An EDA-based Genetic Algorithm for EV Charging Scheduling under Surge Demand

Tianyang Li[1,2] Xiaolong Li[3*], Ting He[4] and Yufeng Zhang[5]
[1]School of Computer Science, Northeast Electric Power University, Jilin, China.
[2]Jiangxi New Energy Technology Institute, Nanchang, China.
[3]Harbin Branch, Bank of Inner Mongolia Co.,Ltd, Harbin, China.
[4]College of Computer Science and Technology, Huaqiao University, Xiamen, China.
[5]Birmingham Business School, University of Birmingham, Birmingham, UK.

2022 IEEE International Conference on Services Computing (SCC) | 978-1-6654-8146-5/22/$31.00 ©2022 IEEE | DOI: 10.1109/SCC55611.2022.00042

***Abstract*—With continually increased Electric Vehicles (EVs), the EVs Charging Scheduling is of great importance to managing multiple charging demands for maximizing user satisfactions and minimizing adverse influences on the grid. However, it is challenging to effectively manage EVs charging schedules when a large number of (on-the-move) EVs are planning to charge at the same time. With this concern, we focus on Charging Station (CS)-selection decision making by the global aggregator that is taken as controller to implement charging management for EVs and CSs. An Estimation of Distribution Algorithm (EDA)-based genetic algorithm is proposed to find constrained charging scheduling plans to maximize the charging efficiency, which may improve user satisfaction and alleviate impacts on the grid. Experimental results under a city scenario with realistic EVs and CSs show the advantage of our proposal, in terms of minimized queuing time and maximized charging performance at both the EV and CS sides. The code and data are available at https://github.com/EV-charging-scheduling-algorithm.**

***Index Terms*—EV charging, charging scheduling, Estimation of Distribution Algorithm, genetic algorithm, surge demand**

I. Introduction

The number of electric vehicles (EVs) increases sharply due to its low-cost and environmentally friendly characteristics. One of the major roadblocks to promote the EVs penetration is the lack of convenient public charging infrastructure, which may result in uncoordinated charging of a large number of EVs with adverse impact on the grid operation and traffic management [1]. Therefore, EVs charging scheduling is of great importance to managing multiple charging demands for maximizing user satisfaction and minimizing adverse influences [2].

Given the likely future high EVs penetration rate and low speed of infrastructure construction, the random increase in charging demand is becoming a common phenomenon. Typically, when the performance of batteries declined in the extreme winter weather, there would be a surge demand in charging stations of certain areas. In this case, uncontrolled EVs charging would cause a significant voltage deviation and transformer aging, as well as long queuing times and traffic congestions [3]. Proper real-time EVs charging schedules are needed to improve the charging efficiency by putting the coordination of charging activities among EVs and charging stations. However, it is challenging to effectively manage EVs charging schedules when a large number of (on-the-move) EVs are planning to charge at the same time [4].

Recently, majority of previous works investigate charging scheduling problems that utilize the global aggregator (GA) or other third party who is interested in EVs charging managements [5]. By monitoring conditions of Charging Stations (CSs), the GA as controller implements the charging scheduling whenever it receives requests from on-the-move EVs. Under this circumstance, with strategies that reduce peak load consumptions and fill valleys in electric load profiles, a few studies applied the deep reinforcement learning algorithms to predict and schedule the multiple charging demands among different CSs based on users' preferences in terms of start times and charging durations [6] [7] [8] [9]. Obviously, aforementioned algorithms can overcome slow convergence and excessive randomness for effectively handling periodic demands, but still requires a tremendous amount of hand labour to adjust mode and lots of training time for coping with the random surge demands properly. Some works employed metaheuristic algorithms to appropriately deal with real-time large-scale uncertain demands in EV charging scheduling problems [10] [11]. In such studies, genetic algorithm [12] [13] [14], particle swarm optimization [15] [16] [17] [18] and artificial bee colony [19] were used to find best charging plans for both customers and CSs side by minimizing the queuing time, charging cost or drivers' trip duration. Although, these simple implement algorithms can effectively solve scheduling problems, when a large-scale requirements increase in a short time, they would be easy to fail to get the better charging plans with premature and falling into local optimums in a large searching space.

To address this issue and the surge in charging demand, we propose an Estimation of Distribution Algorithm (EDA)-based genetic algorithm, which aims to find constrained charging scheduling plans to maximize charging efficiencies and alleviate traffic congestions around CSs. A lightweight probability model for sampling local CS-selection decisions and an EDA searching stage to search global CS-selection decisions are presented to improve the genetic algorithm. Experimental

*The corresponding author. Email: long1378568805@163.com.

2474-2473/22/$31.00 ©2022 IEEE
DOI 10.1109/SCC55611.2022.00042

results under a city scenario with realistic EVs and CSs show advantages of our proposal, in the term of minimized queuing time and maximized charging performance at both the EV and CS sides.

The paper is structured as follows. "Problem Definition" section introduces the entities and assumption of the EV charging scheduling problem, as well as its general mathematical model. "EDA-based Genetic Algorithm" section elaborates the chromosome, EDA sampling mechanism and EDA searching stage of the proposed algorithm. Then, in "Case Study" section, cases studies based on a real-world data set are carried out to validate the effectiveness of proposed algorithm. Finally, "Conclusions" section concludes the paper and suggests future research directions.

II. Problem Definition

A. Definition of Entities

(i) *EV:* Each EV can be expressed as a triple, $EV =< l, tdl, ect >$, where l presents the location of EV, tdl is the travel distance that the EV can drive on the remaining battery capacity, and ect is the expect charging duration. EVs have a status of charge (SOC) threshold. If current energy is below the SOC threshold, EVs start to negotiate with GA to find an appropriate CS for charging. Further to this, Evs also report their charging information to GA, including "how much energy dose it left", "what time it will arrive at the decided CS" and "how long it will take to charge at that CS";

(ii) *CS:* Each CS is also a triple, $CS =< l, nd, vl >$, which locates at a certain location and has multiple charging slots/devices to charge EVs in parallel. l is the location, nd is the number of its charging devices, as well as vl expresses the maximum voltage constraints. The condition information of each CS (number of EVs are changing at the CS and their charging time) is monitored by the GA in real time;

(iii) *CD:* A quadruple is used to present a charging device (CD) $CD =< mt, pr, il_n, eavgct >$, where mt is the maximum charging time for each EV, pr is the price range for EV charging, il_n is the number of EVs in the charging queue, and $eavgct$ is the total expected charging time of all inline EVs. CDs are the unit of EVs scheduling;

(iv) *GA:* It is a centralized third-party entity to manage EVs charging and can be denoted by $GA =< ss, so, sp >$, where sm is the scheduling strategy, so is the scheduling objectives of scheduling problems with different scheduling strategies. sp is the scheduling plans. Usually, GA may use different ss and so under different circumstances, which lead to different sp. Here, the target of solving scheduling problems is to get proper scheduling plans for GA.

B. Assumption

In this paper, we focus on the charging scheduling problem of plug-in EV in a city scenario where CSs are geographically deployed, GA globally manages the charging plans for all EVs in the network.

Without loss of generality, we assume that EVs can easily communicate with GA for request/reply charging services and GA may adopt different scheduling strategies for different situations. Generally, GA will provide different alternatives for each customer to select one they personally prefer.

The underlying EV charging scheduling mode (concerning when/whether to charge EVs) at the CS side, are based on the first come first serve (FCFS) order (the EV with an earlier arrival time will be scheduled with a higher charging priority) or greedy scheduling. The greedy scheduling means that when large amounts of EVs are arriving at exactly the same time or in a same waiting queue, in order to decline the total waiting time and avoid the potential congestion risk, EVs with the short charging duration and early arriving time have a higher charging priority.

We also assume that surges in charging demand are usually unpredictable. Under some circumstance, demands increase sharply in a short term. For instance, there will be a surge in charging demand in some areas under the extreme cold weather conditions due to the nature of batteries. GA may prefer the greedy scheduling in this situation.

C. Problem Formulation

The aim of this work is to find a scheduling plan (solve the scheduling problem) in the initial schedule stage for GA such to maximize the service efficiency on a surge demand while guaranteeing customers' requirements. Improving the service efficiency is convenient whenever CSs have to deal with a great deal of uncertain requirement conflicts.

The problem is formulated in a discrete time setting, where N denotes a large backlog of charging requirements at the sampling time (a time slot) t, I is the number of CSs managed by GA and J is the number of all available CDs. Let $v \in N$, $i \in I$ and $j \in J$ denote a vehicle, CS and CD that involved in the charging process.

For the given vehicle v and CS i, we denote by l_v and l_i the locations of v and i. Then tt_{vi} that denotes the travel time of v to i, which is defined as follow:

$$tt_{vi} = \begin{cases} \frac{d(l_v,l_i)}{(1-\alpha_{vi})*v} & if \quad d(l_v,l_i) < tdl_v \\ 0 & if \quad d(l_v,l_i) > tdl_v \end{cases} \tag{1}$$

where $d(l_v, l_i)$ is the distance between locations of l_v and l_i; $\alpha_{vi} \in$ [0,1] is the traffic congestion coefficient. When the traffic is bad, the coefficient is bigger. Otherwise, the coefficient is 0.

Once the EV is assigned to a CS i, it will use a CD j for charging. If CD j is being used or will be used by users that are in the front of vehicle v within the charging queue, then the expected queuing duration of vehicle v with j can be calculated by:

$$qt_{vj} = \begin{cases} wt_j - tt_{vi} & if \quad wt_j > tt_{vi} \\ 0 & if \quad wt_j < tt_{vi} \end{cases} \tag{2}$$

where wt_j is the duration that CD j may be occupied before it serves the vehicle v.

If the travel time is longer then the occupied duration of all inline EVs that are in the front of vehicle v, some of the time of the CD j is sitting idle before v arriving. The idle duration of the j with vehicle v can be calculated by:

$$lt_{vj} = \begin{cases} tt_{vi} - wt_j & if \quad wt_j < tt_{vi} \\ 0 & if \quad wt_j > tt_{vi} \end{cases} \tag{3}$$

For each vehicle v, the occupied duration of j can be calculated by:

$$wt_j = \begin{cases} tt_{vi} + expt_v & if \quad wt_j < tt_{vi} \\ wt_j + expt_v & if \quad wt_j > tt_{vi} \\ 0 & if \quad N = 0 \end{cases} \tag{4}$$

where $expt_v$ is the expected charging duration of v.

Figure 1 shows an example for calculating these durations. As shown in the figure, three EVs are assigned to the CD j at the same time. The charging sequence is the serial number of EVs. The occupied duration of j before EV_1 arriving is 10 minutes. Consequently, the idle duration of j with EV_1 will be also 10 minutes. When EV_1 is arriving at the station, it will use the CD j immediately. Then the queuing duration of EV_1 is 0 and the occupied duration from the beginning for EV_2 is 50 minutes. Similarity, we can have these duration for EV_2 and EV_3, which are shown in the figure.

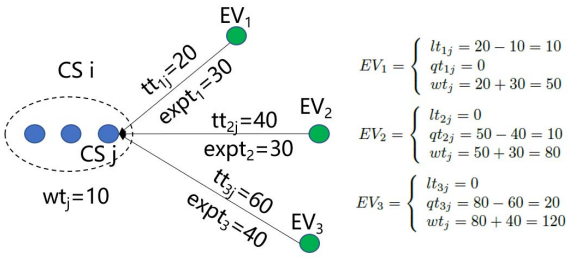

Fig. 1. Example for calculating queuing, idle and occupied durations.

When GA makes CS-selection decisions for all EVs, it should balance both the supply and demand sides, and give consideration to their interests. Typically, the EVs charging scheduling can be formulated as a dynamic multi-objective optimization problem. In different situations, GA will select different sets of objectives to solve the scheduling problem according to different scheduling strategies. Under the surge demand, GA usually use scheduling strategies and objectives that can satisfy all requirements as soon as possible. Therefore, minimizing the queuing and idle durations, as well as maximizing the number of charged EVs per hour are selected as objectives. We further convert the multi-objective optimization problem into a single-objective optimization problem. The objective function can be defined as follows:

$$\min \omega_1 * \frac{\sum_j \sum_v qt_{vj}}{Q_m} + \omega_2 * \frac{\sum_j \sum_v it_{vj}}{It_m} + \omega_3 * (1 - \frac{P_n}{N})$$
$$s.t. \quad \begin{cases} \sum_{v \in i} voli_v <= Volt_i, & (i = 1, \cdots, I) \\ d(l_v, l_i) < tdl_v, & (v = 1, \cdots, N) \\ \sum \omega_n = 1 & (n \in [1, 3]) \end{cases} \tag{5}$$

where $Volt_i$ is the rated voltage of CS i, this means that the voltage of all EVs charging with i at the same time should be less than the rated voltage; $\omega_{n \in [1,3]}$ are weights of objectives; Q_m and It_m are the empirical values that are bigger than $\sum_j \sum_v qt_{vj}$ and $\sum_j \sum_v it_{vj}$.

III. EDA-BASED GENETIC ALGORITHM

Genetic algorithm is one of available methodologies for solving EVs charging scheduling problems. However, with a large searching space, genetic algorithm still owns some disadvantages, like the premature convergence and poor local search capability. In order to overcome these drawbacks and obtain a better solution of EV Charging Scheduling problem under surge demands, we propose an EDA-based genetic algorithm. Its framework is shown in Figure 2.

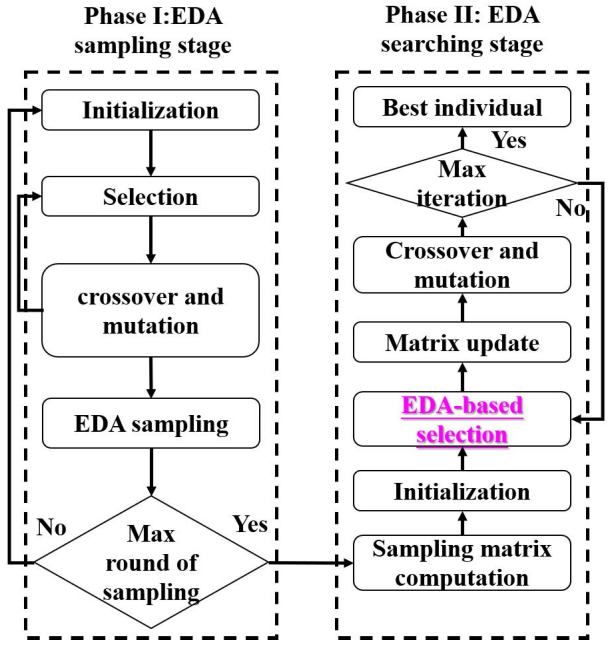

Fig. 2. Framework of EDA-based Genetic Algorithm.

As shown in the figure, EDA-based genetic algorithm has two main stages. One stage is EDA sampling stage which employs the traditional genetic algorithm to obtain sampling data sets. The other stage is the EDA searching stage that implements the improved EDA algorithm to search global optimums.

A. Chromosome and Gene

Two types of chromosomes are designed for case studies with different strategies. One type is suitable for the FCFS scheduling and the other type is appropriate for the greedy scheduling. The infrastructure of two types of chromosomes is same, which is defined as follows:

$$C_{inf} = \begin{bmatrix} CS_i & \dots & CS_i \\ tt_{1i} & \dots & tt_{Ni} \\ EV_1 & \dots & EV_N \end{bmatrix} \tag{6}$$

the gene of chromosomes is presented as a triple $g =< CS_i, tt_{1i}, EV_{v\in N} >$, where $EV_{v\in N}$ is the serial number of vehicle v and CS_i is the serial number of CS i that is assigned to v. The gene is the decision-making information of one vehicle v. The length of chromosomes is the total number of EVs in a sampling time slot t. With the same infrastructure, chromosomes have two main types. The serial number of vehicles in all genes of chromosomes of type one is in order based on the FCFS basis. While the serial number of vehicles in all genes of chromosomes of type two is not, which is used for representing the greedy scheduling. A greedy policy is used to allocate a charging device to a vehicle in the assigned charging station. The vehicle assigned to CS i will be assigned to the queue of CD j that is idle or has the shortest expected waiting time.

B. EDA Sampling Mechanism

As we all know, the Estimation of Distribution Algorithms (EDA) is a new evolution pattern that is able to learn the probability model of solution distributions. In this paper, we establish a probability model of solution distribution by EDA sampling in the selection stage of genetic algorithm. Considering that the probability model based on charging devices and vehicles ($\Omega \in I * J * N$) is too big to sample in a short time, we design a lightweight probability model based on charging stations and vehicles ($\Omega \in I * N$). A probability matrix is used to present the probability model, which is defined as follows:

$$P(i,v) = \begin{bmatrix} p(1,1) & \dots & p(1,v) \\ \vdots & \ddots & \vdots \\ p(i,1) & \dots & p(i,v) \end{bmatrix}_{I\times N} \tag{7}$$

where $p(i,v) = \frac{f_{iv}}{\sum_I f_{iv}}$ is the probability that vehicle v will be assigned to i, and f_{iv} is the frequency that vehicle v is assigned to charging station i.

The calculation of probability matrix is based on samples from each iteration. Different the traditional EDA, we select samples from the population of each iteration of genetic algorithm. Usually, individuals with better fitness will be selected as samples. However, with a large solution space, genetic algorithm is easy to fall into local optimum, so that samples are limited to the local solution space. In order to sample more space for searching the global optimal solution, the fitness-sharing niching technology is subtly used to select samples. This technology amends fitness function based on the fitness value and structural similarity of individuals and can explore more searches in each iteration to maintain the diversity of samples. The amendatory fitness function is defined as follows:

$$F_n(X_i) = \frac{F(X_i)}{\sum_{h\in population} S_{h,i}} \tag{8}$$

here, $F(X_i)$ is the fitness value of individual X_i, and $\sum_{h\in population} S_{h,i}$ represents the crowdedness degree of individual X_i in its niche. $S_{h,i}$ is the sharing value between X_h and X_i and can be calculated as follows:

$$S_{h,i} = \begin{cases} \frac{\sigma - d_{h,i}}{\sigma} & if \quad d_{h,i} < \sigma \\ 0 & if \quad d_{h,i} > \sigma \end{cases} \tag{9}$$

where σ is a defined sharing radius, and $d_{h,i}$ is the number of different genes between X_h and X_i.

With the amendatory fitness function, highly similar individuals are discouraged to be selected as the samples. For example, $X_i = \{x_{12}, x_{23}, x_{37}, x_{42}, x_{51}, x_{66}\}$, $X_j = \{x_{16}, x_{29}, x_{37}, x_{42}, x_{55}, x_{62}\}$ and $X_k = \{x_{16}, x_{29}, x_{37}, x_{43}, x_{55}, x_{62}\}$ are three individuals with fitness values 0.6, 0.8 and 0.9 respectively. $d_{i,j}$=4, $d_{i,k}$=5 and $d_{j,k}$=1; if the σ is set to 2, then $S_{i,j}$=0, $S_{i,k}$=0, $S_{j,k} = 0.5$ and $S_{i,i} = S_{j,j} = S_{k,k} = 1$. By using Eq.(8), $F_n(X_k) = F(X_k)/(S_{i,k} + S_{j,k} + S_{k,k}) = 0.9/1.5 = 0.6$. Similarly, we can have $F_n(X_i)$= 0.6 and $F_n(X_j) = 0.533$. Hence, the two closer individuals X_j and X_k are suppressed to some extent, and individual X_i, which uniquely exploits areas of the search space, is encouraged for selection as samples. In EDA sampling stage, we will select a certain amount of individuals in each iteration of genetic algorithm. For getting more information of solution distributions, we usually implement several rounds of EDA sampling to obtain more local optimal solutions. These selected samples will be used to construct the probability matrix for EDA searching stage to find better solutions. After calculating the probability matrix, the sampling set is cleared out.

C. EDA Searching Stage

In the EDA searching stage, there are several key steps. The first step is the EDA-based selection that is to generate a temporary population. The crossover is used to generate new individuals in the temporary population, which is defined as follows:

$$X_{new} = Cross(X_{random}, X_{EDA}) \tag{10}$$

here, X_{random} and X_{EDA} are two individuals generated in different ways.

X_{random} is generated randomly. While, X_{EDA} is generated with the maximum value of joint distribution probability of all vehicles based on the probability matrix, which is defined as follows:

$$p(X_{EDA}; i, v) = \max \prod_{v\in N} p(i,v) \tag{11}$$

Hence, in the first step, the probability matrix is employed to generate new individuals, which are used to crossover

with individuals that are randomly generated to obtain the temporary population.

The second step is the matrix update step which uses samples selected from the temporary population to update the probability matrix. This means that the EDA sampling mechanism is still working in each iteration of EDA searching stage. The updating function of probabilities is defined as follows:

$$p(i,v)_{new} = p(i,v)_t + \theta * p(i,v)_{t-1} \tag{12}$$

where $\theta \in [0,1]$ is the coefficient for using historical probabilities.

The finally step is the crossover and mutation in which the crossover and mutation of genetic algorithm are further applied to the temporary population to generate the next generation.

Alg 1 shows the pseudocode of EDA-based genetic algorithm.

Alg 1: Pseudocode of EDA-based genetic algorithm

Input: P, T, r, m, s_n, ET, ET_e;
Output: The best individual;
initialization: randomly initialize the initial population, $t = 0, p(i,v) = 0$, $SS = \emptyset$;
fitness evaluation: calculate and amend fitness values;
EDA Sampling stage:
  while $t <= ET$**:**
    selection: select $P * r$ excellent individuals to generate new individuals and add the best of s_n individuals to the sample set SS;
    crossover: cross two individuals with the single-point crossover to generate one new individual;
    mutation: mutate the new individual with partial genes reversal;
    fitness evaluation: calculate/amend fitness values;
    if $t/ET_e == 0$:
      initialization: randomly initialize a population;
      fitness evaluation: calculate/amend the fitness value;
    $t++$;
matrix computation: compute the probability matrix based on SS and set $SS = \emptyset$;
EDA Searching Stage:
  initialization: randomly initialize a population;
  while $t <= T$**:**
    EDA-based selection: generate a temporary generation and add the best of s_n individuals to the sample set SS;
    matrix update: update the probability matrix based on SS and set $SS = \emptyset$;
    crossover and mutation: employ the temporary individual to generate a new generation;
    fitness evaluation: calculate and amend fitness values;
    $t++$;
Return: return the best individual in the last generation.

Here, P is the size of population, T is the number of iterations, r and m are the crossover and mutation rates, S_n is the number of sampling, ET is the total number of iterations for EDA sampling, ET_e is the number of iterations for each round.

IV. Case Studies

A. Yardstick and Dataset

The capability of finding the best solution with limited iterations is used as a main yardstick to evaluate the effectiveness of the proposed algorithm.

Three case studies were implemented by using Jupyter notebook with Python language on a PC: Intel(R) i5-8265U@1.60GHz CPU and 12G Memory. The dataset used in these case studies is collected based on a real-world dataset from a private EVs charging service. More details of the data set are shown in Table I.

There are 34 CSs with different charging prices and each CS has 5 high voltage charging devices. 1000 EVs in different locations are intending to use the charging service in a sampling time slot. But, only 899 EVs can at least arrive the neatest CS with their remaining capacity of batteries. The rest of EVs have to use the batteries exchanging service, which are not considering in case studies.

TABLE I
Details Of Selected Data Set

Data Item	Value
The number of CSs	34
The charge devices of each CS	5
The charging voltage of charge devices	500V-750V
The total number of EV that are intending to use the charge service in a certain time duration	1000
The number of EVs that can arrive the neatest CS with its remaining battery power	899
The number of EVs that use the batteries exchanging service	101

B. Experiment Settings

The first case study is designed to analyze the effectiveness of traditional scheduling methods that GA is not involved. Accordingly, two experiments from the customers' and suppliers' perspectives are implemented separately. The first one runs customer-oriented scheduling method with the first come first service (customer-oriented FCFS) schema. In this scenario, customers will select the nearest CS to charge their EV for minimizing the total cost and consumption of the battery's capacity, which may finally cause a longer average queuing time. From the CSs' (suppliers') perspective, they would like to maximize their revenue by reducing queuing times. Therefore, the other experiment performs a greedy scheduling that the CSs would first serve the nearest customers with a shorter charging duration.

The second case study is designed for evaluating the effectiveness of genetic algorithm for solving the EVs scheduling problem. Three experiments are implemented. The first one performs the genetic algorithm-based FCFS algorithm that customers will randomly select a CS in order for orderly charging their EVs. In the second experiment, the genetic algorithm-based random scheduling algorithm is carried out,

and customers no longer select CSs in order, but still need to orderly use the charging service in each CS. The third experiment runs the genetic algorithm-based greedy scheduling, in which each CS will use the greedy strategic to sort coming customers with their distances and charging durations.

The finally case study is used to assess the proposed algorithm with different rounds of EDA sampling. Accordingly, three experiments are carried out.

Settings of the objective function, genetic algorithm and EDA-based genetic algorithm are shown in Table II.

TABLE II
SETTINGS OF OBJECT FUNCTION AND ALGORITHMS

Variable	Value
Objective function	
a_{vi}	0
$\omega_{n\in[1,3]}$	[0.3, 0.4, 0.3]
Q_m	40000
It_m	3000
Genetic algorithm	
mutation rate m	0.3
crossover rate r	0.7
population size P	200
maximum iteration number T	200,625 and 3000
EDA-based genetic algorithm	
iteration times of sampling ET_e	25
iteration times of sampling ET	75, 500
coefficient θ	1
the number of sampling s_n	1
maximum iteration number T	200, 625 and 3000

It is should be noted that, for improving the sampling efficiency, in each iteration of EDA sampling, only the best individual is added to the sample set.

C. Results and Discussion

Results of all experiments are shown in Table III. As shown in the table, methods in case 1 have poor performances, which implicates that customers oriented methods in surge demand will result in vehicles gathering at charging stations and long queuing times. Moreover, the greedy strategy in charging stations only has a little mitigative effect.

As we can see from the table, results of three methods in case 2 are more better. This means that genetic algorithm-based methods can efficiently solve the scheduling problem. However, with a large search space, these methods are easy to premature or have slow speeds of convergence. For example, algorithms 1 and 3 in case 2 are premature (the convergence times are 52 and 30 respectively) and fall into the local optimal. Algorithm 2 in case 2 have slow speeds of convergence, which cannot explore more areas.

Apparently, results of EDA-based genetic algorithm have shown that it can avoid the "premature" and local optimum. Results of the first experiment in case 3 have shown that the EDA-based genetic algorithm can find a better solution with some extra time in a certain number of iterations, in which the fitness declined to 0.258 (about 19%), the average queuing time of EVs declined to 21.6 minutes (about 14%), the average idle time of EVs declined to 3 minutes (about 28%) and the charged EVs within one hour increased to 769 s(about 5%).

The EDA-based genetic algorithm in the first experiment has three rounds of sampling with 75 iterations. Figure 3 shows the comparison of fitness traces with algorithms in case 2. We can see that the EDA-based genetic algorithm has a good global searching ability and high convergence speed relative to genetic algorithm-based methods. Three troughs in the beginning are sampling processes.

Figure 4 shows more details of the fitness trace of this algorithm. Three inverted triangles in the figure represent sampled local optimums in each round of EDA sampling. Blue circles denote the fitness of EDA searching stage with 125 iterations. They are also taken as the samples in the EDA searching stage to update the probability matrix. As shown in the figure, after 7 iterations, the EDA-based genetic algorithm can get solutions that are better than local optimums. Meanwhile, the trace of blue circles suggests that the EDA sampling mechanism in EDA searching stage makes the algorithm to constantly exploit local optimums for approaching the global optimum, which makes the algorithm obtain a good global searching ability.

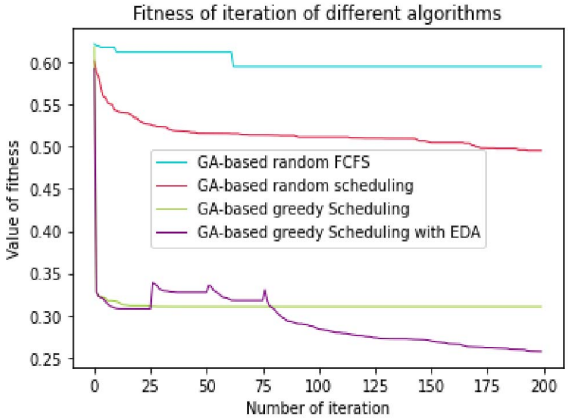

Fig. 3. Fitness trace of different algorithms.

Repeated experiments have implied that local optimums influence the global searching ability. Poor samples will slow the speed of convergence and lower the searching ability. While, there are no clear indications that how the number of sampling influences the searching ability. Therefore, the second experiment in case 3 has 20 rounds of EDA sampling and 125 iterations of EDA searching. Figure 5 shows fitness trace of this experiment. As we can see from the figure, with 20 rounds of EDA sampling, the proposed algorithm only use 3 iterations to find a solution that is better than all local optimums. However, the comparison with figure 4 showed that the algorithm has lower speed of convergence with more rounds of EDA sampling. The reason of this may be that the probability matrix based on more samples from local optimums may lead the algorithm to search local spaces for

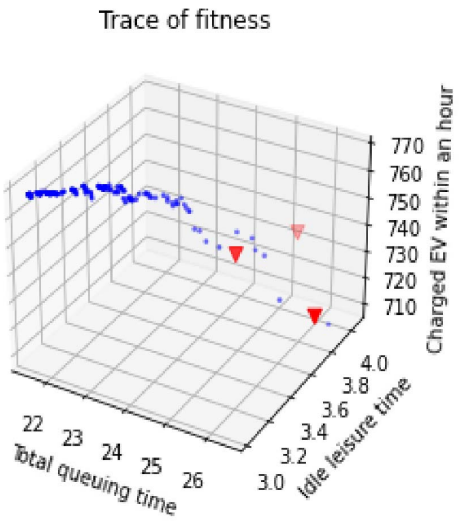

Fig. 4. Fitness trace of EDA-based genetic algorithm.

a while and new experience of samples in the EDA searching stage will be in effect until the algorithm stepping out of local spaces. Lesser rounds of EDA sampling may be an appropriate choice. Using a small coefficient θ can also help to alleviate this "deterioration in convergence".

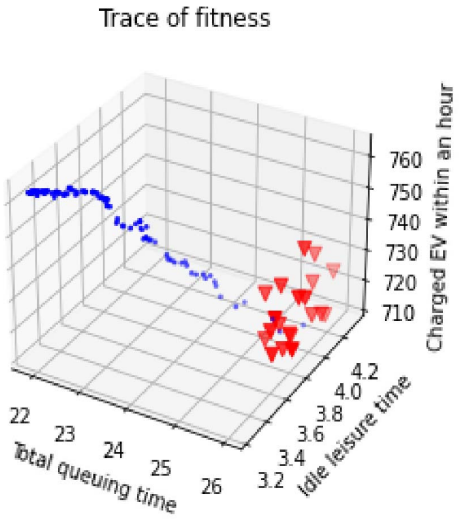

Fig. 5. Fitness trace of EDA-based genetic algorithm with more EDA sampling.

The first and second experiments in case 3 have shown the persistent search ability of the proposed algorithm. For uncovering this characteristic, the last experiment in case 3 is set with 20 rounds rounds of EDA sampling and 2500 iterations of EDA searching. The result of this experiment has also shown that the EDA-based genetic algorithm has the ability to find the global optimum solution with the enough searching time. While, the traditional genetic algorithm does not have this important virtue. The experimental result also indicates that the proposed algorithm has the potential to parallelize for efficiently solving EV charging scheduling problems.

In summery, the EDA sampling mechanism and EDA searching stage can enhance the ability of global convergence of genetic algorithm. The EDA-based genetic algorithm can provide an effective solution for the EVs charging scheduling problem.

V. Conclusions

EV charging scheduling problems under surge demand is becoming more complicated, with ever-increasing EVs and increasing requirements. In this paper, an EDA-based genetic algorithm is proposed in response to the limitation of existing methods to address these new challenges. More specifically, different types of chromosomes are defined, and an EDA sampling mechanism is proposed to construct the matrix of solution distribution. Then, an EDA searching stage of genetic algorithm is presented to find the globally optimal solution. The effectiveness of EDA-based genetic algorithm is validated through Three case studies. In brief, the EDA-based genetic algorithm can help GA employ better scheduling solutions to cope with pressures from surge demands. This algorithm can also provide useful practical guidance and valuable insights for EV scheduling service designers.

In the future, our work will further focus on improving the efficiency and effectiveness of the EDA searching stage, and refine the efficiency of EDA sampling.

Acknowledgment

This work has been supported by the Scientific Research Funds of Northeast Electric Power University (No.BSZT07202107).

References

[1] Long, T., Jia, Q. S., Wang, G., & Yang, Y. (2021). Efficient Real-Time EV Charging Scheduling via Ordinal Optimization. IEEE Transactions on Smart Grid.

[2] Das, R., Wang, Y., Busawon, K., Putrus, G., & Neaimeh, M. (2021). Real-time multi-objective optimisation for electric vehicle charging management. Journal of Cleaner Production, 292, 126066.

[3] Rahman, M. M., Al-Ammar, E. A., Das, H. S., & Ko, W. (2020). Comprehensive impact analysis of electric vehicle charging scheduling on load-duration curve. Computers & Electrical Engineering, 85, 106673.

[4] Jin, J., & Xu, Y. (2020). Optimal policy characterization enhanced actor-critic approach for electric vehicle charging scheduling in a power distribution network. IEEE Transactions on Smart Grid, 12(2), 1416-1428.

[5] Mukherjee, J. C., & Gupta, A. (2016). Distributed charge scheduling of plug-in electric vehicles using inter-aggregator collaboration. IEEE Transactions on Smart Grid, 8(1), 331-341.

[6] Li, H., Wan, Z., & He, H. (2019). Constrained EV charging scheduling based on safe deep reinforcement learning. IEEE Transactions on Smart Grid, 11(3), 2427-2439.

[7] Qian, T., Shao, C., Wang, X., & Shahidehpour, M. (2019). Deep reinforcement learning for EV charging navigation by coordinating smart grid and intelligent transportation system. IEEE Transactions on Smart Grid, 11(2), 1714-1723.

TABLE III
EXPERIMENTAL RESULTS

Case	Algorithm of Experiments	Number of Iteration	Times of EDA sampling	Iteration Number of Convergence	Fitness	Average of queuing time (minute)	Average of idle time (minute)	Number of charged EV within an hour	Time consuming (second)
Case 1	Customer-oriented FCFS	-	-	-	0.755	80.8	3.0	478	0.15
	Supplier-oriented greedy Scheduling	-	-	-	0.681	77.1	1.26	503	0.15
Case 2	Genetic algorithm-based random FCFS	200	-	52	0.605	35.1	12.3	633	411.5
	Genetic algorithm-based random scheduling	200	-	-	0.498	31.8	8.90	659	513
	Genetic algorithm-based greedy Scheduling	200	-	30	0.320	25.1	4.16	729	519
Case 3	EDA-based genetic algorithm	200	3	-	**0.258**	**21.6**	**3.0**	**769**	1195
	EDA-based genetic algorithm	625	20	-	**0.266**	**21.8**	**3.2**	**763**	2664
	EDA-based genetic algorithm	3000	20	-	**0.214**	**19.1**	**2.06**	**784**	21287

[8] Li, S., Hu, W., Cao, D., Dragičević, T., Huang, Q., Chen, Z., & Blaabjerg, F. (2021). Electric vehicle charging management based on deep reinforcement learning. Journal of Modern Power Systems and Clean Energy.

[9] Chang, F., Chen, T., Su, W., & Alsafasfeh, Q. (2020). Control of battery charging based on reinforcement learning and long short-term memory networks. Computers & Electrical Engineering, 85, 106670.

[10] García-Álvarez, J., González, M. A., & Vela, C. R. (2018). Metaheuristics for solving a real-world electric vehicle charging scheduling problem. Applied Soft Computing, 65, 292-306.

[11] Basset, M. (2021). Hybrid Heuristic and Metaheuristic for Solving Electric Vehicle Charging Scheduling Problem. In Evolutionary Computation in Combinatorial Optimization: 21st European Conference, EvoCOP 2021, Held as Part of EvoStar 2021, Virtual Event, April 7-9, 2021, Proceedings (Vol. 12692, p. 219). Springer Nature.

[12] Wang, C., Guo, C., & Zuo, X. (2021). Solving multi-depot electric vehicle scheduling problem by column generation and genetic algorithm. Applied Soft Computing, 112, 107774.

[13] Milas, N. T., Mourtzis, D. A., Giotakos, P. I., & Tatakis, E. C. (2020, September). Two-Layer Genetic Algorithm for the Charge Scheduling of Electric Vehicles. In 2020 22nd European Conference on Power Electronics and Applications (EPE'20 ECCE Europe) (pp. P-1). IEEE.

[14] Hou, S., Jiang, C., Yang, Y., & Xiao, W. (2020, December). Electric Vehicle Charging Scheduling Strategy based on Genetic Algorithm. In Journal of Physics: Conference Series (Vol. 1693, No. 1, p. 012104). IOP Publishing.

[15] Yin, W. J., & Ming, Z. F. (2021). Electric vehicle charging and discharging scheduling strategy based on local search and competitive learning particle swarm optimization algorithm. Journal of Energy Storage, 42, 102966.

[16] Zhang, X., Wang, Z., & Lu, Z. (2022). Multi-objective load dispatch for microgrid with electric vehicles using modified gravitational search and particle swarm optimization algorithm. Applied Energy, 306, 118018.

[17] Bai, X., Wang, Z., Zou, L., Liu, H., Sun, Q., & Alsaadi, F. E. (2021). Electric vehicle charging station planning with dynamic prediction of elastic charging demand: a hybrid particle swarm optimization algorithm. Complex & Intelligent Systems, 1-12.

[18] Wang, N., Li, B., Duan, Y., & Jia, S. (2021). A multi-energy scheduling strategy for orderly charging and discharging of electric vehicles based on multi-objective particle swarm optimization. Sustainable Energy Technologies and Assessments, 44, 101037.

[19] Garcia Alvarez, J., González, M. Á., Rodriguez Vela, C., & Varela, R. (2018). Electric vehicle charging scheduling by an enhanced artificial bee colony algorithm. Energies, 11(10), 2752.