PSACCF: Prioritized Online Slice Admission Control Considering Fairness in 5G/B5G Networks

Miao Dai, Member, IEEE, Long Luo, Member, IEEE, Jing Ren, Member, IEEE, Hongfang Yu, Member, IEEE, and Gang Sun, Member, IEEE

Abstract—5G/B5G is intended to support various services with the assistance of network slices, and each slice requires adequate resources to provide the negotiated service quality. The Slice Admission Control (SAC) algorithm is a necessity for Slice Providers (SPs) to guarantee the Quality of Service (QoS) and Quality of Experience (QoE) of each admitted request. In such circumstances, concerns regarding the priority of services and the fairness of resource allocation are meaningful topics. However, these two issues do not receive sufficient research attention simultaneously in the literature. Here, we study the SAC problem in 5G/B5G networks, aiming to enhance the fairness degree while satisfying the necessary priority requirements. We first reinterpret priority as a higher Cumulative Service Acceptance Ratio (CSAR) and adopt the uniformity of adjacent CSAR gaps to reflect fairness. The SAC problem is formulated as a nonlinear and nonconvex multiobjective optimization problem. Thus, we propose a heuristic algorithm called Prioritized Slice Admission Control Considering Fairness (PSACCF) to solve it. This approach introduces the resource efficiency of services to amend priority violations and then promotes fairness by setting the target CSARs for each service type and improving their actual CSARs. Numerous simulations are conducted to compare the performance of PSACCF with that of Moderate High Priority First (MHPF) and Aggressive High Priority First (AHPF). The results show that PSACCF can achieve at least a 33.6% improvement in fairness degree and a higher minimum average resource utilization while maintaining a nearly identical priority indicator to that of the compared methods.

Index Terms—5G/B5G, online slice admission control, priority, fairness, network slice.

I. INTRODUCTION AND MOTIVATIONS

FUTURE applications of 5G are believed to belong to three broad categories [1] namely, enhanced Mobile Broadband (eMBB), ultra-Reliable Low Latency Communication (uRLLC), and massive Machine Type Communication (mMTC). These categories focus on different performance indicators, and the types and amounts of resources consumed are also distinct. Specifically, eMBB requires support for high mobility and large communication bandwidth, uRLLC has strict latency and reliability limits, and mMTC requires adequate access capacity. To handle the heterogeneity of these applications, the network slice [2], [3] is employed by 5G/B5G networks to perform customized services, resulting in a new paradigm that enables the deployment of virtual networks on generic equipment. By designing and adjusting the structure of slices, slice providers can establish logic networks that possess the corresponding abilities required by Slice Tenants (STs).

The network slice provides a new carrier, replacing tailored devices, to accommodate service requests, which makes the design, deployment, and maintenance of services more flexible and efficient. However, slices provide only a template of networks; only when sufficient resources are allocated, i.e., instantiated, can they effectively play the role STs desire. Since hardware equipment is limited, admission control is needed to match the amount of available physical resources and the admitted workload.

Admission control refers to a policy that determines whether to accept or reject a perceived service request. In early studies [4], [5], requests remain homogeneous, only one kind of resource demand is considered and the optimization objective is unsophisticated. Nevertheless, with the continuous development of applications, the resource requirements of services are diversified, the heterogeneity between services is strengthened and the indicators considered differ. Thus, in sliced networks, call admission control becomes slice admission control, which is expected to be applicable to more complex tasks.

In 5G/B5G networks, many factors should or can be considered as objectives of SAC algorithms [6], mainly resource constraints, the priority of services, fairness of resource sharing, and profits earned [7]. Resource constraints are the most basic targets and must be met to satisfy the Service Level Agreement (SLA) of each slice. The SLA can only be guaranteed when the necessary quantity of resources is obtained [8],

Digital Object Identifier 10.1109/TNSE.2022.3195862

2327-4697 © 2022 IEEE; Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.
Moreover, as applications evolve, the dimension of resource demand also expands. For these reasons, more rigorous and precise resource constraints should be imposed on SAC algorithms.

Priority, an attribute of services, has also received consideration in some admission research. There are many reasons to view it as an important issue. From the perspective of users’ QoE, handoff requests due to mobility should be privileged over new requests; this is because interrupting an ongoing service is greater annoying than rejecting a new service. In addition, services with distinct degrees of urgency ought to be treated differently, and it is reasonable to assign higher priority to requests for emergency communication than to those for general applications. Since each kind of request is served by the corresponding slice, the hierarchical relationship among services is naturally inherited by slices and remains an optional objective of SAC algorithms.

In addition to priority, fairness is an interesting topic in situations where multiple kinds of services coexist and share common substrate networks. Optimization for fairness can achieve more balanced resource allocation and ensure that each kind of request has the opportunity to be served. However, the concept of fairness is traditionally equivalent to that of average and equality [10], which seems inappropriate [11] if the priority is needed as a premise.

In this paper, we focus on the slice admission control problem in 5G/B5G networks and develop a SAC algorithm where priority is taken as the first objective and fairness is controlled simultaneously, which has rarely been investigated in scenarios with multidimensional resources, heterogeneous services, and impatient user behaviors previously. The main contributions of this article are summarized below:

- The heterogeneity of services is considered here; heterogeneity means that the resource requirements and priorities of slices can be quite different. Novel definitions regarding priority and fairness are proposed to ensure that it is reasonable and practical to consider them concurrently in the process of slice admission control.
- Two rational behaviors, i.e., reneging and balking, are taken into account in our work. They represent the impatient psychology of users waiting for services, making the scenario more realistic. We characterize these behaviors with mathematical symbols and model the online slice admission control problem as an integer nonlinear programming problem.
- A multiqueue-based heuristic SAC algorithm called PSACCF is designed. It improves the fairness of admission decisions while respecting differences in service priorities. To construct it, we introduce the ideal CSAR of each queue and push the actual CSAR toward it. Moreover, the degree of priority between adjacent priority levels and the threshold of fairness are controlled by their respective parameters, which makes PSACCF more flexible in dealing with different targets.
- Numerous simulations are conducted to demonstrate the advantages of PSACCF over some existing works.

The remainder of this article is organized as follows. In the next section, we introduce some relevant work performed by other researchers. Section III gives descriptive information about the scenario as well as a reinterpretation of priority and fairness. The system model is given in Section IV, and the details of our slice admission control algorithm are elaborated subsequently in Section V. The simulation results and data analysis are presented in Section VI. Finally, Section VII concludes this article and presents our future plan.

II. RELATED WORK

Tremendous efforts have been made towards admission control in recent decades; a brief review is presented in this part to explain the scope of applications and limitations.

Profit is a common objective favored by many researchers. The authors of [12], [13] focus on maximizing the revenue of the infrastructure provider with the help of reinforcement learning. The paper [14] concentrates on the tradeoff between revenue and service admission rate. This is formulated as a multiple knapsack problem. All requests are roughly divided into premium and best-effort, and an adjustable quota vector is allocated to each kind to control the tradeoff between them. The paper [15] minimizes the network cost of the service provider while satisfying the criteria of requests. Each time the resource constraints are not satisfied, the requests that demand more resources are iteratively rejected until the solution becomes feasible. The authors of [16] propose a multiqueue-based admission control model to pursue higher profit, and it shows advantages in handling heterogeneous requests with rational tenant behavior. These works mainly focus on optimizing the benefits from the service provider’s perspective and ignore the differences in priorities between services, which may affect the experience of some subscribers.

As new applications continue to develop and evolve, they are deeply refined and reclassified on top of the well-known categories, which makes the priority issue nonnegligible. In [17], two kinds of slices are defined: guaranteed service and best-effort service. The former is believed to be more important and should be preferred by the admission policy. A Q-learning approach is proposed, in which higher rewards are attached to the admission of guaranteed services and dropping such a request will incur a penalty. The authors of [18] distinguish the priority of requests according to the network tiers they belong to. Channel allocation and SINR control are the main methods used here to maintain the priority constraint. If a user is provided with its target SINR, then those more privileged than that user should also receive their target SINRs. The papers [19], [20] assume a scenario with multiple strategic tenants. A game theoretic approach is employed to maximize the overall utility of the admission results, where the priority differences are reflected in the multipliers of their respective utilities. The authors in [21] investigate the Base Station (BS) association policy and bandwidth
reservation scheme in vehicular networks. Their goal is to improve the total weighted data rate, and higher weights are assigned to more prioritized vehicles to distinguish them. However, considering only priority differences can easily lead to excessive switching of resources and cause fairness issues, which is outside the scope of these works.

Fairness is always an important issue of admission control, especially when services are provisioned on shared infrastructure. The authors of [10], [22] adopt an equal call blocking probability as the measurement of fairness. In the paper [23], It is assumed that a greater fairness is achieved when the numbers of qualified users in each slice approach each other. Nevertheless, the interpretations above make less sense in scenarios where different applications have distinct QoS requirements. Therefore, the paper [24] defines a utility-based fairness index, and wireless bandwidth is shared among single-hop nodes to achieve approximately the same utility in each connection. The authors of [25] believe that a policy can be considered fair only when the contributions of each tenant are counted. Therefore, the resources in BSs are scheduled for tenants with reference to their contributions. The works [24], [25] offer new ideas for defining fairness, but the services considered are still homogeneous and consume a single type of resource, which means that their methods cannot apply to our scenario.

The literature introduced above optimizes priority and fairness separately. In contrast, the papers [26], [27] combine the two objectives and produce a novel and complex admission problem. In [26], QoS is indicated by the handoff dropping probability, services with higher priorities are protected with higher acceptable QoSs, and the gaps in dropping probabilities between neighbor priorities are restricted to a constant threshold for fairness. Similarly, the authors of [27] use the call blocking probability as the criterion to differentiate priorities. Handoff calls are expected to experience a lower blocking probability than new calls. Fairness is defined differently in the two cases; requests for the same kind of service should obtain a similar blocking probability for different end devices, while requests proposed by the same device for different services should obey a predefined ratio in the average acceptance probability. The concepts of fairness proposed in these two studies are of great significance; fairness is no longer limited to absolute equality but varies depending on the priority. However, traffic patterns are required to be known or derivable conditions for these methods to formulate admission control as a Markov decision process.

Although there have been studies on admission control issues, they have obvious limitations. For example, only a single kind of resource is considered; priority and fairness are rarely handled simultaneously; requests with different priorities actually have homogeneous objectives, i.e., their heterogeneity is not very great; or the rationality of the behaviors of service subscribers is not addressed. Hence, there remains considerable work to do on SAC algorithms.

### III. PRELIMINARY INFORMATION

In this section, we give essential descriptions of the scenario characterization and concept definition employed in our work.

#### A. Scenario Characterization

We introduce a general description of the scenario in which this work is conducted. Two main aspects are included, i.e., the characteristics of services and the impatient behavior of service subscribers.

In addition to the aforementioned service classification criterion, the severity of the resource requirement can act as another measure to further divide services into two classes; elastic and inelastic service [28]:

- **Elastic service.** These services do not have strict real-time resource requirements but a minimum average resource allocation instead. They usually carry an explicit lifetime duration and a total amount of data to transmit or calculate before expiration. A temporary resource shortage will not lead to service interruption as long as it is made up within the subsequent lifetime. In addition, the provision of additional resources can accelerate the completion of services.

- **Inelastic service.** This refers to services that have stringent requirements on the minimum amount of resources at all times to guarantee service stability and continuity. An inadequate supply of resources will drastically degrade their performance [20], or even corrupt them. Inelastic services usually carry no definite lifetime duration and terminate at the wishes of users. For this kind of service, QoS improvement by resource overprovisioning can have a severe marginal effect on QoE [29]. This is because a human is unlikely to perceive quality improvements when quality is already at an extreme level, which means that an excessive supply of resources for these services will reduce the resource utility.

Several typical applications of 5G networks are listed in Table I, together with their categories based on two metrics, i.e., the performance requirement and resource strictness.

| Application          | Performance focus | Resource strictness |
|----------------------|-------------------|---------------------|
| Video call           | eMBB              | Inelastic           |
| VR/AR                | eMBB              | Inelastic           |
| 4K/8K video          | eMBB              | Elastic             |
| Large file transfer  | eMBB              | Elastic             |
| Smart grid           | uRLLC             | Inelastic           |
| Automatic driving    | uRLLC             | Inelastic           |
| Smart home/city      | mMTC              | Elastic             |
| Environmental monitoring | mMTC              | Elastic             |
| Industrial control   | mMTC, uRLLC       | Inelastic           |
| Remote driving       | eMBB, uRLLC       | Inelastic           |
| Remote healthcare    | eMBB, uRLLC       | Inelastic           |
Note that some applications can occupy more than one class under the former criterion, because of their rigorous requirements on multiple performance targets. Those belonging to the same kind among eMBB, uRLLC, and mMTC may continue to be divided into opposite groups according to the stringency of their resource requirements. For example, both 4 K video and VR are classified as eMBB due to their high bandwidth requirements. Thanks to the video caching technology, 4 K video is more elastic in terms of bandwidth resource, but that requirement of VR is much more stringent to response to operations like view rotation at any time. Therefore, one can further divide the two applications into elastic and inelastic services, respectively.

The second thing worth mentioning is the behavior of service subscribers, who are assumed to be rational here. Sometimes, requests cannot be admitted immediately and queues are formed due to resource unavailability. Thus, some subscribers may revoke the proposed requests or simply not submit a request to this overloaded slice provider. We consider such actions to be impatient behaviors and two cases are considered here [30]:

- **Renege.** This describes the behavior whereby a service subscriber cancels his request when it is not pulled to service for a long time after being submitted to the queue. This can be realized by assigning a hold time value to each request that is invisible to the slice provider and begins to decrease once the request joins a queue. As soon as this value saturates, the request departs from the queue.
- **Balk.** This refers to the phenomenon that some subscribers hesitate to submit their requests to their tenants. This can occur when many requests have accumulated in the queues. A long queue indicates that the SP is overwhelmed or this kind of request is not favored by the SP, so the waiting time may exceed the hold time if a request enters the queue, leading to a high risk of reneging. The uncertainty of this behavior can be depicted by a balk probability. It measures the probability that a subscriber initiates and submits a request to the queue and is often formulated as sensitive to queue length; a greater length yields a smaller value to reflect the decline in user enthusiasm for announcing a request.

### B. Adjustment of Concepts

In this subsection, we concentrate on the characterization of priority and definition of fairness employed in this article. First, the reasons the traditional definitions are no longer appropriate are presented; then, new interpretations suitable to the Slice-as-a-Service (SaaS) environment are given.

1) **Characterization of Priority:** The difficulty is how to characterize priority concretely to judge whether priority is guaranteed by the SAC algorithm. In some previous studies [14], [31], priority is expressed by imposing additional exclusive resources on those with higher levels. Although such methods ensure the priority difference between services to a certain extent, they often support only a single kind of resource and thus cannot be applied directly to scenarios where multiple kinds of resources are considered and different services rely on diverse resources with different degrees. Assigning weights is also a common method of maintaining priority. By assigning higher weight to the high-priority service in the optimization goal, the admission algorithm can be prompted to access more high-priority services. However, this approach usually requires a homogeneous target among prioritized services, which is not always the case.

To overcome the difficulty of service heterogeneity when characterizing priorities, the indicator used to measure must be service independent. Thus, we adopt the Cumulative Service Acceptance Ratio (CSAR), which is complementary to the service blocking ratio in [5], to differentiate the priority. If the size relationships of CSARs are consistent with those of priority levels, the priority difference is guaranteed. In this way, regardless of which kind of resources the services mainly consume or what performance indicators they are mostly concerned about, the priority relationships can be explicitly evaluated.

Compared with the above methods, using CSAR to characterize priority neither requires a smart quota distribution of resource capacity nor brings drastic service discrimination because the concept of priority no longer equals absolute privilege but preference instead. Therefore, low-priority services will not always be ignored even when they coexist with high-priority services. Moreover, by adjusting the gap between the CSARs of adjacent priority services, the degree of priority can be conveniently controlled, which enables to make a more precise and flexible admission decision.

2) **Definition of Fairness:** Fairness has long been an enduring topic, and the level of fairness is believed to impact the performance of networks [32] and the QoE of services [33]. However, fairness is not a well-investigated problem because the definition of fairness is extremely scenario-dependent. Optimizing fairness typically conflicts with other objectives pursued simultaneously, such as the priority requirement. Therefore, a reasonable definition of fairness must be provided in advance.

The traditional concept of fairness borders on that of average and equality. Unfortunately, it is increasingly partial to treat fairness as equality [11], [23]. Even for homogeneous services, admission can be considered fair only if the respective contributions of each kind are considered [25] let alone when different services have heterogeneous resource demands, diverse performance preferences, and distinct priority levels. Giving preference to particular services based on their characteristics happens to be fair in the latter case. Evidence can be found in articles [26], [27], where the phenomenon that high-priority service deserves lower blocking probability is taken for granted. Both use a threshold to control fairness. Specifically, [26] sets the threshold to the absolute gap between adjacent dropping probabilities, while [27] sets the threshold to the ratio of adjacent average acceptance probabilities.

Inspired by these works, we place the embodiment of fairness on the uniformity of gaps in CSAR between adjacent
Slice tenants are internet service providers who have no substrate devices and sell virtual network businesses to their subscribers by leasing tailored virtual slices from the SP. One SP can deliver slices to several tenants simultaneously with a renegotiated service level agreement, earn from it, and pay a penalty if the SLA is violated. Assume that there are \( K \) tenants who have signed a contract with the SP and are included in a set \( \mathcal{K} = \{1, 2, \ldots, K\} \). Each tenant corresponds to one type of service, which consumes multiple kinds of resources, such as CPU, GPU, bandwidth, memory, and storage. Suppose there are a total of \( N \) types of resources needed by all tenants under the SP, then the resource demand of tenant \( k \) can be formulated with an \( N \) dimensional nonnegative column vector \( r_k = (r_{k1}, r_{k2}, \ldots, r_{kN})^T \), where each element denotes the amount of each resource. For elastic services, the demand vector gives the average resource demands during their lifetime, while for inelastic services, the vector marks the real-time demands that should be held all the time until expiration. We use an \( N \times K \) matrix \( \mathbf{R} = \{r_1, r_2, \ldots, r_K\} \) to denote the resource demands of all tenants, and the total resource capacity possessed by the SP is \( \mathbf{c} = (c_1, c_2, \ldots, c_N)^T \).

Requests can obtain a priority level that aligns with that of the slice serving them; however, a puzzle remains. For a portion of tenants, requests inside ought to have different priorities due to the mobility of service users, i.e., handoff requests should be prioritized over new requests for the same service type. To solve this problem, a small supplement is required. We treat tenants who may receive handoff requests as two subtenants with the same demand vector but different urgency levels. The same adjustments are made to the corresponding slices and a higher priority is assigned to the more urgent slice. We slightly abuse the set \( \mathcal{K} = \{1, 2, \ldots, K\} \) to label the priority level of each tenant and subtenant or each slice and subslice equivalently. The larger the number, the higher the priority level is.

The SP establishes a queue for each slice to accommodate requests proposed by tenants who cannot be served immediately due to a temporary shortage of resources. Requests inside a queue have the same demand vector and priority level, so the First In First Out (FIFO) principle is employed within each queue when resources are sufficient to pull requests from it. Time is divided into slots that are sufficiently fine to work as a basic unit to measure the lifetime and hold time of requests. For a request \( i \), we use \( T_i^L \) and \( T_i^H \) to record its lifetime and hold time durations, both of them start counting as soon as the request is sensed. These two parameters can differ between requests in the same queue, and only the \( T_i^L \) of an elastic request is revealed to the SP at the moment of initiation by its proposer; the others are invisible. We introduce another symbol \( T_i^P \) to indicate the perceived time of request \( i \) by the SP, i.e., the time slot at which this request joins the corresponding queue. Then, the renge behavior of subscribers can be characterized when Ineq. (1) does not hold, where \( t \) indicates the current time slot sequence. Vector \( d(t) = (d_1(t), d_2(t), \ldots, d_K(t))^T \) is used to denote the number of requests departing from each queue because of reneging.
$T_i^p + T_i^H - 1 \leq t, \quad (1)$

We use $O(t) = \bigcup_{k \in K} O_k(t)$ to represent the set of all ongoing requests in time slot $t$, and $O_k(t)$ corresponds to those in slice $k$. While $I(t) = (I_1(t), I_2(t), \ldots, I_K(t))^T$ is a vector that records the length of all slices in slot $t$, each element matches a specific queue, reflecting the number of requests waiting for service. The admission decision must take into account the status of queues, i.e., $I(t)$, and available resources, i.e., $c^{ava}(t)$. The symbol $r$ is multiplexed to define a function $r(\cdot)$ that takes a request $i$ as input and returns its demand vector if it has not yet been served or its resources actually allocated when it is an ongoing request. Thus, $c^{ava}(t)$ can be computed through (2).

$$c^{ava}(t) = c - \sum_{i \in O(t)} r(i), \quad (2)$$

At the beginning of each slot, some ongoing requests may saturate their lifetime, expire from the system and release the resources they previously occupied. We formulate the condition for these operations briefly as Ineq. (3), where $T_i^A$ is the slot in which request $i$ is admitted by the SP. Note that $T_i^A$ must satisfy Ineq. (4) because the renege behavior is considered. Similar to $O(t)$, we use $E(t) = \bigcup_{k \in K} E_k(t)$ to indicate the set of those requests that died out at time slot $t$. Then, the ongoing set $O(t)$ and available resources $c^{ava}(t)$ should be updated according to (5) and (6), respectively. In addition, we stimulate the balk behavior of subscribers by attaching a probability to each slice $k$ that decreases with the increase in its queue length; see (7), where $p_k(t) \in [0, 1]$ is a slice-dependent parameter to show the willingness of subscribers to wait for type-$k$ slice [34]. New requests initiated by users join the queue with probability $p_k(t)$ rather than entering directly with no hesitation. It is easy to check that when no requests are waiting in the queue, new requests will definitely join. However, if the queue length is excessive, they rarely join because of the scenario typically entails an unacceptable waiting time that elevates the risk of violation of Ineq. (1), i.e., reneging. We use another vector $b(t) = (b_1(t), b_2(t), \ldots, b_K(t))^T$ to count the number of requests lost in time slot $t$ due to balk behavior.

$$T_i^A + T_i^b \leq t, \quad (3)$$

$$T_i^p \leq T_i^A \leq T_i^p + T_i^H - 1, \quad (4)$$

$$O(t) = O(t) \setminus E(t), \quad (5)$$

$$c^{ava}(t) = c^{ava}(t) + \sum_{i \in E(t)} r(i), \quad (6)$$

$$p_k(t) = e^{-\beta_k(t)}, \quad \forall k \in K, \quad (7)$$

The expiration of ongoing requests is captured by the SP as soon as they set resources free, while the loss of requests because of reneging and balk is reported by tenants to the SP periodically, i.e., in each time slot. With this information, the SAC algorithm is responsible for deciding which kind of requests and how many of them should be newly admitted in a given time slot. We use a vector $a(t) = (a_1(t), a_2(t), \ldots, a_K(t))^T$ to term the admission actions, and $a_k(t)$ means that the top $a_k(t)$ requests in the type-$k$ queue are to be pulled to service.

### B. Formulation of Priority Requirement

In our design, the priority of slices is reflected through the measurement of their CSARs. The priority is defined as the quotient of the cumulative number of admitted requests, termed $\text{cum}_k^A(t)$, and the cumulative number of confirmed requests, denoted as $\text{cum}_k^C(t)$. The latter refers to all the requests admitted, reneged, and balked. (8) and (9) give the corresponding formulations. Then, the CSAR of the type-$k$ slice in time slot $t$, denoted as $a_k(t)$, can be computed by (10).

We apply this approach to each slice and form a CSAR vector $\alpha(t) = (\alpha_1(t), \alpha_2(t), \ldots, \alpha_K(t))^T$.

$$\text{cum}_k^A(t) = \sum_{i=1}^{t} a_k(\tau), \quad \forall k \in K, \quad (8)$$

$$\text{cum}_k^C(t) = \sum_{i=1}^{t} a_k(\tau) + d_k(\tau) + b_k(\tau), \quad \forall k \in K, \quad (9)$$

$$a_k(t) = \frac{\text{cum}_k^A(t)}{\text{cum}_k^C(t)}, \quad \forall k \in K, \quad (10)$$

Without loss of generality, we assume the priority level of each slice is labeled by its index. The characterization of priority in Section III-B requires higher CSARs for higher priority levels, so the priority requirement is met if the vector $\alpha(t)$ is monotonously nondecreasing, i.e., sequential Ineq. (11) holds. We introduce an indicator function $\mathcal{I}(\cdot)$ that takes $\alpha(t)$ as input and outputs 1 if $\alpha(t)$ satisfies the priority requirement and 0 otherwise. (12) gives a mathematical description of $\mathcal{I}(\cdot)$.

$$\alpha_1(t) \leq \alpha_2(t) \leq \ldots \leq \alpha_K(t), \quad (11)$$

$$\mathcal{I}(\alpha(t)) = \begin{cases} 1, & \text{if } \alpha_k(t) \leq \alpha_{k+1}(t), \forall k \in [1, K] \\ 0, & \text{otherwise} \end{cases} \quad (12)$$

The above formulations provide a rough picture of whether the priority is preserved. To evaluate the degree of priority between adjacent levels, the gap in CSARs is a good choice, and we denote it as a column vector $\Delta \alpha(t) = (\Delta \alpha_1(t), \Delta \alpha_2(t), \ldots, \Delta \alpha_{K-1}(t))^T$ with $K$-1 dimensions, where $\Delta \alpha_k(t) = \alpha_k(t+1) - \alpha_k(t)$ matches the gap between slice $k$ and slice $k+1$.

Another confusing problem that requires further explanation is which slice is responsible when the priority condition is upset. We elaborate this issue with two CSAR vectors $\alpha(t_1) = (0.7, 0.8, 0.9, 0.85)^T$ and $\alpha(t_2) = (0.95, 0.8, 0.9, 0.92)^T$. In slot $t_1$, the fourth slice breaks the requirement; if we can increase it to 0.9 or even higher, Ineq. (11) holds. While, does the first slice is the one to blame in slot $t_2$ for possessing a high CSAR, the answer is not. In $\alpha(t_2)$, the breakers are actually slice 2, slice 3, and slice 4. What if we treat slice 1 as the culprit, the way to fix the vector is to depress the CSAR of slice 1, it means
that some ongoing requests of slice 1 must be dropped immediately to decrease the numerator in (10), which is taboo in the SLA-guaranteed scenario and thus not allowed in our SAC algorithm. Therefore, the slices that violate the priority requirement refer to those whose CSARs can be promoted to satisfy the previously unsatisfied Ineq. (11). Rather than elaborate on how to do this here, we defer the details to Section V in priority amendment.

C. Formulation of the Fairness Degree

In a prioritized environment, giving more preference to high-priority slices is not only reasonable but also desirable. However, the degree of this preference should not be stretched indefinitely but limited within a certain range, which is why we impose the fairness objective.

The relative degrees of priority between adjacent levels are not necessarily consistent with each other; a weight vector $\mathbf{w} = (w_1, w_2, \ldots, w_{K-1})^T$ is used as a parameter to coordinate the ideal relative size, and each matches an element in $\Delta \mathbf{a}(t)$. This vector can be tailored according to the actual slices offered by the SP. When the tenant population changes, the SP can flexibly adapt to it by adding, removing, or adjusting the corresponding weight values.

Therefore, the aim is for the new definition of fairness to be sensitive to whether the priority requirement is met and to reflect the correlation between $\Delta \mathbf{a}(t)$ and $\mathbf{w}$. The uniformity degree of the zoomed CSAR gaps, denoted $f_j(\mathbf{a}(t))$, is exactly the one needed to ensure that this conception holds. We make minor adjustments to Jain’s fairness index [35] and quantify it in (13).

$$f_j(\mathbf{a}(t)) = \begin{cases} \frac{\left(\sum_{k=1}^{K-1} \Delta a_j(t) \right)^2}{(K-1) \sum_{k=1}^{K-1} \left(\frac{\Delta a_j(t)}{\Delta a_k(t)}\right)^2}, & \text{if } I(\mathbf{a}(t)) > 0, \\ 0, & \text{otherwise} \end{cases}$$

The function $f_j(\mathbf{a}(t))$ is nonnegative and limited to $[0, 1]$. A violation of the priority requirement will suppress $f_j(\mathbf{a}(t))$ to zero. Meanwhile, other positive values can directly reveal the degree of fairness, which is an important property inherited from Jain’s fairness index; the corresponding proofs can be found in [35].

D. Optimization Problem

The SAC module requires three pieces of state information to make an admission decision: the set of ongoing requests $O(t)$, the vector of real-time queue length $I(t)$ and the initial CSAR vector $\mathbf{a}^l(t)$. The first two are used to determine the available resource capacities and the total number of requests in queues in time slot $t$, respectively. The third part is computed as (14), where vector division is defined as the division of counterpart elements. It stands for the initial CSAR of each slice at the beginning of slot $t$, where admission decisions have not been made and executed yet, but the behavior of reneging and balking has already occurred.

$$\mathbf{a}^l(t) = \frac{\text{cum}^A(t-1)}{d(t) + b(t) + \text{cum}^E(t-1)},$$

$$\pi : \mathcal{O} \times \{1\} \times \{\mathbf{a}^l\} \mapsto \mathbb{R}_+^K,$$

$$u(t) = \min \left\{ \sum_{i \in O(t)} r(i) / c \right\},$$

In mathematical terms, slice admission control is an operation, denoted as $\pi$, for implementing a mapping from the state tuple to an action vector, illustrated in (15). During the process, resource constraints should be maintained, and some objectives are pursued, i.e., mainly the priority requirement and fairness in this work. The utilization of resources, see (16), is also considered to avoid excessive idle resources. The final model of our optimization problem is shown in (17).

Maximize $$\left(\sum_{t=1}^{T} I(\mathbf{a}(t)), f_j(\mathbf{a}(t)), u(t)\right),$$

s.t. $\mathbf{a}(t) \leftarrow \pi(O(t), I(t), \mathbf{a}^l(t)),$

$\mathbf{a}(t) \leq I(t),$  

$R \cdot \mathbf{a}(t) \leq c^{\text{ana}}(t),$  

The first objective is the time average of the sum of historical priority requirements, which means we aim for a long-term smooth priority performance. Other objectives have received abundant explanation previously and are not further discussed here. Constraint (17a) indicates that the admission action originates from the current system state, constraint (17b) states that the number of requests to access cannot exceed the amount accumulated in queues, and constraint (17c) implies that the total resource demands of these newly accepted requests should be affordable to the residual resources. Symbol $t$ indicates that this is an online problem with different system states and constraints at different time slots.

V. ALGORITHM DESIGN

The problem formulated in (17) is clearly nonconvex, and the online characteristic results in time-varying constraints, which further complicates the issue. Thus, this problem can hardly be solved with classical mathematical optimization methods in a time-efficient manner. Machine learning-based methods may also not be suitable for solving this problem, as multiple optimization objectives may lead to unstable model convergence. Moreover, adjustments to tenant types and alterations in the resource demands of tenants require reconstruction of the neural network structure and retraining of the neural network parameters, which sometimes hinders a timely and accurate decision output. Therefore, we propose a heuristic-based SAC algorithm called PSACCF, which introduces the resource efficiency of services to amend priority violations, and then promotes fairness by setting the target CSARs for each service type and pushing the admission result to approach the optimal solution.
A. Admission Decision

Eq. (13) illustrates that the degree of fairness relies on whether the priority requirement is satisfied, so PSACCCF can be roughly divided into two parts; priority amendment and fairness enhancement.

1) Priority Amendment: PSACCCF starts with an inspection of the priority requirement, and queues violating the requirement are added into the set \( \mathcal{V}(t) \). Requests belonging to the queues within \( \mathcal{V}(t) \) must be processed first to correct the priority. The resource efficiency, denoted as \( RE_k(t) \), is introduced to determine the admission order of these requests.

\[
RE_k(t) = \frac{\partial \alpha_k(t)}{\partial \alpha_k(t)} / R_k^{DR_k(t)}, \tag{18}
\]

\( RE_k(t) \) is formulated in (18), where \( \frac{\partial \alpha_k(t)}{\partial \alpha_k(t)} \) denotes the derivative of CSAR for each queue with respect to its access volume. This value measures the incremental increase in the CSAR of the service for each new incoming request, as shown in (19). \( R_k^{DR_k(t)} \) is the amount of dominant resource acquired by the request, whose superscript \( DR_k(t) \) indicates the dominant label and is judged by (20). This result marks the label of the first saturated resource when \( e^{d/e}(t) \) in (21) is exclusively used to accommodate type-k requests.

\[
\frac{\partial \alpha_k(t)}{\partial \alpha_k(t)} = \frac{1 + \text{cum}^k_X(t)}{1 + \text{cum}^C(t)} - \alpha_k(t) = \frac{1 - \alpha_k(t)}{1 + \text{cum}^C(t)}, \tag{19}
\]

\[
DR_k(t) = \arg\max_n \left( \frac{r_k}{e^{d/e}(t)} \right), \tag{20}
\]

\[
e^{d/e}(t) = c^{ava}(t) - R \cdot \alpha(t), \tag{21}
\]

The resource efficiency reflects the size of the increment in CSAR resulting from the consumption of the same amount of dominant resource. Therefore, the higher \( RE_k(t) \) is, the faster the priority amendment it brings, which can act as a criterion to sort requests.

**Procedure 1**: Fix the priority requirement.

1 while \( I(\alpha(t)) = 0 \) do
2 Construct \( \mathcal{V}(t) \), descending sort \( \mathcal{V}(t) \) with \( RE(t) \);
3 for each \( k \) in \( \mathcal{V}(t) \) do
4 if \( R \cdot (\alpha(t) + \varepsilon_k) \leq c^{ava}(t) \) then
5 \( \alpha_k(t)++ \), \( l_k(t)-- \), update \( \alpha_k(t) \);
6 break;
7 if \( \alpha(t) \) remains unchanged above then
8 break;

**Procedure 2**: Check the degree of fairness.

1 while \( f_j(\alpha(t)) \geq \psi \) do
2 for each \( k \) \in \{K, \ldots, 1\} \( l_k(t) > 0 \) do
3 if \( R \cdot (\alpha(t) + \varepsilon_k) \leq c^{ava}(t) \) then
4 \( \alpha_k(t)++ \), \( l_k(t)-- \), update \( \alpha_k(t) \);
5 break;
6 if \( \alpha(t) \) remains unchanged above then
7 break;

CSAR update. These steps are repeated until there are no such queues or the priority requirement is satisfied.

2) Fairness Enhancement: Once the priority requirement is satisfied, PSACCCF enters the phase of fairness enhancement. A threshold \( \psi \in [0, 1] \) is used to control the acceptable degree of fairness. When \( f_j(\alpha(t)) \) exceeds \( \psi \), it is regarded as sufficiently fair that admission actions can be skewed towards requests with high priority to further consolidate the priority requirement. That is exactly the job of **Procedure 2**. Line 2 to Line 5 try to admit requests in a high-priority-first manner and are repeated until there are no such requests or the fairness indicator reaches an inadequate level.

When the degree of fairness is not sufficiently high, efforts are made to improve it. First, the target CSARs (denoted \( \alpha^{ava}(t) \)) of each kind of service, under which the fairness indicator reaches 1, should be derived in advance. The following theorem provides the fundamental basis for calculating the target CSARs.

**Theorem 1**: The fairness indicator takes its upper bound if and only if the CSAR gap vector \( \Delta \alpha(t) \) is proportional to the weight vector \( w \).

\[
f(x) = \frac{\sum^n_{i=1} x_i}{n \cdot \sum^n_{i=1} x_i^2}, \quad x_i \geq 0, \tag{22}
\]

\[
\Delta \alpha_j(t) = \frac{\Delta \alpha_j(t)}{w_j}, \quad \forall i \neq j \in [1, K], \tag{23}
\]

**Proof**: It has been proven in [35] that Jain’s fairness index, shown in (22), obtains the maximum value 1 only when the \( x_i \) values are equal to each other. Substituting \( x_i \) with \( \Delta \alpha_j(t)/w_j \), we can obtain that \( f_i(\alpha(t)) \) reaches 1 if and only if the equations in (23) hold. Through a simple transformation, it can be expressed as \( \Delta \alpha(t) = \epsilon \cdot w \), where \( \epsilon \) is the proportion factor.

Based on **Theorem 1**, the principle for calculating the target CSARs is illustrated in **Procedure 3**. The distance between the actual CSARs of the least prioritized and the most prioritized service is compared with the weighted sum of \( \epsilon \). A greater distance means that the span of \( \Delta \alpha(t) \) can be further narrowed by promoting the target CSARs of those less preferred services, which can be realized using (24) and shown in Line 1 to Line 3. Otherwise, the target CSARs of those more prioritized services can be raised to extend the distance between \( \alpha_j(t) \) and \( \alpha_K(t) \). Since the CSAR is bounded between 0 and 1, a secondary judgment is executed in Line 5. Different update formulas are adopted accordingly, which are listed in (25) and (26), with the pseudocode in Line 5 to
Procedure 3: Calculate the target CSAR.

1. If $\alpha_k(t) - \alpha_1(t) \geq \epsilon \cdot \sum_{i=k}^{K-1} w_i$, then
2. For each $k \in K$ do
3. $\alpha_k^{\text{int}}(t) = \alpha_k(t) - \epsilon \cdot \sum_{i=k}^{K-1} w_i$;
4. Else
5. If $\alpha_1(t) + \epsilon \cdot \sum_{i=1}^{K-1} w_i \leq 1$, then
6. For each $k \in K$ do
7. $\alpha_k^{\text{int}}(t) = \alpha_1(t) + \epsilon \cdot \sum_{i=1}^{K-1} w_i$;
8. Else
9. For each $k \in K$ do
10. $\alpha_k^{\text{int}}(t) = \max\{1 - \epsilon \cdot \sum_{i=k}^{K-1} w_i, 0\}$;

Procedure 4: Approach to the target CSAR.

1. Update $e^{\text{int}}(t)$, $DR(t)$, construct $P(t)$;
2. Descending sort $P(t)$ with $RE$;
3. Stably descending sort $P(t)$ with $TD(t)$;
4. For $k \in P(t)$ do
5. If $R \cdot (\alpha(t) + \epsilon_k) \leq e^{\text{int}}(t)$ then
6. $\alpha_k(t)$++,$l_i(t)$--, update $\alpha_k(t)$;
7. break;

Line 7 and Line 8 to Line 10, respectively. During this procedure, the proportion factor $\epsilon$ acts as a vital parameter to control the absolute CSAR gaps between all adjacent priority levels.

\[ \alpha_k^{\text{int}}(t) = \alpha_k(t) - \epsilon \cdot \sum_{i=k}^{K-1} w_i, \quad \forall k \in K, \]

(24)

\[ \alpha_k^{\text{int}}(t) = \alpha_1(t) + \epsilon \cdot \sum_{i=1}^{K-1} w_i, \quad \forall k \in K, \]

(25)

\[ \alpha_k^{\text{int}}(t) = \max\{1 - \epsilon \cdot \sum_{i=k}^{K-1} w_i, 0\}, \quad \forall k \in K, \]

(26)

$\alpha^{\text{int}}(t)$ generated by Procedure 3 exactly satisfies the condition illustrated in Theorem 1, so it provides an ideal reference value for each actual CSAR to pursue. Admission decisions should push $\alpha(t)$ toward $\alpha^{\text{int}}(t)$ so that fairness can be optimized. Considering that dropping an ongoing request will cause a penalty in the SlaaS environment, only those whose actual CSAR is lower than the target CSAR can gain admission. Another set $P(t)$ is used to record this kind of service.

\[ TD_k(t) = \alpha_k^{\text{int}}(t) - \alpha_k(t), \]

(27)

PSACCF relies on the result of double sorts to determine the access order of the queues in $P(t)$. Procedure 4 shows the operations. The elements in $P(t)$ are arranged in descending order with respect to their resource efficiencies and stably sorted again according to the distance between their actual CSARs and target CSARs, termed $TD(t)$ in (27).

Subsequently, queues in $P(t)$ are checked sequentially to see whether the resource constraints still hold when one request in queue $k$ is accepted. If so, the top request in that queue is pulled to service, and $\alpha_k(t)$ is updated concurrently. Then, the For Loop is interrupted, as $\alpha(t)$ has already changed, as shown in Line 4 to Line 7. While ensuring resource feasibility, Procedure 4 promises to admit the request with the largest $TD$, or the one with the highest resource efficiency if several kinds of services have the same $TD$. This strategy facilitates greater fairness at a lower resource cost.

The full pseudocode of our algorithm is given in Algorithm 1. It takes the set of ongoing requests $O(t)$, the queue length vector $l(t)$ and the initial CSAR vector $\alpha^I(t)$ as inputs and outputs the admission decision $\alpha(t)$.

Line 1 initializes variables and counts the loss of requests caused by impatient behavior, thereby updating the queue length, available resources, and the dominant resource labels in Line 2.

Line 3 checks the state of priority and tries to correct violations when they occur. The priority requirement is inspected twice in Line 4. If the amendment in Line 3 fails, it must be due to the shortage of resources, meaning that it is impossible to improve fairness, so the algorithm ends. Otherwise, PSACCF enters the phase of fairness enhancement.

The fairness degree is checked in Line 6, and a second strengthening of the priority requirement is conducted if the fairness is sufficiently high. Then, the calculation of target CSARs is executed in Line 7, closely followed by Procedure 4, which is responsible for pushing the actual CSARs toward the target CSARs. Each time a new request is accepted, $\alpha(t)$ changes accordingly, as does $f_{\text{int}}(\alpha(t))$. Therefore, Procedure 2 to Procedure 4 need to be executed repeatedly until no more optimizations can be made on $\alpha(t)$.

### Algorithm 1: PSACCF

**Input:** 1) Set of ongoing requests $O(t)$; 2) Length of queues $l(t)$; 3) Initial CSAR vector $\alpha^I(t)$.

**Output:** Admission decision $\alpha(t)$.

1. Initiate $\alpha(t) = 0$, $\alpha(t) = \alpha^I(t)$, count $d(t)$ and $b(t)$;
2. Update $l(t)$, $e^{\text{int}}(t)$ and $DR(t)$;
3. //priority amendment
4. Check and amend the priority requirement using Procedure 1;
5. if $I(\alpha(t)) = 1$ then
6. //fairness enhancement
7. while true do
8. //check the degree of fairness
9. Execute Procedure 2;
10. //calculate the target CSAR
11. Execute Procedure 3;
12. //approach to the target CSAR
13. Execute Procedure 4;
14. if $\alpha(t)$ remains unchanged after Procedure 4 then
15. break;
16. return $\alpha(t)$;

### B. Resource Scheduling

An operation that is tightly coupled to admission control is resource allocation. In our scenario, where elastic and inelastic services coexist, requests that originate from different categories are treated differently.
Inelastic requests are granted exactly the amount they demand once admitted and remain unchanged during the whole lifetime. More effort is required regarding elastic requests. Newly admitted elastic requests also receive the requested quantity upon admission but can still be adjusted later; if there are residual resources left when PSACCF is done, they will be evenly shared by ongoing elastic requests with the highest priority level. On the other hand, this allows for better utilization of resources, which echoes the third objective. On the other hand, it enables the effect of relocating the current idle resources to future time slots. Specifically, in the next time slot, overprovisioned elastic requests can be reallocated with the minimum resources that ensure on-time completion, saving more space to accommodate future requests, which looks as if the previous resources were delayed to the subsequent time slot.

C. Complexity Analysis

This subsection explains the complexity of PSACCF. The time costs of Procedure 1 to Procedure 4 are elaborated first, based on which the time consumption of PSACCF is deduced.

In Procedure 1, Line 2 takes $O(K^2)$ and the for Loop operates at most $K$ times, each iteration of which costs $O(N)$. The outer while Loop stops when the priority requirement is met or no resources are available. Since the total capacity and demand vectors are fixed, its upper bound is limited to the maximum number of requests when resources are exclusively consumed by one kind of service. We use a constant $C$, independent of $K$ and $N$, to denote the bound. Then, Procedure 1 takes $O(C \cdot K^2 + C \cdot K \cdot N)$. A similar analysis can be applied to Procedure 2, which decreases to $O(C \cdot K \cdot N)$ due to the absence of the sort operation. For Procedure 3 and Procedure 4, it is easy to see that the time complexities are $O(K)$ and $O(K^2 + K \cdot N)$, respectively.

In Algorithm 5, Line 1 takes $O(K)$ and Line 2 takes $O(K \cdot N)$. The priority amendment takes $O(C \cdot K^2 + C \cdot K \cdot N)$. The While Loop interrupts when fairness is no longer optimized or resources are saturated, which means it also cannot exceed $C$ cycles. Therefore, the time complexity of fairness enhancement is $O(C^2 \cdot K \cdot N + C \cdot K + C \cdot K^2 + C \cdot K \cdot N)$. Thus, the overall time complexity of PSACCF is $O((C + 1) \cdot K + 2 C \cdot K^2 + (C^2 + 2 C + 1) \cdot K \cdot N)$. Considering that $C$ is unrelated to $K$ and $N$, the final result is $O(K^2 + K \cdot N)$.

VI. PERFORMANCE EVALUATION

A. Simulation Settings

Simulations are performed on a laptop with an i5-7300HQ CPU and 16 GB RAM, and the software environment is Windows 10 Professional and C++ 8.1.0.

Two kinds of services are considered during the simulation and both can generate handoff requests, i.e., $K = 4$. Each request is assumed to require two types of resources, i.e., $N = 2$. Similar to the normalized resource vector in [16], the first kind of service is elastic with demand vector $(0.035, 0.03)^T$, the second is inelastic with demand vector $(0.016, 0.02)^T$, and the resource capacity is $(2.0, 2.0)^T$. We use Service-1 to Service-4 to represent these four services; the first two denote new requests, and the last two denote handoff requests, which have higher priority.

We assume that the different requests have their own arrival rates, which are factored against a base arrival rate $\lambda_0$ by $[1.2, 1.5, 0.6, 0.75]$. The lifetime and hold time parameters adopt factors $[0.6, 1.0, 0.3, 0.5]$ and $[1.0, 0.8, 0.3, 0.24]$, respectively. This setting implies that handoff requests usually have a shorter duration and need to be dealt with in a more timely manner.

The parameter $\beta_0$ decreases as the priority level $k$ rises, indicating a higher willingness for a request to join a queue. The service price $p_k$, i.e., the fee a request needs to pay for each time slot, increases with increasing priority. Certain values are summarized in Table II. Note that failure to complete ongoing requests incurs a penalty; we represent this by a penalty ratio, denoted as $pr$, which is set to 1.5.

For convenience of simulation, $w$ is fixed to 1. Different $\lambda_0$, $\epsilon$ and $\varphi$ values are tested to explore the potential abilities of PSACCF, and the values used are also recorded in Table II.

| symbol | value | symbol | value |
|--------|-------|--------|-------|
| $K$    | $4$   | $N$    | $2$   |
| $\tau_0^C$ | $20.0$ | $\tau_0^H$ | $10.0$ |
| $\tau_1^C$ | $0.6 \cdot \tau_1^C$ | $\tau_1^H$ | $1.0 \cdot \tau_1^H$ |
| $\tau_2^C$ | $1.0 \cdot \tau_2^C$ | $\tau_2^H$ | $0.8 \cdot \tau_2^H$ |
| $\tau_3^C$ | $0.3 \cdot \tau_3^C$ | $\tau_3^H$ | $0.3 \cdot \tau_3^H$ |
| $\tau_4^C$ | $0.5 \cdot \tau_4^C$ | $\tau_4^H$ | $0.24 \cdot \tau_4^H$ |
| $p_0$ | $1.0$ | $\beta_0$ | $0.02$ |
| $p_k$ | $(1 + \log(k)) \cdot p_0$ | $\beta_k$ | $\beta_0/(1 + k)$ |
| $pr$ | $1.5$ | $w$ | $(1, 1, \ldots, 1)^T$ |
| $\lambda_1$ | $1.2 \cdot \lambda_0$ | $\tau_1$ | $(0.035, 0.03)^T$ |
| $\lambda_2$ | $1.5 \cdot \lambda_0$ | $\tau_2$ | $(0.016, 0.02)^T$ |
| $\lambda_3$ | $0.6 \cdot \lambda_0$ | $\tau_3$ | $(0.035, 0.03)^T$ |
| $\lambda_4$ | $0.75 \cdot \lambda_0$ | $\tau_4$ | $(0.016, 0.02)^T$ |
| $\lambda_0$ | $\{5, 7, 9, 11, 13, 15\}$ | $\epsilon$ | $(2.0, 2.0)^T$ |
| $\epsilon$ | $\{0.05, 0.10, 0.15\}$ | $\varphi$ | $(0.85, 0.90, 0.95, 1.0)$ |

B. Self-Evaluation

First, we fix $\epsilon$ and $\varphi$ to test the performance of PSACCF under different request arrival rates, and the results are shown in Fig. 2. The curves correspond to the CSARs of each service queue during 1000 time slots. As the load increases, the CSARs of all four services decrease because balk and reneging behavior become more serious. However, the curves become smooth quickly regardless of $\lambda_0$, and the gaps between them remain essentially unchanged, which reflects the effect of $\epsilon$ and $\varphi$.

Fig. 3 further explains how $\epsilon$ and $\varphi$ influence the CSARs by inspecting other combinations of them when $\lambda_0$ is fixed at 9. In more detail, Fig. 3 a relaxes $\varphi$ to 0.85 compared with Fig. 2 a. When the fairness threshold is reached, services with higher priority receive more attention from PSACCF, creating an
expanded gap between Service-4 and Service-3. Fig. 3 b adopts a larger $\varepsilon$, yielding larger absolute gaps in CSAR, while the uniformity is similar to that in Fig. 2 b for a consistent $\varphi$.

It can be concluded from Figs. 2 and 3 that the CSAR of high priority can be promoted at the expense of depressing the CSAR of low priority by raising $\varepsilon$ or suppressing $\varphi$, which makes it seem that more resources are reserved for requests with higher priority. Notably, the amount of resources set aside is no longer immutable but adjusts automatically depending on the request arrival rate, which makes PSACCF display expertise in scheduling resources flexibly and contributes to better resource utilization.

Various combinations of $\varepsilon$ and $\varphi$ are tested in Fig. 4 to observe the response of profit earned by the SP. The phenomena exhibited comply with those in Figs. 2 and 3. When the uniformity of CSAR gaps becomes more rigorous, requests with higher priority are less frequently admitted, leading to a lower total profit. In contrast, a larger $\varepsilon$ makes the CSAR curves sparser, which favors high-priority requests, thereby generating more profitable admission decisions because the principle that higher priority affords higher service fees is assumed in our work.

The figures in this subsection demonstrate that PSACCF not only retains trustworthy stability when the workload level varies but also offers strong flexibility and adaptability for the SP to cope with diverse situations simply by tuning $\varepsilon$ and $\varphi$.

C. **Comparison of Algorithms**

In addition to PSACCF, the algorithms proposed in [33] and [36] are evaluated for comparison below. The former method (denoted as MHPF) groups received requests into clusters according to their characteristics and then accesses them in order of cluster priority. The newly accepted requests cannot interfere with the resources occupied by other in-service requests. High-priority-first processing persists in the latter method, denoted as AHPF. However, a more aggressive resource scheduling scheme is adopted; that is, new requests are permitted to take resources from ongoing services as long as they are more privileged. Since only two priority levels are considered in [36], we slightly expand the setting to our scenario, where multiple levels coexist, by setting the preemption order starting with the lowest priority.

1) **Priority Indicator:** The priority objective of the three algorithms is shown in Fig. 5 by observing the priority indicator defined in (10) for 1000 slots. The fairness threshold $\varphi$ is set to 1.0, and in each subgraph, PSACCF splits into three curves, each with a specific $\varepsilon$. In the beginning, the priority indicator is low because (10) is vulnerable to rational behavior when the cumulative number of accepted requests is small. Then, the curves of PSACCF and MHPF rise sharply in the early stage, during which PSACCF performs better than MHPF, especially under a higher $\delta$. Eventually this indicator approaches the same value for both PSACCF and MHPF.

As the workload becomes heavier, AHPF shows a similar pattern as others and outperforms PSACCF (with small $\varepsilon$) and MHPF, which is in line with intuition and expectation. Nevertheless, when $\lambda_0$ takes a small value, the curve of AHPF changes; it is neither better than any algorithm nor stable. The reason is that resource seizure occurs only in the first two kinds of service to accommodate requests for the last two when the workload is light. However, the arrival rate of
Service-4 is higher than that of Service-3, which means that more balking will occur and will sometimes result in a higher CSAR for the third service than the fourth (see Fig. 6 a). Then, a violation of the priority requirement occurs. When \( \lambda_0 \) becomes larger, preemption continues to spread to a higher level (see Fig. 6 b), and the priority requirement is satisfied again; thus, the indicator improves. Evidence can be found in Fig. 6, where the CSAR curves generated by AHPF for all services are shown.

It can be concluded from Figs. 5 and 6 that PSACCF achieves almost the same priority performance as the comparison methods and even achieves a faster climbing speed in the growth phase (under a large \( \gamma_1 \)). Moreover, PSACCF performs smoothly against changes in the workload, in contrast to AHPF, which reveals its advantages in the presence of balking behavior.

2) Fairness Indicator: The fairness indicator shown in Fig. 7 exhibits different phenomena from that of priority. The parameter \( \epsilon \) is fixed to 0.15 in this graph, and the results yielded by PSACCF under all \( \varphi \) are shown together with those of MHPF and AHPF. MHPF and AHPF are not good at reaching a high level of fairness, and their performance is even worse when \( \lambda_0 \) increases due to excessive resources leaning towards services with higher priority. The instability of AHPF is also evident here for the same reason as in Fig. 5 a. However, PSACCF is more robust when facing a changing arrival rate, and its fairness curves remain roughly at the position where \( \varphi \) resides and remain relatively flat. The improvement over MHPF and AHPF is approximately 33.6% and 82.9%, respectively, when the base arrival rate is 5 and extends to 146.4% and 153.1%, respectively, with \( \lambda_0 \) tripled.

Fig. 6. CSAR of AHPF.

Fig. 7. Fairness indicator of three algorithms when \( \epsilon = 0.15 \).

To show that the odd performance of AHPF at low workload is indeed due to balking, we forbid balking by setting \( \beta_0 \) to zero in Fig. 8. Unsurprisingly, AHPF obtains the best priority performance at all times but is eventually matched by the other methods. Although there are no jumps in the fairness curve of AHPF, it remains the worst, followed by MHPF, and then PSACCF, where the improvements of PSACCF are 39.5% and 75.0% over MHPF and AHPF.

These graphs illustrate that our algorithm is highly controllable and extremely versatile due to its sensitivity to endogenous parameters, stability with changes in environmental settings, and robustness to impatient behaviors. Satisfactory fairness can be achieved by pulling \( \varphi \) to 1, and one can degrade PSACCF to MHPF by suppressing \( \varphi \) to zero when fairness is completely unimportant.

3) Resource Utilization: We further examined the resource utilization of each algorithm in Fig. 9. For the Type-2 resource, all methods achieve almost 100% utilization. In contrast, Type-1 resources are less utilized, and the utilization continues decreasing as \( \lambda_0 \) increases. This is because of the priority requirement of the three algorithms. The Type-2 resource is much more dominant than Type-1 in Service-4, which has the highest priority level, so the second resource will be exhausted first, leaving idle resources of Type-1. When the base arrival rate increases, more high-priority requests are accepted, so the utilization of the first resource decreases accordingly.

Although the average resource utilization of the Type-1 resource under the three algorithms does not reach 100%, that of PSACCF is significantly highest, approximately 3.4% and 2.5% higher than that of MHPF and AHPF in Fig. 9 b, which increases to 9.2% and 9.6% when \( \lambda_0 \) increases in Fig. 9 c. This is a consequence of fairness optimization. The access opportunities are distributed among requests with better balance by PSACCF, so a portion of resources is reserved for Service-1 and Service-3, whose demand for Type-1 resources exceeds that for Type-2 resources, so the minimum resource utilization rises. Moreover, PSACCF can be parameterized with a narrower \( \epsilon \) or larger \( \varphi \) to increase the lifting range, as shown in the columns of Fig. 9. The promotions in Fig. 9 e are 12.4% and 11.5%, and those in Fig. 9 f are 13.0% and 13.4%.

4) Profit and Time Consumption: For the sake of a comprehensive comparison, the profit and time cost of the three algorithms are included in the simulations. Fig. 10 shows the
cumulative profit earned by the SP when a total of 1000 slots have passed. The best curve belongs to MHPF, followed by AHPF when $\lambda_0$ is less than 13. Resource hogging in AHPF will cause SLA violations and incur a penalty. On the other hand, when $\lambda_0$ is too high for preemption to occur, the profit of AHPF and MHPF can hardly be distinguished. PSACCF suffers some loss of earnings, which is one unavoidable cost to pursue extreme fairness (i.e., $\varphi = 1$ in this figure). Fortunately, the amount of loss can be effectively controlled with a larger $\epsilon$, and it becomes even smaller if the relaxation of fairness is permitted, as demonstrated in Fig. 4.

Finally, we provide the time performance in Fig. 11. Each curve consists of a concatenation of the total time spent per slot calculating the admission decision and allocating resources. Both MHPF and AHPF have very low time overhead, as they consider only resource availability; thus, it appears that PSACCF consumes much more time. Actually, the time cost of PSACCF fluctuates mainly between 1 ms and 4 ms, which is sufficiently small (considering the hardware environment of the simulations) for applications where the time slot usually scales in minutes, seconds, and hundreds of milliseconds. Moreover, adjustments of $\epsilon$ and $\varphi$ hardly influence the time cost, ensuring that PSACCF will not suffer from time availability problems when adjusting parameters for different scenarios.

Overall, these simulation results provide sufficient and convincing proof that the algorithm proposed in this work can achieve better fairness and resource utilization while ensuring the priority requirement for slice admission control. Despite a certain sacrifice in terms of time costs, the degradation is relatively small compared with the length of the time slot and thus does not affect the main functionality and usability of our algorithm. In addition, two adjustable parameters grant PSACCF considerable generalization capability to cope with specific requirements in various scenarios.
VII. CONCLUSION AND FUTURE WORK

In this work, we investigate the slice admission control problem in 5G/B5G networks where different services require heterogeneous resource demands, and impatient behaviors of subscribers are considered. The main focus is to optimize the fairness of admission decisions while respecting differences in service priorities.

We began by reinterpreting the concepts of prioritization and fairness in the SlaaS paradigm so that they can more accurately describe the business characteristics and requirements in the new scenario and are compatible with each other. Then, we proposed a heuristic SAC algorithm called PSACCF based on new definitions to take responsibility for making admission decisions. Simulation results show that PSACCF achieves a comparable priority metric to that of the comparison algorithms and obtains significantly better fairness. More noteworthy is that PSACCF shows good stability to changes in external parameters and is gifted in handling various objectives because of the generalizability and adaptability granted by its endogenous parameters.

In the future, we will pay attention to the slice admission control problem in the presence of multiple competing service providers. In that scenario, each SP may carefully adjust its admission parameters to attract more subscribers from its counterparts to address considerations such as profit, which can lead to gaming behavior among SPs. Thus, we believe this setting will be much more complex and interesting for investigations.

REFERENCES

[1] M. Shafi et al., “5G: A tutorial overview of standards, trials, challenges, deployment, and practice,” IEEE J. Sel. Areas Commun., vol. 35, no. 6, pp. 1201–1221, Jun. 2017.

[2] M. Chaabbar, G. Diaz, A. Dandoush, C. Cérin, and K. Ghomimid, “A comprehensive survey on the E2E 5G network slicing model,” IEEE Trans. Netw. Service Manag., vol. 18, no. 1, pp. 49–62, Mar. 2021.

[3] S. Zhang, “An overview of network slicing for 5G,” IEEE Wireless Commun., vol. 26, no. 3, pp. 111–117, Jun. 2019.

[4] E. C. Posner, “Traffic policies in cellular radio that minimize blocking of handoff calls,” in Proc. 11th Telettraffic Congr., vol. 1, 1985, pp. 1–5.

[5] R. Ramjee, D. Towsley, and R. Nagarajan, “On optimal call admission control in cellular networks,” Wireless Netw., vol. 3, no. 1, pp. 29–41, 1997.

[6] M. O. Ojoji and O. E. Falowo, “A survey on slice admission control strategies and optimization schemes in 5G network,” IEEE Access, vol. 8, pp. 14 977–14 990, 2020.

[7] J. Tang, B. Shim, and T. Q. Quek, “Service multiplexing and revenue maximization in sliced C-RAN incorporated with UURLC and multicast eMBB,” IEEE J. Sel. Areas Commun., vol. 34, no. 1, pp. 881–895, Apr. 2019.

[8] X. Li et al., “Automated service provisioning and hierarchical SLA management in 5G systems,” IEEE Trans. Netw. Service Manag., vol. 18, no. 4, pp. 4669–4684, Dec. 2021.

[9] M. A. Nourian, A. Kisedghi, and A. Akbari, “A practical resource management prototype for mobile networks,” in Proc. 26th Int. Comput. Conf. Comput. Soc. Iran, 2021, pp. 1–6.

[10] Y. H. Hwang and S.-K. Noh, “A call admission control scheme for heterogeneous service considering fairness in wireless networks,” in Proc. 4th Ann. ACIS Int. Conf. Comput. Inf., 2005, pp. 686–692.

[11] W. Orygczak et al., “Fair optimization and networks: A survey,” J. Appl. Math., vol. 2014, pp. 1–25, 2014.

[12] S. Bakri, B. Brik, and A. Ksentini, “On using reinforcement learning for network slice admission control in 5G: Offline vs. online,” Int. J. Commun. Syst., vol. 34, no. 7, 2021, Art. no. e4757.

[13] W. F. Villota Jácome, O. M. Caicedo Rendon, and N. L. S. da Fonseca, “Admission control for 5G network slicing based on (deep) reinforcement learning,” 2021. [Online]. Available: https://doi.org/10.36227/techrxiv.14498190.v1

[14] R. Challa, V. V. Zalyubovskyi, S. M. Raza, H. Choo, and A. De, “Network slice admission model: Tradeoff between monetization and rejections,” IEEE Syst. J., vol. 14, no. 1, pp. 657–660, Mar. 2020.

[15] S. Ebrahimi, A. Zakeri, B. Akbari, and N. Mokari, “Joint resource and admission management for slice-enabled networks,” in Proc. IEEE/IPIN Netw. Operations Manage. Symp., 2020, pp. 1–7.

[16] B. Han et al., “Multiservice-based network slicing orchestration with impatient tenants,” IEEE Trans. Wireless Commun., vol. 19, no. 7, pp. 5010–5024, Jul. 2020.

[17] B. Han et al., “Admission and congestion control for 5G network slicing,” in Proc. IEEE Conf. Standards Commun. Netw., 2018, pp. 1–6.

[18] M. Monemi, M. Rasti, and E. Hossain, “Low-complexity SINR feasibility checking and joint power and admission control in prioritized multitenant cellular networks,” IEEE Trans. Wireless Commun., vol. 15, no. 3, pp. 2421–2434, Mar. 2015.

[19] P. Caballero, A. Banchs, G. De Veciana, X. Costa-Pérez, and A. Azzorra, “Network slicing for guaranteed rate services: Admission control and resource allocation games,” IEEE Trans. Wireless Commun., vol. 17, no. 10, pp. 6419–6432, Oct. 2018.

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

[21] A. A. Al-Khatib and A. Kheli, “Priority and reservation-based slicing for future vehicular networks,” in Proc. 6th IEEE Conf. Netw. Softwarization, 2020, pp. 36–42.

[22] R.-H. Hwang and C.-F. Chi, “Fairness in QoS guaranteed networks,” in Proc. IEEE Int. Conf. Commun., 2003, pp. 218–222.

[23] X. Yang et al., “Genetic algorithm in resource allocation of ran slicing with QoS isolation and fairness,” in Proc. IEEE Latin-Americ. Conf. Commun., 2020, pp. 1–6.

[24] M. Dianati, X. Shen, and S. Naik, “A new fairness index for radio resource allocation in wireless networks,” in Proc. IEEE Wireless Commun. Netw. Conf., 2005, pp. 712–717.

[25] 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.

[26] N. Nasser and H. Hassanein, “An optimal and fair call admission control policy for seamless handoff in multimedia wireless networks with QoS guarantees,” in Proc. IEEE Glob. Telecommun. Conf., 2004, pp. 3926–3930.

[27] G. Liu et al., “Fairness-based joint call admission control for heterogeneous wireless networks: An SMDP approach,” Sci. China Inf. Sci., vol. 57, no. 8, pp. 1–12, 2014.

[28] D. M. Chiu and A. S. Tam, “Network fairness for heterogeneous applications,” in Proc. ACM SIGCOMM ASIA Workshop, 2005, pp. 1–10.

[29] Y. Li, Y. Wang, and X. Sun, “Que-based energy saving resource allocation for video streaming in wireless networks,” in Proc. IEEE Int. Conf. Commun. Workshops, 2016, pp. 547–552.

[30] K. Wang, N. Li, and Z. Jiang, “Queueing system with impatient customers: A review,” in Proc. IEEE Int. Conf. Serv. Operations Logistics, Inform., 2010, pp. 82–87.

[31] Y. Li, Y. Wang, Y. Jin, X. Cheng, L. Xu, and G. Liu, “Research on wireless resource management and scheduling for 5G network slice,” in Proc. Int. Wireless Commun. Mobile Comput., 2021, pp. 508–513.

[32] S. Haryadi and D. R. Aryanti, “Queueing system with impatient tenants,” in Proc. Int. Wireless Commun. Mobile Comput., 2021, pp. 1–6.

[33] A. Perveen et al., “Dynamic traffic forecasting and fuzzy-based optimized admission control in federated 5G-open ran networks,” Neural Comput. Appl., pp. 1–19, 2021.

[34] Y. Han, V. Sciancalepore, D. Feng, X. Costa-Pérez, and H. D. Schotten, “A utility-driven multi-queue admission control solution for network slicing,” in Proc. IEEE Conf. Comput. Commun., 2019, pp. 55–63.

[35] R. K. Jain et al., A Quantitative Measure of Fairness and Discrimination for Resource Allocation in Shared Computer System, Hudson, MA, USA: Eastern Res. Lab., Digit. Equip. Corporation, 1984.

[36] A. Kushchazli et al., “Model of radio admission control for URLLC and adaptive bit rate eMBB in 5G network,” in Proc. IEEE Conf. Comput. Commun., 2021, pp. 74–84.