Evolutionary Optimization for Proactive and Dynamic Computing Resource Allocation in Open Radio Access Network

Gan Ruan, Leandro L. Minku, *Senior Member, IEEE,* Zhao Xu, and Xin Yao, *Fellow, IEEE*

***Abstract*—In Open Radio Access Network (O-RAN), intelligent techniques are urged to achieve the automation of the computing resource allocation, so as to save computing resources and increase their utilization rate, as well as decrease the network delay. However, the existing formulation of this problem as an optimization problem defines the capacity utility of resource in an inappropriate way and it tends to cause much delay. Moreover, the only algorithm proposed to solve this problem is a greedy search algorithm, which is not ideal as it could get stuck into local optima. To overcome these issues, a new formulation that better describes the problem is proposed. In addition, an evolutionary algorithm (EA) is designed to find a resource allocation scheme to proactively and dynamically deploy the computing resource for processing upcoming traffic data. A multivariate long short-term memory model is used in the proposed EA to predict future traffic data for the production of deployment scheme. As a global search approach, the EA is less likely to get stuck in local optima than greed search, leading to better solutions. Experimental studies carried out on real-world datasets and artificially generated datasets with different scenarios and properties have demonstrated the significant superiority of our proposed EA over a baseline greedy algorithm under all parameter settings. Moreover, experimental studies with all afore-mentioned datasets are performed to compare the proposed EA and two variants under different parameter settings, to demonstrate the impact of different algorithm choices.**

***Index Terms*—Evolutionary algorithms, Resource allocation, Open Radio Access Network (O-RAN), Computational intelligence.**

I. Introduction

Open Radio Access Network (O-RAN) is a newly proposed wireless network architecture in the fifth generation (5G) or beyond 5G wireless systems [1], [2] with fundamental aspects: Openness and Intelligence [3]. It provides open interfaces for different operators and vendors to deploy their own infrastructures and offer tailored services. Despite being beneficial, these aspects cause the network to become more complex and heavier than any networks before. It is therefore impossible for operators and vendors to depend on human intensive means of optimizing, deploying and operating the O-RAN. Therefore, novel intelligent technologies are welcome to assist the automation of the system [3].

Gan Ruan, Leandro L.Minku and Xin Yao are with CERCIA, School of Computer Science, University of Birmingham, Edgbaston Birmingham B15 2TT, UK (e-mail: GXR847@student.bham.ac.uk, L.L.Minku@bham.ac.uk, xinyao@ln.edu.hk).

Zhao Xu are with NEC Laboratories Europe, 69115 Heidelberg, Germany. (email: zhao.xu@neclab.eu)

Xin Yao is also with the Department of Computing and Decision Sciences, Lingnan University, Hong Kong, China.

O-RAN, as a type of RAN [4], involves two major components, i.e. remote radio head (RRH) for radio resources management and baseband unit (BBU) for signal data processing. One of the most challenging problems in O-RAN is how to allocate the computing resources (i.e, BBUs) to different RRHs such that they can handle traffic data in the network optimally, which in essence is an NP-hard problem. Furthermore, the vendors and operators need to save the computing resources to decrease their capital expenditures and deployment cost. At the same time, the network should ideally have little delay, helping them to provide competitive Internet services and increase user experience. Therefore, it is a significant problem how to proactively allocate the computing resources in the network to achieve the optimization goals of decreasing the required number of computing units, increasing the resources utilization rate, and decreasing the delay. Note that this problem can be also seen as a clustering problem with these objectives.

In order to address this problem, a problem formulation was proposed and a greedy algorithm was suggested to solve it in [4]. However, both the problem definition and the algorithm have limitations. Specifically, the problem formulation has one incorrect factor, since optimizing it has no relation to the three above-mentioned optimization goals, which will be detailed in Section III. Besides, the existing problem formulation is defined in a way that the computing resources may be unable to cope with high traffic in some hours of the day, while being underutilized in other hours of the day. Moreover, the formulation is very likely to induce solutions causing much network delay. In addition, greedy algorithms are known to easily get stuck into local optima [5], [6]. Evolutionary algorithm (EA), as an algorithm with good ability of global search [7], is an alternative to be selected as the algorithm to solve the new problem formulation.

Considering that existing EA operators are not applicable to this resource allocation problem, one of the challenges is how to design tailored EA operators for solving this problem with non-greedy algorithms to avoid getting stuck in local optima. In addition, the RRH traffic data changes from one day to the next and the existing algorithm [4] optimizes the problem of each day from scratch. This would cause redundant computational effort during the O-RAN problem-solving process. Therefore, tailored operators able to avoid redundant computational effort would be desirable to improve allocations for this problem.

To overcome these problems, the following research questions are answered in this paper:

1) How to appropriately formulate the computing resource allocation problem?
2) How to design an EA tailored for solving the new formulation?
    - Does the proposed EA outperform a greedy algorithm? Under what conditions?
    - What is the influence of different algorithm's design choices on its performance?

To answer the first research question, a new problem formulation is firstly proposed to overcome the weaknesses of the existing formulation. Specifically, the proposed problem formulation is defined in a way that an optimal solution would not lead to delays in certain periods of the day while other periods may have resources underutilized. In addition, it equally considers the effect of the delay and computing resource utilization rate on the fitness value of solutions. Besides, the new problem formulation removes the incorrect metric in the existing formulation. A mathematical analysis of the reasons why the proposed problem formulation is more adequate than the existing one will be presented in Section III.

Considering the characteristics of the problem, existing EAs are not suitable for the computing resource allocation problem in this paper, as their representation and evolutionary operators are not directly applicable. Therefore, we propose an EA (called SplitEA) tailored for solving the resource allocation problem in this paper. SplitEA includes problem-specific solution representation, initialization to randomly generate a set of feasible solutions, mutation operator to produce feasible solutions and random cluster splitting to transfer [8], [9] O-RAN resource allocation knowledge and generate feasible solutions so as to speed up optimization.

To simplify the problem, we assume traffic data of all RRHs come in a daily manner. Same to [4], given the traffic data of all RRHs in 24 hours of current day in the network, we use a multivariate Long Short-Term Memory (LSTM) model to predict their traffic data in 24 hours of the next day, such that optimal solutions can be found and deployed beforehand based on the predicted traffic data. We use LSTM following [4] because it can effectively learn the temporal dependency and spatial correlation among base station traffic patterns, and make accurate traffic forecast [4].

Since the RRHs position does not change over time, the problems of two days are similar, only with the traffic data of these RRHs different. Therefore, in SplitEA, except for the problem at the first day where the random initialization is used, we propose a knowledge transfer-based strategy specifically designed for the computing resource allocation problem, called random cluster splitting, which transfers the optimal solutions found for the problem of the previous day to produce an initial population for the problem of the next day.

Given that optimal solutions in the population for the problem of the previous day have converged, the population may have few diversity on the problem of the next day if we directly transfer all of them without any mechanism on them. Therefore, our proposed random cluster splitting also needs to enhance population diversity for the problem of the next day, except for transferring optimal solutions of the old problem. Specifically, it randomly selects a cluster from a solution in the optimized population of the previous day and randomly split it into two clusters (note a solution is a clustering scheme with a set of clusters). Our research hypothesis is that our proposed SplitEA would perform better than the approach based on the greedy search. Experimental studies have been carried out on a series of real-world and artificial datasets to answer two sub-questions of the second research question, which successfully validates the superiority of our proposed SplitEA over the greedy algorithm and also validates the effectiveness our proposed knowledge transfer technique.

The novel contributions of this paper are as follows:

1) A novel problem formulation that better describes the computing resource allocation problem than existing work [4], [10]–[12].
2) A new EA (called SplitEA) specifically designed to solve the new problem formulation, which includes a novel population initialization, mutation operator and random cluster splitting, so as to tackle the challenge of the greedy search regarding easily getting stuck into local optima. Note that all SplitEA operators directly generate feasible solutions via iteratively grouping point(s) into cluster(s) and/or breaking up existing cluster(s) after considering the characteristics of the clustering problem tackled in this paper, unlike most existing EAs that use generic operators to generate solutions without considering constraints and then rely entirely on penalties in the objective functions to eliminate infeasible solutions. The novelties of our proposed SplitEA can be summarized as follows:
    - A novel population initialization: this operator randomly generates feasible solutions via iteratively grouping point(s) into cluster(s), after considering the characteristics of the clustering problem that any two points in a cluster should be no larger than a distance parameter;
    - A novel mutation operator: this operator generates a feasible offspring solution mutated from a parent solution via breaking up existing cluster(s) and/or iteratively grouping point(s) into cluster(s), after considering the characteristics of the clustering problem that any two points in a cluster should be no larger than a distance parameter;
3) More importantly, a novel operator of random cluster splitting is proposed and used in our SplitEA to reduce redundant computational effort during the O-RAN problem-solving process. This operator innovatively makes use of knowledge from previous optimization rounds to generate feasible solutions. It benefits from knowledge gained in the previous optimization process, being able to improve optimization, compared to optimizations from scratch and completely copying all previous solutions.

The rest of this paper is organized as follows: Section 2 introduces the background on the computing resource allocation problem in O-RAN and general approaches to resources allocation problems. Section 3 presented the proposed problem

formulation. Section 4 describes the details of the proposed evolutionary algorithm. Experimental studies are presented in Section 5. Section 6 summarizes this paper and gives an outlook of the future work.

II. Background

A. Related Work on the Computing Resource Allocation Problem

Considering that computing resources are required to be deployed beforehand to tackle the future coming data in O-RAN, it is significant to proactively allocate the computing resources in the network to decrease required number of BBUs, increase the resources utilization rate, and decrease the delay. In order to solve this problem, it is firstly necessary to predict the traffic data of a set of RRHs in the network. The authors in [4] designed a sequence to sequence model that uses unified multivariate LSTM model. The model receives the traffic data of all RRHs in the network at 24 hours of the previous day as the input and outputs the traffic data of all RRHs at each of the 24 hours of the next day.

The authors in [4] also created a problem formulation. They clustered RRHs with complementary traffic patterns together and assigned each cluster to one BBU, such that the BBU capacity can be shared by those clustered RRHs. To define the complementarity of RRHs, the authors considered two aspects: peak distribution and so-called capacity utility, as introduced in [4].

Peak distribution: suppose points represent RRHs and those RRHs assigned with the same BBU are in one cluster. Given a set of clustered n points $C = (r_1, ..., r_i, ..., r_n)$, peak hours for each point in C are defined as follows, which is calculated based on the predicted traffic volumes for the next day,

$$T(r_i) = \{t_{i_1}, t_{i_2}, ..., t_{i_m}\}, 1 \leq i_m \leq 24, \tag{1}$$

where t_{i_m} denotes the m_{th} peak time of r_i. Then the Shannon entropy of the peak hours of the set of clustered points $T(C) = \cup T(r_i)$ is calculated as follows:

$$H(C) = -\sum_{j=1}^{J} p_j log(p_j), \tag{2}$$

where $J = |T(C)|$ is the total quantity of peaks in C and p_k is the probability of observing the corresponding peak hour in the set $T(C)$. A larger entropy value of a cluster indicates that the points are more complementary in the cluster w.r.t. traffic patterns.

Capacity utility aims to make the aggregated traffic of all RRHs in each cluster close to the BBU capacity, which is defined as in Equation (3). To describe the problem formulation in a more concise way, we assume the utility of BBU capacity is 1.

$$U(C) = [mean\ \mathbf{f}(C)]^{-ln(mean\ \mathbf{f}(C))}, \tag{3}$$

where $\mathbf{f}(C) = \sum_{i=1}^{n} \mathbf{f}(r_i) = \left(\sum_{i=1}^{n} f_1(r_i), ..., \sum_{i=1}^{n} f_h(r_i), ..., \sum_{i=1}^{n} f_{24}(r_i) \right)$ is the aggregated traffic volume of the cluster C. Therefore, $mean\ \mathbf{f}(C) = \frac{1}{24} \sum_{h=1}^{24} \sum_{i=1}^{n} f_h(r_i)$. Fig. 2(a) displays the curve of the so-called capacity utility function, which achieves its maximum when the mean aggregated traffic volume is equal to the BBU capacity 1. Therefore, the complementarity of the RRHs cluster C is calculated as follows [4],

$$\begin{aligned} M(C) &= U(C) * H(C) \\ &= -[meanf(C)]^{-ln(meanf(C))} \sum_{j=1}^{J} p_j log(p_j). \end{aligned} \tag{4}$$

Note that quality-of-service requirements may be affected by the propagation delay as the distance between RRHs and BBU increases. Besides, there may be also communication delay among RRHs when enabling machine to machine communications such as handover [13] in the mobile network. Therefore, to alleviate the delay in the network, when allocating the computing resources in the network, the distance between any two RRHs assigned to the same BBU should be constrained within a range.

To solve this problem formulation, the authors proposed a Distance-Constrained Complementarity-Aware algorithm [4] to cluster close RRHs to the same BBU. The basic idea of the algorithm is to iteratively cluster close RRHs into groups, such that the complementarity of RRHs in all clusters is maximized. Specifically, the authors designed a fitness function considering the complementarity of RRHs and the distance of any two RRHs in the cluster. The algorithm greedily assigns randomly selected RRHs to the adjacent cluster with highest fitness value until the termination criteria is satisfied. More details about the algorithm can be found in [4].

However, there are several limitations in the existing framework [4]. Firstly, in the definition of the so-called capacity utility, there are two main limitations. It is clear from Fig. 2(a) that the so-called capacity utility is maximum when $mean\ \mathbf{f}(C)$ reaches BBU capacity 1. However, $mean\ \mathbf{f}(C)$ is the mean traffic volume of all RRHs in this cluster under 24 hours. It is possible that a clustering scheme causing much delay at some hours while very low BBU occupation rate at other hours may be still a good one. This is not captured by this framework. Besides, according to the curve, when $mean\ \mathbf{f}(C)$ is extremely large, it still gets the same capacity utility as that of an almost unoccupied BBU. To this end, the clustering scheme found by this problem formulation may bias towards solutions that cause much delay.

Another limitation of the existing problem formulation is that the peak distribution is incorrect as this definition does not have a direct relationship to the three optimization goals. A simple example will be presented to state why it is not required in Section III. Besides the limitations in the problem formulation, the used greedy algorithm is known to easily get stuck in the local optima [5] [6].

Note that the RRH traffic data changes from one day to the next and the existing algorithm [4] tend to optimize the problem of each day from scratch. Therefore, one challenge of solving this resource allocation problem is how to reduce redundant computational effort during the O-RAN problem-solving process, targeting better solution quality under a given fixed computational budget. Our proposed approach in Section IV will tackle this challenge.

The existence of recent work on the resource allocation problem in 5G network demonstrates that this is still a timely topic worth studying [10]–[12] working on this recently. Even though [10]–[12] are related studies, they are actually different O-RAN problems with different focus and scenarios. Specifically, there is no explicit problem formulation in [10]; instead, the Silhouette score is used as the optimization goal to select the best clustering scheme by clustering approaches from machine learning. Obviously, the three optimization goals, i.e., minimizing the number of computing resources, increasing their utilization rate and decreasing the network delay, were not considered in [10], since a clustering scheme with best Silhouette score does not necessarily mean a best resource allocation to achieve the three optimization goals. In [11], the scenario is to allocate multiple computing resources to process the traffic data at each of all RRHs each time, to minimize wasted resources and unsatisfied user demand. However, this formulation does not consider minimizing the number of used resources, which is important to avoid high costs to the O-RAN providers. In [12], the scenario is to allocate multiple computing resources to each of a set of RRHs by adding or removing one computing resource each time, so as to minimize the number of computing resources and the cost of movements while must satisfying the demands requested by the RRHs (i.e., the allocated resource capacity must be no less than the demand). However, this formulation does not consider increasing the utilization rate of computing resources, which is important for avoiding computing resource waste, especially in the period of demand decreasing rapidly.

Three clustering approaches from machine learning, i.e., K-means, density-based spatial clustering of applications with noise and Agglomerative clustering, were studied in [10] to see which one achieves a clustering scheme with the best Silhouette score. However, none of these clustering approaches is able to consider the objectives of minimizing the number of resources, maximizing their utilization rate and decreasing the network delay. As for [11], [12], different reinforcement learning algorithms were proposed in their paper. However, to include the reinforcement learning algorithm in our comparison, we would need to design totally novel state space, action space, reward functions, etc. for the reinforcement learning algorithms, which would lead to a different algorithm from those proposed in [11], [12].

B. Time Series Prediction Methods

Given that in the computing resource problem traffic data needs to be predicted beforehand for the proactive allocation of the computing resource, several commonly used time series prediction models in the literature [14]–[16] are presented.

Autoregressive integrated moving Average (ARIMA) is one of the most widely used time series analyses model and has been successfully applied to solve many short-term forecasting problems [14]. However, it gets worse prediction errors and confidence on long-term forecasting problems where multiple future steps need to be predicted [15]. In addition, Artificial Neural Network (ANN) models are also leveraged to learn time series properties and predict the trend of the data in the future through using a sliding-window-based strategy [17], which has been used in many domains like operation research [18] and financial market [19]. However, it is difficult for ANNs to model the temporal dependency between the elements in each time series window as analyzed in [4]. Moreover, a multivariate LSTM is proposed in [4] to learn the temporal dependency and spatial correlation of RRH traffic data.

C. Existing Optimization Algorithms for Resource Allocation Problems

There are three types of methods for solving resource allocation problems in the literature, which are linear programming [20] [21], heuristic and metaheuristic methods [22] [23] and other methods [24], [25]. All of them have their advantages and disadvantages. Linear programming tends to find an exact solution for resource allocation problems [20]. However, considering that most problems are highly complex, nonlinear and with constraints, people always simplify the modeling of the linear programming, thus results in inaccurate solutions for the problem. As for a kind of heuristic algorithm, greedy methods are a representative approach to solve the resource allocation problem [26]–[28]. However, it is easy for the greedy algorithm to get stuck into local optima. Many metaheuristic algorithms like evolutionary approaches have been widely applied into addressing the resource allocation problem due to its competent global search ability [29]–[36].

Given that the problem we deal with in this paper is nonlinear and has complex objective function and constraint, it is unable for any linear programming methods to solve. If the linear programming is used to solve this problem, the problem formulation needs to be simplified, thus resulting inaccurate solution for the problem, which is not a good trial. In addition, greedy algorithms, as a heuristic algorithm, easily get stuck into the local optimal, which has been verified and stated in [5] [6]. Therefore, the greedy algorithm that has been used in the existing work [4] may prevents the search of better solutions, which is not ideal.

As far as we know, the comparison algorithm in [4] is the state-of-the-art method to solve the resource allocation problem that firstly predicts the traffic data of RRHs and then optimizes the problem with an approach based on a greedy search. There are recently proposed optimization methods [10]–[12] for the resource allocation problem in 5G network that are based on deep reinforcement learning. They firstly model the dynamic resource allocation problem as a Markov decision process and then solve it using deep reinforcement learning. Even though reinforcement learning achieved some success, it requires many data to train such that they can learn the environment well. Evolutionary algorithms, which do not need any training process and data, are an appealing alternative. As far as we know, there is no work using evolutionary algorithm to solve the problem.

Evolutionary algorithm (EA) [7] is a generic population-based metaheuristic optimization algorithm, which is inspired by the principles of natural evolution and genetics. EA is a valuable tool in the field of computational intelligence and

optimization due to their versatility, robustness, and ability to handle diverse problem types. A conventional EA has the following detailed steps:

1) Initialization: generate an initial population of candidate solutions in the search space as a parent population.
2) Evaluation: calculate the fitness of each individual in the population based on the objective function(s).
3) Reproduction: produce an offspring population via genetic operators on the parent population.
4) Environmental selection: combine the parent and offspring populations to select one population with the fittest individuals as the population for the evolution of the next generation.
5) Termination: check if termination criteria are met; if not, repeat the steps 3 and 4 until a stopping condition is satisfied.

Note that most state-of-the-art EAs [37]–[39] for single objective constrained optimization problems are not suitable for the computing resource allocation problem in this paper. The reason is that the problems they are solving are continuous problems and they use real-codes in the solution representation, which cannot be directly used for solving the discrete resource allocation problem targetted in this paper. In particular, their genetic operators are not suitable for discrete problems.

Moreover, existing state-of-the-art evolutionary algorithms for other resource allocation problems are not applicable to our targetted problem, since different resource allocation problems have different problem characteristics and need specifically designed algorithms. Compared to the resource allocation problems in [40], [41], the problem in this paper have different problem characteristics (including objective function and constraint). The number of resources in the problem of our paper is to be minimized, while this number is pre-given in [40], [41]. In addition, in the problem of our paper, the distance between any two points/tasks assigned to the same resource should not exceed a distance parameter, which is not required in [40], [41]. Such differences require different initialization procedures, genetic operators and constraint handling.

In particular, standard initialization, crossover and mutation operators would lead to abundant infeasible solutions or inferior solutions, especially in problems with small feasible regions. A toy example is used to reveal this, as shown in Fig. 1. Even though existing constraint handling techniques [42] could be used, different techniques have their own limitations on the problem in this paper:

- Feasbility first: this type of methods do not consider solutions with constraint violation. If this type of constraint handling techniques are used, the algorithm would run slowly or even stuck in the initialization process without using any targetted initialization procedure such as the one proposed in our paper, since initialization would be conducted by many times to generate fill a population with feasible solutions or even no feasible solutions would be generated.
- Constraint violation as penalty: this type of methods remove the constraint and add it as a penalty to objective. It is difficult to set the appropriate penalty parameter, since the objective value and the constraint violation have different scale. Moreover, the most appropriate parameter is different for the same problem in this paper under different datasets, since different dataset have different points locations. Therefore, a penalty parameter should be re-tuned when the same algorithm is applied a new dataset. It is also not guaranteed that final optimized solutions are feasible.
- Constraint violation as objective: this type of methods put constraint violation into one of the objectives instead. This method changes the single-objective optimization problem to a multiple one, which makes it more complex. In addition, it might spend many function evaluations on infeasible solutions that might not be of interest. Moreover, if the exact boundary solution is not found, where the first objective (i.e., the constraint violation) is minimized, the best solution might still be infeasible.

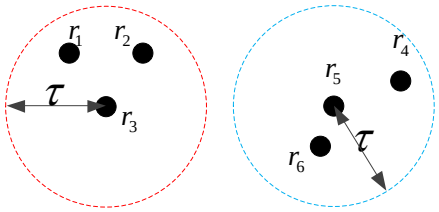


Fig. 1. Given 6 points (r_1 to r_6), suppose $dis(r_i, r_j) < \tau$ ($i, j \in [1, 3]$ or $i, j \in [4, 6]$) and $dis(r_i, r_j) > \tau$ ($i \in [1, 3]$ and $j \in [4, 6]$). We use solution vector $\boldsymbol{X} = \{x_1, x_2, ..., x_6\}$ be a solution vector, where $x_i \in [1, 6]$ corresponds to the identifier of the cluster to which point r_i belongs. (a). Standard initialization would easily initialize an infeasible solution, as long as two points from different circles are assigned to the same cluster (e.g., $x_{inf} = (1, 1, 2, 2, 3, 3)$). In addition, standard initialization would initialize a solution $X_{ini} = \{x_1, x_2, ..., x_6\}$ to make $x_i \in [1, 6]$, since the maximal number of clusters is equal to the number of points. The worse case of a standard initialization would initialize a solution $x_w = (1, 2, 3, 4, 5, 6)$, which is a solution with the maximal number of resources. (b). Given a parent solutions $x_p = (1, 1, 2, 3, 4, 4)$, standard one point mutation would easily generate an infeasible offspring solution such as $x_{o1} = (1, 3, 2, 3, 4, 4)$. This issue of standard initialization and mutation operators becomes much severe when the number of points is large, since any one point is more likely to be assigned to its far clusters. However, our proposed initialization and mutation operators will not generate such solutions.

Therefore, most existing constraint handling techniques are not appropriate to tackle the constraint of the resource allocation problem in this paper.

Moreover, the existing algorithm [4] for the resource allocation problem tends to conduct the optimization from scratch when the traffic data of the next day come. This would cause redundant computational effort during the O-RAN problem-solving process, which is one of the gap in the literature. Our proposed approach in Section IV will overcome this gap.

III. Proposed Formulation of the Computing Resource Allocation Problem

The limitations of the existing problem formulation have been stated in Section II-A. Therefore, in this section, a new problem formulation is proposed to remedy the weaknesses of the existing problem formulation, answering the first research question: How to appropriately formulate the computing resource allocation problem?

To make the problem formulation more mathematical, points represent RRHs and the data of each point means the traffic data of each RRH. In addition, assume those RRHs assigned to the same BBU are in one cluster. Suppose there are N points $R = (r_1, ..., r_i, ..., r_N)$; each point r_i has a fixed position (r_i^1, r_i^2) (where r_i^1 and r_i^2 are the longitude and latitude, respectively) and traffic data at 24 hour of a day $\boldsymbol{f} = \left(f_0(r_i), ..., f_h(r_i), ..., f_{23}(r_i)\right)^T$, where $f_h(r_i)$ is the traffic volume of RRH i in time span between h-th to $(h+1)$-th hour at the current day.

At the end of each day, the clustering scheme of the next day needs to be found to deploy the BBUs to the RRHs in the network. Considering this background, the aim is to proactively cluster these N points to K clusters, so as to maximize the BBU utilization rate, and minimize the delay in the network and the required number of BBUs. Therefore, this problem needs to firstly predict the traffic data of all points and then dynamically find an optimal solution through an optimization algorithm based on the predicted traffic data. This problem can be also regarded as a time series clustering problem.

There may be communication delay among RRHs when enabling machine to machine communications such as handover [13] in the mobile network. Therefore, to alleviate the delay in the network, when allocating the computing resources to the base stations in the network, the distance between any two RRHs assigned to the same BBU should be constrained within a range τ.

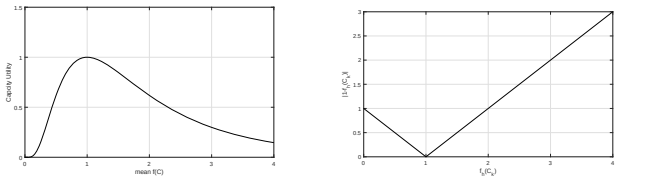
Let $\boldsymbol{X} = \{x_1, x_2, ..., x_N\}$ be a solution vector, where $x_i \in \boldsymbol{X}$ corresponds to the identifier of the cluster to which point r_i belongs. Since the identifier of one cluster is an integer, the value of x_i is an integer, which means that this problem is a discrete optimization problem. The total number of clusters K is equal to the number of unique cluster identifiers in $\boldsymbol{X}$. We will refer to the k-th cluster as C_k, where $1 \leq k \leq K$. The objective function is presented as follows:

$$\begin{cases} \min \mathbf{F}(\mathbf{X}) = w * K + \frac{1}{K} \sum_{k=1}^{K} U(C_k), \\ s.t.\ dist(r_u, r_v) \leq \tau\ (\forall\ r_u, r_v \in C_k), \end{cases} \tag{5}$$

where $w \in (0, 1]$ is a parameter that controls the weight balancing K and $\frac{1}{K} \sum_{k=1}^{K} U(C_k)$; when increasing the value of 'w', it means that operators and vendors tend to minimize the required BBUs more; otherwise, they bias the problem towards the maximization of the BBU utilization rate and minimization of the network delay. $dist(r_u, r_v)$ is the maximum distance of any two points r_u and r_v in k-th cluster C_k; τ is a threshold controlling the distance of neighboring points; $U(C_k)$ is defined as:

$$U(C_k) = \frac{1}{24} * \sum_{h=0}^{23} |f_h(C_k) - 1|, \tag{6}$$

where $f_h(C_k)$ is the sum of the traffic of all points in k-th cluster in time span between h-th to $(h+1)$-th hour of a day, which is defined as:



(a) (b)

Fig. 2. Curve comparison of the existing so-called capacity utility function in [4] and the proposed function in equation 6. (a) The curve of the so-called capacity utility function in [4], which reaches its maximum when the cluster traffic volume equals 1. (b) The curve of $|1 - f_h(C_k)|$ function in equation 6, which reaches its minimal value when the clustered traffic volume equals 1.

$$f_h(C_k) = \sum_{r_m \in C_k} f_h(r_m), \tag{7}$$

where C_k is the set of all points in the k-th cluster, which is presented as follows:

$$C_k = \{r_m | x_m = k\}. \tag{8}$$

Note that in equation (6), the definition of U(C_k) is the summation of the resource utilization rate and the delay of the $k-$th cluster C_k. In this equation, $|f_h(C_k) - 1|$ represents the delay of the $k-$th cluster C_k in time span between h-th to $(h+1)$-th hour of a day when $f_h(C_k) > 1$ while it represents the resource utilization rate when $f_h(C_k) < 1$.

One of the most challenging problems in O-RAN is to allocate the computing resources (i.e, BBUs) to different RRHs such that they can handle traffic data in the network optimally, which in essence is an NP-hard problem. This problem is very likely consisting of a landscape with local optima, such that avoiding these local optima is important. However, no operators currently exist for solving this problem with non-greedy algorithms to avoid getting stuck in local optima. In addition, the RRH traffic data changes from one day to the next and existing algorithms tend to optimize the problem of each day from scratch. Therefore, the other challenge of solving this resource allocation problem is how to reduce redundant computational effort during the O-RAN problem-solving process with the aim of improving solution quality under a given fixed computational budget.

Note that the resource allocation problem is a real world problem in 5G networks, which are currently well known for being widely used as the most advanced wireless technology for mobile phones. This problem is still being studied in 5G networks [10]–[12], being of current practical relevance. The problem formulation in equation (5) is built based on the real world variables and constraints, such that solving it could potentially enable significant savings in computing resources and reductions in network delay.

Remark 1: The reason why we add the K as one part in the fitness function is that the number of clusters is one of the main goals to be minimized to decrease the capital investment of operators and vendors. w is a parameter controlling the weight of number of clusters compared to the averaged summation of delay and BBU utilization rate over all clusters, which gives operators and vendors a choice to make a balance between the investment and the user experience such that they can set it according to their own spirit.

TABLE I
THE VALUES OF $1-U(C_k)$, $H(C_k)$ AND THEIR MEAN VALUE OF DIFFERENT CLUSTERING SCHEMES FOR DIFFERENT DATASETS DEMONSTRATING WHY THE $H(C_k)$ IS INCORRECT.

Datasets	Cluster	1-UC1	HC1	1-UC2	HC2	1-UC3	HC3	$\overline{1-U}$	$\overline{M}$
1h, 2h, 3h	12, 3	0.733	1	0.5	0	N	N	0.617	0.367
[**0.8**, 0.5, 0.3]	13, 2	0.967	1	0.333	0	N	N	**0.65**	0.483
[0.2, **0.7**, 0.1]	1, 23	0.533	0	0.633	1	N	N	0.583	0.317
[0.2, 0.6, **0.7**]	1, 2, 3	0.533	0	0.333	0	0.5	0	0.456	0
	123	0.633	1.585	N	N	N	N	0.633	**1.004**
1h, 2h, 3h	12, 3	0.533	0	0.5	0	N	N	0.517	0
[**0.8**, 0.5, 0.3]	13, 2	0.967	1	0.333	0	N	N	0.65	0.483
[**0.7**, 0.2, 0.1]	1, 23	0.533	0	0.833	1	N	N	**0.683**	0.417
[0.2, 0.6, **0.7**]	1, 2, 3	0.533	0	0.333	0	0.5	0	0.456	0
	123	0.633	0.918	N	N	N	N	0.633	**0.581**
1h, 2h, 3h	12, 3	0.533	0	0.5	0	N	N	0.517	0
[**0.8**, 0.5, 0.3]	13, 2	0.633	0	0.333	0	N	N	0.483	0
[**0.7**, 0.2, 0.1]	1, 23	0.533	0	0.567	1	N	N	**0.55**	0
[**0.7**, 0.6, 0.2]	1, 2, 3	0.533	0	0.333	0	0.5	0	0.456	0
	123	0.367	0	N	N	N	N	0.367	0
1h, 2h, 3h	12, 3	0.287	1	0.15	0	N	N	0.218	0.143
[**0.18**, 0.15, 0.13]	13, 2	0.303	1	0.133	0	N	N	0.218	0.152
[0.12, **0.17**, 0.11]	1, 23	0.153	0	0.283	1	N	N	0.218	0.142
[0.12, 0.16, **0.17**]	1, 2, 3	0.153	0	0.133	0	0.15	0	0.146	0
	123	0.437	1.585	N	N	N	N	**0.437**	**0.692**
1h, 2h, 3h	12, 3	0.287	0	0.15	0	N	N	0.218	0
[**0.18**, 0.15, 0.13]	13, 2	0.303	1	0.133	0	N	N	0.218	0.152
[**0.17**, 0.12, 0.11]	1, 23	0.153	0	0.283	1	N	N	0.218	0.142
[0.12, 0.16, **0.17**]	1, 2, 3	0.153	0	0.133	0	0.15	0	0.146	0
	123	0.437	0.918	N	N	N	N	**0.437**	**0.401**
1h, 2h, 3h	12, 3	0.287	0	0.15	0	N	N	0.218	0
[**0.18**, 0.15, 0.13]	13, 2	0.303	0	0.133	0	N	N	0.218	0
[**0.17**, 0.12, 0.11]	1, 23	0.283	0	0.153	0	N	N	0.218	0
[**0.17**, 0.16, 0.12]	1, 2, 3	0.153	0	0.133	0	0.15	0	0.146	0
	123	0.437	0	N	N	N	N	**0.437**	0

In the column of 'Datasets', each line represents the traffic data of each point at three hour, in which the peak traffic value is highlighted in bold face. In the column of 'Clustering', points with the numbers divided by the comma are in the same cluster. For example, '12,3' means first and second point are in the same cluster while the third point is an isolated cluster. In the last column, the best values obtained by the corresponding clustering scheme are highlighted in bold face. There are five different clustering choices shown in the second column. The clustering scheme with the largest values selected by $mean(1-U)$ or $meanM$ is also highlighted with bold font.

Remark 2: The $U(C_k)$ in equation (6) makes more sense than the one proposed in [4]. The reasons are two-fold. Firstly, minimizing equation (6) is trying to make the summed data of each cluster close to the BBU capacity of 1 at 24 hours of a day rather than the existing one where the average summed traffic data of 24 hours in each cluster is close to 1. In addition, the curve of $y = |f_h(C_k) - 1|$ is shown in Fig. 2(b). It can be seen from this figure that when minimizing equation (6), one solution with underutilized BBU (e.g. $f_h(C_k)$ = 0.5) and another solution with delay (e.g. $f_h(C_k)$ = 1.5) are given the same y value (i.e. 0.5), without any bias towards to BBU utilization rate or delay. In particular, solutions with much delay (i.e. $f_h(C_k) > 2$) are given large y value.

Remark 3: Why we do not need the peak distribution $H(C)$ in the fitness function? It is because optimizing $H(C)$ does not have a direct relation with the three above-mentioned optimization goals. A simple example with several points and their traffic data at several hours is given to explain the reason. In this example, several datasets with different cases are generated, each of which has three points to be clustered and each point has the traffic data at three hours to make $U(C_k)$ and $H(C_k)$ easily calculated. As the $U(C_k)$ in the fitness function is to be minimized and $H(C_k)$ is to be maximized, let $M(C_k) = (1 - U(C_k)) * H(C_k)$ to be maximized. Therefore, the clustering that gets the largest value is the best one.

The clustering schemes for each dataset of this example and the values of $1 - U(C_k)$, $H(C_k)$ and the mean of them is shown in Table I. Each dataset with three points has five different clustering schemes by clustering one, two or three points together, respectively. Among six generated datasets, when doing the clustering on the first three datasets, there might be delay while there is no delay on the last three datasets. For example, when clustering all of the three points in the first dataset together, there are delays in three hours. However, no matter how to cluster the three points on the last three datasets, there is no delay. In addition, the difference among the first three and the last three datasets is in the peak distribution of the traffic data and the peak traffic data among three hours is highlighted with bold font.

It is clear from this table that $meanM$ cannot distinguish all choices on datasets where all points have the same peak hour. In addition, both $meanM$ and $mean(1-U)$ can get the best clustering on datasets where there is no delay in any clustering scheme. However, on datasets where there might be delay, $mean(1-U)$ gets best solution that makes more sense while $meanM$ tends to cluster all points together no matter whether there is delay. To conclude, the peak distribution in the existing problem formulation is incorrect.

IV. SPLITEA: AN EVOLUTIONARY APPROACH TO THE COMPUTING RESOURCE ALLOCATION PROBLEM

This section aims to answer the second research question: How to design an evolutionary algorithm tailored for solving the new formulation? Specifically, the proposed SplitEA tailored for solving the new problem formulation is presented in detail.

As a meta-heuristic algorithm with capability of global search, evolutionary algorithms have been successfully applied to solve many resource allocation problems [29]–[36]. Despite this, the existing evolutionary algorithms for solving resource allocation problems in the literature are not suitable for the problem in this paper, since the problem formulation in this paper is different from that in the literature and evolutionary operators of existing evolutionary algorithms are not directly applicable. More specifically, genetic operators to generate offspring solutions and constraints handling mechanisms in existing evolutionary algorithms are not appropriate for the problem formulation in this paper, since they tend to generate infeasible solutions which cannot be handled well by existing constraint handling techniques. Besides the proposed EA components, the process of random cluster splitting in SplitEA is designed to produce an initial population for the problem of the next day, through randomly selecting a cluster from a solution in the optimized population of the previous day and randomly splitting it into two clusters.

The main novelty our proposed SplitEA is that the proposed EA operators are designed considering the problem characteristics. Different from standard EA operators which operate solutions on solution representation (e.g., vector), our proposed operators operate solutions on the clustering scheme first and then update the solution representation (i.e., vector). Regarding the resource allocation problem as a clustering problem, the operators in SplitEA use the principle of iteratively grouping point(s) into an existing cluster and/or breaking up existing cluster(s) to generate feasible solutions.

Algorithm 1: Proposed SplitEA framework

Input: A set of N points $R = (r_1, ..., r_i, ..., r_N)$ with their position coordinate (r_i^1, r_i^2) (where r_i^1 and r_i^2 are the longitude and latitude, respectively); neighborhood threshold τ; population size $popsize$; maximum number of generations $maxgen$; start date $startdate$ and end date $enddate$.

Output: The found best solutions $\boldsymbol{x}^*$ for days from $startdate + 1$ to $enddate + 1$.

1 Calculate the distance of any two points $dist_{i,j}$ to fill the distance matrix $distM = dist_{i,j}$;
2 Initialize parameters: set the count of generation $gencount$ = 1; set the count of days d = $startdate$;
3 **InitialPop()**: randomly generate a population **P** with a set of $popsize$ feasible solutions;
4 **while** $d \leq enddate$ **do**
5 Input the traffic data of each point $\boldsymbol{f}_i^d = \left(f_0^d(r_i), ..., f_h^d(r_i), ..., f_{23}^d(r_i)\right)^T$ on the current day, where $f_h^d(r_i)$ is sum of data of r_i from h-th to $(h+1)$-th hour at the current day;
6 Utilize a forecasting model to predict the traffic data of each point on the next day ($(d+1)-$th) $\boldsymbol{fp}_i^{d+1} = \left(f_0^{d+1}(r_i), ..., f_h^{d+1}(r_i), ..., f_{23}^{d+1}(r_i)\right)^T$ based on the current day's ($d-$th) traffic data $\boldsymbol{f}_i^d$;
7 Calculate the fitness value of each solution in population **P** based on the predicted traffic data $\boldsymbol{fp}_i^{d+1}$ using the equation (5);
8 **while** $gencount \leq maxgen$ **do**
9 Generate an offspring population $\mathbf{P}_{off}$ using the mutation operator based on the parent population **P**: **Mutation(P)**;
10 Calculate the fitness value of each solution in population $\mathbf{P}_{off}$ based on the predicted traffic data $\boldsymbol{fp}_i^{d+1}$ using the equation (5);
11 Combine two populations **P** and $\mathbf{P}_{off}$ and select the top $popsize$ solutions as the **P** for the next iteration;
12 $gencount$ = $gencount$ + 1;
end
13 Select $\boldsymbol{x}_{d+1}$ with the smallest fitness value from **P** as the solution to be adopted/deployed for day d+1;
14 **RandomClusterSplitting(P)**: conduct random cluster splitting on **P** to produce a new population as the initial population **P** for the next day;
15 d = d + 1;
end

A. Overview of the Proposed Evolutionary Algorithm SplitEA

This section introduces the framework of the proposed evolutionary algorithm, which is exhibited in Algorithm 1. Given a set of N points with their position coordinate in the network, the algorithm firstly calculates the distance of any two points to fill the adjacent matrix in line 1 of Algorithm 1. Then, in line 2, the initialization process of parameters is conducted to set the initial values for the parameters used in SplitEA. The initial values of $gencount$ and d are set as 1 and $startdate$, respectively. After that, the initialization process of the algorithm randomly generates a population **P** with a set of feasible solutions in line 3. Then, allocate the computing resources to all points day to day from the start date $startdate$ to the end date $enddate$.

Within the outer loop of the while from line 4 to line 15, the first step is to input the traffic data of all points on the current day in line 5. Then in line 6, leverage a prediction model to forecast the traffic data of all points on the next day. The input of the prediction model is the traffic data of all points on the current day. Note that, in reality, the traffic data of all points is only completely collected and therefore available at the end of each day. Next step in line 7 is to calculate the fitness value of all solutions in the initial population **P** for this day on the predicted traffic data using the equation (5).

Algorithm 2: InitialPop(): procedures of the population initialization.

Input: A set of N points $R = (r_1, ..., r_i, ..., r_N)$ with their position coordinate (r_i^1, r_i^2) (where r_i^1 and r_i^2 are the longitude and latitude, respectively); distance matrix $disM$; neighborhood threshold τ;

Output: The initialized population **P**.

1 **for** *j:=1 to popsize* **do**
2 Set the cluster count $clucount$ as 0;
3 **while** *there is a point not clustered* **do**
4 Randomly select a point r from the set of points R;
5 Find all points from the rest points to get a set $CloseSet$, in which the distance of any point to r is smaller than τ, the size of the set $CloseSet$ is N_c;
6 Randomly pick points from $CloseSet$ with the number of less than or equal to N_c and group them with r as a cluster;
7 Set the value of those clustered points as: x_k=$clucount$, where k is the index of those points grouped into the same cluster;
8 $clucount = clucount + 1$;
end
9 Set the $j-$th solution as $\mathbf{P}_j = (x_1^j, ..., x_N^j)$;
end
10 **Return P**.

After evaluating the initialized population in line 7, the algorithm iterates until the $gencount$ reaches the pre-setting maximum number of generations within the loop of while from line 8 to line 12. At each generation, the proposed mutation operator is used to produce an offspring population $\mathbf{P}_{off}$ with feasible solutions based on the parent population **P** in line 9. Then in line 10, solutions in the offspring population are also evaluated using the fitness function in equation (5). After that, the parent population **P** and the offspring population $\mathbf{P}_{off}$ are combined together in line 11. Then also in line 11, the top $popsize$ solutions with the smallest fitness values are selected

from the combined population as the **P** for the next generation.

After the iteration before line 13, a solution with the smallest fitness value is selected from **P** as the found best solution for the resource allocation problem at the $d+1$-th day to be deployed. Then, in line 14, conduct the process of random cluster splitting on **P** to produce a new population as the initial population **P** for the next day. This process is trying to improve the clustering structure diversity of the whole population such that better solutions can be found for the next day.

B. Description of Each Step of the Algorithm

This section describes the details of three procedures in the framework of the proposed evolutionary algorithm: the initialization process of the population InitialPop(), mutation operator to produce feasible offspring solutions Mutation(**P**) and the process of random cluster splitting to produce the initial population for the next day RandomClusterSplitting(**P**). These procedures are specifically tailored for solving the resource allocation problem. Notably, all these three procedures are designed in a way that ensures that no solution generated will violate the constraints.

1) Population initialization: Different from standard EA initialization which randomly generate a value within its range for each element of the solution vector, the population initialization process in the proposed SplitEA randomly generates a clustering scheme through iteratively grouping the points into a set of clusters and then assigning the points within the same cluster with the same value in the solution representation. This ensures that no infeasible solutions are generated.

The idea of initializing a feasible solution in the population is illustrated in Fig. 3, which is to iteratively group one randomly selected unclustered point and a set containing a random number of its neighboring points together, until all points are clustered. As shown in the figure, suppose r_1 is firstly selected. Two neighboring points (with the distance to r_1 smaller than τ) and r_1 are grouped into one cluster (black points). A similar procedure is followed for r_2 and r_3.

The detailed procedures of the population initialization are stated in Algorithm 2, which aims to produce a population with $popsize$ feasible solutions. The specific steps for producing each feasible solution are as follows. Firstly, a point r is randomly selected from the set of points in line 4. Then, find all neighboring points of r to form a neighboring set of this point $CloseSet$ in line 5. All those neighboring points have the distance to r smaller than τ. After that, in line 6, randomly pick several points with the number of no larger than N_c to cluster them to point r, where N_c is the size of $CloseSet$. Repeat these steps until all points in the set of all points are clustered. This initialization process aims to generate a population with the number of cluster smaller than the number of points and potentially increase the clustering structure diversity as much as possible, such that the optimal clustering structure could be found in later optimization process.

2) Mutation operator: Different from standard mutations operators which usually mutate one or more gene points of a solution vector to generate mutated solutions, the mutation operator in the proposed SplitEA generates an offspring solution through randomly choosing a cluster from a parent solution (i.e., a clustering scheme) to break up and randomly re-cluster the points. As our problem is a resource allocation problem that can be seen as a clustering problem, our operator will guarantee that the offspring also represents valid clusters.

The idea of how to generate a feasible solution using the designed mutation operator is presented in Fig. 4. As shown in the figure, the process of the mutating a solution is classified into two situation depending on whether there is an isolated cluster in the parent solution. As shown in Fig. 4 (a), when there is an isolated cluster (i.e. the point x as a single cluster), it is grouped to any one adjacent cluster of x (the cluster with black points). Otherwise, as shown in Fig. 4 (b), randomly

Algorithm 3: Mutation(P): procedures of the mutation operator.

Input: The parent population **P** distance matrix $disM$; neighborhood threshold τ; the probability of preferentially clustering isolated points $prob$.
Output: The offspring population $\mathbf{P}_{off}$.

1 **for** *j:=1 to* $popsize$ **do**
2 Select the isolated points as the set $isoPoint$ from $\mathbf{P}_j$, each of which solely forms a cluster;
3 Random generate a number $randNum$ between 0 and 1: $randNum = rand(0,1)$;
4 **if** $randNum < prob$ *and* $isoPoint$ *is not null* **then**
5 Randomly select an isolated point x with the index k from the set $isoPoint$;
else
6 Randomly select a point from all points;
end
7 Find all adjacent clusters of x as $mutClusters$, in which all points have the distance to point x smaller than or equal to τ;
8 **if** $mutClusters$ *is not NULL* **then**
9 Group x into a random cluster $randCluster$ in $mutClusters$ and set the value of x_k as the cluster number of $randCluster$: $x_k = randCluster$;
else
10 Randomly select an adjacent cluster C and put those points whose distance to x is smaller than or equal to τ in the set C_{close}, of which the size is Num;
11 Randomly pick a random number of points between 1 and Num from C_{close}, and group them and x into a new cluster;
12 Set the value of those points in the new cluster as $max(x_i)+1, (x = 1, ..., N)$ (It is impossible that some existing clusters are eliminated since the size of cluster C is larger than Num. The reason is $mutClusters$ is NULL.);
end
13 Get the $j-$th mutated solution $\mathbf{P}_j = (x_1^j, ..., x_N^j)$;
end
14 **Return** $\mathbf{P}_{off}$.

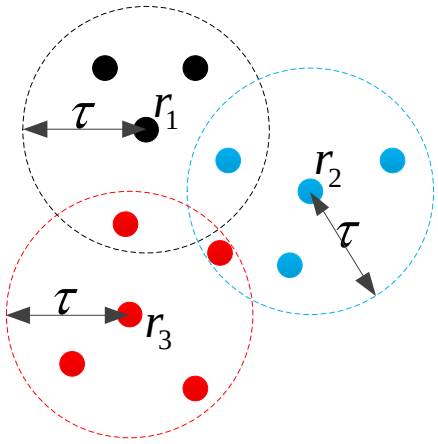


Fig. 3. The process of initializing a feasible solution in the procedure of population initialization. All points are to be clustered. Suppose the first randomly selected point is r_1, find all neighboring points of r_1 whose distance to r_1 is smaller than or equal to τ, and put them in $CloseSet$ (i.e. points within the black dash circle) with the number of N_c (N_c=4); randomly pick several points (2 black points) from $CloseSet$ with the number of less than or equal to N_c and group them with r_1 as a cluster. The above similar procedures can be conducted on r_2 and r_3 to get other two clusters with blue and red points, respectively.

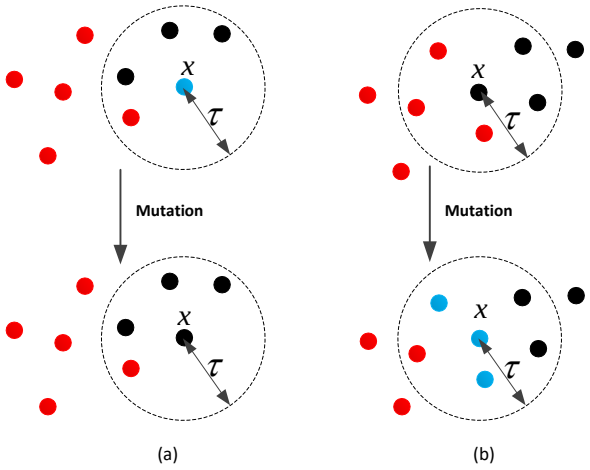


Fig. 4. Illustration of how to conduct mutation on a solution: (a) there is an isolated cluster with only one point. Suppose x is the point in the isolated cluster, x has two adjacent clusters (red and black). Note that the set of adjacent clusters of x, where all points in the cluster have the distance smaller or equal to x, is denoted as mutClusters. Therefore, only the adjacent cluster of x with black points is the mutClusters. Then, x could be grouped to the cluster of black points. (b) There is no cluster with only one point. Suppose x is the selected point, x has one adjacent cluster with red points. The red cluster has three (i.e. Num=3) points with the distance smaller than τ to x. Then, randomly pick less than Num points (supposing 2) and group them and x to form a new cluster (i.e. blue points).

select a random number of points (2 red points) from its x adjacent cluster to group x and selected points as a new cluster (the cluster with blue points).

Specific procedure of the mutation is presented in Algorithm 3. Given a parent population $\mathbf{P}$, the mutation operator produces an offspring population $\mathbf{P}_{off}$ with $popsize$ feasible solutions. For each solution of $\mathbf{P}$, the operator firstly selects the isolated point as the set of $isoPoint$ in line 2, in which each point is formed as one isolated cluster. Here, isolated points are those that are not grouped into any clusters with other points and each of them forms a single cluster. Then, in line 3, randomly generate a random number $randNum$ between 0 and 1. If $randNum$ is smaller than the pre-setting probability $prob$ and the set $isoPoint$ is not empty, randomly select an isolated point x with the index k from the set $isoPoint$ in line 5; else, randomly select a point from all points in line 6. Here, $prob$ controls the weight of decreasing the number of clusters and increasing the diversity of the clustering scheme. For example, if $prob$ is large, more isolated points could be given more priority to be grouped to its adjacent clusters.

Afterwards, the mutation process is conducted on the selected point x, trying to set x_k as the cluster number to produce a feasible solution. Firstly, find all adjacent clusters of x as $mutClusters$ in line 7, in which all points have the distance to point x smaller than or equal to τ. This is to ensure whether x can be directly grouped to the existing cluster. If the set $mutClusters$ is not null, just directly group x into one of the randomly picked cluster in the $mutClusters$ to decrease the number of clusters in line 9; else, in line 10 randomly select an adjacent cluster C of x and put those points whose distance to x is smaller than or equal to τ in the set C_{close} with the number of Num.

After that, randomly pick a random number of points between 1 and Num from C_{close} group them and x into a new cluster in line 11. Following this way, feasible solutions in the offspring population $\mathbf{P}_{off}$ can be produced. In the second case, a new cluster is produced, which results in one more cluster while increase the diversity of the clustering. Therefore, solutions with more resource utilization rate and less delay might be searched.

3) Random Cluster Splitting: Problems at different days are similar to each other, since they have the same position for all points. Therefore, transferring useful information from solving one problem instance to solve another related problem instance could potentially speed up the optimization process for the new instance. However, considering that optimal solutions in the population for the problem of the previous day have converged, the population may have limited diversity on the problem of the next day if we directly transfer all of them without any mechanism on them. Therefore, we propose a knowledge transfer-based strategy random cluster splitting to transfer optimal solutions of the old problem and to generate an initial population with enough diversity on the problem of the next day. The main idea is, for each solution in the population obtained previously, to split a randomly selected cluster into two clusters, in which each cluster has a random number of points.

The process of the random cluster splitting is shown in Fig. 5. As shown in the figure, the red cluster is randomly split into two clusters (red one and the blue one). The reason why not just copying the individuals from the previous population is that copied population has very few diversity regarding the number of cluster, which may prevent the search of solutions with better fitness value and few BBU underutilization and delay. The specific procedures of the random cluster splitting is described in Algorithm 4.

For each solution in the population $\mathbf{P}$, conduct the following random cluster splitting on it to increase the structure diversity of the existing clustering scheme, such that each solution

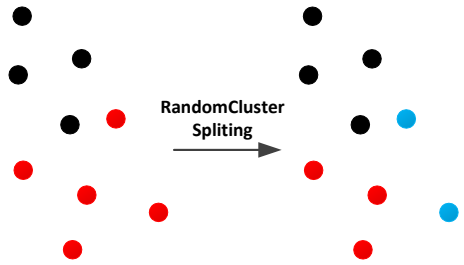


Fig. 5. The process of Random Cluster Splitting. Suppose the solution in the left side has two clusters (black one and red one) and suppose the randomly selected cluster is the red one, there are 5 point in the red cluster. Randomly generate a number between 1 and 5 (suppose it is 2). Then, the red cluster is split to two clusters with two points in the blue (new) cluster.

has more chance to reach better fitness in later optimization. Firstly, in line 2, randomly pick a cluster C with more than one point as the cluster to be split, since there is no need to split with just one point. Then, calculate the number of points in the cluster C: NR in line 3.

Next, in line 4, randomly generate a number between 1 and $\lfloor NR/2 \rfloor$: $Nsplit = rand(1, \lfloor NR/2 \rfloor)$. This also enables the diversity of the clustering structure. After that, in line 5, split the cluster C into two clusters with size of $Nsplit$ and $NR - Nsplit$, respectively. Lastly, set the value of those points in the cluster with the size of $NR - Nsplit$ as $max(x_i) + 1, (x = 1, ..., N)$. Through this random cluster splitting process, the produced solutions can decrease the number of cluster by 1 and therefore maintain most clustering structure. At the same time, the diversity of the clustering structure can be increased through splitting the cluster, which might generate better solutions for the problem at the next day.

Algorithm 4: RandomClusterSplitting(P): procedures of the random cluster splitting.

Input: The population **P** of the problem at the previous day; distance matrix $disM$; neighborhood threshold τ;
Output: The initialized population **P** for the problem at the next day.

1 **for** *j:=1 to popsize* **do**
2 Randomly select a cluster C with more than one point;
3 Calculate the number of points in the cluster C: NR;
4 Randomly generate a number: $Nsplit = rand(1, \lfloor NR/2 \rfloor)$;
5 Split the cluster C into two clusters with size of $Nsplit$ and $NR - Nsplit$, respectively;
6 Set the value of those points in the cluster with the size of $NR - Nsplit$ as $max(x_i) + 1, (i = 1, ..., N)$;
7 Set the $j-$th solution as $\mathbf{P}_j = (x_1^j, ..., x_N^j)$;
end
8 **Return P**.

C. Time Complexity of the Proposed Evolutionary Algorithm

This section analyzes the time complexity of the proposed knowledge transfer-based evolutionary algorithms. The processes that consume main time complexity will be analyzed. Step 1 in Algorithm 1 takes $O(N^2)$ computations, where N is the number of points. The while loop from Step 3 to Step 8 in Algorithm 2 takes less than $O(N)$. Therefore, the population initialization consumes maximum $O(N * popsize)$ computations. At each generation of the evolutionary algorithm, mutation process consumes the most time complexity. Step 7 of the Algorithm 3 consumes $O(N)$ computations while all other steps in the loop from Step 2 to Step 12 in Algorithm 3 consumes less than $O(N)$ computations. Therefore, the mutation process consumes $O(N * popsize)$. Steps 2 to 7 in the Algorithm 4 consumes $O(NR - Nsplit)$, as Step 6 consists a for loop taking $O(NR - Nsplit)$ computations. Therefore, the process of random cluster splitting takes $O(popsize)$ computations. To conclude, the proposed evolutionary algorithm except for the prediction model consumes $O(days * maxgen * N * popsize)$ computations, where $days$ is the number of days in the dataset and $popsize$ is the population size.

V. Experimental Studies

This section introduces datasets, compared algorithms, parameter settings for them and performance metrics, so as to validate the effectiveness of our proposed evolutionary algorithm (i.e. SplitEA). Specifically, Section V-B1 answers the first sub-research question of the second research question: Does proposed EA outperform a greedy algorithm? Under what conditions? Then, the second sub-research question (i.e. what is the influence of different algorithm's design choices on its performance) is answered in Section V-B2.

A. Experimental Setup

1) Datasets Description: There are two sets of datasets to be used, one of which is the real-world datasets found online while another is the artificial datasets. The real-world datasets are used to verify whether the proposed approach is able to solve the real-world problems better than the greedy algorithm. However, the available real-world datasets have limited types of location and traffic dataset. In order to verify the effectiveness of the proposed algorithm on datasets with more properties, several artificial datasets with the combination of different RRH location and different traffic dataset are generated.

a) Real-world Datasets: There are four real datasets found in the literature: Milan [43], Songliao Basin [44], C2TM [45] and Archive [11]. Note that the Archive dataset does not have the location information, three location datasets are generated to complement it, via randomly selecting from the location dataset of Milan and Songliao Basin and randomly generating within a range. Therefore, there are six real-world datasets in total. The specific description of datasets are explained in the Supplementary File.

b) Artificial Datasets: This paragraph describes the generated artificial datasets and how they are combined with different types of generated location dataset and generated traffic dataset. There are 8 generated artificial datasets with seven days, which includes 1a, 2a, 3a 100/158, 3a 120/158,

1c-Milan, 1c-Songliao, 2b-Np=10 (Nt=174) and 2b-Np=5 (Nt=185), where Np is the maximal number of points in each group for the location dataset, Nt is the total number of points in the solution. Note that in dataset 2a, the maximal number of points (Np) in each group for the location dataset is set as 5. Among those datasets, '1', '2' and '3' means the first, second and third location dataset, respectively; 'a', 'b' and 'c' means the first, second and third traffic dataset, respectively. The specific description of artificial datasets are explained in the Supplementary File. These datasets and the codes on specifically generating those datasets will be released online.

2) Model Specification: In the experiments on real-world datasets with the prediction, we use the LSTM model to do the prediction, which is used in Step 6 of Algorithm 1. The reason why we use LSTM is that it can capture the temporal dependency and spatial correlation among base station traffic patterns [4] and it has been proved to achieve better prediction results than ARIMA and WANN in [4]. The specification of the model and the model parameters are the same as those used in [4]. The model has two stacked LSTM layers. The encoder layer $L1$ contains N_{en} encoder memory units, which accepts a traffic snapshot of shape [24;182] as input, and outputs an encoded sequence for the decoder. The decoder contains N_{de} decoder memory units, which accepts the encoded sequence as input and outputs the forecast of the traffic snapshot.

Considering that only the Milan and Archive datasets have many days to do the prediction, compared algorithms on those two days will use the LSTM to do the prediction. The division of the training set and testing set is also the same as that in [4], i.e. the first 70% days of the dataset is regarded as the training set with the remaining days as the testing set. For other datasets except Milan and Archive which do not have enough days, no splitting process is conducted on them, which are not used to do the prediction. To save space, the prediction errors of the LSTM model on two realworld datasets Milan [43] and Archive [11] are shown in Fig. 1 of the Supplementary File.

3) Compared Algorithms: There are four compared algorithms in the following experiments. In order to verify the effectiveness of our proposed SplitEA in solving the resource allocation problem, we compare SplitEA to the greedy algorithm [4] as it is the state-of-the-art algorithm for solving the computing resource allocation problem in open RAN of this paper. Moreover, to verify the effectiveness of the proposed knowledge transfer-based strategy (i.e., random clustering splitting) in SplitEA, two algorithms (i.e., the following RandEA and CopyEA) are modified from SplitEA to be compared to SplitEA.

1) The greedy algorithm (GreedyAlg): modified from the greedy algorithm in the existing work [4] through applying the fitness function of the proposed problem formulation and setting the termination criteria as the pre-set maximum number of evaluations. In the greedy algorithm of the existing work [4], it calculates the fitness value of clusters formed by a selected point to its adjacent clusters, of which more details can be found in [4]. However, in the greedy algorithm here, it calculates the fitness function of the cluster schemes (solutions) formed by a selected point to its adjacent clusters as well as other clusters. The greedy algorithm here is regarded as a baseline algorithm to verify the effectiveness of the proposed evolutionary algorithms. The aim of setting the maximum number of evaluations as the termination criterion is to make a fair comparison to the proposed evolutionary approach through setting the evaluation times the same for all compared algorithms.
2) RandEA: replace step 14 of Algorithm 1 with Algorithm 2: it randomly generated an initial population whenever solving the problem of the next day; other procedures remain the same. The reason why to introduce this algorithm is to verify whether the random cluster splitting is better than optimization from scratch.
3) CopyEA: replace step 14 of Algorithm 1 with the following process: just copy **P** as the initial population **P** for the next day. CopyEA is regarded as the compared algorithm to prove whether the random cluster splitting process is better than simply copying all previous solutions.
4) SplitEA: our proposed evolutionary algorithm.

In order to verify the performance of our proposed SplitEA over the greedy algorithm, we compare SplitEA against the greedy algorithm on six real-world datasets and eight artificial datasets. In addition, we analyze the impact of different parameter settings on the performance of SplitEA and the greedy algorithm on Milan dataset. The comparison results are shown in Section V-B1. Similarly, the performance of different algorithm design choices and the influence of parameter settings on them are also analyzed. The comparison results are shown in Section V-B2.

4) Parameter Settings: In this section, we describe the used specific parameter settings, like the mutation probability and the parameters of the problem.

- $prob$=0.5 in the improved mutation, which makes the probability of clustering isolated points into a cluster and that of searching for different clustering structure equal.
- Neighborhood threshold $\tau = 3d$, where d is the average distance of all point to their closest point, which is selected based on the experience.
- Evaluation time in GreedyAlg = 1500.
- w = 0.01 in the problem formulation; $popsize$ = 10 and $generation$ = 150 for all compared EAs including SplitEA, RandEA and CopyEA, which are set to make the evaluation times = 1500, equal to that of the greedy algorithm to enable fair comparison. The evaluation time, generation and population size are selected based on the experience.
- Mutation probability: 1, as there is only one operator to evolve the population;

The impact of these parameters will also be analyzed in Section V-B1 and Section V-B2.

5) Performance Metrics: Metrics to evaluate the performance of found solutions by the greedy algorithms and the evolutionary algorithms are shown as follows. They are set according to the optimization objectives of the problem, so that we can investigate how well the algorithm is able to optimize these objectives.

1) The number of clusters: K;

2) The average difference between all points data and 1 for all clusters: U, which is defined as follows:

$$U = \frac{1}{K}\sum_{k=1}^{K} U(C_k). \tag{9}$$

3) The average difference of all points data and 1 when there is delay: Udelay, which is described as follows:

$$Udelay = \frac{1}{K}\frac{1}{24}\sum_{k=1}^{K}\sum \big(f_h(C_k) - 1\big), \quad when\ f_h(C_k) > 1. \tag{10}$$

4) The average difference of all points data and 1 when there is NO delay: Uunder1, which is described as follows:

$$Uunder1 = \frac{1}{K}\frac{1}{24}\sum_{k=1}^{K}\sum \big(1 - f_h(C_k)\big), \quad when\ f_h(C_k) \leq 1. \tag{11}$$

Note that all metrics are required to be minimized. Minimizing Uunder1 means maximizing the utilization rate.

For the computational studies, each algorithm is given 30 independent runs. Friedman and Nemenyi statistical tests [46] with the significance level 0.05 are used to indicate the statistical significance of all compared algorithms. The metric value obtained by a given algorithm on one dataset is regarded as an observation to compose that algorithms group for the test, following Demsars guidelines [46]. Therefore, there are 30 observations in each group.

B. Experimental Results

This section firstly presents the comparing results between the proposed SplitEA and the greedy algorithm on the real-world datasets, to show the superiority of our proposal. Then, the comparison results of SplitEA and the greedy algorithm on artificial dataset are shown to verify the performance of them under different scenarios, like different distribution of points and different traffic pattern. Later on, three EAs (RandEA, CopyEA and SplitEA) are compared on both real-world and artificial datasets to verify the benefit of the process of random cluster splitting in SplitEA.

TABLE II
COMPARISON RESULTS OF SPLITEA AND THE GREEDY ALGORITHM ON REAL-WORLD DATASETS.

Datasets	Milan		Songliao		C2TM-sub1	
Algorithms	GreedyAlg	SplitEA	GreedyAlg	SplitEA	GreedyAlg	SplitEA
K	81.3907	52.3019	104.7381	54.3619	59.6792	51.7708
U	0.8388	0.7644	0.8715	0.8042	0.9961	0.9956
Udelay	0.0003	0.0076	0.0001	0.0234	1.25E-07	2.97E-07
Uunder1	0.8385	0.7568	0.8714	0.7760	0.9961	0.9956
f	1.6527	1.2874	1.9189	1.3478	1.5929	1.5133

Datasets	Archive-Milan		Archive-Songliao		Archive-Random	
Algorithms	GreedyAlg	SplitEA	GreedyAlg	SplitEA	GreedyAlg	SplitEA
K	15.3396	13.0352	15.6082	13.2629	15.2642	12.0069
U	0.7839	0.7536	0.7880	0.7538	0.7823	0.7260
Udelay	0.0001	0.0039	0.0002	0.0022	0.0001	0.0009
Uunder1	0.7837	0.7497	0.7878	0.7517	0.7822	0.7250
f	0.9372	0.8840	0.9440	0.8865	0.9349	0.8460

The values in this table are the mean value of the metrics under 30 independent runs. Friedman and Nemenyi statistical tests [46] with the significance level 0.05 are used to indicate the statistical significance between compared algorithms. The significantly better values obtained by the algorithm are highlighted in red color.

1) Comparison Results of SplitEA and the Greedy Algorithm: This section tries to answer the first sub-research question of the second question: Does proposed EA outperform a greedy algorithm? Under what conditions? Specifically, this section presents the comparison results of SplitEA and the greedy algorithm on several real-world and artificial datasets, to show the superiority of our proposal over the greedy algorithm.

Firstly, SplitEA and the greedy algorithm are tested on the real-world datasets by using the real traffic data in the fitness function instead of the predictions of such traffic data. This has the aim of evaluating the optimization mechanisms themselves in the proposed SplitEA. Then, two algorithms are tested on two real-world traffic datasets (Milan and Archive) with predictions to see whether the prediction error affects the performance of SplitEA through using the predicted traffic data in the fitness function. Lastly, two compared algorithms are tested on artificial datasets with different scenarios, to further verify the effectiveness of the optimization mechanisms themselves in the proposed SplitEA on more datasets beyond the real-world datasets. Note for those experiments without using the prediction model, the Step 6 in Algorithm 1 is deleted and in Step 7, the fitness value of solutions is calculated on the real traffic data.

a) Comparison results of SplitEA and greedy algorithm on real datasets: The comparison results of the greedy algorithm and SplitEA on the real dataset are shown in Table II. It is clear that SplitEA gets significant better fitness values on all datasets, which shows that the SplitEA is able to get better solutions as expected. In addition, SplitEA achieves significant better values of all 4 metrics on all datasets except for the metric $Udelay$.

The reason why the compared two algorithms get equal $Udelay$ on the dataset $C2TM-sub$ has been analyzed that the traffic data of most points at most hours of each day is too small to cause delay. This can be reflected by the results of the computing resource utilization rate ($Uunder1$), as the traffic data is too small to make the BBU rather underutilization. For all other datasets, SplitEA gest worse significant $Udelay$. It is intuitive to get the reason that SplitEA tends to cluster more points together due to the less required number of clusters for fixed number of points and this would inevitably increase the delay in the network.

b) Comparison results of SplitEA and greedy algorithm on real-world datasets with the prediction: The comparison results of the greedy algorithm and SplitEA on the real-world datasets with predicted traffic data are shown in Table III. Note that only Milan and Archive datasets have enough days to do the prediction. Therefore, there are only four datasets used to test the ability of compared greedy algorithm and SplitEA on the predicted traffic datasets. It is clear from Table III that SplitEA gets significant better fitness value on all predicted datasets.

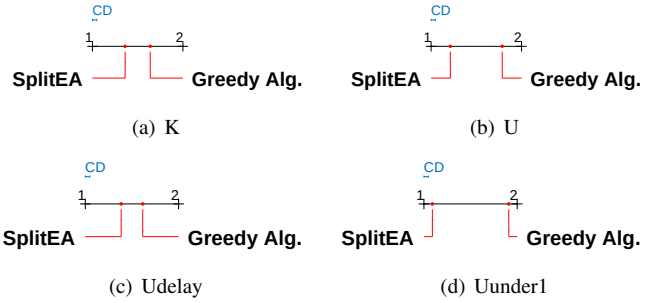
In addition, SplitEA gets significant better metrics value regarding all four metrics on $Archive-Songliao$. On other datasets, SplitEA gets significant better metrics regarding all metrics except for $Udelay$. It is intuitive to get the reason that SplitEA tends to cluster more points together due to the less

TABLE III
COMPARISON RESULTS OF SPLITEA AND THE GREEDY ALGORITHM ON REAL-WORLD DATASETS WITH THE PREDICTION.

Datasets	Milan		Archive-Milan	
Algorithms	GreedyAlg	SplitEA	GreedyAlg	SplitEA
K	81.4481	52.4222	14.9969	13.0181
U	0.8389	0.7673	0.7811	0.7540
Udelay	0.0003	0.0089	0.0013	0.0043
Uunder1	0.8386	0.7585	0.7799	0.7497
f	1.6534	1.2915	0.9311	0.8842

Datasets	Archive-Songliao		Archive-Random	
Algorithms	GreedyAlg	SplitEA	GreedyAlg	SplitEA
K	81.4481	52.4222	14.9969	13.0181
U	0.8389	0.7673	0.7811	0.7540
Udelay	0.0003	0.0089	0.0013	0.0043
Uunder1	0.8386	0.7585	0.7799	0.7497
f	1.6534	1.2915	0.9311	0.8842

The values in this table are the mean value of the metrics under 30 independent runs. Friedman and Nemenyi statistical tests [46] with the significance level 0.05 are used to indicate the statistical significance between compared algorithms. The significantly better values obtained by the algorithm are highlighted in red color.



(a) K (b) U

(c) Udelay (d) Uunder1

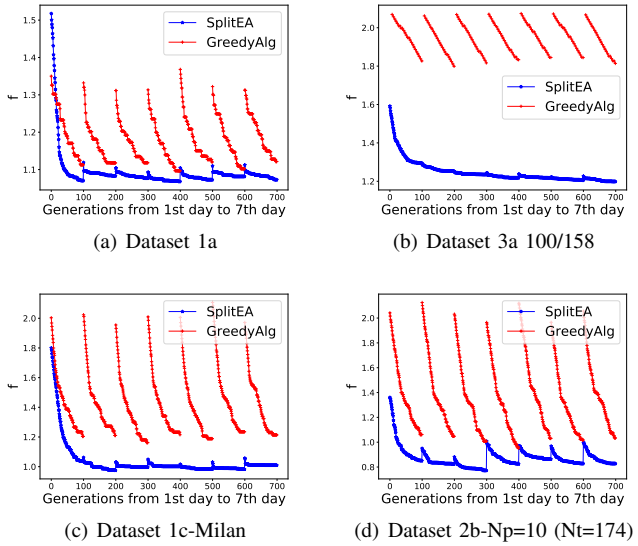
Fig. 6. Nemenyi post-tests results among four metrics by SplitEA and the greedy algorithm on all artificial datasets.

required number of clusters for fixed number of points and this would inevitably increase the delay in the network.

c) Comparison results of SplitEA and greedy algorithm on artificial datasets: In order to test the ability of the SplitEA in searching for good solutions over the greedy algorithm on datasets with more properties beyond those of the existing datasets, like different types of location information and traffic patterns, six artificial datasets with different points distribution and different traffic pattern have been generated to do the verification. The property of those datasets are described in V-A1.

To show the significant superiority of our proposed SplitEA over the greedy algorithm on artificial datasets in general, Friedman and Nemenyi statistical tests [46] are adopted across all artificial datasets regarding the four metrics (HV, GD and MS). The metric value of each independent run that each algorithm gets on one dataset at each day of seven days is regarded as an observation of the test. Therefore, there are 1680 (8 artificial datasets, 7 days and 30 independent run runs) observations for each algorithm in the Friedman and Nemenyi tests.

Fig. 6 presents the Nemenyi post-tests results among four metrics by SplitEA and the greedy algorithm. Friedman test detects significant differences in average accuracy for all metrics with a p-value of 3.1628E-30, 2.5955E-122, 2.9012E-21 and 4.4523E-247, respectively. It is clear from this figure that our proposed SplitEA outperforms the greedy algorithms



(a) Dataset 1a (b) Dataset 3a 100/158

(c) Dataset 1c-Milan (d) Dataset 2b-Np=10 (Nt=174)

Fig. 7. Curve of the fitness value obtained by SplitEA and the greedy algorithm on four artificial datasets across the whole evolution process.

in general regarding all metrics, which prove the effectiveness of our proposed SplitEA over the greedy algorithms. Only the test results are shown here to save space. More details can be found in Table 1 of the Supplementary File.

d) Fitness value curve of SplitEA and greedy algorithm on selected artificial datasets: In order to have a better understanding of the behavior of SplitEA against the greedy algorithm in the optimization process over time, the curve of the fitness value obtained by SplitEA and greedy algorithm on four representative artificial datasets (1a, 3a 100/158, 1c-Milan and 2b-Np=10 (Nt=174)) are drawn to be shown in Fig. 7. In other words, we need to check if the better performance of the proposed SplitEA than that of the greedy algorithm is consistent over time.

It is clear from Fig. 7 that SplitEA gets better fitness values at all days of dataset $3a100/158$ and dataset $2b - Np = 10(Nt = 174)$. In addition, SplitEA only gets worse fitness value at the beginning of the first day of dataset $1a$ and dataset $1c - Milan$. Therefore, it has been checked that there are not some periods of time for which SplitEA is actually much worse than the greedy algorithm, despite being better overall. To be more specific, SplitEA is able to get better fitness values than the greedy algorithm in the optimization process over time.

e) The effect of different parameter settings on the performance of SplitEA and greedy algorithm: In order to check the influence of different parameter settings on the performance of SplitEA and greedy algorithm, two methods are tested on the Milan Dataset which sets different values for two problem-related parameters w and τ. The comparison results of SplitEA and greedy algorithm on dataset Milan under different parameter settings are presented in Table 2 of the Supplementary File. It is clear from this table that under different settings of w, SplitEA significantly performs better than the greedy algorithm on all metrics and the fitness value. In addition, SplitEA gets significantly better results the greedy algorithm on all metrics under all settings of τ except for the setting of $\tau = 1500$ on the metric $Udelay$. The reason might

be that if τ is too large, SplitEA tends to cluster many points together, which inevitably causes more delay than the greedy algorithm causes.

2) Analyses of Random Cluster Splitting in SplitEA: This section tries to answer the second sub-research question of the second question: What is the influence of different algorithms design choices on its performance? Considering that process of random cluster splitting is an importance mechanism in SplitEA and there are two intuitive alternatives, this section analyzes the effect of the process of random cluster splitting on SplitEA. Therefore, we replace the process of random cluster splitting with two different processes to create two variants of SplitEA, named as RandEA and CopyEA, and compare three of them on several real-world and artificial datasets. The details of the replacement in RandEA and CopyEA can be found in Section V-A3.

Firstly, SplitEA and two variants are tested on the real-world datasets to verify that whether the proposed SplitEA gets best optimization results than two variants on real-world datasets. Then, three algorithms are tested on predicted traffic datasets to see whether the prediction error affects the performance of SplitEA. Lastly, three compared algorithms are tested on artificial datasets with more properties beyond the existing datasets. Note for those experiments without using the prediction model, the Step 6 in Algorithm 1 is deleted and in Step 7, the fitness value of solutions is calculated on the real traffic data.

TABLE IV
COMPARISON RESULTS OF THREE EAS ON REAL-WORLD DATASETS.

Datasets	Milan			Songliao		
Algorithms	RandEA	CopyEA	SplitEA	RandEA	CopyEA	SplitEA
K	55.5481	**53.3444**	52.3019	53.4524	**53.8810**	54.3619
U	0.7766	**0.7671**	0.7644	0.8187	**0.8109**	0.8042
Udelay	0.0067	0.0069	0.0076	0.0351	**0.0274**	0.0234
Uunder1	0.7699	**0.7602**	0.7568	0.7836	**0.7778**	0.7760
f	1.3321	**1.3005**	1.2874	1.3532	**1.3497**	1.3478

Datasets	C2TM-sub1			Archive-Milan		
Algorithms	RandEA	CopyEA	SplitEA	RandEA	CopyEA	SplitEA
K	**54.3792**	52.0458	51.7708	13.5239	**13.1157**	13.0352
U	**0.9958**	0.9956	0.9956	0.7587	**0.7548**	0.7536
Udelay	1.70E-07	2.84E-07	2.97E-07	0.0020	**0.0038**	**0.0039**
Uunder1	**0.9958**	0.9956	0.9956	0.7567	**0.7510**	0.7497
f	**1.5396**	1.5160	1.5133	0.8939	**0.8859**	0.8840

Datasets	Archive-Songliao			Archive-Random		
Algorithms	RandEA	CopyEA	SplitEA	RandEA	CopyEA	SplitEA
K	13.5308	12.9308	**13.2629**	**12.1497**	12.0057	12.0069
U	**0.7572**	0.7522	0.7538	0.7289	**0.7268**	0.7260
Udelay	0.0011	0.0043	**0.0022**	0.0008	**0.0014**	0.0009
Uunder1	**0.7560**	0.7479	0.7517	0.7280	**0.7254**	0.7250
f	0.8925	0.8815	**0.8865**	0.8504	**0.8469**	0.8460

The values in this table are the mean value of the metrics under 30 independent runs. The best and the second best values obtained by the algorithm are highlighted in red and bold face, respectively. Friedman and Nemenyi statistical tests [46] with the significance level 0.05 are used to indicate the statistical significance between compared algorithms. If results obtained by two or three algorithms are highlighted with the same mark, it meas there is no significant difference between them.

a) Comparison results of SplitEA and two variants on real-world datasets: The comparison results of three algorithms on the real-world dataset are shown in Table IV. It is clear that SplitEA gets significant overall best results on all datasets except for the dataset Archive-Songliao. The reason why CopyEA gets significant better results than SplitEA on the dataset Archive-Songliao is due to the location dataset of Archive-Songliao, as three datasets (Archive-Milan, Archive-Songliao and Archive-Random) have the same traffic dataset. It is possible on this dataset that when the solutions from the problem of the previous day are still good for the problem of the next day. Therefore, when SplitEA splits one cluster into two clusters, it makes the solutions worse.

On all datasets except for Archive-Songliao, SplitEA gets significantly best values on three metrics. Specifically, SplitEA is the best regarding K, U and $Uunder1$ on datasets Milan and $Archive-Milan$. It is easy to understand that when SplitEA gets the smallest number of clusters while resulting more delay. As for the reason why CopyEA and SplitEA perform equally on C2TM, it is due to the traffic dataset, which is so few that the SplitEA could not find better solutions than CopyEA. With regards to the results on dataset Songliao, SplitEA gets the worse K while RandEA gets the best. It might be that when further increasing the resource utilization rate and decreasing the delay, the number of clusters inevitably increases.

TABLE V
COMPARISON RESULTS OF THREE EAS ON REAL-WORLD DATASETS WITH PREDICTION.

Datasets	Milan			Archive-Milan		
Algorithms	RandEA	CopyEA	SplitEA	RandEA	CopyEA	SplitEA
K	**13.3635**	13.6233	13.0340	13.4692	**13.0962**	13.0181
U	**0.7591**	0.7611	0.7547	0.7594	**0.7546**	0.7540
Udelay	**0.0036**	0.0021	0.0044	0.0029	**0.0039**	0.0043
Uunder1	**0.7555**	0.7590	0.7502	0.7566	**0.7507**	0.7497
f	**0.8928**	0.8973	0.8850	0.8941	**0.8855**	0.8842

Datasets	Archive-Songliao			Archive-Random		
Algorithms	RandEA	CopyEA	SplitEA	RandEA	CopyEA	SplitEA
K	13.2068	**12.8242**	12.8106	**12.0794**	12.0019	12.0019
U	0.7608	0.7535	**0.7553**	0.7284	**0.7268**	0.7267
Udelay	0.0060	**0.0061**	0.0072	**0.0014**	**0.0014**	0.0013
Uunder1	0.7548	0.7473	**0.7480**	0.7270	**0.7254**	0.7253
f	0.8929	0.8817	**0.8834**	0.8492	**0.8468**	0.8467

The values in this table are the mean value of the metrics under 30 independent runs. The best and the second best values obtained by the algorithm are highlighted in red and bold face, respectively. Friedman and Nemenyi statistical tests [46] with the significance level 0.05 are used to indicate the statistical significance between compared algorithms. If results obtained by two or three algorithms are highlighted with the same mark, it meas there is no significant difference between them.

b) Comparison results of SplitEA and two variants on datasets with prediction: The comparison results of three algorithms on the real-world datasets with prediction are shown in Table V. It is clear that SplitEA gets the best fitness values on all four dataset except the Archive-Songliao, which is similar to those without prediction. The reason might be similar, due to the location dataset of Archive-Songliao. In addition, SplitEA significantly performs best on datasets Milan and Archive-Milan regarding all four metrics except for the $Udelay$.

The only difference between their performance on Milan with prediction and without prediction is that RandEA performs better than CopyEA on Milan with prediction. The performance of three algorithms on Archive-Random is similar to that without prediction. Therefore, it can be concluded that the prediction error does not have much influence on the performance of SplitEA.

c) Comparison results of SplitEA and two variants on artificial datasets: In order to test the performance of the SplitEA in searching for good solutions over RandEA and CopyEA on different scenarios with different points distribution and different traffic pattern, several artificial dataset have been generated to do the verification. The property of those datasets are described in V-A1.

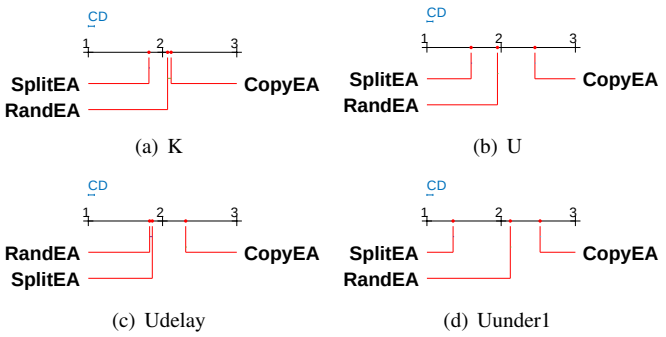


Fig. 8. Nemenyi post-tests results among four metrics by SplitEA, RandEA and CopyEA on all artificial datasets.

To show the significant superiority of our proposed SplitEA over the RandEA and CopyEA on artificial datasets in general, Friedman and Nemenyi statistical tests [46] are adopted across all artificial datasets regarding the four metrics (HV, GD and MS). The metric value of each independent run that each algorithm gets on one dataset at each day of seven days is regarded as an observation of the test. Therefore, there are 1680 (8 artificial datasets, 7 days and 30 independent run runs) observations for each algorithm in the Friedman and Nemenyi tests.

Fig. 8 presents the Nemenyi post-tests results among four metrics by SplitEA, RandEA and CopyEA. Friedman test detects significant differences in average accuracy for all metrics with a p-value of 3.7973E-25, 1.5423E-159, 4.1767E-63 and 9.8020e-303, respectively. It is clear from this figure that our proposed SplitEA outperforms RandEA and CopyEA in general regarding three metrics except for Udelay, which prove the effectiveness of our proposed SplitEA over RandEA and CopyEA Only the test results are shown here to save space. More details can be found in Table 3 of the Supplementary File.

d) Fitness value curve of SplitEA and two variants on artificial datasets: In order to have a better understanding of the proposed SplitEA and its two variants in the evolution process, the fitness value curve of three EAs on all artificial datasets in a randomly selected run is drawn and presented in Figs. 2 and 3 of the Supplementary File. As three EAs all initialize the population randomly at the first day, the curve of all days except for the first day is presented.

It is clear from those two figures that during the whole optimization process, the fitness vale of SplitEA is always better than that of RandEA and CopyEA on four datasets 3as and 1cs. As for the comparison results on datasets 1a and 2a, SplitEA is only worse than CopyEA on first day of all days. The reason might be that the spitting of clusters at the day cannot provide enough diversity as the RandEA gets the best on the first day. This case of insufficient diversity introduced by SplitEA becomes worse on two datasets 3as on which SplitEA performs best at initial generations while RandEA performs best at later generation of most days among all days.

e) The effect of different parameter settings on the performance of SplitEA and two variants: In order to check the influence of different parameter settings on the performance of SplitEA and two variants, three algorithms are tested on the Milan Dataset which sets different values for two problem-related parameters w and τ and three EA-related parameters $prob$, G and $popsize$. The comparison results of SplitEA and two variants on dataset Milan under different parameter settings are presented in Table 4 of the Supplementary File. It is clear from this table that under different settings of w, SplitEA significantly performs better than the greedy algorithm on all metrics and the fitness value. It is clear from this table that under different settings of all parameters, SplitEA gets best results, which shows that the proposed SplitEA is not sensitive with parameter setting.

In the proposed random cluster splitting, the superiority of the solution is not affected so much, even though the diversity is introduced. The reason is that in the random cluster splitting, only a random one cluster in a solution is selected and splitted into two clusters with other clusters unchanged, which maintains the superiority of the solution regarding the summation of the resource utilization rate and the delay on other non-splitted clusters. Moreover, it can be seen from Fig. 2 of the Supplementary File that the proposed SplitEA has better fitness value after each day than CopyEA (copy all solutions from the previous iterations without any modification) in most cases through comparing the curves of them (i.e., red and blue curves). This further verifies that the superiority of the solution in the random cluster splitting is not affected so much from the perspective of experimental results.

VI. Conclusion and Future Work

In this paper, we firstly mathematically revealed limitations of the existing problem formulation for the computing resource allocation problem in O-RAN. To overcome these limitations, we proposed a novel formulation for a better representation of the problem.We also proposed a novel evolutionary algorithm (called SplitEA), enabling better solutions to be found than the existing greedy algorithm for the problem. In particular, three novel operators were designed in SplitEA tailored for the new problem formulation. They are population initialization, mutation operator and knowledge transfer-based random cluster splitting. Experimental studies demonstrated that our proposed SplitEA significantly outperformed the greedy algorithm under all parameter settings on all real-world datasets and most artificial datasets. The comparison between SplitEA and two variants has shown that our proposed transfer technique can achieve better fitness over time than transferring all previous solutions and optimization from scratch.

In the future, this work can be improved in several directions. Firstly, different size of computing resource can be considered in different clusters. Secondly, the communication delay between the BBU pool and base stations could be also taken into consideration. Lastly, more artificial intelligence techniques like prediction methods or optimization approaches can be leveraged to help the automation of O-RAN.

Acknowledgment

This work has received funding from the European Unions Horizon 2020 research and innovation programme under grant

agreement number 766186. The work was also supported by the Program for Guangdong Introducing Innovative and Enterpreneurial Teams (Grant No. 2017ZT07X386), Shenzhen Peacock Plan (Grant No. KQTD2016112514355531) and the Program for University Key Laboratory of Guangdong Province (Grant No. 2017KSYS008).

REFERENCES

[1] S. K. Singh, R. Singh, B. Kumbhani, The evolution of radio access network towards open-ran: challenges and opportunities, in: 2020 IEEE Wireless Communications and Networking Conference Workshops (WCNCW), IEEE, 2020, pp. 1–6.

[2] Y. Liu, X. Wang, G. Boudreau, A. B. Sediq, H. Abou-zeid, Deep learning based hotspot prediction and beam management for adaptive virtual small cell in 5g networks, IEEE Transactions on Emerging Topics in Computational Intelligence 4 (1) (2020) 83–94.

[3] L. Gavrilovska, V. Rakovic, D. Denkovski, From cloud ran to open ran., Wirel. Pers. Commun. 113 (3) (2020) 1523–1539.

[4] L. Chen, D. Yang, D. Zhang, C. Wang, J. Li, T.-M.-T. Nguyen, Deep mobile traffic forecast and complementary base station clustering for c-ran optimization, Journal of Network and Computer Applications 121 (2018) 59–69.

[5] B. I. Simmons, C. Hoeppke, W. J. Sutherland, Beware greedy algorithms, Journal of Animal Ecology 88 (5) (2019) 804–807.

[6] W. Gao, T. Friedrich, F. Neumann, C. Hercher, Randomized greedy algorithms for covering problems, in: Proceedings of the Genetic and Evolutionary Computation Conference, 2018, pp. 309–315.

[7] A. E. Eiben, J. E. Smith, et al., Introduction to evolutionary computing, Vol. 53, Springer, 2003.

[8] A. Gupta, Y.-S. Ong, L. Feng, Insights on transfer optimization: Because experience is the best teacher, IEEE Transactions on Emerging Topics in Computational Intelligence 2 (1) (2017) 51–64.

[9] K. C. Tan, L. Feng, M. Jiang, Evolutionary transfer optimization-a new frontier in evolutionary computation research, IEEE Computational Intelligence Magazine 16 (1) (2021) 22–33.

[10] S. P. Jayakumar, A. Conte, Framework: Clustering-driven approach for base station parameter optimization and automation (ceda-batop), in: 2024 IEEE 21st Consumer Communications & Networking Conference (CCNC), IEEE, 2024, pp. 1026–1029.

[11] R. T. Rodoshi, T. Kim, W. Choi, Deep reinforcement learning based dynamic resource allocation in cloud radio access networks, in: 2020 International Conference on Information and Communication Technology Convergence (ICTC), IEEE, 2020, pp. 618–623.

[12] A. Staffolani, V.-A. Darvariu, L. Foschini, M. Girolami, P. Bellavista, M. Musolesi, Prorl: Proactive resource orchestrator for open rans using deep reinforcement learning, IEEE Transactions on Network and Service Management.

[13] S. Tekinay, B. Jabbari, Handover and channel assignment in mobile cellular networks, Communications Magazine IEEE 29 (11) (1991) 42–46.

[14] A. Sang, S.-q. Li, A predictability analysis of network traffic, Computer Networks 39 (4) (2002) 329–345.

[15] G. E. Box, G. M. Jenkins, G. C. Reinsel, G. M. Ljung, Time series analysis: forecasting and control, John Wiley & Sons, 2015.

[16] A. N. Jahromi, S. Hashemi, A. Dehghantanha, R. M. Parizi, K.-K. R. Choo, An enhanced stacked lstm method with no random initialization for malware threat hunting in safety and time-critical systems, IEEE Transactions on Emerging Topics in Computational Intelligence 4 (5) (2020) 630–640.

[17] G. Dorffner, Neural networks for time series processing, in: Neural Network World, Citeseer, 1996.

[18] G. P. Zhang, M. Qi, Neural network forecasting for seasonal and trend time series, European Journal of Operational Research 160 (2) (2005) 501–514.

[19] E. M. Azoff, Neural network time series forecasting of financial markets, John Wiley & Sons, Inc., 1994.

[20] H. Moens, F. De Turck, Vnf-p: A model for efficient placement of virtualized network functions, in: 10th International Conference on Network and Service Management (CNSM) and Workshop, IEEE, 2014, pp. 418–423.

[21] A. Gupta, M. F. Habib, P. Chowdhury, M. Tornatore, B. Mukherjee, On service chaining using virtual network functions in network-enabled cloud systems, in: 2015 IEEE International Conference on Advanced Networks and Telecommuncations Systems (ANTS), IEEE, 2015, pp. 1–3.

[22] Z.-J. Lee, C.-Y. Lee, A hybrid search algorithm with heuristics for resource allocation problem, Information Sciences 173 (1-3) (2005) 155–167.

[23] P. Samimi, Y. Teimouri, M. Mukhtar, A combinatorial double auction resource allocation model in cloud computing, Information Sciences 357 (2016) 201–216.

[24] A. Mebrek, A. Yassine, Intelligent resource allocation and task offloading model for iot applications in fog networks: A game-theoretic approach, IEEE Transactions on Emerging Topics in Computational Intelligence PP (99) (2021) 1–15.

[25] X. Liu, M. Jia, Intelligent spectrum resource allocation based on joint optimization in heterogeneous cognitive radio, IEEE Transactions on Emerging Topics in Computational Intelligence 4 (1) (2020) 5–12.

[26] R. Riggio, T. Rasheed, R. Narayanan, Virtual network functions orchestration in enterprise wlans, in: 2015 IFIP/IEEE International Symposium on Integrated Network Management (IM), IEEE, 2015, pp. 1220–1225.

[27] R. Mijumbi, J. Serrat, J.-L. Gorricho, N. Bouten, F. De Turck, S. Davy, Design and evaluation of algorithms for mapping and scheduling of virtual network functions, in: Proceedings of the 2015 1st IEEE Conference on Network Softwarization (NetSoft), IEEE, 2015, pp. 1–9.

[28] A. A. Khan, M. Abolhasan, W. Ni, J. Lipman, A. Jamalipour, A hybrid-fuzzy logic guided genetic algorithm (h-flga) approach for resource optimization in 5g vanets, IEEE Transactions on Vehicular Technology 68 (7) (2019) 6964–6974.

[29] A. Perveen, R. Abozariba, M. Patwary, A. Aneiba, Dynamic traffic forecasting and fuzzy-based optimized admission control in federated 5g-open ran networks, Neural Computing and Applications (2021) 1–19.

[30] E. Robinson, P. McBurney, X. Yao, Market niching in multi-attribute computational resource allocation systems, in: Proceedings of the 13th International Conference on Electronic Commerce, 2011, pp. 1–10.

[31] P. R. Lewis, P. Marrow, X. Yao, Evolutionary market agents and heterogeneous service providers: Achieving desired resource allocations, in: 2009 IEEE Congress on Evolutionary Computation, IEEE, 2009, pp. 904–910.

[32] P. R. Lewis, P. Marrow, X. Yao, Resource allocation in decentralised computational systems: an evolutionary market-based approach, Autonomous Agents and Multi-Agent Systems 21 (2) (2010) 143–171.

[33] Z. Wang, K. Tang, X. Yao, Multi-objective approaches to optimal testing resource allocation in modular software systems, IEEE Transactions on Reliability 59 (3) (2010) 563–575.

[34] P. R. Lewis, P. Marrow, X. Yao, Evolutionary market agents for resource allocation in decentralised systems, in: International Conference on Parallel Problem Solving from Nature, Springer, 2008, pp. 1071–1080.

[35] S. Salcedo-Sanz, J. A. Portilla-Figueras, E. G. Ortiz-García, A. M. Pérez-Bellido, C. Thraves, A. Fernández-Anta, X. Yao, Optimal switch location in mobile communication networks using hybrid genetic algorithms, Applied Soft Computing 8 (4) (2008) 1486–1497.

[36] S. Salcedo-Sanz, X. Yao, Assignment of cells to switches in a cellular mobile network using a hybrid hopfield network-genetic algorithm approach, Applied Soft Computing 8 (1) (2008) 216–224.

[37] H. Huang, Y. Xu, Y. Xiang, Z. Hao, Correlation-based dynamic allocation scheme of fitness evaluations for constrained evolutionary optimization, IEEE Transactions on Evolutionary Computation.

[38] Y. Wang, J.-P. Li, X. Xue, B.-C. Wang, Utilizing the correlation between constraints and objective function for constrained evolutionary optimization, IEEE Transactions on Evolutionary Computation 24 (1) (2020) 29–43.

[39] B.-C. Wang, H.-X. Li, Q. Zhang, Y. Wang, Decomposition-based multiobjective optimization for constrained evolutionary optimization, IEEE Transactions on Systems, Man, and Cybernetics: Systems 51 (1) (2021) 574–587.

[40] S. Salcedo-Sanz, X. Yao, A hybrid hopfield network-genetic algorithm approach for the terminal assignment problem, IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) 34 (6) (2004) 2343–2353.

[41] A. Mousa, M. El-Shorbagy, M. Farag, K-means-clustering based evolutionary algorithm for multi-objective resource allocation problems, Appl. Math. Inf. Sci 11 (6) (2017) 1681–1692.

[42] I. Rahimi, A. H. Gandomi, F. Chen, E. Mezura-Montes, A review on constraint handling techniques for population-based algorithms: from

single-objective to multi-objective optimization, Archives of Computational Methods in Engineering 30 (3) (2023) 2181–2209.
[43] G. Barlacchi, M. De Nadai, R. Larcher, A. Casella, C. Chitic, G. Torrisi, F. Antonelli, A. Vespignani, A. Pentland, B. Lepri, A multi-source dataset of urban life in the city of milan and the province of trentino, Scientific Data 2 (1) (2015) 1–15.
[44] Z. Du, Y. Yang, Z. Ertem, C. Gao, L. Huang, Q. Huang, Y. Bai, Inter-urban mobility via cellular position tracking in the southeast songliao basin, northeast china, Scientific Data 6 (1) (2019) 1–6.
[45] X. Chen, Y. Jin, S. Qiang, W. Hu, K. Jiang, Analyzing and modeling spatio-temporal dependence of cellular traffic at city scale, in: 2015 IEEE International Conference on Communications (ICC), IEEE, 2015, pp. 3585–3591.
[46] J. Demšar, Statistical comparisons of classifiers over multiple data sets, Journal of Machine Learning Research 7 (Jan) (2006) 1–30.