Honda Research Institute EU
Honda Research Institute Europe GmbH
https://www.honda-ri.de/

A New Co-evolutionary Optimization Method for the Dynamic Vehicle Routing Problem

Xiaofen Lu, Ke Tang, Stefan Menzel, Xin Yao

2020

Preprint:

This is an accepted article published in IEEE Symposium Series on Computational Intelligence (SSCI). The final authenticated version is available online at: https://doi.org/10.1109/SSCI47803.2020.9308367 Copyright 2020 IEEE

A Competitive Co-evolutionary Optimization Method for the Dynamic Vehicle Routing Problem

Xiaofen Lu [1], Ke Tang [1], Stefan Menzel [2], Xin Yao [1]
1 *Guangdong Provincial Key Laboratory of Brain-inspired Intelligent Computation*
Department of Computer Science and Engineering
Southern University of Science and Technology
Shenzhen, China
Emails: luxf@sustech.edu.cn, tangk3@sustech.edu.cn, xiny@sustech.edu.cn
2 *Honda Research Institute Europe*
Offenbach/Main, Germany
Email: stefan.menzel@honda-ri.de

***Abstract*—In the basic vehicle routing problem, the best routes need to be found for a fleet of vehicles to serve a set of customers. The dynamic vehicle routing problem is a variant of the vehicle routing problem, in which part or all the information defining the routing problem might change over time. This requires an optimizer to perform a fast search of new routes once the change happens. For its robustness with respect to noise, evolutionary algorithms have shown great potential for the dynamic vehicle routing problem. However, the existing evolutionary methods only evolve a population of solutions online for the ever-changing problem, which might not be very efficient. In this work, we propose a new evolutionary method which combines offline computation and online optimization to solve the dynamic vehicle routing problem. It first searches for a set of good solutions for possible environmental changes in an offline way, and then searches for the new optimal solution online for the ever-changing problem through doing local search on this solution set. Competitive co-evolution is applied to search for the solution set in the proposed method. Experimental study has been conducted on one dynamic vehicle routing benchmark with the change of customers' demands. The experimental results show the efficiency of the proposed method.**

***Index Terms*—Dynamic vehicle routing problem, Dynamic optimization, Evolutionary algorithms, Competitive co-evolution**

I. Introduction

Vehicle routing plays a central role in the areas of logistics, transportation and other related industries like retail, ride hailing and food delivery industries [1]. Therefore, the vehicle routing problem (VRP) has attracted a lot of researchers' and practitioners' interests [2]. In the basic VRP problem, a fleet of vehicles are routed to serve a set of customers. Each vehicle starts and ends at a central depot. Each customer is visited once by only one vehicle. A set of multiple routes (each route for one vehicle) with minimum cost (e.g. travelling distance, time, fuel etc.) needs to be found while satisfying all constraints [3].

Considering the nature of real-world systems, researchers have studied several variants of the VRP including Capacitated VRP, Capacitated Arc Routing Problem [4], [5], [6], [7], Multi-Depot VRP, VRP with time windows, and VRP with Pick-Up and Delivery (VRPPD) [8], [9]. Dynamic VRP (DVRP) is a complex variant of static VRP [10]. In the DVRP, the number of vehicles, the vehicle load or speed might change over time due to the conditions of the vehicles, while the customers' requirements or the traffic conditions also change over time, and these will result in the constant change of the best route [11]. For dynamic VRP, an optimizer is required to be able to react quickly to the changes and continuously search for the best solution under new environments. In the literature, there are two kinds of optimization models for DVRPs, the periodic re-optimization model and continuous re-optimization model. In a periodic re-optimization model, the DVRP is treated as a sequence of static VRP instances which contain all the customers' information known at particular time. In a continuous re-optimization model, a DVRP is optimized throughout the working day and the optimized vehicle routes are maintained in an adaptive manner [12].

As a class of nature-inspired optimization methods, evolutionary algorithms (EAs) have shown good robustness with respect to noise [13], and thus have been widely studied in the field of dynamic optimization [14], [15], [16], [17], [18]. EAs have also shown great potential for the dynamic VRP. In [9], a genetic algorithm was applied to solve dynamic periodic VRP. In [19], an ant colony system algorithm was proposed to solve periodic VRP. In [20], the ant colony optimization (ACO) algorithm was integrated with immigrants schemes to solve dynamic VRP. In [21], the authors proposed a self-adaptive EA which evolves a set of configurations including parameter values, operator types, combination of operators and order of operation invocation.

However, the existing evolutionary methods only evolve a population of solutions in an *online* fashion to track the changing optimum. Since EAs usually need a lot of fitness evaluations to achieve a near-optimum solution, the evolutionary dynamic methods might be inefficient when the problem changes fast. Considering a fast response to environmental changes is the key requirement for dynamic VRP, this paper proposes a new evolutionary method to solve DVRP. In this method, a set of good solutions for possible environmental changes are obtained in an *offline* way before the whole system runs. After this, the method searches *online* for the

new optimum under dynamic environments by doing local search on the obtained solution set. We assume that the range of environmental changes are known beforehand. Considering that the environment space might be large, it is computationally unacceptable to consider all the possible environmental parameter values during the search process. To solve this issue, this paper proposes a method which searches the solution set and a representative set of environmental parameter values in a typical competitive co-evolution scheme [22], [23]. As a set of good solutions have been prepared beforehand, fast response to environmental changes can be expected. Moreover, the online search part can be done in a parallel way as only local search is conducted on each solution. This will further accelerate the response to new environments. An experimental study has been conducted on 10 DVRP instances generated from one stationary VRP benchmark with change of customers' demands. The experimental results showed the efficiency of the proposed method.

The remainder of this paper is organized as follows. Section 2 introduces the DVRP considered in this paper and the related work. Section 3 details the proposed method for solving DVRP. In Section 4, the experimental studies on 10 DVRP instances are presented. Section 5 concludes this paper and points out future work directions.

II. Problem Description and Related Work

In this paper, we consider the dynamic capacitated VRP problem. In the capacitated VRP problem, a fleet of vehicles with limited capacity are routed to serve a set of customers at a minimum cost, e.g. the total travel time. Each vehicle starts and ends at a central depot. Each customer should be visited once by only one vehicle. This means that an efficient set of multiple routes with minimum cost needs to be found (Fig. 1). In Fig. 1, v_0 denotes the central depot. Note that the total demand of each route can not exceed the vehicle capacity Q. Based on these constraints, we formulate the formulation for the basic VRP problem as follows:

$$\begin{aligned} &\min cost(R) = \sum_{i=1}^{k} c(C_i) \\ &s.t. \;\; l(C_i) \leq Q \end{aligned} \tag{1}$$

where k is the number of vehicles, $c(C_i)$ calculates the sum of the travel cost in route C_i, and $l(C_i)$ denotes the load of the vehicle on the route C_i, i.e. the total demand of customers on route C_i.

We consider the periodic re-optimization DVRP model with dynamic demands, i.e. changes in the number of ordered goods. In this problem, a fleet of vehicles serves multiple fixed customers periodically, e.g. on daily basis. At the beginning of each period, the demand information of each customer is given and the routes of each need to be re-optimized. The problem can be formulated as follows:

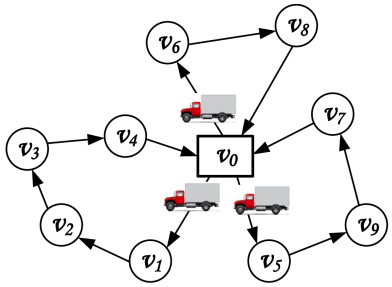

Fig. 1. The Static VRP Problem, taken from [12]

$$\begin{aligned} &\min cost(R,t) = \sum_{i=1}^{k} c(C_{i,t}) \\ &s.t. \;\; l(C_i,t) \leq Q \end{aligned} \tag{2}$$

where k is the number of vehicles, t denotes time and $C_{i,t}$ denotes the route for the i-th vehicle at time t. The cost of each route is defined as the total distance of that route multiplied by half of the total load for the vehicle that serves this route. In this work, we assume the range for the demand of each customer is known beforehand but how the demand of each customer changes is not known beforehand.

This problem is in fact strongly related to some real-world applications like courier service, vendor-managed distribution, storekeeper service systems and so on. In the courier service problem, parcels are collected from a local depot and brought back to a central depot for further shipping. In the vendor-management distribution system, distribution companies need to replenish vendors before they run out of stock. In the storekeeper service problem, a grocery depot needs to ship supplies daily to storekeepers [11]. For these problems, the customers are all static but the demands of them are changing over time.

The DVRP problem belongs to the category of dynamic optimization problems (DOPs) for which an optimizer is expected to identify the new optimum efficiently when the problem changes. In the literature, many evolutionary dynamic optimization methods have been proposed for DOPs, including diversity-driven methods, memory methods, and prediction methods [17]. Diversity-driven methods mainly include introducing random immigrants, i.e. individuals, into the current population after a change happens [24] or maintaining the diversity of population during the evolutionary process [15]. Memory methods store and use good-performing solutions in previous search [25]. Prediction methods predict new optimal solutions based on the best solutions obtained before and use them for population re-initialization once the problem changes [26]. Since memory methods and prediction methods are more appropriate for cyclic and predictable problems, diversity-driven methods are most commonly used methods, which have shown promising results for DVRP [20], [27].

Although these methods were developed for DOPs, challenges still exist in the field of DOPs. When only limited time

is available to make a decision, these methods might not work efficiently. The diversity-driven methods do not have enough time to find a satisfactory solution. The memory methods lack of optimal solutions to archive. The prediction methods are short of useful samples for accurate prediction. Therefore, efficient dynamic handling methods are still needed for DOPs.

III. A New Method for DVRP

When applying EAs to solve problems that change over time, good initial solutions are essential for fast reaction to each environmental change. Suppose we can explore as many different environmental changes as possible and find good solutions under such environmental changes beforehand. Then, fast optimization can be expected for each truly happening environmental change by conducting an efficient local search starting from one of these found solutions. Based on this motivation, this paper proposes a competitive evolutionary method which combines *offline* computation and *online* optimization to solve DVRP.

The proposed evolutionary method for DVRP is called set-based dynamic vehicle routing (setDVR for short). The general idea is to search a solution set for the DVRP in an *offline* fashion that contains good solutions for different environmental parameter values before the whole system runs. In the DVRP considered in this paper, the environmental parameters are the demands of customers. In the *online* optimization phase, each time the demands of the customers in the DVRP change, the method triggers a local search on the obtained solution set to find an optimal solution for the changed problem. Algorithm 1 details the procedure of the proposed setDVR method.

Algorithm 1 The Proposed setDVR Framework

```
Do set search to get a solution set X considering the possible values for the demand of each customer
Archive the solution set X
Initialize a population pop with X
Evaluate pop with the current cost function
while stopping criteria is not satisfied do
    Do local search on each individual of pop
    if the demand of each customer changes then
        Re-intialize the population pop with X
        Evaluate pop with the current cost function
    end if
end while
```

A. Offline Search of Solution Set

In the offline set search process, the setVRP method aims to find a solution set that can cover possible changes of the demand for each customer. Considering that the range for each customer's change might be large, it is computationally unacceptable to consider all environmental changes during the process of searching the solution set. To solve this issue, competitive co-evolution [23] is used to search the solution set and a representative set of customers' demanding values in an adversarial way. Inspired from biology, CC co-evolves a prey population and a predator population in an adversarial way to solve problems without explicit objective function. During the evolutionary process, the fitness of each individual in the prey population depends on its performance on the individuals in the predator population and vice versa. Through this adversarial way, both of the two populations can be enhanced.

In the co-evolutionary set search process, the prey population is used to represent the solution set and the predator population is used to represent the demand parameters. The prey population is denoted as SP (solution population) and the predator population is denoted as DP (demand population). Algorithm 2 details the competitive co-evolutionary framework for SP and DP. At the beginning, setVRP initializes both SP and DP by randomly generating m individuals from the search space of $\boldsymbol{x}$, i.e. routes for all vehicles, and n individuals from the variation range of $\boldsymbol{d}$, i.e. demands for all customers. The evolution of SP targets to obtain better and better solutions for individuals in DP, while the evolution of DP targets to obtain more and more challenging demands for individuals in SP. Through this adversarial scheme, the SP is expected to have good solutions for more and more problems with different demands.

Algorithm 2 Co-evolutionary Framework for SP and EP

Require: Search space for the solution $\boldsymbol{x}$, variation range for the demand parameters $\boldsymbol{d}$, maximum number of generations: G_{max}

```
Set g = 0
Initialize solution population SP_0
Initialize environment population DP_0
while g ≤ G_max do
    Evolve SP_g to get SP_{g+1} based on DP_g
    Evolve DP_g for one generation based on SP_{g+1}
    Set g = g + 1
end while
```

Ensure: Output $SP_{G_{max}}$ as the soution set $\boldsymbol{X}$

Algorithm 3 provides details on the evolutionary process of SP for one generation. At every generation g, two individuals $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ are randomly selected from SP at the current generation, i.e. SP_g). Then, a new individual $\boldsymbol{x}_{new}$ is generated through evolutionary operation on $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ and added to SP_g to get SP'_g. After this, the worst individual $\boldsymbol{x}_{worst}$ that has the smallest fitness will be deleted from SP'_g and the remaining individuals in SP'_g will enter the next generation as SP_{g+1}.

Algorithm 3 SP Evolution for One Generation

Require: Solution pop: SP_g, demand pop: DP_g

```
Evaluate the fitness of each x in SP_g based on EP_g
Randomly select x_1 and x_2 from SP_g
Generate x_new based on x_1 and x_2
Set SP'_g = SP_g ∪ x_new
Evaluate the fitness of each x in SP'_g
Set SP_{g+1} = SP'_g / x_worst
```

The fitness of $\boldsymbol{x}_i$ in SP is defined as the performance drop of SP on DP after deleting $\boldsymbol{x}_i$ from SP. Suppose $F(SP, \boldsymbol{d})$ denotes the performance of SP for the VRP under the demands $\boldsymbol{d}$. $F(SP, \boldsymbol{d})$ is calculated as the fitness value

of $\boldsymbol{x}_{best}$ that performs best on $\boldsymbol{d}$ among all $\boldsymbol{x}_i$ in SP. If $\boldsymbol{x}_{best}$ is feasible for the problem under the demands $\boldsymbol{d}$, then $F(SP, \boldsymbol{d}) = cost(\boldsymbol{x}_{best}, \boldsymbol{d})$; otherwise, $F(SP, \boldsymbol{d})$ is set to the sum of constraint violation values of $\boldsymbol{x}_{best}$ on $\boldsymbol{d}$, i.e. the load beyond the capacity of each vehicle. Considering the problem scaling, we define the fitness of $\boldsymbol{x}_i$ in SP as the number of $\boldsymbol{d}$ in DP on which the performance changes if deleting $\boldsymbol{x}_i$ from SP. That is:

$$Fit(\boldsymbol{x}_i) = \sum_{\boldsymbol{d} \in EP} [F(SP, \boldsymbol{d}) \neq F(SP/\boldsymbol{x}_i, \boldsymbol{d})] \tag{3}$$

When comparing two individuals in SP that have the same $Fit(\boldsymbol{x}_i)$, one individual is randomly selected.

Algorithm 4 details the evolutionary process of DP for one generation. At every generation g, a new population DP_{new} is generated based on DP_g using evolutionary operation. Then, DP_g and DP_{new} are combined to obtain DP'_g. After this, DP_g and DP_{new} will be compared using a pair-wise comparison based on their challenging degree. All winners enter the next generation as DP_{g+1}.

Algorithm 4 DP Evolution for One Generation

Require: Demand pop: DP_g, solution pop: SP_g
1: Generate a new population DP_{new} based on DP_g using crossover and mutation
2: Set $DP'_g = DP_g \cup DP_{new}$
3: Evaluate each $\boldsymbol{d}$ in DP'_g
4: **for** i=1,2,...,$size_of_DP$ **do**
5: **if** DP^i_g is more challenging than DP^i_{new} **then**
6: $DP^i_{g+1} = DP^i_g$
7: $DP^i_{worse} = DP^i_{new}$
8: **else**
9: $DP^i_{g+1} = DP^i_{new}$
10: $DP^i_{worse} = DP^i_g$
11: **end if**
12: **end for**

An environment $\boldsymbol{d}$ is identified as being more challenging if individuals in SP perform worse under this environment. Thus, to evaluate the challenging degree of an individual $\boldsymbol{d}_j$ in EP, setDVR randomly generates a set of solutions (denoted as I) and uses the improvement obtained by SP over I on $\boldsymbol{d}_j$ to calculate the fitness. According to whether the best solutions for $\boldsymbol{d}_j$ in I and SP are feasible and whether an improvement is investigated, an individual $\boldsymbol{d}_j$ can be categorized into the following four cases:

1) The best solution for $\boldsymbol{d}_j$ in SP is infeasible. In this case, the fitness of $\boldsymbol{d}_j$ is defined as the sum of constraint violation values of the best solution.
2) The best solution for $\boldsymbol{d}_j$ in SP is feasible, but it is worse than the best solution for $\boldsymbol{d}_j$ in I. In this case, the improvement obtained by I over SP on $\boldsymbol{d}_j$ is used to evaluate it. That is,

$$Fit(\boldsymbol{d}_j) = \frac{\min_{\boldsymbol{x} \in SP} F(\boldsymbol{x}, \boldsymbol{d}_j) - \min_{\boldsymbol{x} \in I} F(\boldsymbol{x}, \boldsymbol{d}_j)}{\max(|\min_{\boldsymbol{x} \in I} F(\boldsymbol{x}, \boldsymbol{d}_j)|, |\min_{\boldsymbol{x} \in SP} F(x, \boldsymbol{d}_j)|)} \tag{4}$$

The larger the improvement is, the more challenging $\boldsymbol{d}_j$ is.
3) The best solution for $\boldsymbol{d}_j$ in SP is feasible and the best solution in I is infeasible. In this case, as the improvement is hard to calculate, the fitness of $\boldsymbol{d}_j$ is defined as the function value of the best solution in SP for $\boldsymbol{d}_j$ (i.e. $\min_{\boldsymbol{x} \in SP} F(x, \boldsymbol{d}_j)$).
4) Both best solutions for $\boldsymbol{d}_j$ in SP and I are feasible but the best solution in SP is better than the best solution in I. $\boldsymbol{d}_j$ is considered less challenging if a larger improvement is obtained by SP over I. Thus, the fitness of $\boldsymbol{d}_j$ is defined as the negative of the improvement obtained by SP over I on $\boldsymbol{d}_j$. That is,

$$Fit(\boldsymbol{d}_j) = \frac{\min_{\boldsymbol{x} \in SP} F(\boldsymbol{x}, \boldsymbol{d}_j) - \min_{\boldsymbol{x} \in I} F(\boldsymbol{x}, \boldsymbol{d}_j)}{\max(|\min_{\boldsymbol{x} \in I} F(\boldsymbol{x}, \boldsymbol{d}_j)|, |\min_{\boldsymbol{x} \in SP} F(\boldsymbol{x}, \boldsymbol{d}_j)|)} \tag{5}$$

In each case, the larger the fitness is, the more challenging $\boldsymbol{d}_j$ is. When comparing individuals belonging to different cases, an individual belonging to case 1 is considered more challenging than an individual that belongs to case 2, an individual belonging to case 2 is considered more challenging than an individual belonging to case 3 or case 4. When comparing an individual belonging to case 3 and an individual belonging to case 4, a random one is selected.

In the evolution of DP, the individuals that do not enter the next generation are saved into DAr as they are less challenging. At the beginning, all $\boldsymbol{\alpha}$ individuals that do not enter the next generation are saved directly into DAr. When the number of added individuals achieves the size of DAr, each new individual $\boldsymbol{d}$ are added according to the following replacement rules:

1) Set $DAr' = DAr \cup \boldsymbol{d}$,
2) Calculate the diversity contribution of each individual in DAr',
3) Delete the individual that has the least contribution to the diversity of the archive.

The diversity contribution of each individual is calculated as the reduction of the diversity of the archive when deleting this individual. The diversity of the archive is calculated as the mean value of the difference between each pair of individuals in the archive. The difference of two individuals $\boldsymbol{\alpha}_i$ and $\boldsymbol{\alpha}_j$ is calculated as follows:

$$Diff(\boldsymbol{d}_i, \boldsymbol{d}_j) = -\text{Spearman}(Ranks(SP, \boldsymbol{d}_i), Ranks(SP, \boldsymbol{d}_j)) \tag{6}$$

where Spearman denotes the Spearman's rank correlation coefficient or Spearman's rho [28] between $Ranks(SP, \boldsymbol{d}_i)$ and $Ranks(SP, \boldsymbol{d}_j)$. $Ranks(SP, \boldsymbol{d}_i)(Ranks(SP, \boldsymbol{d}_j))$ denotes the performance rank vector of all individuals in SP on $\boldsymbol{d}_i(\boldsymbol{d}_j)$. If $\boldsymbol{x}_i$ is the individual in SP that peforms best on $\boldsymbol{d}$, then its rank is 1. If it performs second best, its rank is 2, and so on. When evolving SP, the DP along with DAr will be used together to evaluate the fitness of each $\boldsymbol{x}$ of SP in Algorithm 3.

B. Solution Representation and Evolutionary Operation

Suppose the considered DVRP has 44 customers and 4 vehicles. The solution representation for the DVRP is shown in Fig. 2. A solution includes two parts, the first part denotes

the permutation of the indices of customers and the second part denotes the number of customers that are allocated to each vehicle. Moreover, n_i does not only denote the number of customers but also the route for each vehicle i. For each example, n_1 denotes that vehicle 1 will serve the first n_1 customers from the permutation part and the route among these customers is the same as the permutation, n_2 denotes that vehicle 2 will serve the following n_2 customers with the route from the permutation part, and so on. The demand representation for the DVRP is shown in Fig. 3. Each d_i denotes the demand of customer i.

Note that the objective of this paper is not about the most efficient encoding for the VRP, but about a novel approach to solve dynamic VRPs when the environmental changes are unknown in advance and may change fast. The key research question is: how do we deal with fast unknown changes in dynamic VRPs when we do not have enough time to run a time-consuming optimization or re-optimization algorithm like an EA. Our key contribution is a novel combination of offline competitive co-evolution and fast online greedy search. The reason why we use the encodings in Figs. 2 and 3 in this work is that it is easy to implement when evolutionary operations are involved.

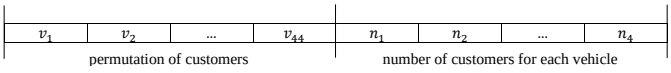

Fig. 2. Solution Representation for DVRP

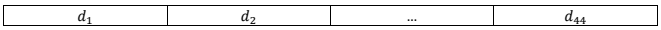

Fig. 3. Demand Parameter Representation for DVRP

In the set search process, two mutation operations and one crossover operation are used to evolve the solution set. The first mutation operation is to randomly select two customers from the permutation part and switch them. The second mutation operation is to randomly select two elements n_i and n_j and then increase one by 1 and decrease the other one by 1. In the crossover operation, we select two individuals from the population as the parents and then exchange the first part of them, e.g. the permutation of the indices of customers to generate two offsprings.

The evolution of environmental parameter is conducted based on the demands of customers. For this evolutionary process, Gaussian mutation is applied. After obtaining the solution set, for each change of the DVRP, the evolution starts with the solution set, and only mutation operation is conducted on each solution to perform a local search in every generation.

C. Online Optimization Process

In the online optimization process, only the two mutation operations are applied to the current population for a local search. When doing local search, an offspring is generated for each individual in the population. After evaluating the newly generated offspring individuals, comparsions are made between each pair of parent and offspring individuals. The one with the lower cost values are allowed to enter the next generation.

IV. Experimental Study

We generated the problem instance based on a stationary benchmark, F-n45-k4, which was taken from a real-life application, i.e. grocery deliveries from the Peterboro terminal [29]. In F-n45-k4, there are 44 customers and 4 vehicles. For each vehicle, its capacity is 2010. The range for the demands of the first 20 customers is [200, 250] and the range for the last 24 customers is [85, 126]. This is to keep the number of vehicles required unchanged. The number of change is 10. In each change, the demand for each customer was randomly generated from its given range. The objective of this problem is to minimize the total cost of 4 routes. For each route, the load of the corresponding vehicle can not exceed its capacity. The cost of each route is defined as the total distance of that route multiplied by half of the total load for the vehicle that serves this route.

To evaluate the performance of setDVR on DVRP, we made experiments to compare setDVP on the generated DVRP instances with two other methods, one is Restart-EA and the other is EA with random immigrants (RI-EA). For Restart-EA, it re-initializes the popualtion with randomly generated individuals each time a change happens, and then mutation and crossover operations are applied to evolve the population until the next problem change. EAs with random immigrants have shown promising results on dynamic VRP [20]. For RI-EA, each time the problem changes, half of the population is initialized with randomly generated individuals while the second half with the current population. Adding random immigrants is a commonly used method in the field of DOPs as well as DVRPs [29].

We randomly generate 10 cases for the DVRP instance. For each test case, three settings for the change frequency were considered. They are 500, 200 and 100 objective function evaluations (FEs). The population size for setDVR, Restart-EA and RI-EA was set to 10. The mutation rate for setDVR, Restart-EA, RI-EA was set to 1.0 and the crossover rate was set to 0.5. The maximum number of generations for the solution set search process in setDVR was set to 200. The size of solution set obtained offline was set to 20. The best half of them was used to initialize the population in setDVR each time the problem changes. Each algorithm has been run for 30 times on each case to allow statistical analysis. The Wilcoxon ranksum test at a confidence level of 0.05 was applied to make comparisons among the three algorithms.

We recorded the total cost of the best routes obtained by each algorithm at each generation on each case. Tables I,II and III summarizes the average and standard deviation of the cost of the best routes over every generation and each run under the change frequency of 500 FEs, 200FEs and 100FEs, respectively. The best results were marked in bold. It can be

observed that setDVR generally performed better than Restart-EA and RI-EA on each of the 10 cases.

TABLE I
STATISTICS (MEAN AND STANDARD DEVIATION) OF ROUTE COST FOR RESTART-EA, RI-EA AND SETDVR OVER 30 RUNS ON EACH TEST CASE UNDER THE CHANGE FREQUENCY OF 500FES. THE BEST RESULTS ARE MARKED IN **BOLD**.

Function	Restart-EA	RI-EA	setDVR
case-1	1.80e+06±2.16e+04	1.38e+06±6.17e+04	**1.20e+06±7.95e+04**
case-2	1.81e+06±1.59e+04	1.37e+06±7.20e+04	**1.20e+06±8.37e+04**
case-3	1.80e+06±1.91e+04	1.38e+06±6.06e+04	**1.19e+06±7.86e+04**
case-4	1.80e+06±2.02e+04	1.36e+06±5.99e+04	**1.20e+06±8.16e+04**
case-5	1.79e+06±1.71e+04	1.37e+06±7.78e+04	**1.19e+06±7.94e+04**
case-6	1.80e+06±1.80e+04	1.34e+06±5.03e+04	**1.20e+06±8.09e+04**
case-7	1.78e+06±2.06e+04	1.34e+06±7.30e+04	**1.19e+06±7.99e+04**
case-8	1.78e+06±1.98e+04	1.35e+06±6.64e+04	**1.18e+06±8.02e+04**
case-9	1.78e+06±1.82e+04	1.35e+06±5.10e+04	**1.19e+06±8.02e+04**
case-10	1.80e+06±1.75e+04	1.37e+06±4.63e+04	**1.19e+06±7.96e+04**

TABLE II
STATISTICS (MEAN AND STANDARD DEVIATION) OF ROUTE COST FOR RESTART-EA, RI-EA AND SETDVR OVER 30 RUNS ON EACH TEST CASE UNDER THE CHANGE FREQUENCY OF 500FES. THE BEST RESULTS ARE MARKED IN **BOLD**.

Function	Restart-EA	RI-EA	setDVR
case-1	1.92e+06±2.28e+04	1.63e+06±6.44e+04	**1.20e+06±8.00e+04**
case-2	1.92e+06±2.25e+04	1.64e+06±6.77e+04	**1.21e+06±8.46e+04**
case-3	1.91e+06±2.58e+04	1.60e+06±6.57e+04	**1.20e+06±7.96e+04**
case-4	1.91e+06±2.37e+04	1.62e+06±4.65e+04	**1.20e+06±8.53e+04**
case-5	1.90e+06±2.11e+04	1.60e+06±6.55e+04	**1.19e+06±7.97e+04**
case-6	1.92e+06±2.15e+04	1.62e+06±5.77e+04	**1.20e+06±8.36e+04**
case-7	1.90e+06±2.25e+04	1.59e+06±6.98e+04	**1.19e+06±8.18e+04**
case-8	1.89e+06±2.20e+04	1.59e+06±5.84e+04	**1.19e+06±8.17e+04**
case-9	1.90e+06±1.89e+04	1.59e+06±6.69e+04	**1.20e+06±8.23e+04**
case-10	1.91e+06±2.16e+04	1.62e+06±6.18e+04	**1.20e+06±8.07e+04**

TABLE III
STATISTICS (MEAN AND STANDARD DEVIATION) OF ROUTE COST FOR RESTART-EA, RI-EA AND SETDVR OVER 30 RUNS ON EACH TEST CASE UNDER THE CHANGE FREQUENCY OF 500FES. THE BEST RESULTS ARE MARKED IN **BOLD**.

Function	Restart-EA	RI-EA	setDVR
case-1	1.97e+06±2.49e+04	1.80e+06±7.37e+04	**1.20e+06±8.08e+04**
case-2	1.98e+06±2.27e+04	1.77e+06±7.41e+04	**1.21e+06±8.43e+04**
case-3	1.97e+06±2.54e+04	1.75e+06±7.16e+04	**1.20e+06±8.09e+04**
case-4	1.97e+06±2.42e+04	1.78e+06±5.72e+04	**1.20e+06±8.39e+04**
case-5	1.96e+06±2.45e+04	1.76e+06±6.35e+04	**1.20e+06±7.95e+04**
case-6	1.97e+06±2.20e+04	1.76e+06±7.14e+04	**1.21e+06±8.42e+04**
case-7	1.95e+06±2.58e+04	1.76e+06±6.20e+04	**1.19e+06±8.20e+04**
case-8	1.95e+06±2.61e+04	1.77e+06±5.48e+04	**1.19e+06±8.24e+04**
case-9	1.96e+06±2.23e+04	1.76e+06±6.49e+04	**1.20e+06±8.30e+04**
case-10	1.97e+06±2.56e+04	1.75e+06±6.88e+04	**1.20e+06±8.03e+04**

Figs. 4, 5 and 6 show the evolution curves for Restart-EA, RI-EA and setDVR on one representative case under the change frequency of 500FEs, 200FEs, and 100 FEs, respectively. It can be seen that the performance of Restart-EA and RI-EA deteriorates as the problem changes faster and faster. In contrast, setDVR always react quickly to demand changes. This demonstrate the potential of the proposed setDVR method.

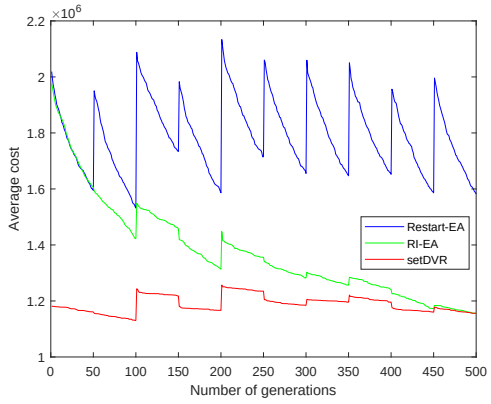

Fig. 4. Evolutionary Curves of Restart-EA, RI-EA and setDVR on case 5 under the change frequency of 500FEs

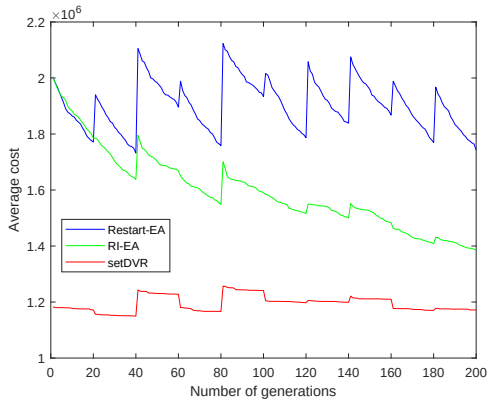

Fig. 5. Evolutionary Curves of Restart-EA, RI-EA and setDVR on case 5 under the change frequency of 200FEs

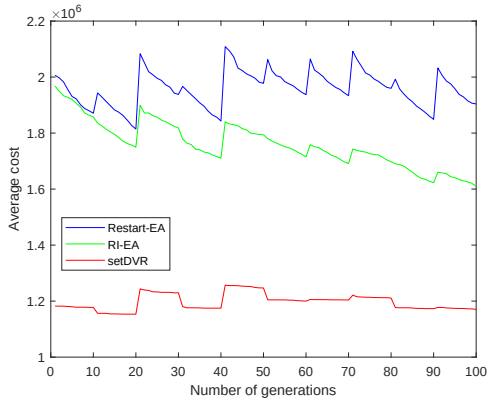

Fig. 6. Evolutionary Curves of Restart-EA, RI-EA and setDVR on case 5 under the change frequency of 100FEs

The advantage of setDVR attributes to the offline co-evolutionary set search process. As the offline set search process needs additional computational resources, in total, setDVR needs higher computational overhead than the compared algorithms. However, this allows setDVR to react to environmental changes and output high-quailty solutions much faster, this is an essential requirement in real-time DVRPs. Thus, the setDVR is still a promising method for DVRPs, which also represents a new mode in designing algorithms for DVRPs.

V. Conclusions and Future Work

In this work, we proposed a new dynamic optimization method based on competitive co-evolution, called setDVR, to solve dynamic VRP with the change of customers' demands. It maintains a solution set which is obtained through competitive co-evolution at the beginning, and conducts online local search by using this solution set as initial population once a change of DVRP happens. To evaluate the efficiency of setDVR, experiments have beeen made on 10 DVRP instances. Comparisons have been made among setDVR, Restart-EA and RI-EA. The comparison results highlights that the efficiency of the setDVR method is superior among all tested algorithms.

In future work, we will test the performance of setDVR on more test instances and modify it to solve DVRP with changing numbers and positions of customers. The success of setDVR mainly depends on the solution set obtained by competitive co-evolution. Therefore, the setDVR method is more suitable for DVRPs in which a limited set of solutions exist for all possible changes. In future work, we will investigate whether a solution set of limited size exist for which optimal solutions for any given environmental change can be found by doing local search on them for a DVRP and whether competitive co-evolution is the most efficient method to find such a solution set.

Acknowledgment

This work was partially supported by Honda Research Institute Europe (HRI-EU), the National Natural Science Foundation of China (Grant No. 61906082), 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] K. Braekers, K. Ramaekers, and I. Van Nieuwenhuyse, "The vehicle routing problem: State of the art classification and review," *Computers & Industrial Engineering*, vol. 99, pp. 300–313, 2016.

[2] P. Toth and D. Vigo, *Vehicle routing: problems, methods, and applications*. SIAM, 2014.

[3] A. M. F. AbdAllah, D. L. Essam, and R. A. Sarker, "On solving periodic re-optimization dynamic vehicle routing problems," *Applied Soft Computing*, vol. 55, pp. 1–12, 2017.

[4] Y. Mei, X. Li, and X. Yao, "Cooperative coevolution with route distance grouping for large-scale capacitated arc routing problems," *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 3, pp. 435–449, 2013.

[5] R. Shang, Y. Wang, J. Wang, L. Jiao, S. Wang, and L. Qi, "A multi-population cooperative coevolutionary algorithm for multi-objective capacitated arc routing problem," *Information Sciences*, vol. 277, pp. 609–642, 2014.

[6] J. Wang, K. Tang, J. A. Lozano, and X. Yao, "Estimation of the distribution algorithm with a stochastic local search for uncertain capacitated arc routing problems," *IEEE Transactions on Evolutionary Computation*, vol. 20, no. 1, pp. 96–109, 2015.

[7] Y. Liu, Y. Mei, M. Zhang, and Z. Zhang, "A predictive-reactive approach with genetic programming and cooperative coevolution for the uncertain capacitated arc routing problem," *Evolutionary computation*, vol. 28, no. 2, pp. 289–316, 2020.

[8] V. Pillac, M. Gendreau, C. Guéret, and A. L. Medaglia, "A review of dynamic vehicle routing problems," *European Journal of Operational Research*, vol. 225, no. 1, pp. 1–11, 2013.

[9] M. Elhassania, B. Jaouad, and E. A. Ahmed, "Solving the dynamic vehicle routing problem using genetic algorithms," in *2014 International Conference on Logistics Operations Management*. IEEE, 2014, pp. 62–69.

[10] A. Larsen, O. B. Madsen, and M. M. Solomon, "Recent developments in dynamic vehicle routing systems," in *The vehicle routing problem: Latest advances and new challenges*. Springer, 2008, pp. 199–218.

[11] G. Ghiani, F. Guerriero, G. Laporte, and R. Musmanno, "Real-time vehicle routing: Solution concepts, algorithms and parallel computing strategies," *European Journal of Operational Research*, vol. 151, no. 1, pp. 1–11, 2003.

[12] L. Zhou, L. Feng, A. Gupta, Y.-S. Ong, K. Liu, C. Chen, E. Sha, B. Yang, and B. W. Yan, "Solving dynamic vehicle routing problem via evolutionary search with learning capability," in *Proceedings of the 2017 IEEE Conference on Evolutionary Computation*, 2017.

[13] D.-C. Dang and P. K. Lehre, "Efficient optimisation of noisy fitness functions with population-based evolutionary algorithms," in *Proceedings of the 2015 ACM Conference on Foundations of Genetic Algorithms XIII*, 2015, pp. 62–68.

[14] H. G. Cobb, "An investigation into the use of hypermutation as an adaptive operator in genetic algorithms having continuous, time-dependent nonstationary environments," DTIC Document, Tech. Rep., 1990.

[15] T. T. Nguyen and X. Yao, "Continuous dynamic constrained optimization–the challenges," *IEEE Transactions on Evolutionary Computation*, vol. 16, no. 6, pp. 769–786, 2012.

[16] M. Campos and R. Krohling, "Bare bones particle swarm with scale mixtures of gaussians for dynamic constrained optimization," in *Proceedings of the 2014 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2014, pp. 202–209.

[17] T. T. Nguyen, S. Yang, and J. Branke, "Evolutionary dynamic optimization: A survey of the state of the art," *Swarm and Evolutionary Computation*, vol. 6, pp. 1–24, 2012.

[18] X. Lu, K. Tang, and X. Yao, "Speciated evolutionary algorithm for dynamic constrained optimisation," in *International Conference on Parallel Problem Solving from Nature*. Springer, 2016, pp. 203–213.

[19] R. Montemanni, L. M. Gambardella, A. E. Rizzoli, and A. V. Donati, "Ant colony system for a dynamic vehicle routing problem," *Journal of Combinatorial Optimization*, vol. 10, no. 4, pp. 327–343, 2005.

[20] M. Mavrovouniotis and S. Yang, "Ant algorithms with immigrants schemes for the dynamic vehicle routing problem," *Information Sciences*, vol. 294, pp. 456–477, 2015.

[21] N. R. Sabar, A. Bhaskar, E. Chung, A. Turky, and A. Song, "A self-adaptive evolutionary algorithm for dynamic vehicle routing problems with traffic congestion," *Swarm and evolutionary computation*, vol. 44, pp. 1018–1027, 2019.

[22] W. D. Hillis, "Co-evolving parasites improve simulated evolution as an optimization procedure," *Physica D: Nonlinear Phenomena*, vol. 42, no. 1-3, pp. 228–234, 1990.

[23] C. D. Rosin and R. K. Belew, "New methods for competitive coevolution," *Evolutionary computation*, vol. 5, no. 1, pp. 1–29, 1997.

[24] J. J. Grefenstette *et al.*, "Genetic algorithms for changing environments," in *PPSN*, vol. 2, 1992, pp. 137–144.

[25] H. Richter, "Memory design for constrained dynamic optimization problems," in *Applications of Evolutionary Computation*. Springer, 2010, pp. 552–561.

[26] P. Filipiak and P. Lipinski, "Infeasibility driven evolutionary algorithm with feed-forward prediction strategy for dynamic constrained optimization problems," in *Applications of Evolutionary Computation*. Springer, 2014, pp. 817–828.
[27] M. Mavrovouniotis and S. Yang, "Ant colony optimization with immigrants schemes for the dynamic vehicle routing problem," in *European Conference on the Applications of Evolutionary Computation*. Springer, 2012, pp. 519–528.
[28] F. Schmid and R. Schmidt, "Multivariate conditional versions of spearman's rho and related measures of tail dependence," *Journal of Multivariate Analysis*, vol. 98, no. 6, pp. 1123–1140, 2007.
[29] M. Mavrovouniotis and S. Yang, "Ant algorithms with immigrants schems for the dynamic vehicle routing problem," *Information Sciences*, vol. 294, pp. 456–477, 2015.