Online Parameter Tuned SAHiD Algorithm for Capacitated Arc Routing Problems

Changwu Huang*†, Yuanxiang Li† and Xin Yao*
*Guangdong Provincial Key Laboratory of Brain-inspired Intelligent Computation,
University Key Laboratory of Evolving Intelligent Systems of Guangdong Province,
Department of Computer Science and Engineering,
Southern University of Science and Technology, Shenzhen 518055, China
†School of Computer Science, Wuhan University, Wuhan 430072, China
Email: huangcw3@sustech.edu.cn, yxli@whu.edu.cn, and xiny@sustech.edu.cn (the corresponding author)

***Abstract*—The Capacitated Arc Routing Problem (CARP) is a general and challenging arc routing problem. As the problem size increasing, exact methods are not applicable, and heuristic and meta-heuristic algorithms are promising approaches to solve it. To obtain good performance, parameter values of heuristics or meta-heuristics should be properly set. In recent years, automatic parameter tuning, which includes off-line and online parameter tuning, has attracted considerable attention in the evolutionary computation community. At present, parameters are usually determined through simple off-line parameter tuning, such as empirical analysis or grid search, when designing algorithms for CARP. However, using off-line parameter tuning on CARP has some disadvantages, among which the computational cost is the serious one. This work proposed an online parameter tuning approach using exponential recency-weighted kernel density estimation (ERW-KDE), and combines it with the SAHiD algorithm, which is an hierarchical decomposition based algorithm for CARP, to constitute the online parameter tuned SAHiD (OPT-SAHiD) algorithm. The experimental results show that OPT-SAHiD significantly outperforms the compared algorithms on two CARP benchmark sets owing to the proposed online automatic parameter tuning approach. The proposed online automatic parameter tuning approach based on ERW-KDE not only improves the performance of SAHiD algorithm, but also removes the additional computational overhead required for off-line parameter tuning.**

***Keywords*—Capacitated arc routing problem, automatic parameter tuning, online parameter tuning, kernel density estimation.**

I. Introduction

The Capacitated Arc Routing Problem (CARP) [1] is the most general arc routing problem and a difficult combinatorial optimization problem. Specifically, CARP is the routing problem of servicing a set of edge (or arc) on a graph (or network) using a fleet of capacity constrained vehicles which are initially located at a node called depot. Usually, each edge in CARP stands for a street in the real world and the graph represents a street network. The goal of the problem is to minimize the total routing cost. Practical examples and applications of the CARP include street sweepers, snow removal vehicles, waste collection, and so forth. Because of its wide application , CARP has received more and more attention in the past few decades and many heuristic and meta-heuristic algorithms have been designed to solve it.

Theoretically, the CARP and its variants are NP-hard [2]. Thus, exact methods are only suitable to small-size CARP instances, and heuristic and metaheuristic algorithms are effective alternatives to address CARP. A large number of heuristic and meta-heuristic algorithms, for instances, Path-Scanning [3], Ulusoy's method [4], genetic algorithm [5], and etc., have been proposed or introduced to solve CARP during the past few decades. Most of the existing approaches are designed for relatively small-scale CARP instances which usually consist of up to a few hundred edges and tasks. With the rapid development of cities, real-world CARPs usually involve a large number of roads and tasks, such as several hundred and thousand edges. Consequently, efficient approaches for large-scale CARPs are highly desired. It is more computationally expensive to find a near-optimal or sub-optimal solution for large-scale CARPs than that for small-scale ones due to the problem size increasing. There are, to the best of our knowledge, only a few recent works focus on large-scale CARPs. Tang *et el.* [6] proposed the SAHiD algorithm for large-scale CARPs that consists of thousands of edges (or arcs).

To efficiently solve a problem or to provide high-quality solutions, in addition to designing effective strategies or schemes to create candidate solutions and to improve their quality, the value of control parameters should be properly set by algorithm designers or end-users since parameter setting has strong impact on performance of parameterized algorithm. The so-called parameter setting problem [7], that is, to identify proper parameter setting for algorithms, is an important and challenging task in the design and application of algorithm. Parameter setting problem is mainly divided into off-line parameter tuning (usually referred to as parameter tuning) [8], where suitable parameter values are identified before the algorithm starts to run, and online parameter tuning (also known as parameter control) [9], where the values of parameters are changing dynamically during the execution of algorithm according to some strategies. Recently, automatic parameter setting methods, that is, automatic approaches to handle parameter setting problems, has draw considerable attention from algorithm designers and end-users. Many automatic off-line parameter tuning approaches have been proposed and

978-1-7281-6929-3/20/$31.00 ©2020 IEEE

applied to parameterized algorithms for hard problems. Interested readers can find more methods and details in [10] which comprehensively reviewed off-line parameter tuning approaches. Online parameter tuning, however, is still an open issue that requires further research and development since it is more challenging than off-line tuning problem. [9] briefly summarized the development of parameter control in evolutionary computation and pointed out the challenges in this area.

Automatic parameter tuning approaches, however, are rarely proposed and employed in CARP area. At present, algorithm's parameter values are commonly set by means of empirical analysis when designing or applying algorithms to CARP. When designing the SAHiD algorithm in [6], empirical tests of 25 different parameter settings were conducted to determine a proper one from them. This approach is commonly known as grid search, which is one of the simplest off-line parameter tuning methods. To improve above simple off-line parameter tuning approach and further enhance the performance of SAHiD algorithm, Huang *et el.* [11] adopted Bayesian optimization as an off-line parameter tuning approach to automatically tune two parameters of the SAHiD algorithm. The results showed that the off-line automatic parameter tuning approach significantly improve the performance of the SAHiD algorithm. However, off-line parameter tuning approaches have some disadvantages. The most serious one is that off-line parameter tuning is usually time-consuming and computational expensive because a large number of possible parameter value combinations need to be explored before finding a proper parameter setting. This is especially the case for computationally expensive problems, such as large-scale CARPs. Additionally, the obtained parameter setting from off-line tuning may be not proper or optimal for problem instances at hand. Since the parameter values are optimized according to the empirical performance estimated on training problem instances in off-line tuning, the obtained parameter setting usually cannot achieve peak performance on problem instances that are significantly different from the training instances. Additionally, in off-line parameter tuning, the obtained parameter setting keeps unchanged in solving the problem, but the proper or optimal parameter values may changes at different steps or stages of the search process. Online parameter tuning methods can eliminate or alleviate above drawbacks. Therefore, this paper focuses on introducing online parameter tuning approach to SAHiD algorithm with the goal of improving algorithm performance and alleviating above described disadvantages suffered by off-line parameter tuning.

In this work, an online parameter tuning approach based on exponential recency-weighted kernel density estimation (ERW-KDE) is proposed and applied to the SAHiD algorithm to constitute an online parameter tuned SAHiD (OPT-SAHiD) algorithm. In OPT-SAHiD, its parameter values are determined by the online parameter tuning procedure and dynamically changed during the problem solving process, thus the parameter setting task is addressed in an online fashion and the computational overhead for exploring different parameter value combinations required in off-line parameter tuning is removed. To examine the effectiveness of the proposed online parameter tuning approach and the OPT-SAHiD algorithm, experimental studies are conducted on two CARP benchmark sets. The experimental results show that the OPT-SAHiD algorithm achieves better solution quality than the investigated algorithms, because of the use of the proposed online parameter tuning approach.

The rest of the paper is organized as follows. First, the background, including the definition of CARP and the SAHiD algorithm, are introduced in Section II. Then, the proposed online parameter tuning approach ERW-KDE and the OPT-SAHiD algorithms are presented in Section III. Section IV provides experimental studies to evaluate the performance of our proposed approach. Finally, a brief conclusion is given in Section V.

II. Background

A. The Capacitated Arc Routing Problem (CARP)

The CARP considered in this paper is defined on a connected undirected graph (or network) $G(V, E)$, where V denotes the set of nodes (or vertices) and E represents the set of edges (or arcs). Each edge $e \in E$ associates with a deadheading cost $c(e) > 0$ and a demand $d(e) \geq 0$. An edge with positive demand is referred to as a task, and all the edges that have positive demands form the so-called task set $T = \{\tau \in E | d(\tau) > 0\}$. A fleet of identical vehicles with capacity Q are located at the node called depot, which is a predefined node $v_0 \in V$. The goal of CARP is to minimize the total cost of a set of routes for the fleet of vehicles to serve all the edges in task set T, subject to the following constraints:

1) each route must start and end at the depot v_0,
2) each task $\tau \in T$ is served exactly once,
3) the total demand of tasks served in each route, i.e., the total demand of tasks served by a vehicle, cannot exceed the vehicle capacity Q.

B. The SAHiD Algorithm

In this subsection, the SAHiD algorithm proposed in [6] is briefly described. The SAHiD algorithm is an iterative algorithm that involves the loop of create new solution based on previous one and improve it by local search. To handle large-scale CARPs, SAHiD adopts the hierarchical decomposition scheme which decomposes the tasks into subgroups and solves the induced sub-problems recursively. The general procedure of the SAHiD algorithm taken from [12] is presented in Algorithm 1.

Before describing the execution process of the SAHiD algorithm, its three main operators, namely, the HDU, LS, and Reconstruction operators, are introduced. The HDU operator, which is a combination of the hierarchical decomposition (HD) and the Ulusoy's split [4] procedures, is to create candidate solutions for CARP based on the task set. In HDU operator, firstly, the HD procedure dedicates to finding a permutation of all tasks (a giant tour that passes all tasks) regardless of the capacity constraint. Next, the Ulusoy's split procedure is

Algorithm 1 The SAHiD Algorithm [12]

1: **input**: parameter setting (β, α), CARP instance G, computational budget b_{max}.
2: apply HDU(β, G) to create an initial solution s;
3: apply LS(s) on solution s to improve it;
4: record the best solution: $s^* \leftarrow s$;
5: **while** budget b_{max} is not used up **do**
6: apply Reconstruction(α, β, s) to create new solution s';
7: apply LS(s') on solution s' to improve it;
8: **if** s' is accepted **then**
9: $s \leftarrow s'$;
10: **if** s' is better than s^* **then**
11: update the best solution $s^* \leftarrow s'$;
12: **end if**
13: **end if**
14: **end while**
15: **return**: s^*.

applied on the obtained permutation to split it into a set of (several) feasible routes that satisfy the capacity constraint, so that a candidate solution is achieved. The LS (local search) operator, which combines the reverse operator and merge-split operator [13], aims at improving the quality of the created candidate solution. The Reconstruction operator creates a new solution based on an existing solution. Specifically, it firstly splits an existing solution into a set of virtual tasks and then the HDU operator is applied on this virtual task set to create a new solution.

Next, the executing process of SAHiD algorithm is briefly described. Firstly, the HDU operator is used to generate an initial solution s of CARP before entering the loop (Line 2). Then, the initial solution s is improved by the LS operator (Line 3). After that, the SAHiD algorithm enters into the iterative procedure (Line 5 to 14). In each iteration, a new solution s' is generated by applying the reconstruction operator on current solution s (Line 6). This new solution s' is then improved by using LS operator on it (Line 7). Lastly, the threshold accepting idea is used to determine whether the solution s' will be selected to enter in the next iteration or not (Line 8 to 13). Above iterative procedure is terminated until the computational resource, such as maximum iterations or run time, is exhausted. The current best solution s^* is returned as the obtained solution (Line 15). More details about the SAHiD algorithm can be found in [6].

In the original SAHiD algorithm, the two parameters, that is, the scale parameter β for HDU operator and the probability α of splitting a route into sub-routes for reconstruction operator, were set by the authors via experimental analysis. Specifically, the performance of SAHiD algorithm with 25 different α and β value combinations were evaluated on 4 representative CARP instances, and the parameter setting that achieved best performance was selected. The two parameters β and α both are real-valued parameters that take values within [0, 1]. In the original SAHiD algorithm given in [6], these two parameters are set as $\beta = 0.1$ and $\alpha = 0.1$. In the following of this paper, the SAHiD algorithm using $\beta = 0.1$ and $\alpha = 0.1$ is named as the Original-SAHiD algorithm for the sake of simplicity.

III. Online Parameter Tuning using Exponential Recency-Weighted Kernel Density Estimation

A. Kernel Density Estimation

Kernel density estimation (KDE) [14] is a popular unsupervised learning approach to estimate the unknown probability density function (PDF) of random variables based on a set of observations or samples [15]. It can estimate the PDF for both univariate and multivariate (or multidimensional) data set [16]. As a nonparametric density estimation method, KDE does not require the assumption of the underlying distribution, it automatically learns the PDF from the observed data set. This nonparametric nature and its flexibility make KDE become a very popular and widely-used density estimation approach.

Let $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^d$ denote a set of independent and identically distributed (IID) samples drawn from some distribution with unknown density function p. The KDE model or estimator of the density function $p(\mathbf{x})$ is expressed as,

$$\hat{p}_n(\mathbf{x}) = \frac{1}{nh^d} \sum_{i=1}^{n} K(\frac{\mathbf{x} - \mathbf{x}_i}{h}), \tag{1}$$

where $K : \mathbb{R}^d \mapsto \mathbb{R}$ is known as kernel function and $h > 0$ is the bandwidth (also known as smoothing parameter). One of the most commonly used kernel functions [17] is the Gaussian kernel,

$$K(\mathbf{x}) = \frac{1}{\sqrt{2\pi}} e^{-\frac{\mathbf{x}^2}{2}}. \tag{2}$$

KDE firstly smooths out each data sample into a bump, which is defined by the kernel function $K(\mathbf{x})$. Then, it obtain a density estimator by summing up all these bumps. Consequently, KDE model will output a high density value at the regions where many data points are located, since many bumps are around there. On the contrary, for regions where only a few data samples are observed, the density value will be low.

B. Exponential Recency-Weighted KDE

In this paper, we attempt to use KDE to estimate the probability distribution function of promising or successful parameter values for SAHiD based on the used parameter settings during the execution of the algorithm. Since the promising parameter value may change at different stages of the algorithm execution, its probability density changes over time. In other words, the probability density of successful or promising parameter values is nonstationary. It is well known that nonstationary problems are often encountered in reinforcement learning problem, for instance, the nonstationary multi-armed bandit (MAB) problems [18]. An effective and commonly used approach to address nonstationary MAB is to use the so-called exponential recency-weighted average action value [18] in stead of classical average action value

in stationary MAB. In this work, we adopt the idea of exponential recency-weighted average and bring the exponential recency-weights into KDE, which is referred to as Exponential Recency-Weighted Kernel Density Estimation (ERW-KDE).

Given that the samples of parameter setting $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^d$ are collected incrementally during the run of algorithm, that is, $\mathbf{x}_k$ is acquired after $\mathbf{x}_{k-1}$, each sample $\mathbf{x}_i$ is assigned a exponential recency-weight (ERW) $w_i = \alpha^{n-i}$, where α is called the discount which takes value within (0, 1). Since α is less than 1, the weight given to the very last sample $\mathbf{x}_n$ is 1, and weights given to prior samples $\mathbf{x}_i (i < n)$ decreases exponentially. In other words, weights on newly collected samples are larger than that of the older ones. The density is estimated by KDE based on $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^d$ with consideration of the assigned weights. In this way, more emphasis is put on the last samples. Consequently, the density estimator in Equaton 1 is changed into Equation 3 in ERW-KDE.

$$\hat{p}_n(\mathbf{x}) = \frac{1}{nh^d} \sum_{i=1}^{n} w_i K(\frac{\mathbf{x} - \mathbf{x}_i}{h}), \tag{3}$$

C. Online Parameter Tuned SAHiD Algorithm

The basic idea of online parameter tuned SAHiD (OPT-SAHiD) algorithm using ERW-KDE is to learn probability distribution of promising parameter value from the previously used parameter settings and sample new parameter setting during the execution of the algorithm. Firstly, different parameter values are used in the beginning iterations of SAHiD and the successful or promising parameter settings are identified and collected. Here, the parameter setting with which the algorithm generates a candidate solution with better quality than that of the previous iteration is marked as successful or promising. When enough data is collected, the density distribution function of the successful parameter values is learned by ERW-KDE based on the collected data set. New parameter settings are then draw from the distribution function estimator from ERW-KDE. By this means, the parameter values are dynamically sampled during the execution of the algorithm. The OPT-SAHiD is illustrated in Figure 1.

It is assumed that, before the algorithm starts running on a CARP instance, no prior knowledge about promising parameter values, i.e., which parameter values will performs well, is available. Thus, in the first few iterations, two parameters α and β are uniformly sampled within (0, 1). So that, different parameter values are applied and the parameter space is explored to identify promising regions. If a new solution generated by using a parameter setting is better than the best solution of the previous iteration, the used parameter setting is considered as successful. The successful parameter settings are collected and saved in the data set $\mathcal{D}$. When there are enough data samples in $\mathcal{D}$ (in this paper, at least 10 samples in $\mathcal{D}$), the probability distribution $p(\alpha, \beta)$ is estimated by training a ERW-KDE model on data set $\mathcal{D}$. Then, parameter setting for next iteration is sampled from the learned distribution $\hat{p}(\alpha, \beta)$. Once there is a new sample, i.e., a successful parameter setting, added to $\mathcal{D}$, the ERW-KDE model is retrained and the density distribution estimator is updated. The two parameters α and β are considered together as a two-dimensional variable for ERW-KDE, so that, the jointed effects of these two parameters on algorithm performance is considered in the online parameter tuning in this work.

IV. Experimental Studies

To validate the performance of our proposed OPT-SAHiD algorithm, empirical studies are conducted to evaluate the performance of OPT-SAHiD by comparing it against the Original-SAHiD on two benchmark sets. Furthermore, to conform the advantage of online parameter tuning approach using ERW-KDE, another two algorithms the Random-SAHiD, in which ERW-KDE is not used and (α, β) are uniformly sampled within (0, 1) in each iteration, and the Adaptive-SAHiD proposed in [12] are also compared with OPT-SAHiD. The difference between OPT-SAHiD and Random-SAHiD is that the latter dose not use ERW-KDE to sample (α, β). And the difference between OPT-SAHiD and Adaptive-SAHiD is that the latter uses KDE rather than ERW-KDE. In other words, in Adapive-SAHiD, when estimating the distribution of promising parameter value by KDE, all the data samples taken equal weights as 1 and the exponential recency-weights (ERW) are not used.

A. Experimental Setting

Two CARP benchmark sets, that is, the EGL-G [19] benchmark set of a medium-scale CARP instances, and the Hefei [6] benchmark set of large-scale CARP instances, are used in experimental studies. The stopping criterion of the investigated algorithms are set the same as the maximum iterations $b_{max} = 500$. The experimental results are collected by executing each algorithm for 25 independent runs on each CARP instance. Meanwhile, all the investigated algorithms are started from the same initial solution, which make the comparison fair. For online parameter tuning approach using ERW-KDE in OPT-SAHiD, the Gaussian kernel is adopted and the bandwidth is set as $h = 0.2$, the discount is set as $\alpha = 0.9$.

B. Experimental Results

The total cost of solution, i.e., the costs of final solutions achieved by four investigated algorithms on two benchmark sets, are listed in Tables I and II, respectively. The columns headed "Best", "Average" and "Std" present the best, average and standard deviations of total costs among the 25 runs, respectively. The minimal average costs, i.e., the average quality of solutions, on each test instance among the investigated algorithms are marked with "*" and in bold. The OPT-SAHiD is compared with the Original-SAHiD, the Random-SAHiD and the Adaptive-SAHiD by using Wilcoxon rank-sum test with the level of significance 0.05 over 25 runs on each test instance. If the results of a compared algorithm is worse than that of OPT-SAHiD on test instances according to the statistical test, these results in columns headed "Average" are marked with underline. Otherwise, no symbol is marked on

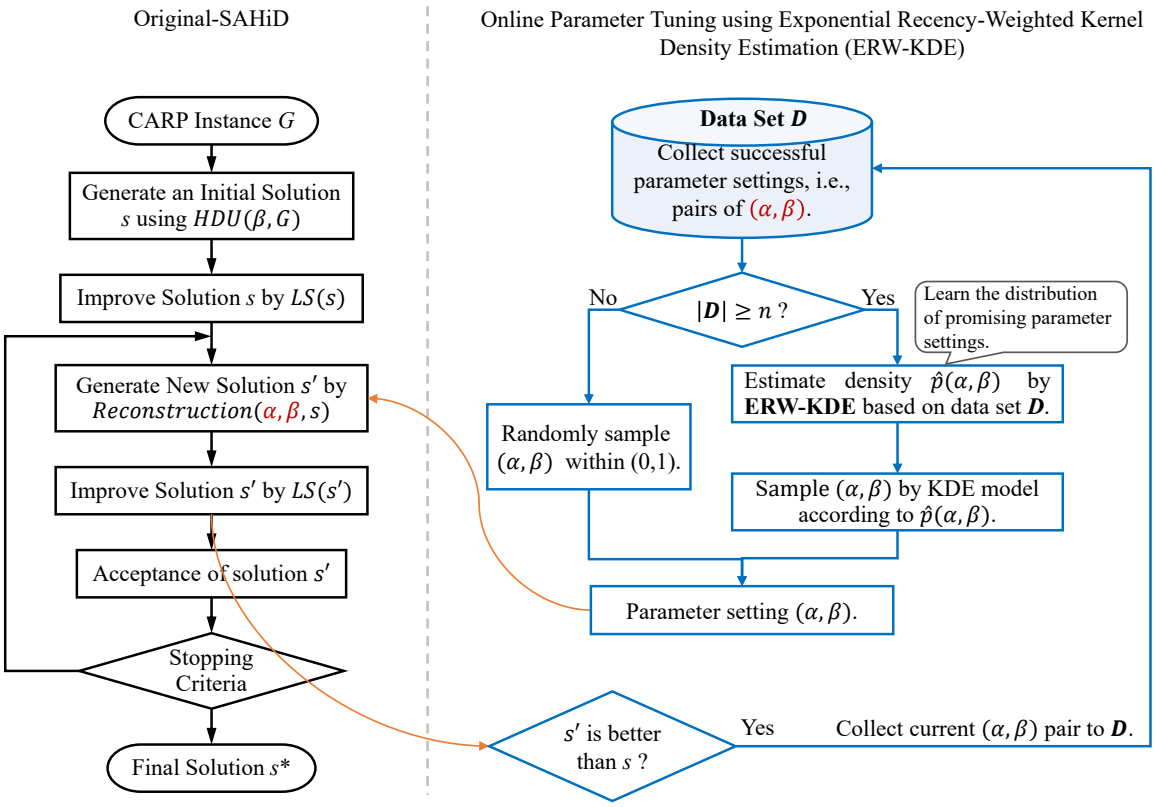

Fig. 1. Illustration of the Online Parameter Tuned SAHiD (OHT-SAHiD).

the results. For the three compared algorithms, the # of "w-d-l" summarizes the number of "Win-Draw-Loss" from the Wilcoxon rank-sum test.

The experimental results given in Tables I and II show that the OPT-SAHiD apparently achieves the best performance among the investigated algorithms. In terms of average cost of the solution, the OPT-SAHiD achieves the smallest average costs, i.e., higher solution quality, on all 10 instances of EGL-G test set and on 9 out of 10 instance of Hefei test set. Considering the results of statistical tests, the OPT-SAHiD performs significantly better than the other three compared algorithms on the majority of the 20 test CARP instances.

Firstly, according to the results of statistical tests between the OPT-SAHiD and the Original-SAHiD, the OPT-SAHiD performs better than the Original-SAHiD on 9 instances of EGL-G test set and 6 instances of Hefei test set. Considering that the values of α and β in Original-SAHiD are determined by off-line parameter tuning, which takes some additional computational cost, this computational overhead is not required in OPT-SAHiD since it performs parameter tuning in an online method. Thus, with the aid of online parameter tuning, the OPT-SAHiD not only performs better than the Original-SAHiD, but also removes the computational overhead required by off-line parameter tuning.

Furthermore, according to the results of statistical tests between the OPT-SAHiD and the Random-SAHiD, the OPT-SAHiD performs better on all 10 instances of EGL-G test set and 8 instances of Hefei test set. Additionally, in the comparison between the OPT-SAHiD and the Adaptive-SAHiD, the OPT-SAHiD performs better on 7 instances of EGL-G test set and 6 instances of Hefei test set. It is obvious that the OPT-SAHiD performs better than the Random-SAHiD and the Adaptive-SAHiD. Since the only difference between OPT-SAHiD and Random-SAHiD is that the latter dose not use ERW-KDE and the main difference between OPT-SAHiD and Adaptive-SAHiD is the latter dose not use exponential recency-weight (ERW) in KDE, it is no doubt that the advantages of the OPT-SAHiD over the three compared algorithm, namely the Original-SAHiD, Random-SAHiD and Adaptive-SAHiD, is credited to the proposed online automatic parameter tuning method using ERW-KDE. Consequently, this indicates that the proposed online autmatic parameter tuning approach based on ERW-KDE is effective, and this method brings significant performance improvement for SAHiD algorithm on CARPs.

Additionally, several average convergence curves of each algorithm on some test instancces from EGL-G and Hefei test sets are presented in Fig. 2. In these convergence plots,

TABLE I
RESULTS ON EGL-G BENCHMARK SET IN TERMS OF THE TOTAL SOLUTION COSTS.

Instance	$\|T\|$	Original-SAHiD			Random-SAHiD			Adaptive-SAHiD			OPT-SAHiD		
		Best	Average	Std	Best	Average	Std	Best	Average	Std	Best	Average	Std
EGL-G1-A	347	1046510	1077890	12684	1067200	1109020	19349	1056330	1077510	13073	1061720	**1076100***	9099
EGL-G1-B	347	1192080	1215120	11590	1215730	1255710	21219	1191650	1217030	12339	1182670	**1207030***	10284
EGL-G1-C	347	1305990	1360630	16903	1368280	1403510	17003	1326620	1352670	14858	1319410	**1346070***	14626
EGL-G1-D	347	1485660	1519800	18187	1517850	1560110	26845	1469520	1497770	15625	1456160	**1482270***	12827
EGL-G1-E	347	1667960	1699810	19750	1670280	1730880	26316	1620220	1649810	15980	1616210	**1636630***	12403
EGL-G2-A	375	1168060	1193310	13593	1187810	1235780	19847	1157570	1191610	15758	1159270	**1184970***	13371
EGL-G2-B	375	1286780	1326010	18270	1340910	1365080	15497	1291350	1319070	17361	1277030	**1308690***	13898
EGL-G2-C	375	1435220	1480480	20537	1475310	1526860	26747	1417920	1460010	13950	1421880	**1447970***	12296
EGL-G2-D	375	1623460	1657090	20641	1653120	1696170	19289	1577690	1609310	16963	1553820	**1592900***	20359
EGL-G2-E	375	1784170	1833590	22534	1809000	1857940	21189	1745260	1771290	16646	1729440	**1748860***	13532
# of "w-d-l"			9-1-0			10-0-0			7-3-0				

TABLE II
RESULTS ON HEFEI BENCHMARK SET IN TERMS OF THE TOTAL SOLUTION COSTS.

Instance	$\|T\|$	Original-SAHiD			Random-SAHiD			Adaptive-SAHiD			OPT-SAHiD		
		Best	Average	Std	Best	Average	Std	Best	Average	Std	Best	Average	Std
Hefei-1	121	251848	261543	5243	251122	258205	4279	250632	**257650***	4625	250869	259812	5150
Hefei-2	242	457984	473186	7395	464145	473861	6729	459629	473151	5929	457993	**471981***	6555
Hefei-3	364	616723	628681	7810	614935	631914	7287	609746	628383	8070	611008	**626548***	8058
Hefei-4	485	795461	808768	7344	808976	822577	7924	793902	807645	6222	791898	**805728***	7491
Hefei-5	606	1023600	1040980	9536	1038900	1061850	12980	1025290	1033030	4412	1014880	**1026250***	6452
Hefei-6	727	1181250	1208900	13909	1188050	1221970	15589	1161250	1179940	12118	1152040	**1172020***	9994
Hefei-7	848	1423620	1443690	14810	1412690	1446270	15520	1375310	1392270	9811	1366130	**1379110***	10151
Hefei-8	970	1640400	1665010	11777	1632290	1662510	16861	1572420	1592380	14304	1548750	**1578230***	17165
Hefei-9	1091	1833870	1860820	19245	1818680	1857910	22398	1754030	1791920	17619	1730590	**1768220***	24882
Hefei-10	1212	2016250	2041750	14987	1987010	2025830	17547	1920690	1965560	20587	1902580	**1936600***	22237
# of "w-d-l"			6-4-0			8-2-0			6-4-0				

it can be observed that Original-SAHiD performs better at the beginning which may thanks to off-line tuning, but the convergence speed of OPT-SAHiD and Adaptive-SAHiD is gradually accelerated owing to the online parameter tuning approach. And finally OPT-SAHiD and Adaptive-SAHiD outperform the Original-SAHiD. Although, OPT-SAHiD generally starts to speed up later than Adaptive-SAHiD, it converges more quickly than Adaptive-SAHiD in late stage and finally get better solutions. Because Random-SAHiD does not use online parameter tuning, it just randomly sample parameter values during the run of algorithm, no evident enhancement in convergence speed is observed during the run. This also validates that the proposed online parameter tuning approach based on ERW-KDE is an effective method to solve parameter setting problem.

V. CONCLUSION

This paper proposed an online parameter tuning approach based on exponential recency-weighted kernel density estimation (ERW-KDE) and introduced the so-called OPT-SAHiD algorithm for CARP. This online parameter tuning method firstly learns the distribution of successful or promising parameter values by using ERW-KDE and then samples new parameter values according to the estimated distribution function during the execution of the algorithm. Experimental studies are performed to evaluate the performance of our proposed OPT-SAHiD and the effectiveness of online parameter tuning approach based on ERW-KDE. The experimental results prove that the OPT-SAHiD significantly outperforms the compared algorithms, that is, the Original-SAHiD, Random-SAHiD, and Adaptive-SAHiD. This validates the effect of the proposed online automatic parameter tuning approach. By using the online automatic parameter tuning approach, OPT-SAHiD not only obtains significant performance improvement but also eliminates the parameter setting problems for end-users.

ACKNOWLEDGMENTS

This work was supported by Guangdong Basic and Applied Basic Research Foundation (Grant No. 2019A1515110575), Guangdong Provincial Key Laboratory (Grant No. 2020B121201001), the Program for Guangdong Introducing Innovative and Enterpreneurial Teams (Grant No. 2017ZT07X386), Shenzhen Science and Technology Program (Grant No. KQTD2016112514355531), the Program for University Key Laboratory of Guangdong Province (Grant No. 2017KSYS008).

REFERENCES

[1] S. Wøhlk, *A Decade of Capacitated Arc Routing*. Boston, MA: Springer US, 2008, pp. 29–48.

[2] R. van Bevern, R. Niedermeier, M. Sorge, and M. Weller, "Chapter 2: The complexity of arc routing problems," in *Arc Routing*. Society for Industrial and Applied Mathematics, oct 2013, pp. 19–52.

[3] B. Golden, J. Dearmon, and E. Baker, "Computational experiments with algorithms for a class of routing problems," *Computers & Operations Research*, vol. 10, no. 1, pp. 47–59, jan 1983.

[4] G. Ulusoy, "The fleet size and mix problem for capacitated arc routing," *European Journal of Operational Research*, vol. 22, no. 3, pp. 329–337, dec 1985.

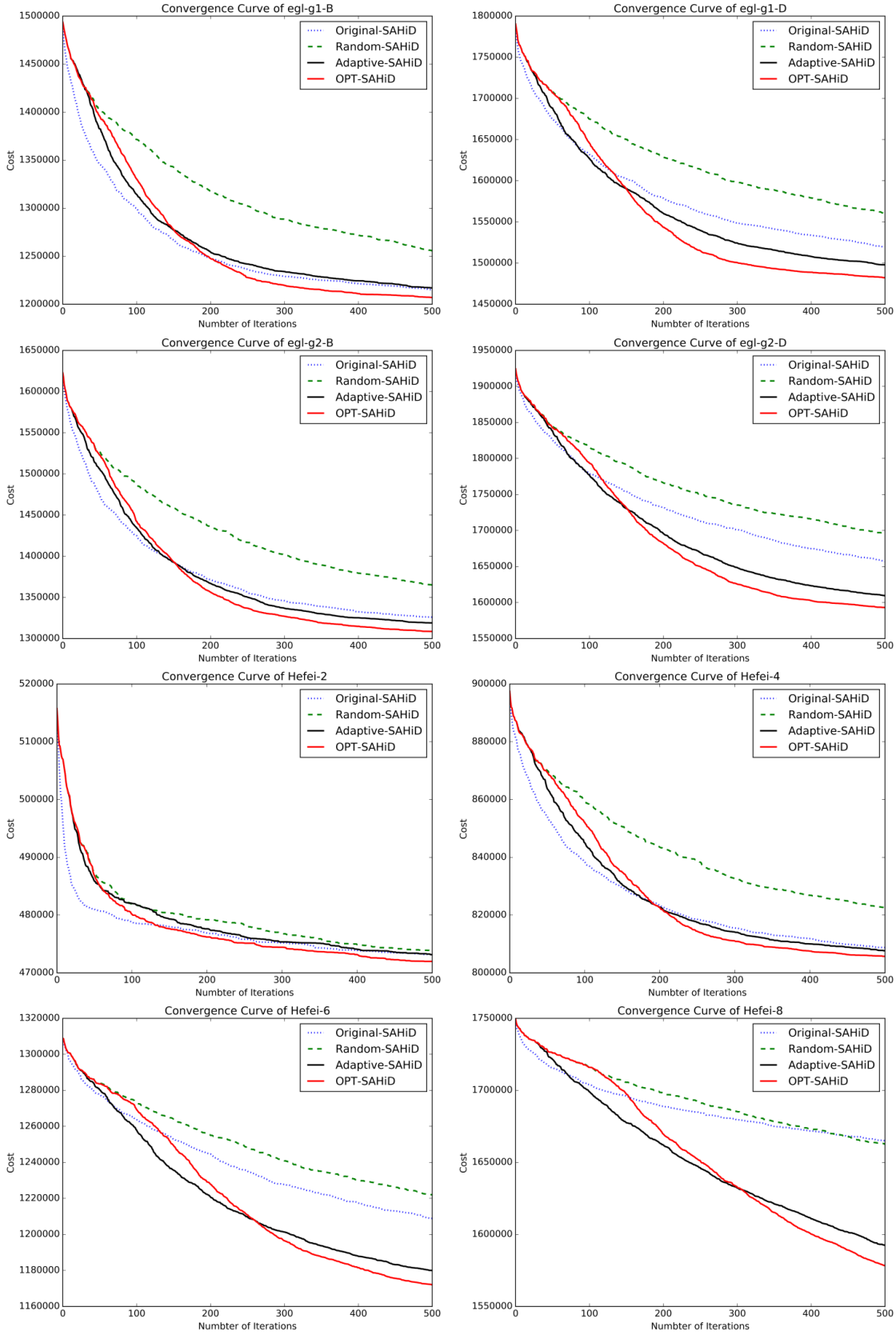

Fig. 2. Several convergence curves on test CARP instances. Each curve shows the average costs obtained over 25 runs from each investigated algorithm on a CARP instance.

[5] P. Lacomme, C. Prins, and W. Ramdane-Chérif, "Evolutionary algorithms for periodic arc routing problems," *European Journal of Operational Research*, vol. 165, no. 2, pp. 535–553, sep 2005.

[6] K. Tang, J. Wang, X. Li, and X. Yao, "A scalable approach to capacitated arc routing problems based on hierarchical decomposition," *IEEE Transactions on Cybernetics*, vol. 47, no. 11, pp. 3928–3940, nov 2017.

[7] H. H. Hoos, *Automated Algorithm Configuration and Parameter Tuning*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 37–71.

[8] A. E. Eiben and S. K. Smit, "Parameter tuning for configuring and analyzing evolutionary algorithms," *Swarm and Evolutionary Computation*, vol. 1, no. 1, pp. 19–31, 2011.

[9] G. Karafotias, M. Hoogendoorn, and A. E. Eiben, "Parameter control in evolutionary algorithms: Trends and challenges," *IEEE Transactions on Evolutionary Computation*, vol. 19, no. 2, pp. 167–187, April 2015.

[10] C. Huang, Y. Li, and X. Yao, "A survey of automatic parameter tuning methods for metaheuristics," *IEEE Transactions on Evolutionary Computation*, vol. 24, no. 2, pp. 201–216, 2020.

[11] C. Huang, B. Yuan, Y. Li, and X. Yao, "Automatic parameter tuning using bayesian optimization method," in *2019 IEEE Congress on Evolutionary Computation (CEC)*, June 2019, pp. 2090–2097.

[12] C. Huang, Y. Li, and X. Yao, "Adaptive-SAHiD algorithm for capacitated arc routing problems," in *2019 IEEE Symposium Series on Computational Intelligence (SSCI)*, December 2019, pp. 1668–1675.

[13] K. Tang, Y. Mei, and X. Yao, "Memetic algorithm with extended neighborhood search for capacitated arc routing problems," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 5, pp. 1151–1166, oct 2009.

[14] E. Parzen, "On estimation of a probability density function and mode," *Ann. Math. Statist.*, vol. 33, no. 3, pp. 1065–1076, sep 1962.

[15] Y.-C. Chen, "A tutorial on kernel density estimation and recent advances," *Biostatistics & Epidemiology*, vol. 1, no. 1, pp. 161–187, 2017.

[16] T. A. O'Brien, K. Kashinath, N. R. Cavanaugh, W. D. Collins, and J. P. O'Brien, "A fast and objective multidimensional kernel density estimation method: fastkde," *Computational Statistics & Data Analysis*, vol. 101, pp. 148–160, 2016.

[17] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and Édouard Duchesnay, "Scikit-learn: Machine learning in python," *Journal of Machine Learning Research*, vol. 12, no. 85, pp. 2825–2830, 2011.

[18] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[19] J. Brandão and R. Eglese, "A deterministic tabu search algorithm for the capacitated arc routing problem," *Computers & Operations Research*, vol. 35, no. 4, pp. 1112–1126, apr 2008.