IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received May 9, 2018, accepted June 8, 2018, date of publication June 12, 2018, date of current version July 6, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2846543

Slice as an Evolutionary Service: Genetic Optimization for Inter-Slice Resource Management in 5G Networks

BIN HAN, (Member, IEEE), JI LIANGHAI, (Student Member, IEEE), AND HANS D. SCHOTTEN, (Member, IEEE)

Institute of Wireless Communications, Technische Universität Kaiserslautern, 67655 Kaiserslautern, Germany

Corresponding author: Bin Han (binhan@eit.uni-kl.de)

This work was supported in part by the European Union Horizon-2020 Project 5G-MoNArch under Grant Agreement 761445 and in part by the Network for the Promotion of Young Scientists (TU-Nachwuchsring), Technische Universität Kaiserslautern (individual funding).

ABSTRACT In the context of fifth-generation mobile networks, the concept of "Slice as a Service" promotes mobile network operators to flexibly share infrastructures with mobile service providers and stakeholders. However, it also challenges with an emerging demand for efficient online algorithms to optimize the request-and-decision-based inter-slice resource management strategy. Based on genetic algorithms, this paper presents a novel online optimizer that efficiently approaches toward the ideal slicing strategy with maximized long-term network utility. The proposed method encodes slicing strategies into binary sequences to cope with the request-and-decision mechanism. It requires no *a priori* knowledge about the traffic/utility models and therefore supports heterogeneous slices while providing solid effectiveness, good robustness against non-stationary service scenarios, and high scalability.

INDEX TERMS 5G mobile communication, business model, communication system operations and management, genetic algorithms, network slicing, optimal scheduling, optimization, resource management.

I. INTRODUCTION

Network slicing was proposed by the *Next Generation Mobile Networks (NGMN) Alliance* [1]. Since then, it has become one of the hottest topics in the filed of future 5th Generation (5G) mobile communication networks. Generally, the concept of network slicing can be understood as creating and maintaining multiple independent logical networks on a common physical infrastructure platform, every slice operates a separate business service with certain Quality of Service (QoS) requirements. Enabled and supported by the emerging technologies of software defined networks (SDN) and network function virtualization (NFV), network slicing exhibits great potentials – as indicated in [2] – not only in supporting specialized applications with extreme performance requirements, but also in benefiting the mobile network operators (MNOs) with increased revenue. A sliced mobile network manages its infrastructure and virtual resources in independent scalable slices, each slice runs a homogeneous service with a specific business model. Thus, an MNO can dynamically and flexibly create, terminate and scale its slices to optimize the resource utilization.

In a previous paper [3], we have proposed a profit optimization model for sliced mobile networks that applies on the traditional business mode: the MNOs with network resources implement the slices and provide all network services directly to their end-users. In this case, a MNO is fully aware of a priori knowledge about the service demands and the cost/revenue models of every slice. It is able to scale the slices in real time according to their utility efficiencies (which can be flexibly defined as such like some QoS or the revenue rate), in order to achieve the maximal overall network utility under the resource constraints. This is a classical multi-objective optimization problem (MOOP), in which the main challenge is to solve the optimal resource allocation, or at least to find a satisfactory solution, with affordable computing efforts.

Unfortunately, this model does not apply on the slices operated by tenants such as mobile virtual network operators (MVNOs), which are considered to play an important role in 5G networks [4]. Tenants are third-parties that provide services without owning any network infrastructure, some instances are utility/automotive companies and over-the-top service providers such as *YouTube*®. To provide

connection services, they have to be granted by MNOs with network resources, including radio/infrastructure resources and virtualized resource blocks, i.e. computation resources. In legacy networks, every tenant makes its contractual agreement with the MNO(s), to pay a fixed and coarsely estimated annual/monthly fee for these resource sharing concepts. In the context of network slicing, in contrast, the resources are first bundled into slices before granted to tenants upon demand. Depending on the slice type, different slices have various utility efficiencies and periodical payments. For example, with the same amount of resource, slices for mobile broadband (MBB) servicesrequire high average user throughput, while slices for massive Machine-Type Communication (mMTC) services focus more to simultaneously serve more low-traffic devices [5]. Even with the same service type, elastic slices can be defined to guarantee an average QoS level for a lower payment, while inelastic slices provide guaranteed minimal QoS level for a higher payment [6]. This approach, usually known as "Slice as a Service" (SlaaS) [7], improves the sharing efficiency and the resource utilization rate. However, as such slices are operated by tenants, and their scales shall be formulated and protected by contractual agreements, a new agreement between the MNO and tenant may therefore be essential to flexibly rescale or terminate a slice at arbitrary time, which leads to extra operations expenditure (OPEX). As an efficient alternative, the MNO can offer resources to implement slices of different types, and macroscopically optimize its resource allocation by choosing if to accept or decline every request from tenant for slice creation. On the other hand, as every slice is logically isolated from the others, a tenant has no access to slices operated by other tenants, but only the administration over its own resources by requesting new slices or terminating active slices of its own. In this case, neither the MNO nor the tenants can jointly optimize all slices in a fully dynamic approach, which disables most classical techniques of resource allocation and proposes a new challenge of network resource management.

First efforts have been made recently on this emerging topic. On the one hand, focusing on how the tenants adjust slice parameters to reduce cost while maintaining the quality of service, a game theory model has been proposed in [8]. On the other hand, taking the MNO's point of view, the authors of [6] have proposed to optimize the slicing strategy in order to maximize the overall revenue. In this paper, we focus on the latter problem, and propose a novel online genetic approach of slicing strategy optimization. Compared to existing methods, our proposed approach encodes every feasible slicing strategy into an individual binary sequence, so that it copes with the binary-decision based inter-slice control mechanism. Furthermore, it requires no pre-knowledge about the utility model, and therefore allows heterogeneous utility functions for different slice types, in order to better support the coexistence of heterogeneous slices with highly various QoS requirements in 5G networks.

The rest part of this paper is organized as follows: In Sec. II, we setup the system model to describe the business process of SlaaS. Then in Sec. III we review the existing methods of resource allocation, especially the Q-Learning method in [6], and discuss about their limits. Afterwards, Sec. IV briefly introduces genetic algorithms to help readers understand our proposed method which we present in Sec. V. Subsequently, we evaluate the performance of our approach through numerical simulations in Sec. VI, before Sec. VII closes the paper with conclusions and some outlooks.

II. SYSTEM MODEL

A. SPACE OF RESOURCE FEASIBILITY

Consider a MNO with M different types of resources to support the maintenance of up to N different types of slices. The resource pool can be therefore described with a M-dimensional vector $\mathbf{r} = [r_1, r_2, \ldots, r_M]^{\mathrm{T}}$. Every slice type $n \in \{1, \ldots, N\}$ is characterized by its resource cost vector $\mathbf{c}_n = [c_{1,n}, c_{2,n}, \ldots, c_{M,n}]^{\mathrm{T}}$. At any time instance, the active slice set can be represented by a N-dimensional vector $\mathbf{s} = [s_1, s_2, \ldots, s_N]^{\mathrm{T}}$, where s_n denotes the number of active slices of type n. Correspondingly, the resource assignment can be described as

$$\mathbf{a} \triangleq [a_1, a_2, \ldots, a_M]^{\mathrm{T}} = \mathbf{C} \times \mathbf{s}, \tag{1}$$

where $\mathbf{C} = [\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_N]$. Thus, the *space of resource feasibility* is given by

$$\mathcal{S} = \{\mathbf{s} : r_m - a_m \geq 0, \forall 1 \leq m \leq M\}, \tag{2}$$

which is illustrated bottom-left in Fig. 1.

It should be noticed that this is a highly abstracted definition of resource for keeping the generality. In practice, network slicing can be applied both on physical resources, i.e. radio/infrastructure resources [9], and on virtualized resource blocks, i.e. computational capacity [10]. The practical design of resource pool, therefore, depends on the use case specification. Generally, all virtualized resource blocks on the same server or server cluster, no matter exploited by which virtual network function (VNF), can be considered as homogeneous and therefore modeled with one dimension of the resource vector $\mathbf{r}$. In contrast, heterogeneous physical resources such as frequency bands and transmission power, must be distinguished with different orthogonal dimensions in $\mathbf{r}$.

It also worths to note, that the linear resource assignment (1) formally excludes any resource multiplexing over different slices, which is, especially for physical resources, not only common in practice but also essential for realizing slice elasticity. Nevertheless, as derived in [6], in the context of inter-slice resource management, an elastic slice that shares resources with other homogeneous slices is equivalent in resource consumption to an inelastic slice with downscaled utility efficiency. Therefore, in this work we consider only inelastic slices for simplification, as elastic slices can be modeled as their inelastic equivalents.

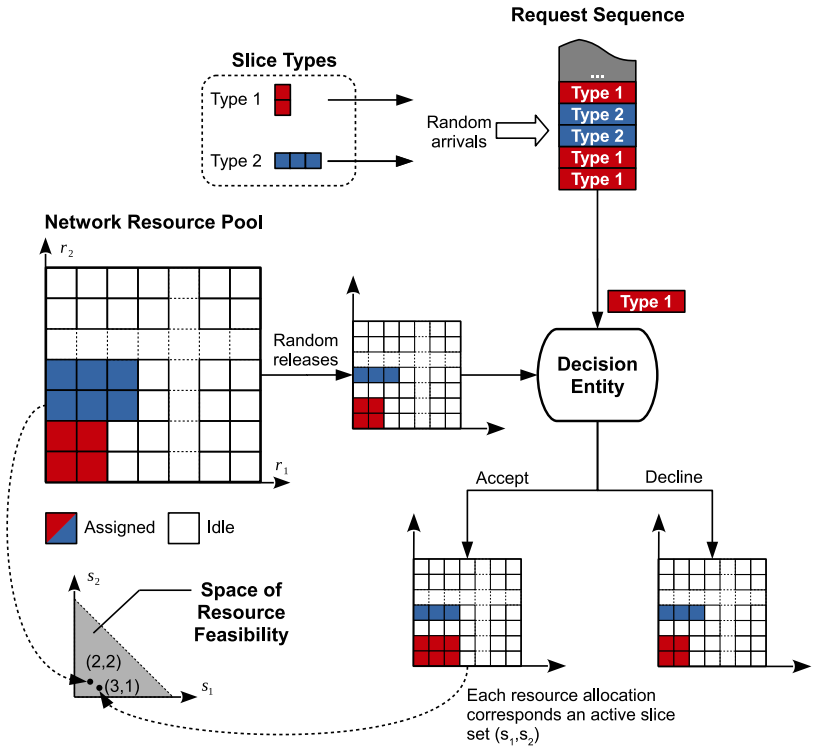


FIGURE 1. System model of inter-slice resource management based on tenant requests and binary decisions. $M = N = 2$ taken for the illustration.

B. RESOURCE ASSIGNMENT AND RELEASE

In SlaaS, resource requests for implementing slices of different types randomly arrive from various tenants. Here we consider a time-frame-based processing of tenant requests: in every time frame, a random number of requests for every type of slices are proposed by tenants. In this paper, we use the term *operations period* to denote the length of this time frame. Once a request for slice type n arrives, the MNO checks if its idle resources can support it to create such a slice. If not, the request will be immediately declined. Otherwise, the MNO can decide if to accept the request or to decline it. Upon acceptance, the MNO creates a new slice of type n, and allocates a corresponding resource bundle $\mathbf{c}_n$ from its idle resource pool to the new slice.

If the request is declined, no slice will be created. This implies that some tenant may fail to immediately obtain the required network resource for their service, especially when the resource pool is highly occupied. To solve this problem, a mechanism of delayed service upon request decline is required instead of a simple denial. For example, a random-access-alike protocol can be designed to let the tenant resubmit its declined slice creation request after a random delay. Alternatively, the MNO can buffer all declined requests in a waiting queue for future decision. In case a bidding mechanism is valid as proposed in [8], the declined tenant can also reattempt with a raised bid for a better opportunity of acceptance. With such approaches, the binary decision mechanism is able to eventually accept every request after some delay. As the scope of this paper focuses mainly on the optimization of binary decision, we consider the random delay approach which keeps the request arrivals Poisson distributed, and do not discuss its impact on the latency of slice creation, which worths further studies in future.

Depending on the business mode, the termination of a slice can either be planned in the request, or randomly happen upon cancellation by the tenant. In this work, we consider the latter case, where every slice of type n has a random lifetime. When a slice of type n is terminated, its resource bundle $\mathbf{c}_n$ will be released and returned to the MNO's idle resource pool.

To simplify the analysis and simulation, in this work we assume that all requests for slice termination only arrive and get handled at the beginning/end of operation periods, while requests for slice termination can arrive any time and will be responded by the MNO immediately. Fig. 1 briefly concludes this procedure of releasing and assigning resources.

C. SPACE OF FREE DECISION AND SLICING STRATEGY

So far, we can describe the MNO's decision on an incoming resource request of slice type n with a binary variable $d \in \{0, 1\}$, where $d = 0$ denotes decline and $d = 1$ stands for acceptance. Upon the decision, the active slice set is updated from its previous value $\mathbf{s}$ to

$$g(\mathbf{s}, n, d) = \begin{cases} \mathbf{s} & d = 0, \\ [s_1, \ldots, s_{n-1}, s_n + 1, s_{n+1} \ldots, s_N] & d = 1. \end{cases} \tag{3}$$

Given a certain space of resource feasibility $\mathcal{S}$, if the decision d is a function of the current active slice set $\mathbf{s}$ and the incoming request n, we say that the MNO has a consistent *slicing strategy*

$$d(\mathbf{s}, n) : \mathcal{S} \times \{1, 2, \ldots, N\} \rightarrow \{0, 1\}. \quad (4)$$

In this case, $g(\mathbf{s}, n, d) = g(\mathbf{s}, n)$, i.e. the new active slice set is uniquely determined by the current active slice set and the incoming request.

As the incoming request n is independent of the current active slice set $\mathbf{s}$, the amount of all different possible constructions of the mapping described by Eq. (3) is $2^{\|\mathcal{S}\| \times N}$, where $\|\mathcal{S}\|$ is the number of all $\mathbf{s} \in \mathcal{S}$. However, as discussed earlier in Sec.II-B, the MNO cannot accept but only decline the request if its idle resources are not sufficient. Hence, we can further restrict the domain of slicing strategy d to the *space of free decision*:

$$\mathcal{D} = \{(\mathbf{s}, n) : \forall \mathbf{s} \in \mathcal{S}, \forall 1 \leq n \leq N, d(\mathbf{s}, n) \in \mathcal{S}\}, \quad (5)$$

whose size $\|\mathcal{D}\|$ is slightly smaller than $\|\mathcal{S}\| \times N$.

D. UTILITY MODEL AND LONG-TERM STRATEGY OPTIMIZATION

Depending on the slice type n, every active slice generates a certain utility in every operations period, which we denote with u_n. Depending on the use case, the utility can be flexibly defined in the tenant's point of view as a function of some specified key performance indicator (KPI) such as the network throughput, the average latency or the network reliability. Alternatively, it can also be defined in the MNO's point of view as a direct payoff such as the payment for renting the network resource bundle. The overall utility generated by all slices in an arbitrary operations period t is

$$u_\Sigma(t) = \sum_{n=1}^{N} s_n(t) \cdot u_n. \quad (6)$$

In this work, our interests focus on selecting the optimal slicing strategy that maximize the expected average overall utility over a long term of T operations periods:

$$d_{\text{opt}} = \arg\max_{d} \mathrm{E}\left\{\frac{1}{T}\sum_{t=1}^{T} u_\Sigma(t)\right\}. \quad (7)$$

Such a strategy is supposed to optimize, depending on the selection of utility function, either the overall performance of the entire sliced network, the economic revenue of the MNO, or other target metric. This is a non-convex optimization problem, where no analytic solution is available and heuristic techniques are therefore needed.

III. EXISTING METHODS AND LIMITS

SlaaS shall be considered as a specific form of cloud computing. The problem of network resource management also commonly exists in the classical cloud environments, including Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS). Since over a decade, various approaches have been proposed to schedule and allocate physical and logical resources over different cloud clients [11], [12].

Nevertheless, the ubiquitous features of 5G network slicing are challenging the deployment of classical cloud resource allocation schemes in SlaaS. First, it usually considers almost homogeneous instances and simple resource pool in classical public cloud environments, which simplifies the resource constraints to one-dimensional [13], [14], or two-dimensional [15]. In 5G networks, as indicated in [3], a large number of slice types N can be required to support highly heterogeneous mobile services, and the dimension of resource pool M can also be considerably large. These can lead to a high computational complexity of global optimizing algorithms with cascaded loops such as [14], and reduce their feasibilities. Second, depending on the use scenario, different slices in 5G networks can even have highly heterogeneous constructions of the utility function u_n. For instance, the energy efficiency is a critical term in the utility function of mMTC slices, the delay is more important for ultra-low-latency reliable communications (URLLC) slices, while the MBB slices are more evaluated regarding the throughput. Classical cloud resource allocating approaches that mostly consider one or few homogeneous cost functions, such as power [16], throughput [17] or resource utilization rate [15], for all instances, can be hardly applied in 5G SlaaS. Novel methods are therefore called for, which are expected to be flexible with various constructions of resource constraints and heterogeneous utility functions. Recently, two numerical algorithms have been proposed in [6] to obtain the global optimum of inter-slice resource management : the Value Iteration which is an iterative full-search approach, and the Q-Learning which is a model-free online machine learning algorithm. Compared to the Value Iteration, the Q-Learning approach is not only capable to support a flexible selection of optimization target, i.e. the utility function, but also proven to effectively reduce the computational cost while approximating the optimal performance.

However, the Q-Learning approach has a drawback intrinsically rooting in its action-based optimization framework, that it intends to maximize the average reward that the MNO receives from every decision it makes. This "decision reward" lacks of intuitiveness in the business view, and is difficult to map onto common business metrics such as the overall network utility defined in Eq. (6).

Furthermore, the Q-Learning algorithm is limited in scalability. The method needs to keep a value table for all possible "actions" that the system can take, and to update the values online through an exploration-exploitation process. In the exploration-exploitation process, the algorithm has a chance of δ to intentionally make a wrong or unevaluated decision, so that it guarantees to traverse all possible actions in long-term. By modifying the value of δ, the algorithm takes its preference between the converging speed and the exploration efficiency. In our case, an action refers to a

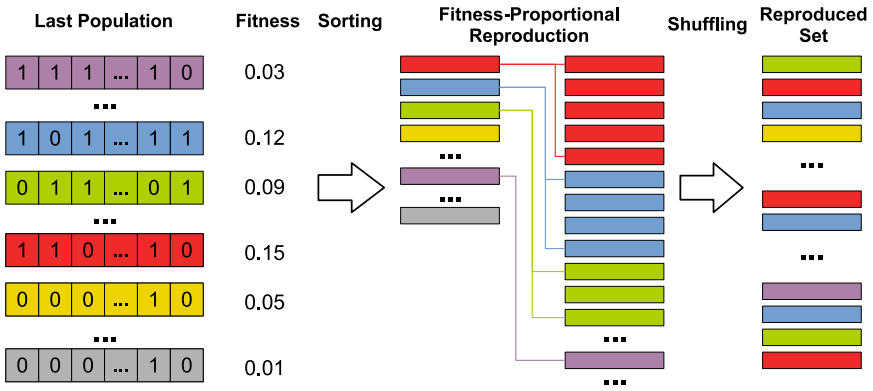


FIGURE 2. The reproduction procedure in standard GAs.

combination of an arbitrary state in the space of free decision and an arbitrary binary decision, so the size of value table is $2\|\mathcal{D}\|$. When $\|\mathcal{D}\|$ grows to a large number, despite of the exploration-exploitation process, the essential time for convergence increases linearly [18] – which reduces the applicability of the algorithm in practice. This problem can become even worse, when the environment, i.e. the statistical behavior of request arrivals and / or slice terminations, is non-stationary.

IV. GENETIC ALGORITHM

Since the 1980s, a category of evolutionary hill-climbing algorithms, known as genetic algorithms (GAs), have been widely applied on various search and optimization problems in the fields of engineering and operations research [19]. They have been proved to be efficient in addressing some difficult challenges in such problems, including large state spaces, incomplete state information and non-stationary environments [20].

A GA is intrinsically integrated with a specified encoder, which maps every candidate strategy to an individual binary sequence (code) of a certain length. At the initialization step, a random set of sequences are selected from the codebook, corresponding to the so-called initial *population*. Each candidate strategy is evaluated to obtain its *fitness*, i.e. the value of objective function to optimize. Subsequently, according to the fitness values of the current population, new populations are iteratively generated. In a standard GA [19], every iteration consists of three sequential steps:

1) *Reproduction*: in this step, every individual strategy in the last population is copied into a new set according to its fitness. The number of copies occurring in the reproduced set is proportional to the fitness value of origin in the last population. The reproduced set has the same size as the last population – so that the better candidates proliferate through the reproduction, while the worst outperformed candidates are eliminated. The procedure is briefly illustrated in Fig. 2.
2) *Crossover*: in this step, all sequences in the reproduced set are randomly paired. Each pair has a chance to randomly swap a subsequence with each other. By doing so, new sequences are randomly generated, where each "child" has a chance to inherit and combine advanced "genes" from its both "parents." A larger chance of swap (*crossover rate*) leads to a faster convergence to the optimum, while also increasing the risk of premature convergence to local maximums. The procedure is shown in the left part of Fig. 3.
3) *Mutation*: where every candidate sequence has a chance to invert one or several random bits of it, which encourages an exploration in the codebook. An increase in either the number of mutation rounds β or the chance of one-bit-mutation per round γ leads to a reduced risk of premature convergence, while also aggravating the meandering during the convergence – and therefore raising the risk of drifting away from the global optimum. The procedure is shown in the right part of Fig. 3.

By iterating these steps, a GA is able to approach to the optimum through a winding process. It shall be noted that, differing from classical optimization techniques, GAs do not guarantee to converge at the global optimum due to an endogenous risk of premature convergence. Nevertheless, such risk can be minimized with a variety of techniques. To the readers with further interest on the converging performance of GAs, we recommend the empirical and analytical studies in [19] and [21], respectively.

Similar to the Q-Learning algorithm, GAs also possess the advantages of model-free and can be applied online. But differing from most reinforcement learning techniques including the Q-Learning, GAs rely on the quantized "fitness" values of different overall strategies instead of the reward value of every single action. This is sometimes considered as a drawback of GAs, because in some applications the fitness function can be difficult to appropriately select [22]. Nevertheless, this is hardly a flaw in the context of SlaaS, because business metrics such as the long-term average network utility defined by Eq. (7) are available as fitness functions. On the contrary, it even benefits the deployment

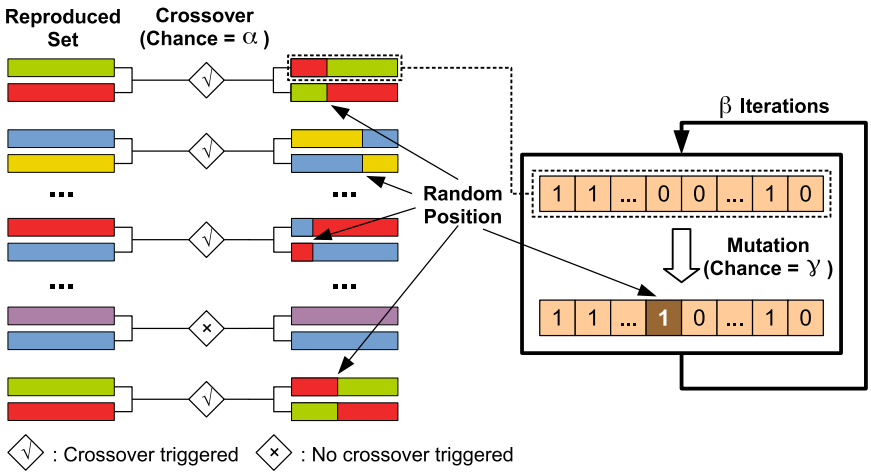


FIGURE 3. The crossover (left) and mutation (right) procedures in standard GAs.

of heterogeneous slices to customize the fitness value with different utility functions for various slice types, as discussed earlier in Sec. III.

Another common complain about GAs is that the strategy encoder can be challenging to design. However, in our case of SlaaS here, the binary nature of slicing strategy $d(\mathbf{s}, n)$ enables a simple and effective encoder design, which is an important and essential novelty of our work in comparison to existing applications of GA on resource allocation, as we will discuss in the next section.

V. PROPOSED METHOD

A. SLICING STRATEGIES AS BINARY SEQUENCE CODES

This is not the first attempt to deploy GA for optimization of resource allocation. Mature solutions have been proposed for allocation of generic resource [23], [24], radio resource [25] and cloud resource [26], [27]. All these approaches consider the problem of global resource optimization, where the system allocates resource blocks from a certain pool to a known set of targets (activities, links, users, etc.). Therefore, they generally aim to optimize the static resource schedule, which is a sequence of resource-target pairs, so every code of theirs represents an individual schedule. In SlaaS, as discussed in Sec. I, the MNO does not jointly rescale existing slices but make binary decisions to every arriving request for a new slice of random type. The target of optimization here is the slicing strategy $d(\mathbf{s}, n)$ as defined in Eq. (4), and the classical encoding scheme in literature therefore does not apply.

Noticing from Eq. (3) that $d(\mathbf{s}, n)$ is a binary function over a limited domain, every individual slicing strategy can be simply encoded into a binary sequence of length $\|\mathcal{S}\| \times N$, where each bit represents the MNO's decision to a request for new slice of specific type $1 \leq n \leq N$ with a given active slice set $\mathbf{s} \in \mathcal{S}$. Furthermore, as discussed earlier in Sec. II-C, the MNO can only make a free binary decision when its current active slice set falls in the space of free decision $\mathcal{D}$. In any other case, the MNO has to decline all incoming requests for new slice creation. Thus, the set of all *feasible* slicing strategies can be enumerated with a codebook of $\|\mathcal{D}\|$-bit-long binary sequences.

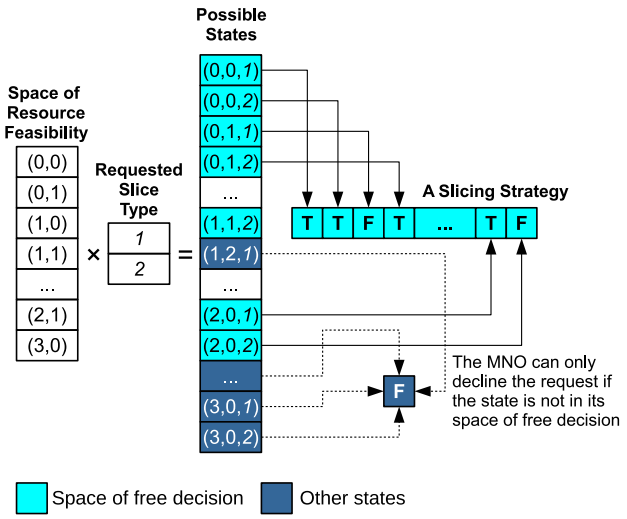


FIGURE 4. Every feasible slicing strategy can be uniquely encoded into a binary sequence by a look-up table, where 'T' stands for True (accept) and 'F' for False (decline). The dark states do not map into the codebook, as they are not in the known space of free decision and therefore always map to 'F.'

Therefore, as the first step to encode slicing strategies, we computed the MNO's space of free decision $\mathcal{D}$, which is a limited enumerable set. Subsequently, we mapped $\mathcal{D}$ to the integer set $[0, \|\mathcal{D}\| - 1]$, which represents the bit positions of a codeword, as illustrated in Fig. 4. By always declining in states outside the space of free decision, i.e. not indexed in the codebook, it guarantees that no decision will break the resource feasibility as far as both the overall resource pool and the list of slice types remain consistent. In case that either of them variates, e.g. when the network infrastructure is

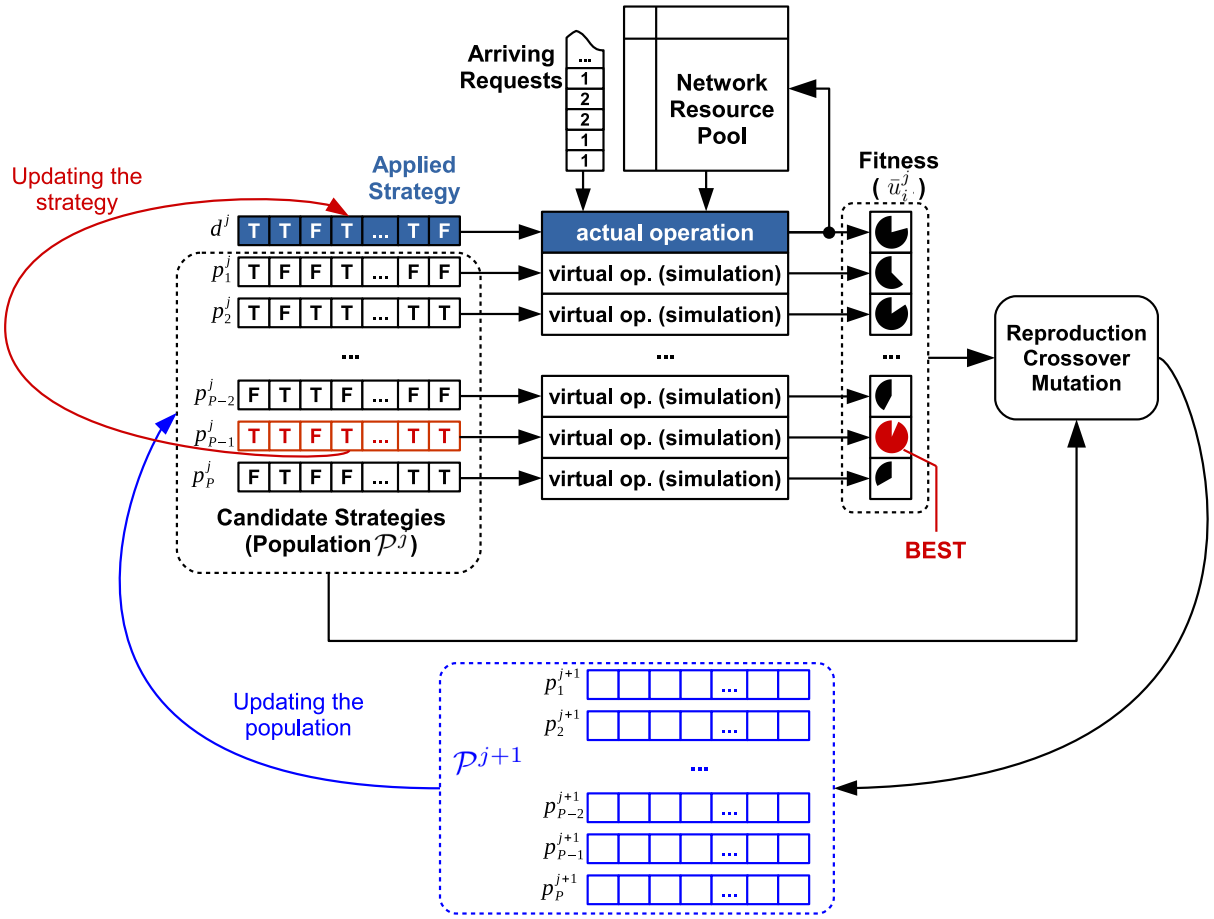


FIGURE 5. Diagram of the proposed genetic slicing strategy optimizer. Both d^j and $\mathcal{P}^j$ are randomly initialized at $j = 1$.

maintained or upgraded, both $\|\mathcal{S}\|$ and $\|\mathcal{D}\|$ have to be recalculated, and the codebook must be correspondingly updated as well.

B. GENETIC SLICING STRATEGY OPTIMIZER

With the code designed above, we implemented a slicing strategy optimizer based on the standard genetic algorithm, which runs in a online mode as Fig. 5 illustrates.

1) INITIALIZATION

An initial population of candidate strategies, $\mathcal{P}^1$ with certain size P, are randomly selected from the pre-generated codebook and kept by the MNO in background for "*virtual*" operation. Meanwhile, an initial strategy d^1 is randomly generated and applied by the MNO for *actual* operation.

2) ONLINE FITNESS EVALUATION

The MNO sets an evolution term $T > 1$ (normalized to one operations period), and records its active slice set at the beginning of every evolution term. As the network runs, the MNO responses every incoming tenant request according to its currently applied slicing strategy, and meanwhile makes an individual "virtual" decision in the background according to every candidate strategy in the current population. For every single candidate strategy, the MNO tracks the simulated utility every operations period. At the end of the j^{th} evolution term, letting $\mathcal{P}^j$ denote the current population, the optimizer evaluates every strategy candidate $p_i^j \in \mathcal{P}^j$ with the average utility it generated (or simulated) over the last evolution term, i.e. the last T operations period:

$$\bar{u}_i^j = \frac{1}{T} \sum_{t=(j-1)T+1}^{jT} u_{\Sigma, p_i^j}(t), \quad \forall 1 \le i \le P, j \in \mathbb{N}^+, \tag{8}$$

which is taken as the *fitness* value (see Fig. 2). This implies that, the more utility efficient a candidate strategy performed in the i^{th} evolution term, the higher fitness value it gets.

3) EVOLUTION

First, the best candidate in $\mathcal{P}^j$ with respect to the fitness is selected to update the strategy for *actual* operation in the next evolution term:

$$d^{j+1} = \arg\max_{p_i^j \in \mathcal{P}^j} \bar{u}_i^j. \tag{9}$$

TABLE 1. The spaces of resource feasibility and free decision under the simulation specification.

Elements in $\mathcal{S}$	Elements in $\mathcal{D}$
[0,0], [0,1], [0,2], [0,3], [1,0], [1,1], [1,2], [2,0], [2,1], [3,0]	[0,0,1], [0,0,2], [0,1,1], [0,1,2], [0,2,1], [0,2,2], [1,0,1], [1,0,2], [1,1,1], [1,1,2], [2,0,1], [2,0,2]

TABLE 2. The model parameters of slice request arrivals and slice terminations in different scenarios.

Scenario	λ_1	λ_2	μ_1	μ_2
#1	0.5	2	2	10
#2	0.3	1	2	3
#3	1	0	2	5

That is, the candidate strategy that had generated / simulated the most utility in the j^{th} evolution term is applied by the MNO for its actual operation in the $(j+1)^{\text{th}}$ evolution term.

Afterwards, a reproduction $\tilde{\mathcal{P}}^j$ of $\mathcal{P}^j$ is generated with respect to the normalized fitness values. The reproduction numbers of an arbirary $p_i^j \in \mathcal{P}^j$ in $\tilde{\mathcal{P}}^j$ is

$$A_i^j = \text{round}\left\{P \times \frac{\bar{u}_i^j + \epsilon}{\sum_{i=1}^{P} \bar{u}_i^j + P\times\epsilon}\right\}, \tag{10}$$

where ϵ is a small number to mitigate error in the rare case that $\sum_{i=1}^{P} \bar{u}_i^j = 0$. The rounding operation here intends to ensure the number of copies A_i^j to be an integer. As shown in Figs.2–3, the elements in $\tilde{\mathcal{P}}^j$ are then shuffled and paired, each pair has a chance of α to execute the crossover operation. After the crossover, every candidate strategy in the new population experiences β turns of mutation, in each turn the candidate strategy has an independent chance of γ to invert one random bit of it, as shown in Fig. 3. The resulted set of strategies is taken to update the population $\mathcal{P}^{j+1}$ for *virtual* operation in the next evolution term.

VI. PERFORMANCE EVALUATION

A. SETUP OF SIMULATION ENVIRONMENT

Towards a brief and convincing demonstration with minimized computational complexity, we considered a MNO with one-dimensional normalized resource pool:

$$\mathbf{r} = [r_1] = [1], \tag{11}$$

which accepts two different slice types, i.e. $M = 1$ and $N = 2$. Thus, the resource cost vector of each slice type $\mathbf{c}_n$ is also one-dimensional, which we set to $\mathbf{c}_1 = \mathbf{c}_2 = [0.3]$. A small space of resource feasibility $\mathcal{S}$ with size of 10 and a small space of free decision $\mathcal{D}$ with size of 12 can be then obtained, as listed in Tab.1. Under this specification, the number of all feasible slicing strategies d sums to $2^{12} = 4096$.

We considered the periodical utilities of the two slice types as $u_1 = 2$ and $u_2 = 1$, respectively, so that the slice type 1 is twice so utility efficient as the slice type 2. Furthermore, we set the length of an evolution term to $T = 6$ operations periods.

B. DEFINITION OF SERVICE SCENARIOS

Similar to [6], we assumed the arrivals of requests for slice creation as Poisson processes, i.e. for every slice type $n \in \{1, 2\}$, the number of arriving requests k_n over one operations period is Poisson distributed:

$$P(k_n \text{ requests arrive}) = e^{-\lambda_n}\frac{\lambda_n^{k_n}}{k_n!}, \quad \forall 1 \le n \le N. \tag{12}$$

Meanwhile, we assumed every slice of arbitrary type $n \in \{1, 2\}$ to have a random lifetime (normalized to one operations period) that obeys the exponential distribution:

$$f(\tau_n = t_n) = \frac{1}{\mu_n}e^{-\frac{t_n}{\mu_n}}, \quad \forall t_n \in \mathbb{N}^+, \ \forall 1 \le n \le N. \tag{13}$$

For our simulations, we defined three service scenarios with different parameter sets $[\lambda_1, \lambda_2, \mu_1, \mu_2]$, as listed in Tab. 2.

C. EFFECTIVENESS

To demonstrate the effectiveness of our proposed method, we simulated two genetic slicing strategy optimizers in scenario #1: one with 10 candidate strategies in every generation, and the other with a larger population size of 50. Each optimizer was initiated with a random population of candidate strategies and a fully idle resource pool, then evolved 20 generations. Aiming at a fast convergence, both genetic optimizers were set to have full crossover rate $\alpha = 1$ and $\beta = 1$ round of mutation with rate of $\gamma = 0.1$. We repeated this simulation 500 times for Monte-Carlo test, and tracked the long-term average network utility defined in Eq. (6). Meanwhile, as a benchmark, the global optimum out of all 4096 feasible strategies was obtained by full-search through 500 times of the same Monte-Carlo test. Besides, we also tested three "naive" reference strategies as baselines for performance comparison:

- *Greedy*: accepting all incoming requests, so long as the resource pool supports
- *Conservative*: accepting all requests for type 2 slices, while declining all requests for type 1 slices
- *Opportunistic*: accepting all requests for type 1 slices, while declining all requests for type 2 slices

As the benchmark strategies do not evolve, they remain constant over all generations. The results are illustrated in Fig. 6. It can be observed that both genetic optimizers started on poor initial utility levels, but then converged to competitive slicing strategies with satisfying performances quickly (within 4 generations). The genetic optimizers failed to achieve the global optimum of utility efficiency within the simulated progress, converging to a local maximum. Nevertheless, from the fourth generation of evolution on, i.e. after evaluating 30 or 150 out of the 4096 strategies, both genetic optimizer outperformed all three static naive reference strategies with long-term average network utilities over 90% with

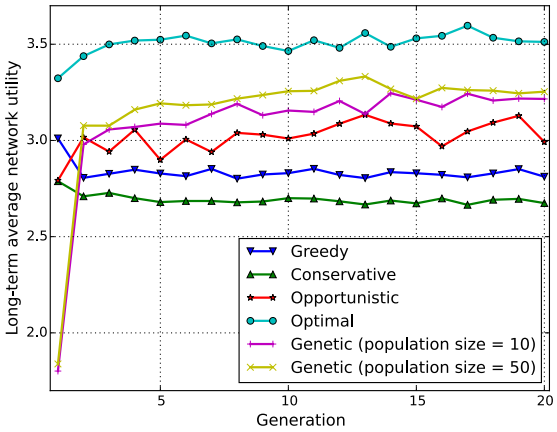


FIGURE 6. Network utilities generated by proposed genetic optimizer in comparison to those under reference slicing strategies.

respect to the global optimum. Additionally, comparing the two optimizers with each other, it can be observed that the increase in population size boosts the convergence.

D. EVOLUTION OF THE ENTIRE POPULATION

An important feature of GA is that not only the best candidate but also the entire population evolve in every iteration. Fig. 7 shows the performance distribution of the genetic optimizer's population with 50 strategies in different generations. A significant approach towards an overall "good" strategy set can be observed. This phenomenon reveals a potential of our genetic optimizer in generating training sets for initialization and updating, when it is jointly applied with other machine learning methods.

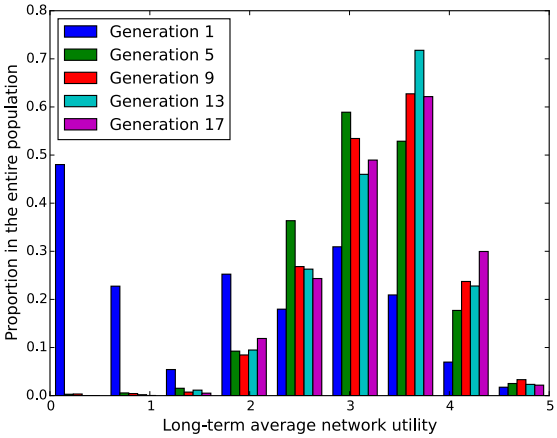


FIGURE 7. As the genetic optimizer runs, the entire population generally approach towards a "good" strategy set.

E. ROBUSTNESS AGAINST NON-STATIONARITY

As mentioned earlier in Sec. IV, GAs are known to be robust against non-stationary environments. In the context of slicing strategy optimization, this refers to time-varying statistical behavior of resource requests and slice terminations. To test our genetic slicing strategy optimizer under such conditions, we conducted a simulation over 60 generations of evolution, i.e. 360 operations periods. For the first 20 generations, the scenario was set to #1, so that the same global optimal strategy obtained in Sec. VI-C remained valid; during the generations 21 to 40, the scenario was set to #2; during the generations 41 to 60, the scenario was updated again to #3. We deployed a genetic optimizer in this scenario, which was specified to the parameters $[\alpha, \beta, \gamma] = [1, 1, 0.1]$ and a population size of 50. Its performance was compared with those of the global optimal strategy obtained in scenario #1, as well as of the three aforementioned naive reference strategies. The results given by 500 times of Monte-Carlo tests are illustrated in Fig. 8. It can be observed that the genetic optimizer succeeded to quickly adapt with environmental variations, and hence remained on a high performance level. In contrast, the scenario-specified optimum gave a poor dynamic performance when the environment changed. Similarly, the performances of all static reference strategies also turned out to strongly rely on the environment.

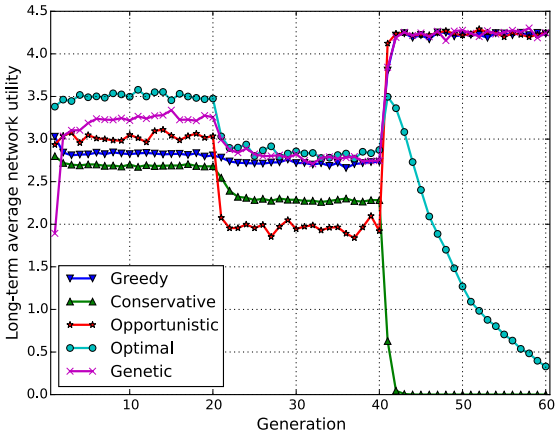


FIGURE 8. Genetic optimizers are feasible in non-stationary scenarios, outperforming all static reference strategies.

F. SCALABILITY AND ENHANCEMENTS

To test the computational scalability of our genetic slicing strategy optimizer, we set a complexer environment with significantly smaller slice scales $\mathbf{c}_1 = \mathbf{c}_2 = 0.03$. Under this specification, the MNO has a space of resource feasibility $\mathcal{S}$ with size 595, a space of free decision $\mathcal{D}$ with size 1122. The amount of its possible slicing strategies d therefore sums to an astronomical figure of 2^{1122}. We also correspondingly scaled the utility efficiencies to $[u_1, u_2] = [0.2, 0.1]$, and set the service scenario parameters to $[\lambda_1, \lambda_2, \mu_1, \mu_2] = [2.5, 10, 2, 10]$.

Then we tested two genetic optimizers with population sizes of 10 and 50, respectively. Both optimizers were set to $[\alpha, \beta, \gamma] = [1, 1, 0.1]$. Once again, we took the reference strategies "Greedy," "Conservative" and "Opportunistic" as benchmarks. No global optimum was evaluated, as the computational cost of full-search for the optimum is

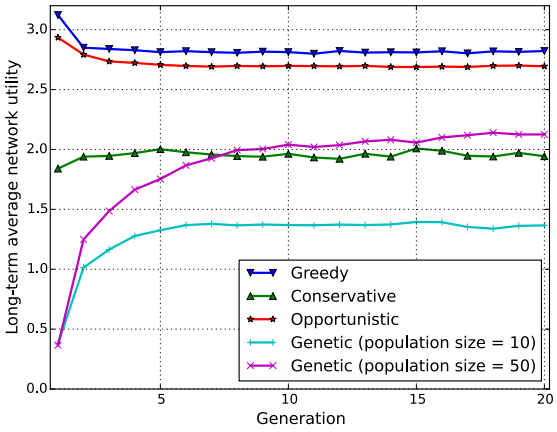


FIGURE 9. When the solution space is huge, genetic optimizers are still able to evolve fast, but can easily converge to poor local maximums.

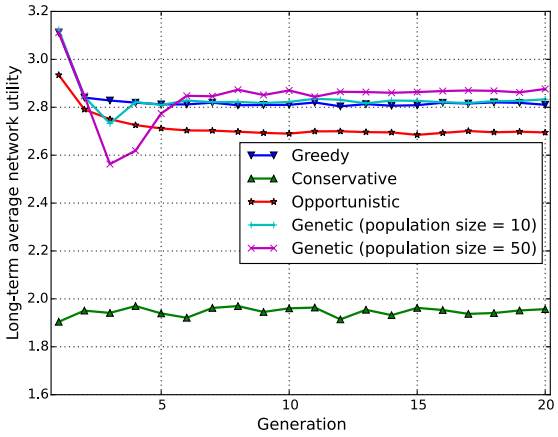


FIGURE 10. With minor enhancements, genetic optimizers guarantee to outperform any certain strategy. The convergence can be further improved at the expense of population size without significantly increased time complexity.

unbearably high. As illustrated in Fig. 9, both optimizers succeeded to quickly converge within 10 generations, but only to reach local maximums that are much worse than all reference strategies.

This phenomenon has its origin in the fact, that as the size of strategy space grows, both the amount of local maximums and the average distance between a random strategy and the global optimum increase. As a consequence, the risk of premature convergence also rises. Additionally, as the GA initiates with a random population, it can easily converge to a poor level.

To counter this effect, efforts can be made in two aspects: 1) to improve the initial population, and 2) to mitigate early convergences at local maximums. The first one can be achieved by involving one or several reference strategies into the initial population, so that the performance evolves from the benchmark level. For the second, either a lower crossover rate or a higher mutation rate can help. Additionally, it is a common technique in GAs to preserve one or several "elite" individuals in every generation from the crossover and mutation operations, and directly put it into the next generation, in order to suppress the random degradation that may caused by mutations [28].

So we repeated the aforementioned simulation, manually involving the reference strategy "Greedy" in the initial random populations of both optimizers. Both optimizers were configured to $[\alpha, \beta, \gamma] = [0.9, 1, 0.1]$ and to preserve one best individual in every generation of population. The results are depicted in Fig. 10. It can be observed that the genetic optimizers either outperformed the benchmark or at least draw it with these simple enhancements. Both optimizers converged within 6 generations.

Another phenomenon that worths to notice is that, the genetic optimizer with smaller population may temporarily outperform the one with larger population in the first generations, as Fig. 10 exhibits. This is determined by the stochastic and winding nature of evolving process, and the fact that the initial population of candidate strategies are randomly selected as well. Nevertheless, as we can learn from Figs. 9 and 10, a large population brings a long-term utility gain when the optimizer eventually converges.

G. SUMMARY

So far, we can assert that our genetic optimizer guarantees to converge to outperform any certain static strategy, while the essential time for convergence only slightly increases with the size of solution space. It is also worth to note that we can improve the convergence by extending the population. As the evaluation of different candidate strategies in the same generation can be easily parallelized [19], the upscaling of population impacts little on the time complexity of our proposed method, making it highly scalable and practical for complex realistic applications.

VII. CONCLUSION AND OUTLOOKS

In this paper, we have presented a novel online genetic slicing strategy optimizer to maximize the long-term network utility in SlaaS. The proposed approach has been evaluated through numerical simulations, exhibiting a satisfying approximate to the global optimum, a fast convergence, a timely adaptation to environment variation and a good scalability. It encodes slicing strategies instead of resource schedules into binary sequences, which enables genetic optimization for inter-slice resource management based on tenant requests and MNO's binary decisions. Besides, it requires no a priori knowledge about the traffic or utility model.

As follow-up work, it remains interesting to enhance the convergence performance of the proposed slicing strategy optimizer with advanced operations and techniques in genetic search, such as fitness scaling, diploid evolution and sequence reordering [19]. Especially, it worths an attempt to ameliorate the rate of convergence of GA with heuristic searching as reported in [23], in order to meet the real-time requirement of network resource management. Besides, as referred in Sec. VI-D, there is also a great potential to combine our

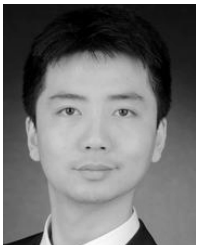
genetic slicing strategy optimizer with other machine learning approaches such as reinforcement learning and artificial neural networks. Additionally, as mentioned in Sec.II-B, different mechanisms to grant declined tenants slices after delays and their impacts on the business case of SlaaS worth further studies, as well.

ACKNOWLEDGMENT

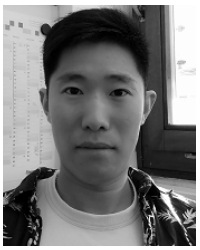
The authors would like to thank the contributions of their colleagues. This information reflects the consortium's view, but the consortium is not liable for any use that may be made of any of the information contained therein.

REFERENCES

[1] *5G White Paper*, Next Gener. Mobile Netw., Frankfurt, Germany, 2015.
[2] R. Peter *et al.*, "Network slicing to enable scalability and flexibility in 5G mobile networks," *IEEE Commun. Mag.*, vol. 55, no. 5, pp. 72–79, May 2017.
[3] B. Han, S. Tayade, and H. D. Schotten, "Modeling profit of sliced 5G networks for advanced network resource management and slice implementation," in *Proc. 22nd IEEE Symp. Comput. Commun. (ISCC)*, Jul. 2017, pp. 576–581.
[4] P. Rost *et al.*, "Mobile network architecture evolution toward 5G," *IEEE Commun. Mag.*, vol. 54, no. 5, pp. 84–91, May 2016.
[5] S. Redana *et al.*, "View on 5G architecture," 5G PPP Archit. Work. Group, Heidelberg Germany, White Paper, Jul. 2016. [Online]. Available: https://5g-ppp.eu/white-papers/
[6] 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. 36th IEEE Int. Conf. Comput. Commun. (INFOCOM)*, May 2017, pp. 1–9.
[7] V. Sciancalepore, F. Cirillo, and X. Costa-Perez, "Slice as a service (SlaaS) optimal IoT slice resources orchestration," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2017, pp. 1–7.
[8] P. Caballero, A. Banchs, G. de Veciana, and X. Costa-Pérez, "Network slicing games: Enabling customization in multi-tenant networks," in *Proc. 36th IEEE Int. Conf. Comput. Commun. (INFOCOM)*, May 2017, pp. 1–9.
[9] S. Vural, N. Wang, P. Bucknell, G. Foster, R. Tafazolli, and J. Müller, "Dynamic preamble subset allocation for RAN slicing in 5G networks," *IEEE Access*, vol. 6, pp. 13015–13032, 2018.
[10] V. N. Ha and L. B. Le, "End-to-end network slicing in virtualized OFDMA-based cloud radio access networks," *IEEE Access*, vol. 5, pp. 18675–18691, 2017.
[11] S. S. Manvi and G. K. Shyam, "Resource management for infrastructure as a service (IaaS) in cloud computing: A survey," *J. Netw. Comput. Appl.*, vol. 41, pp. 424–440, May 2014.
[12] B. Jennings and R. Stadler, "Resource management in clouds: Survey and research challenges," *J. Netw. Syst. Manage.*, vol. 23, no. 3, pp. 567–619, 2015.
[13] A. Inomata *et al.*, "Proposal and evaluation of a dynamic resource allocation method based on the load of VMs on IaaS," in *Proc. 4th IFIP Int. Conf. New Technol., Mobility Secur. (NTMS)*, Feb. 2011, pp. 1–6.
[14] S. He, L. Guo, and Y. Guo, "Real time elastic cloud management for limited resources," in *Proc. IEEE Int. Conf. Cloud Comput. (CLOUD)*, Jul. 2011, pp. 622–629.
[15] T. Tomita and S.-I. Kuribayashi, "Congestion control method with fair resource allocation for cloud computing environments," in *Proc. IEEE Pacific Rim Conf. Commun., Comput. Signal Process. (PacRim)*, Aug. 2011, pp. 1–6.
[16] J. Chabarek, J. Sommers, P. Barford, C. Estan, D. Tsiang, and S. Wright, "Power awareness in network design and routing," in *Proc. IEEE 27th Conf. Comput. Commun. (INFOCOM)*, Apr. 2008, pp. 457–465.
[17] Y. Mei, L. Liu, X. Pu, and S. Sivathanu, "Performance measurements and analysis of network I/O applications in virtualized cloud," in *Proc. IEEE 3rd Int. Conf. Cloud Comput. (CLOUD)*, Jul. 2010, pp. 59–66.
[18] M. G. Azar, R. Munos, M. Ghavamzadeh, and H. J. Kappen, "Speedy Q-learning," in *Proc. 24th Int. Conf. Neural Inf. Process. Syst.* New York, NY, USA: Curran Associates Inc., 2011, pp. 2411–2419.
[19] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*. Reading, MA, USA: Addison-Wesley, 1989.
[20] D. E. Moriarty, A. C. Schultz, and J. J. Grefenstette, "Evolutionary algorithms for reinforcement learning," *J. Artif. Intell. Res.*, vol. 11, pp. 241–276, Sep. 1999.
[21] G. Rudolph, "Convergence analysis of canonical genetic algorithms," *IEEE Trans. Neural Netw.*, vol. 5, no. 1, pp. 96–101, Jan. 1994.
[22] V. Rieser, D. T. Robinson, D. Murray-Rust, and M. Rounsevell, "A comparison of genetic algorithms and reinforcement learning for optimising sustainable forest management," in *Proc. 11th Int. Conf. GeoComput.*, London, U.K., 2011, pp. 20–24.
[23] Z.-J. Lee, S.-F. Su, C.-Y. Lee, and Y.-S. Hung, "A heuristic genetic algorithm for solving resource allocation problems," *Knowl. Inf. Syst.*, vol. 5, no. 4, pp. 503–511, 2003.
[24] Y. Liu, S.-L. Zhao, X.-K. Du, and S.-Q. Li, "Optimization of resource allocation in construction using genetic algorithms," in *Proc. Int. Conf. Mach. Learn. Cybern.*, vol. 6, Aug. 2005, pp. 3428–3432.
[25] S. H. da Mata and P. R. Guardieiro, "A genetic algorithm based approach for resource allocation in LTE uplink," in *Proc. Int. Telecommun. Symp. (ITS)*, Aug. 2014, pp. 1–5.
[26] E. Arianyan, D. Maleki, A. Yari, and I. Arianyan, "Efficient resource allocation in cloud data centers through genetic algorithm," in *Proc. 6th Int. Symp. Telecommun. (IST)*, Nov. 2012, pp. 566–570.
[27] E. Hachicha, K. Yongsiriwit, M. Sellami, and W. Gaaloul, "Genetic-based configurable cloud resource allocation in QoS-aware business process development," in *Proc. IEEE Int. Conf. Web Services (ICWS)*, Jun. 2017, pp. 836–839.
[28] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Trans. Evol. Comput.*, vol. 6, no. 2, pp. 182–197, Apr. 2002.



BIN HAN (M'15) received the B.E. degree in electronic science and technology from Shanghai Jiao Tong University, China, in 2009, the M.Sc. degree in electrical and information engineering from the Technische Universität Darmstadt, Germany, in 2012, and the Dr. Ing. degree in electrical and information engineering from the Kalsruhe Institute of Technology, Germany, in 2016. Since July 2016, he has been with the Institute of Wireless Communication, University of Kaiserslautern, Germany, where he is currently a Senior Lecturer. He has been participating in multiple EU Horizon 2020 research projects for 5G mobile networks. His current research interests are in the broad area of wireless communication systems and signal processing, with special focus on inter-slice resource management.



JI LIANGHAI (S'17) received the B.Sc. degree from Shandong University, China, in 2010, and the M.Sc. degree from the University of Ulm, Germany, in 2012. He is currently pursuing the Ph.D. degree with the Institute of Wireless Communication, Technische Universität Kaiserslautern, Germany. He was involved in European 5G flagship projects METIS and METIS-2 and some other 5G projects with industrial partners. He is currently representing the University of Kaiserslautern in German 5G Project 5G-NetMobil.

HANS D. SCHOTTEN (S'93–M'97) received the Diploma and Ph.D. degrees in electrical engineering from the Aachen University of Technology RWTH, Germany, in 1990 and 1997, respectively. Since 2007, he has been a Full Professor and the Head of the Institute of Wireless Communication, Technische Universität Kaiserslautern. Since 2012, he has been the Scientific Director with the German Research Center for Artificial Intelligence, where he is the Head of the Intelligent Networks Department.