces significantly. The phenomena can also be found in Fig. 6. For online scenarios, the phenomena is amplified by the randomness of arrival sequence of tenants. Thus, online algorithms perform more unsatisfied. Even though, the advantage of DPoS is clear. In the worst case, i.e., when $C = 9$, the ratio α_{CVX} is 2.37, which is still acceptable for online algorithms. It even outperforms the offline algorithm Heuristic when C is 5 and 7 by 18.00 and 13.40 percent, respectively.

Fig. 7 demonstrates the impact of scales of tenants and resource types on the performance of DPoS comprehensively. In general, the gap between DPoS and the offline optimal increases with the increasing scale of the problem. When C is 1 and N is 50, what DPoS achieves is exactly the offline optimal. When C is 9 and N is 500, the gap is the highest, which reaches $1.57\times$. Further, we can find that the ratio grows faster with resource types than with tenant size. We leave the design of resource type-scalable pricing functions as future work. Table 3 comapares all the algorithms from multiple angles, including social welfare achieved, cross-agent communica-

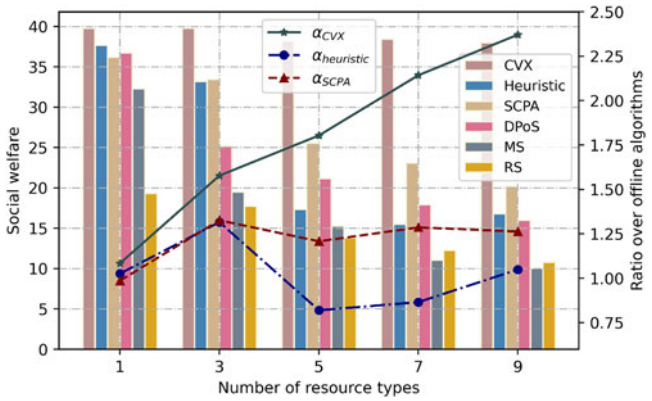


Fig. 5. The social welfare achieved by each algorithm and the ratio of social welfare achieved by each offline algorithm to DPoS, under different number of resource types.

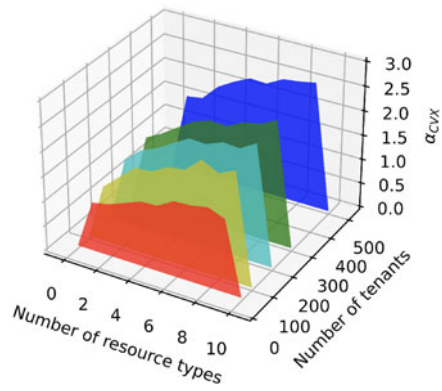


Fig. 7. The ratio of social welfare achieved by DPoS to the optimal, CVX, under different scales of tenants and resource types.

TABLE 3
Comparsion of Transferred Data Size and Algorithm's Running Time Under Default Parameter Settings

	CVX	Heuristic	SCPA	DPoS	MS	RS
inputform	offline	offline	offline	online	online	online
architecture	centralized	centralized	decentralized	decentralized	decentralized	-
transferreddatasize	4.16KB	4.16KB	4.16KB	1.92KB	1.92KB	-
runningtime	78.81	2172.35	24.43	1	0.93	0.48
α_{CVX}	1	1.199	1.189	1.578	2.04	2.47

tion data size, and algorithm running time. The amount of data transferred by the decentralized online algorithm refers to the amount of data communicated between tenants and the MVNO. Meanwhile, the amount of data transferred by the centralized algorithm is all data related to problem $\mathcal{P}_1$. The data size is calculated as 4 bytes for each value. Note that we normalize the running time of DPoS to 1. We can find that the superiority of CVX and Heuristic are based on a lot of computing time overhead. By contrast, DPoS achieves a satisfactory balance between the overheads the performance. In addition to the 4-th line of Table 3, Figs. 8 and 9 also verify the linear algorithmic runtime of DPoS intuitively.

5.3 Sensitivity Analysis

In this subsection, we analyze the sensitivity of DPoS under different environment parameters settings.

Fig. 10 demonstrates the impact of tenants' resource requirements. The x-axis is the mean value μ of the Normal distribution $N(\mu, \sigma = \frac{1}{N^2})$ where N is 100. We find that when the resource requirements increase, the transaction success rate decreases, which further decreases the social welfare achieved. It is interesting that the social welfare achieved by CVX also decreases significantly when tenants' resource requirements increase. This phenomenon indicates that the competition among tenants for resources significantly reduces the feasible solution space. Even so, the ratio on social welfare is stable no matter how the resource requirements change.

Figs. 11 and 12 demonstrate the impact of $\{q_c\}_{\forall c \in \mathcal{C}}$ and $\{l_n\}_{\forall n \in \mathcal{N}}$. We can find that the ratio on social welfare has a smooth variation. Considering that their impacts are minor, no more detailed discussion will be launched.

All the experiment results in this subsection show the robustness of DPoS.

6 Related Works

Network slicing is widely accepted as an architectural enabling technology for 5G by industry and standardization communities [1], [2], [3], [4]. The idea is to *slice* the physical resources of the mobile networks into logical network functions, and orchestrate them to support diversified over-the-top services. Previous works on network slicing mainly focus on the architectural aspects, while efficient resource allocation and sharing, which has been identified as a key issue by the Next Generation Mobile Network (NGMN) alliance [50], lags behind.

A number of studies have emerged in recent years to fill the gap, especially for mobile network slicing [10], [11] [17] and core network slicing [14], [15], [16]. Overall, these works formulate a non-convex combinatorial problem to maximize the utilities of involved business players. Take [10] as an example, the authors defined the utility according to the satisfaction of multiple slice resource demands (SRDs). They formulated the resource sharing problem as a Mixed Integer Linear Programming (MILP) and proposed a two-step approach (provisioning-and-deployment) to solve it efficiently. Similarly, Caballero *et al.* proposed a dynamic resource allocation algorithm based on the weighted proportionally fairness, also for the RAN resources [11]. Based on this algorithm, they devised a practical approach with limited computational information and handoff overheads. Further, the authors verified the approximate optimality of the approach with both theoretical proof and extensive simulations. In addition to the heuristics designed by the above mentioned works, AI-based optimization has been gaining in popularity. For example, Yan *et al.* resorted to deep reinforcement learning (DRL) to formulate an intelligent resource scheduling strategy, iRSS, for 5G RAN slicing [21]. They take deep neural networks to perform large

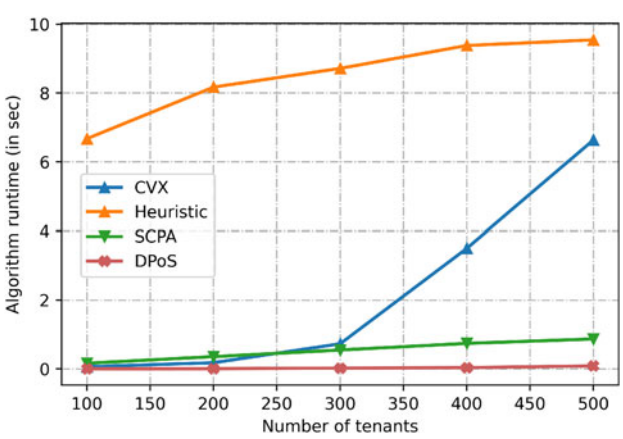


Fig. 8. The runtime of each algorithm under different number of tenants.

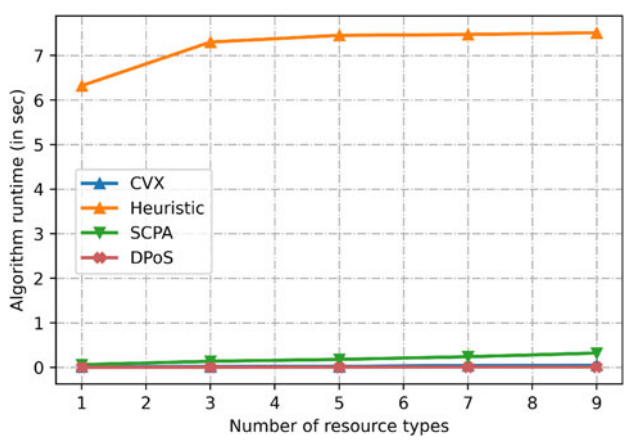


Fig. 9. The runtime of each algorithm under different number of resource types.

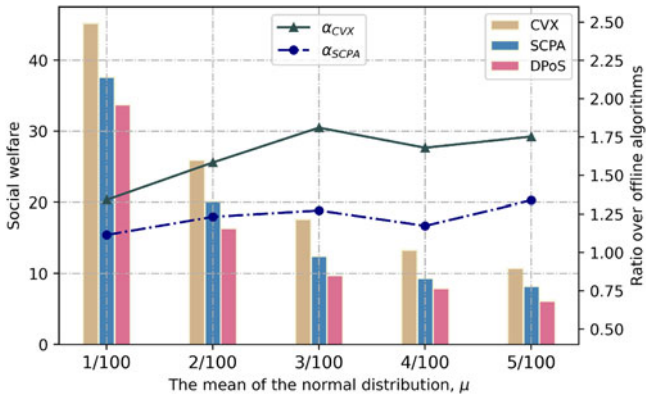


Fig. 10. The social welfare achieved by each algorithm and the ratio of social welfare achieved by CVX and SCPA to DPoS, under different sampling of $\{d_n^c\}_{\forall c \in \mathcal{C}}$.

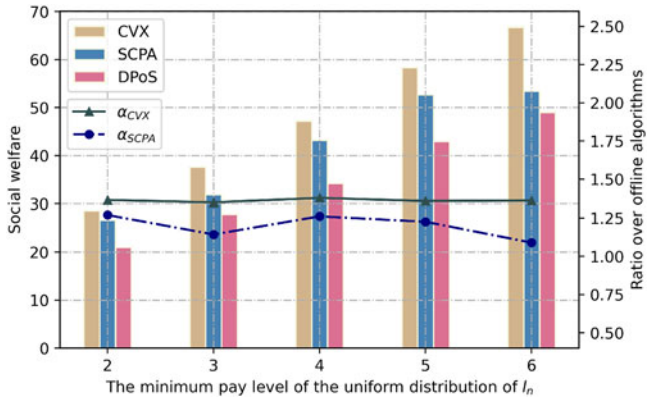


Fig. 12. The social welfare achieved by each algorithm and the ratio of social welfare achieved by CVX and SCPA to DPoS, under different sampling of pay levels $\{l_n\}_{\forall n \in \mathcal{N}}$.

time-scale resource allocation while the reinforce agent performs online resource scheduling to predict network states and dynamics. Likewise, the authors of [18] also designed a DRL-based algorithm, to perform corss-slice resource sharing.

In addition to the centralized and fine-tuned algorithms, a substantial literature designed the network slicing algorithms based on economic frameworks, especially the auction-related mechanisms [5], [7] [8], [33], [41], [51]. These algorithms are usually decentralized, easy-to-use and simply constructed. In these works, the tenants sequentially compete and bid for the network resources. The utilization of auction mechanism usually integrate tightly with dynamic pricing and game model [35]. For example, Wang *et al*. solved the joint efficiency and revenue maximization problem with a varying-pricing policy [30]. They designed a decentralized algorithm, run by each player, to maximize the net social welfare. In [41], the authors designed a non-cooperative game where each tenant reacts to the user allocations of the other tenants so as to maximize its own utility selfishly. Existing works mainly resort to Fisher market [27], where strategic players anticipate the impact of their bids. Besides, VCG-Kelly mechanisms and their derivatives [29] are also popular for slice resource allocation and sharing [41], [49]. In Kelly's mechanism, the bidders bid for prices, and the resources are allocated to them according to their bids. In VCG mechanism, in a different way, the bids are the utility of involved players. We find that existing auction-based works are mainly designed for offline markets, where all the tenants participate the auction and bid for their interests sequentially. Even so, we still discover an online auction-based resource allocation algorithm, proposed in [49]. The authors model the slicing resource allocation problem as an online winner determination problem, with aim to maximize the social welfare of auction participants. However, what the authors of [49] proposed is a centralized algorithm, where the bidding and privacy-relevant information has to be collected by the MVNO.

Our work is based on the posted price mechanism [36], under the principle of *take-it-or-leave-it*. Compared with fined-tuned heuristics and DRL-based works, our algorithm has fairly low complexity and is well-suited for online network slicing scenarios. Besides, the time-consuming repeat bidding between tenants and the MVNO is not required compared with auction-based works. In addition, our algorithm provides each business player an agent, which can be deployed in a realistic online market directly without any modification.

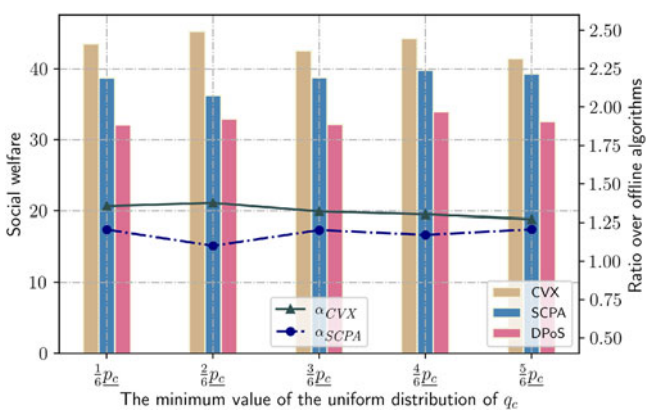


Fig. 11. The social welfare achieved by each algorithm and the ratio of social welfare achieved by CVX and SCPA to DPoS, under different sampling of $\{q_c\}_{\forall c \in \mathcal{C}}$.

7 Concluding Remarks

We presented a decentralized and low-complexity online slicing algorithm, DPoS, by virtue of the primal-dual approach and posted price mechanism. Our goal was to address the problem of the high complexity, privacy leakage, and unrealistic offline setting of current network slicing algorithms. We first presented the global offline social welfare maximization problem. Then, we relax the original combinatorial problem to a convex primal problem and give its dual. Based on the alternative update of primal and dual variables, DPoS maximizes the social welfare with a $O\big(\max_{c\in\mathcal{C}}\{\ln\sum_{c'\in\mathcal{C}}(\overline{p_{c'}}-q_{c'})-\ln(\underline{p_c}-q_c)\}\big)$ gap in worst case. By giving back the decision-making power to each player, DPoS stops the privacy leakage from the source. This decentralized property also erases the heavy burden to solve a centralized offline optimization algorithm, which is often of high complexity. In addition to the efficiency, the competitive ratio of DPoS is the optimal over all the online algorithms. The extensive simulation further verify that DPoS can not only achieve close-to-offline-optimal performance, but also have much lower algorithmic overheads compared with contrast algorithms.

ACKNOWLEDGMENTS

This work was supported in part by the National Science Foundation of China under Grants U20A20173 and 61772461, in part by the National Key Research and Development Program of China under Grant 2019YFD1101105, and in part by the Natural Science Foundation of Zhejiang Province under Grant LR18F020003. The work of Schahram Dustdar was supported by the Zhejiang University Deqing Institute of Advanced technology and Industrilization.

REFERENCES

[1] 5G PPP Architecture Working Group, "View on 5G architecture: Version 3.0," Feb. 2020. [Online]. Available: https://doi.org/10.5281/zenodo.3265031

[2] M. Richart, J. Baliosian, J. Serrat, and J. Gorricho, "Resource slicing in virtual wireless networks: A survey," *IEEE Trans. Netw. Serv. Manage.*, vol. 13, no. 3, pp. 462–476, Sep. 2016.

[3] X. Foukas, G. Patounas, A. Elmokashfi, and M. K. Marina, "Network slicing in 5G: Survey and challenges," *IEEE Commun. Mag.*, vol. 55, no. 5, pp. 94–100, May 2017.

[4] I. Afolabi, T. Taleb, K. Samdanis, A. Ksentini, and H. Flinck, "Network slicing and softwarization: A survey on principles, enabling technologies, and solutions," *IEEE Commun. Surv. Tut.*, vol. 20, no. 3, pp. 2429–2453, Jul.–Sep. 2018.

[5] D. Bega, M. Gramaglia, A. Banchs, V. Sciancalepore, K. Samdanis, and X. Costa-Perez, "Optimising 5G infrastructure markets: The business of network slicing," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, 2017, pp. 1–9.

[6] C. Marquez, M. Gramaglia, M. Fiore, A. Banchs, and X. Costa-Perez, "How should i slice my network?: A multi-service empirical evaluation of resource sharing efficiency," in *Proc. Annu. Int. Conf. Mobile Comput. Netw.*, New York, NY, USA, 2018, pp. 191–206.

[7] X. Zhang, Z. Huang, C. Wu, Z. Li, and F. C. M. Lau, "Online stochastic buy-sell mechanism for VNF chains in the NFV market," *IEEE J. Sel. Areas Commun.*, vol. 35, no. 2, pp. 392–406, Feb. 2017.

[8] S. Gu, Z. Li, C. Wu, and C. Huang, "An efficient auction mechanism for service chains in the NFV market," in *Proc. IEEE INFOCOM 35th Annu. Int. Conf. Comput. Commun.*, 2016, pp. 1–9.

[9] S. Vassilaras *et al.*, "The algorithmic aspects of network slicing," *IEEE Commun. Mag.*, vol. 55, no. 8, pp. 112–119, Aug. 2017.

[10] Q. Luu, S. Kerboeuf, A. Mouradian, and M. Kieffer, "A coverage-aware resource provisioning method for network slicing," *IEEE/ACM Trans. Netw.*, vol. 28, no. 6, pp. 2393–2406, Dec. 2020.

[11] P. Caballero, A. Banchs, G. de Veciana, and X. Costa-Pérez, "Multi-tenant radio access network slicing: Statistical multiplexing of spatial loads," *IEEE/ACM Trans. Netw.*, vol. 25, no. 5, pp. 3044–3058, Oct. 2017.

[12] H. Zhao, S. Deng, Z. Liu, J. Yin, and S. Dustdar, "Distributed redundancy scheduling for microservice-based applications at the edge," *IEEE Trans. Serv. Comput.*, early access, Aug. 3, 2020, doi: 10.1109/TSC.2020.3013600.

[13] S. Deng *et al.*, "Optimal application deployment in resource constrained distributed edges," *IEEE Tran. Mobile Comput.*, vol. 20, no. 5, pp. 1907–1923, May 2021.

[14] D. A. Chekired, M. A. Togou, L. Khoukhi, and A. Ksentini, "5G-slicing-enabled scalable SDN core network: Toward an ultra-low latency of autonomous driving service," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 8, pp. 1769–1782, Aug. 2019.

[15] M. R. Sama, X. An, Q. Wei, and S. Beker, "Reshaping the mobile core network via function decomposition and network slicing for the 5G era," in *Proc. IEEE Wirel. Commun. Netw. Conf.*, 2016, pp. 1–7.

[16] D. Sattar and A. Matrawy, "Optimal slice allocation in 5G core networks," *IEEE Netw. Lett.*, vol. 1, no. 2, pp. 48–51, Jun. 2019.

[17] P. L. Vo, M. N. H. Nguyen, T. A. Le, and N. H. Tran, "Slicing the edge: Resource allocation for RAN network slicing," *IEEE Wirel. Commun. Lett.*, vol. 7, no. 6, pp. 970–973, Dec. 2018.

[18] X. *et al.*, "Multi-tenant cross-slice resource orchestration: A deep reinforcement learning approach," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 10, pp. 2377–2392, Jun. 2019.

[19] S. Deng, C. Zhang, C. Li, J. Yin, S. Dustdar, and A. Y. Zomaya, "Burst load evacuation based on dispatching and scheduling in distributed edge networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 8, pp. 1918–1932, Aug. 2021.

[20] B. Han, J. Lianghai, and H. D. Schotten, "Slice as an evolutionary service: Genetic optimization for inter-slice resource management in 5G networks," *IEEE Access*, vol. 6, pp. 33 137–33 147, Feb. 2018.

[21] M. Yan, G. Feng, J. Zhou, Y. Sun, and Y. Liang, "Intelligent resource scheduling for 5G radio access network slicing," *IEEE Trans. Veh. Technol.*, vol. 68, no. 8, pp. 7691–7703, Aug. 2019.

[22] X.-L. Huang, X. Ma, and F. Hu, "Machine learning and intelligent communications," *Mobile Netw. Appl.*, vol. 23, no. 1, pp. 68–70, 2018.

[23] S. Deng, H. Zhao, W. Fang, J. Yin, S. Dustdar, and A. Y. Zomaya, "Edge intelligence: The confluence of edge computing and artificial intelligence," *IEEE Internet Things J.*, vol. 7, no. 8, pp. 7457–7469, Aug. 2020.

[24] J. Sakuma, S. Kobayashi, and R. N. Wright, "Privacy-preserving reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2008, pp. 864–871.

[25] X. Liu, R. H. Deng, K. K. RaymondChoo, and Y. Yang, "Privacy-preserving reinforcement learning design for patient-centric dynamic treatment regimes," *IEEE Trans. Emerg. Top. Comput.*, vol. 9, no. 1, pp. 456–470, Jan.–Mar. 2021.

[26] Y. K. Cheung, R. Cole, and Y. Tao, "Dynamics of distributed updating in fisher markets," in *Proc. ACM Conf. Econ. Comput.*, 2018, pp. 351–368.

[27] R. Cole *et al.*, "Convex program duality, fisher markets, and nash social welfare," in *Proc. ACM Conf. Econ. Comput.*, 2017, pp. 459–460.

[28] A. Mu'Alem and N. Nisan, "Truthful approximation mechanisms for restricted combinatorial auctions," *Games Econ. Behav.*, vol. 64, no. 2, pp. 612–631, 2008.

[29] S. Yang and B. Hajek, 'VCG-kelly mechanisms for allocation of divisible goods: Adapting VCG mechanisms to one-dimensional signals," *IEEE J. Sel. Areas Commun.*, vol. 25, no. 6, pp. 1237–1243, Aug. 2007.

[30] G. Wang, G. Feng, W. Tan, S. Qin, R. Wen, and S. Sun, "Resource allocation for network slices in 5G with network resource pricing," in *Proc. IEEE Global Commun. Conf.*, 2017, pp. 1–6.

[31] M. Jiang, M. Condoluci, and T. Mahmoodi, "Network slicing in 5G: An auction-based model," in *Proc. IEEE Int. Conf. Commun.*, 2017, pp. 1–6.

[32] Y. Zhang and M. Guizani, *Game Theory for Wireless Communications and Networking*. New York, NY, USA: Cambridge Univ. Press, 2011.

[33] S. D. Oro, F. Restuccia, T. Melodia, and S. Palazzo, "Low-complexity distributed radio access network slicing: Algorithms and experimental results," *IEEE/ACM Trans. Netw.*, vol. 26, no. 6, pp. 2815–2828, Dec. 2018.

[34] GSM Association, "Official documentNG.116 – Generic network slice template v4.0," Nov. 2020. [Online]. Available: https://www.gsma.com/newsroom/wp-content/uploads//NG.116-v4.0-2.pdf

[35] L. Einav, C. Farronato, J. Levin, and N. Sundaresan, "Auctions versus posted prices in online markets," *J. Political Econ.*, vol. 126, no. 1, pp. 178–215, 2018.

[36] J. Correa, P. Foncea, R. Hoeksma, T. Oosterwijk, and T. Vredeveld, "Posted price mechanisms for a random stream of customers," in *Proc. ACM Conf. Econ. Comput.*, 2017, pp. 169–186.

[37] N. Buchbinder and J. S. Naor, "The design of competitive online algorithms via a primal–dual approach," *Found. Trends Theor. Comput Sci*, vol. 3, no. 2–3, pp. 93–263, 2009.

[38] X. Tan, B. Sun, A. Leon-Garcia, Y. Wu, and D. H. Tsang, "Mechanism design for online resource allocation: A unified approach," in *Proc. Abstracts SIGMETRICS/Perform. Joint Int. Conf. Meas. Model. Comput. Syst.*, 2020, pp. 11–12.

[39] Z. Huang and A. Kim, "Welfare maximization with production costs: A primal dual approach," *Games Econ. Behav.*, vol. 118, pp. 648–667, 2019.

[40] M. Leconte, G. S. Paschos, P. Mertikopoulos, and U. C. Kozat, "A resource allocation framework for network slicing," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, 2018, pp. 2177–2185.

[41] P. Caballero, A. Banchs, G. De Veciana, and X. Costa-Pérez, "Network slicing games: Enabling customization in multi-tenant mobile networks," *IEEE/ACM Trans. Netw.*, vol. 27, no. 2, pp. 662–675, Apr. 2019.

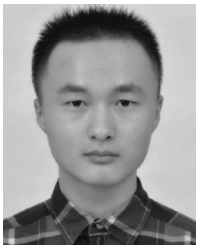
[42] W. Ma and D. Simchi-Levi, "Tight weight-dependent competitive ratios for online edge-weighted bipartite matching and beyond," in *Proc. ACM Conf. Econ. Comput.*, 2019, pp. 727–728.

[43] R. Wang, "Auctions versus posted-price selling," *Amer. Econ. Rev.*, vol. 83, no. 4, pp. 838–851, 1993.

[44] L. Einav, C. Farronato, J. Levin, and N. Sundaresan, "Auctions versus posted prices in online markets," *J. Political Econ.*, vol. 126, no. 1, pp. 178–215, 2018.
[45] R. Cooke and V. Arnold, *Ordinary Differential Equations*. Berlin, Germany: Springer, 1992.
[46] A. Borodin and R. El-Yaniv, *Online Computation and Competitive analysis*. New York, NY, USA: Cambridge Univ. Press, 2005.
[47] S. Boyd and L. Vandenberghe, *Convex Optimization*. New York, NY, USA: Cambridge Univ. Press, 2004.
[48] GUOSHENG Securities, "The way for video platforms to raise the ARPPU value is expected to co-exist with the pressure," May 2020. [Online]. Available: http://pg.jrj.com.cn/acc/Res/CN_RES/INDUS/2020/5/17/53fbbf09–3e88-4495-98fc-ba9ae7dbdf81.pdf.
[49] L. Liang, Y. Wu, G. Feng, X. Jian, and Y. Jia, "Online auction-based resource allocation for service-oriented network slicing," *IEEE Trans. Veh. Technol.*, vol. 68, no. 8, pp. 8063–8074, Aug. 2019.
[50] NGMN Alliance, "5G white paper 2," 2020. [Online]. Available: https://www.ngmn.org/wp-content/uploads/NGMN-5G-White-Paper-2.pdf
[51] H. Ding, J. Huang, H. Cao, and Y. Liu, "Improving cold music recommendation through hierarchical audio alignment," in *Proc. IEEE Annu. Sympo. Found. Comput. Sci.* 2016, pp. 77–82.

Hailiang Zhao received the BS degree from the School of Computer Science and Technology, Wuhan University of Technology, Wuhan, China, in 2019. He is currently working toward the PhD degree with the College of Computer Science and Technology, Zhejiang University, Hangzhou, China. His research interests include edge computing, service computing, and machine learning. He was the recipient of the Best Student Paper Award of the IEEE ICWS 2019.

Shuiguang Deng (Senior Member, IEEE) received the BS and PhD degrees in computer science from the College of Computer Science and Technology, Zhejiang University, China, in 2002 and 2007, respectively. He is currently a full professor with the College of Computer Science and Technology, Zhejiang University. He was a visiting scholar with the Massachusetts Institute of Technology in 2014 and Stanford University in 2015. He has authored or coauthored more than 100 papers in journals and refereed conferences. His research interests include edge computing, service computing, cloud computing, and business process management. He is currently an associate editor for the journal *IEEE Transactions on Services Computing*, *Knowledge and Information Systems*, *Computing*, and *IET Cyber-Physical Systems: Theory and Applications*. In 2018, he was the recipient of the Rising Star Award by the IEEE TCSVC. He is also a fellow of the IET.



Zijie Liu received the BS degree from the School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, China, in 2018. He is currently working toward the master's degree with the College of Computer Science and Technology, Zhejiang University, Hangzhou, China. His research interests include edge computing and software engineering.

Zhengzhe Xiang (Member, IEEE) received the BS and PhD degree of computer science and technology from Zhejiang University, Hangzhou, China. He was a visiting studentwith Karlstad University, Sweden, in 2018. He is currently a lecturer with Zhejiang University City College, Hangzhou, China. His research interests include service computing, cloud computing, and edge computing.

Jianwei Yin received the PhD degree in computer science from Zhejiang University (ZJU) in 2001. He was a visiting scholar with the Georgia Institute of Technology. He is currently a full professor with the College of Computer Science, ZJU. He has authored or coauthored more than 100 papers in top international journals and conferences. His current research interests include service computing and business process management. He is currently an associate editor for the *IEEE Transactions on Services Computing*.

Schahram Dustdar (Fellow, IEEE) is currently a full professor of computer science (informatics) with a focus on Internet technologies heading the Decentralized Systems Group, Vienna University of Technology. Since December 9, 2016, he has been the chairman of the Informatics Section, Academia Europaea. From 2004 to 2010, he was an honorary professor of information systems with the Department of Computing Science, University of Groningen, The Netherlands. From December 2016 to January 2017, he was a visiting professor with the University of Sevilla, Spain and from January to June 2017, he was a visiting professor with the University of California Berkeley, USA. He has been a member of the IEEE Conference Activities Committee (since 2016), of the Section Committee of Informatics of the Academia Europaea (since 2015), and a member of the Academia Europaea: The Academy of Europe, Informatics Section (since 2013). He is currently an associate editor for the *IEEE Transactions on Services Computing*, the*ACM Transactions on the Web*, and the *ACM Transactions on Internet Technology* and on the editorial board of *IEEE Internet Computing*. He is the editor-in-chief of *Computing* (an SCI-ranked journal of Springer). He was the recipient of the ACM Distinguished Scientist Award (2009) and the IBM Faculty Award (2012).

Albert Y. Zomaya (Fellow, IEEE) is currently a charted engineer, and also the chair professor of high performance computing and networking with the School of Computer Science, University of Sydney, and the director of the Centre for Decentralized and High Performance Computing. He has authored or coauthored more than 600 scientific papers and articles, and has authored or coauthored or edited more than 30 books. His research interests include parallel and decentralized computing and complex systems. He is currently the editor-in-chief of the *IEEE Transactions on Sustainable Computing and ACM Computing Surveys.* He is currently an associate editor for several leading journals. From 2011 to 2014, he was an editor-in-chief of the *IEEE Transactions on Computers*. He is also a fellow of the AAAS and IET.