# An Evolutionary Algorithm with Heuristic Longest Cycle Crossover for Solving the Capacitated Vehicle Routing Problem

Thammarsat Visutarrom and Tsung-Che Chiang
Department of Computer Science and Information Engineering
National Taiwan Normal University
Taipei, Taiwan, R.O.C.
thammarsat@gmail.com, tcchiang@ieee.org

**<< This paper is included in the Proceedings of IEEE Congress on Evolutionary Computation (CEC 2019), held in New Zealand during Jun. 11-13, 2019. >>**

***Abstract*—Crossover is one of the most important parts of an evolutionary algorithm (EA) for solving optimization problems. Many crossover operators have been proposed for solving the capacitated vehicle routing problem (CVRP), a classical NP-hard problem in the field of operations research. This paper aims to improve the search ability of the cycle crossover (CX). The longest cycle selection and the nearest neighbor heuristic are utilized to improve the performance. Experimental results show that the proposed heuristic longest cycle crossover (HLCX) outperforms the original CX and four other operators. Additionally, we apply a search reduction strategy in the local refinement procedure to reduce the computation time at a little cost of solution quality.**

**Keywords—evolutionary algorithm, capacitated vehicle routing problem, cycle crossover, cycle length, nearest neighbor**

## I. Introduction

The vehicle routing problem (VRP) [1] was first introduced by Dantzig and Wright in 1959 [2] as an NP-hard combinatorial optimization problem [3]. The VRP is to manage a fleet of vehicles to serve customers under a set of constraints to optimize the concerned objectives, for example, the total travel distance. Solving the VRP well is helpful for companies to reduce the transportation cost. There are many VRP variants [4], such as the VRP with time windows (VRPTW) and the multi-depot VRP (MDVRP), depending on the constraints and objectives.

The capacitated VRP (CVRP) is a classical variant of VRP. There are $M$ homogeneous vehicles and $N$ customers. The demand of customer $i$ is denoted by $q_i$. The maximum capacity ($Q$) of each vehicle defines the main problem constraints. The total load, i.e., the sum of demand of served customers of a vehicle should not exceed $Q$. There is a single depot, and each vehicle starts and ends the route at the depot. Fig. 1 illustrates an example of a CVRP with nine customers and a solution with three vehicles. The distance between customers $i$ and $j$ is denoted by $d_{ij}$. The decision variable $x_{ijk}$ is 1 if and only if the vehicle $k$ travels from customer $i$ to customer $j$. The objective is to minimize the total travel distance of all vehicles. The mathematical model of this problem can be formulated as follows.

$$f: \min \sum_{i=0}^{N} \sum_{j=0}^{N} \sum_{k=1}^{M} d_{ij} x_{ijk} \quad (1)$$

Subject to

$$\sum_{i=1}^{N} x_{i0k} = \sum_{i=1}^{N} x_{0ik} \le 1, \quad k = 1, 2, \dots, M \quad (2)$$

$$\sum_{k=1}^{M} \sum_{i=1}^{N} x_{0ik} \le M \quad (3)$$

$$\sum_{i=0}^{N} \sum_{j=0}^{N} q_i x_{ijk} \le Q, \quad k = 1, 2, \dots, M \quad (4)$$

$$\sum_{i=0}^{N} x_{ijk} = \sum_{i=0}^{N} x_{jik}, \quad k = 1, 2, \dots, M, j = 1, 2, \dots, N \quad (5)$$

$$\sum_{k=1}^{M} \sum_{i=0}^{N} x_{ijk} = \sum_{k=1}^{M} \sum_{i=0}^{N} x_{jik} = 1, j = 1, 2, \dots, N \quad (6)$$

The transportation cost is calculated by the total travel distance defined in (1). Constraints (2) guarantee that the routes start and end at the depot. The number of vehicles is limited in constraint (3). Constraints (4) guarantee that the load of each vehicle does not exceed the maximum capacity. Each vehicle must leave after visiting each customer, and each customer can be visited only one time by a single vehicle, modeled by constraints (5) and (6), respectively.

The rest of this paper is organized as follows. Related research is presented in the next section. Section III describes thoroughly the proposed algorithm. Experiments and results are discussed in Section IV. Finally, the paper is concluded in the last section.

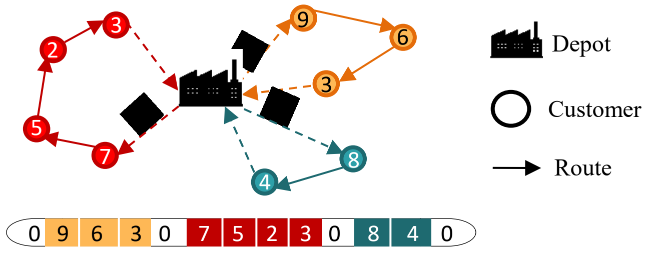

Fig. 1. An example of a CVRP with nine customers and a solution with three vehicles

## II. Literature Review

The CVRP attracts many researchers after it was introduced. Many approaches including exact, approximation, and metaheuristic methods have been proposed in solving the problem [5]. Long computation time required by exact methods makes many researchers turn to apply the approximation and metaheuristic methods [6]. As this paper utilizes the metaheuristic method in solving the CVRP, related studies applying metaheuristics are reviewed in this section.

Tabu search (TS) is one of the first metaheuristic methods utilized to solve NP-hard optimization problems. There are many papers about using it to deal with the CVRP. Toth and Vigo [7] combined TS with granular neighborhoods that used only a small number of “promising” moves. Their experimental results showed that their strategy was able to provide good solutions within acceptable computation time. Cordeau and Maischberger [8] introduced the parallel iterated tabu search together with heuristics to search for the optimal solution for different routing problems. They concluded that using heuristics could achieve better performance than the use of TS alone.

Simulated annealing (SA) is another popular algorithm for handling the CVRP. Xiao *et al.* [9] presented SA for handling different-sized CVRPs. They integrated variable neighborhood search (VNS) into SA to generate neighboring solutions and improved the algorithm performance. SA has also been used to solve other types of VRPs such as the VRPTW [10] and the CVRP for fuel optimization [11].

Multi-agent optimization algorithms have also been studied. Bin *et al.* [12] proposed an ant-weight strategy combined with swap and 2-opt exchange operators to improve the search performance of ant colony optimization (ACO). Szeto *et al.* [13] presented an artificial bee colony (ABC) algorithm together with several mutation operators including random swaps and inversion for handling the CVRP. The results showed that ABC could perform efficiently.

Evolutionary algorithms (EAs), which are characterized by crossover, mutation, and selection, have shown good search ability in solving optimization problems. Prins [14] proposed an EA for solving medium- and large-scale CVRP, and the optimal split method was utilized to decode the permutation-encoded chromosomes into routes. The linear order crossover and local search were employed. The algorithm found the best-known solutions of many problem instances with little computation time. Nazif and Lee [15] proposed a genetic algorithm (GA), in which an optimized crossover together with inversion and swap sequence operators were used. They showed that the optimized crossover could be one of the competitive operators. Baker and Ayechew [16] applied the neighborhood search with GA to improve search ability and found that their solutions were better than the solutions of TS and SA.

Crossover is the main operator to generate new (and better) solutions in EAs. Several classical crossover operators, for example, two-point crossover [17], partially mapped crossover (PMX) [18], and order crossover (OX) [19], have been proposed to solve combinatorial optimization problems in the early times. They are still consistently used, and meanwhile researchers [15][20] continued improving the performance of classical operators and developing new operators.

In addition to the reproduction operator, the initialization procedure is also another important part that can help EAs find high-quality solutions faster. There are many approaches to generate the initial solutions. The randomization technique is the simplest way and was adopted in many papers [15][21][22]. However, the randomization technique usually generates solutions with low quality. Thus, some better techniques were proposed. For example, Wang and Lu [19] clustered customers into groups based on polar angles and combined the concept of the nearest addition method and the sweep algorithm to generate initial solutions. Shanmugam *et al.* [23] generated half of the initial population by the nearest neighbor heuristic and the other half by randomization. Instead of clustering customers based only on polar angles, Tiwari and Chang [24] also considered customer demand during clustering.

## III. Proposed Algorithm

In this paper, the EA is utilized for solving the CVRP. The whole process of our EA is presented in Table I. Chromosome encoding is explained in subsection A. The initialization procedure and chromosome decoding are thoroughly described in subsections B and C, respectively. Brief descriptions of the fitness calculation are given in subsection D. Duplicate elimination technique is presented in subsection E. Subsection F elaborates our proposed heuristic longest cycle crossover, and subsection G details the local refinement procedure. Subsections H and I describe the mutation operator and the 2-opt operator, respectively.

TABLE I. THE PROPOSED EVOLUTIONARY ALGORITHM

```
Notations
$P$: the current population
$O$: the set of offspring solutions
$CT$: the set of the best $2 \cdot N_P$ solutions found so far
$p$: parent
$N_P$: population size
$I_{Max}$: Maximum number of iterations
$BS$: Best-found solution

Initialization($P$, $N_P$)
for $i = 1$ to $I_{Max}$
    $O \leftarrow \emptyset$
    $P \leftarrow$ DuplicateRemoval($P$)
    for $j = 1$ to $N_P/2$
        $\{p_1, p_2\} \leftarrow$ TournamentSelection($P$)
        $O \leftarrow O \cup$ Crossover($p_1$, $p_2$)
    end
    $CT \leftarrow$ Update($P \cup O \cup CT$)            %Keep the best $2N_P$ solutions
    for $j = 1$ to 3
        $CT \leftarrow$ LocalRefinement($CT$)
    end
    $P \leftarrow$ EnvironmentalSelection($CT$)    %Return the best $N_P$ solutions
    $P \leftarrow$ Mutation($P$)
    if $i = I_{Max}$
        $BS \leftarrow$ FindBest($P$)               %Take the best solution in $P$
        $BS \leftarrow$ 2-Opt($BS$)
    end
end
return $BS$
```

## A. *Chromosome Encoding*

A chromosome encodes a sequence of customers, which represents the order in which customers are visited by vehicles. Each element inside the sequence is an integer number from 1 to the number of customers ($N$). The sequence will be decoded into a set of routes, as illustrated in Fig. 1, where 0 denotes the depot and a subsequence between depots represents a route. The detailed steps to decode a customer sequence into a set of routes will be described in subsection C.

## B. *Initialization*

Customers are separated into groups based on the polar angle. The number of groups is fixed as eight, and the angle of each group is 360°/8 = 45°. The first group is located between [0°, 45°), the second group is between [45°, 90°), and so on. Two customer sequences are generated by putting groups of customers clockwise and counterclockwise. Such a concept makes customers that are close geographically also close in the sequence. The first group that is put to the customer sequence is selected randomly, and its customers are also ordered randomly. This helps to increase the diversity of initial solutions. Fig. 2 illustrates the idea of angle-based clustering and two examples of initial solutions.

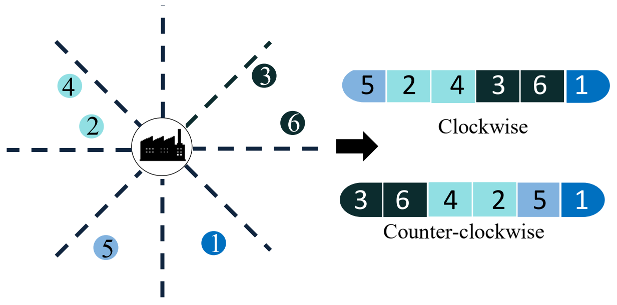

Fig. 2. Clustering of customers and generation of initial solutions

### C. *Chromosome Decoding*

The decoding procedure consists of two steps: vehicle assignment and customer sequencing. Each step is described as follows.

*1) Vehicle assignment:* this step starts with the first vehicle and checks whether it can serve unassigned customers without violating the capacity constraint (see constraints (4)) following the customer sequence. Taking the clockwise sequence in Fig. 2 as an example, customer 5 is checked at first, then customers 2, 4, 3, 6, and 1. If the vehicle can serve a customer, the customer is assigned to the vehicle; otherwise, the next customer is checked. If there is at least one unassigned customer after checking the whole customer sequence, the same procedure is repeated with a new vehicle.

*2) Customer sequencing:* After assigning customers to vehicles, we need to determine the route of each vehicle, i.e., the order in which each vehicle serves its customers. The greedy insertion heuristic is applied to construct the routes. This step starts with an empty route for each vehicle. Then, the customers served by each vehicle are inserted one by one in the order the customers appear into the chromosome sequence. Each customer is inserted in the best position, which is the position that incurs the minimal extra distance.

### D. *Fitness Function*

There are many ways to measure the cost in the CVRP, such as total travel time, fuel consumption, the number of vehicles used, and so on. In this paper, total travel distance is the objective function (see equation (1)) and is also taken as the fitness function. The smaller the fitness value is, the better the chromosome is.

### E. *Duplicate Elimination*

Population diversity is important for an EA to perform with full potential. The existence of duplicate solutions might lead the search to get stuck. Here, solutions are regarded as duplicate if they have the same fitness value. These duplicate solutions will be replaced by randomly generated new solutions.

### F. *Crossover*

The cycle crossover (CX) is one of the most well-known crossover operators in solving sequencing problems [25][26]. However, only a few papers about the VRP considered the CX operator [27][28]. From our preliminary experiments, CX did not perform well when it was compared with other classical operators like heuristic crossover (HX) or PMX. The results motivated us to investigate how to improve the performance of CX. Two improving strategies are proposed in this paper – the *longest cycle selection* and the *nearest neighbor heuristic*. The longest cycle selection, as the name indicates, picks up the longest cycle of customers (the largest-sized set of customers whose union of gene positions is the same in both parents) from the parents to relocate in the offspring. It intends to cause a sufficiently large difference between parents and offspring. On the other hand, the nearest neighbor heuristic relocates the chosen customers by selecting the nearest neighbor. It intends to keep the solution quality (i.e. to keep short travel distance) during the big change. We call our improved operator the heuristic longest cycle crossover (HLCX). Detailed steps are described in the following:

1. Select an unmarked customer $i$ randomly from one parent, say $p_1$. Mark $i$ in $p_1$, and let the position of $i$ be $pos(i)$.
2. Identify the customer $j$ in the position $pos(i)$ in the other parent, say $p_2$. Mark $j$ in $p_2$.
3. Search for the customer $j$ in $p_1$. Let $i = j$. Mark $i$ in $p_1$, and let the position of $i$ be $pos(i)$.
4. Repeat steps 2 and 3, and the procedure ends with a cycle when we meet a marked customer in $p_1$ again.
5. Repeat steps 1-4 to find all cycles.
6. Select the longest cycle and copy all customers in the cycle to a *reloc* array.
7. Copy the whole customer sequence from parents to the offspring, except the customers in the longest cycle. Thus, there will be some empty positions.
8. Fill in the leftmost empty position by choosing from the *reloc* array the customer that is closest to the previous customer in the sequence. (See the rightmost part in Fig. 3, where customer 9 is chosen to fill in position 4.) Remove the chosen customer from the *reloc* array.
9. Repeat step 8 until the *reloc* array is empty.

Note that in step 8 there is a special case. If the first position is empty, the customer from the *reloc* array that is closest to the next customer is chosen since the first position has no previous customer (See the middle part in Fig. 3). In case the second position is also empty, the customer in the position of the parent is temporarily used as the next customer.

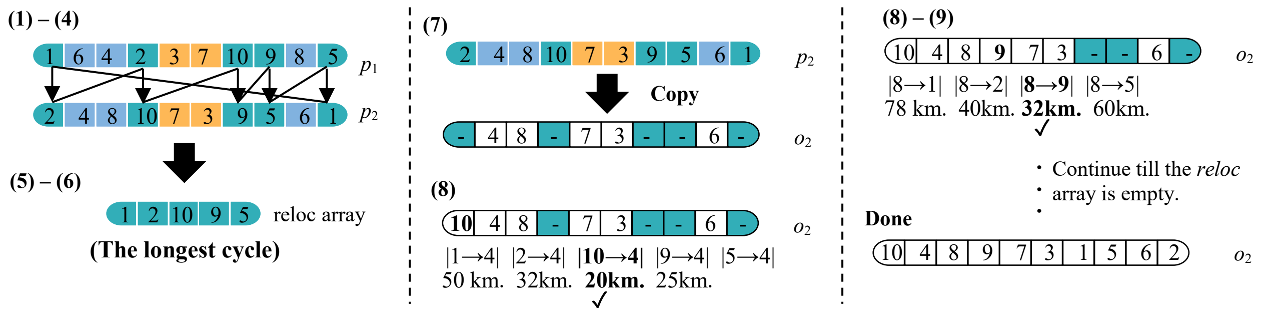

Fig. 3. Heuristic Longest Cycle Crossover (HLCX)

## G. *Local Refinement*

EAs are known for their high exploration ability but also for the lack of exploitation ability. Thus, a local refinement is added to our algorithm in every generation to improve search ability. Our idea is to refine high-quality solutions and avoid useless computation due to doing the same operator to the same solution. Table II presents the pseudo code of the local refinement procedure.

TABLE II. LOCAL REFINEMENT PROCEDURE

```
Notations
R: the set of better solutions
CT: the set of the best 2·N_P solutions found so far
N_P: population size
f: objective function, i.e., total distance
NB_i: neighborhood function, NB_1 is NEH, NB_2 is swap, and NB_3 is ejection.

for i = 1 to 3
    R ← ∅
    j ← N_P
    for e in CT in ascending order of total distance
        if all customers in e have been used in local refinement
            continue
        for x in NB_i(e)
            if f(x) < f(e)
                R ← R ∪ {x}
        j ← j – 1
        if j = 0
            break
    end
    CT = Update(R ∪ CT)
end
return CT
```

We maintain *CT*, the set of the best $2 \cdot N_P$ solutions found during the search process (no two solutions in *CT* have the same total distance). We check the solutions in *CT* one by one in ascending order of total distance. Whenever a neighborhood function is applied to a solution *e*, an unused customer is chosen randomly and marked as used. If all customers in *e* are already used, skip the solution *e*. Neighboring solutions are stored in *R* if they are better than *e*. Three neighborhood functions, NEH [29], swap, and ejection, are adopted. Fig. 4 illustrates how these three neighborhood functions work. After one neighborhood function is applied to $N_P$ solutions successfully, we change to the next neighborhood function. Once all three neighborhood functions are finished, the local refinement procedure is done. Detailed steps of the neighborhood functions are described in the following.

NEH: First, an unused customer *i* is chosen randomly. Then, the set *A* of the top $\alpha$% customers that are closest to *i* is identified. Finally, neighboring solutions are generated by relocating *i* in all positions before and after each customer in *A*.

Swap: First, an unused customer *i* is chosen randomly. Let *r* denote the route to which *i* belongs. Then, the set *A* of the top $\alpha$% customers that are closest to *i* from a different route *r*′ is identified. Each customer *j* in *A* is swapped with *i*, and *j* and *i* are relocated in the best position in *r* and *r*′, respectively. Note that *i* and *j* can be swapped only when the capacity constraints are not violated in *r* and *r*′.

Ejection: First, an unused customer *i* is chosen randomly. If *i* is not among the top 10% customers with the highest demand, skip doing ejection to *e*. Let *r* denote the route to which *i* belongs. A different route *r*′ is chosen randomly. A set *B* is initialized to be empty. Customers *j* in *r*′ are added to *B* one by one from the highest demand to the lowest demand if the demand of *j* is not greater than the demand of *i*. The set *B* stops growing when the total demand of customers in *B* is not smaller than the demand of *i* or when all customers in *r*′ are added to *B*. Finally, the set *B* of customers are swapped with customer *i* if the total demand of *B* is equal to the demand of *i*. The set *B* of customers are relocated into the best positions in *r*, and customer *i* is relocated into the best position in *r*′.

If the total demand of $B$ is not equal to the demand of $i$, another route is chosen randomly as $r'$ again. If it still fails to find a set $B$ with equal demand to $i$, skip doing ejection to the solution $e$.

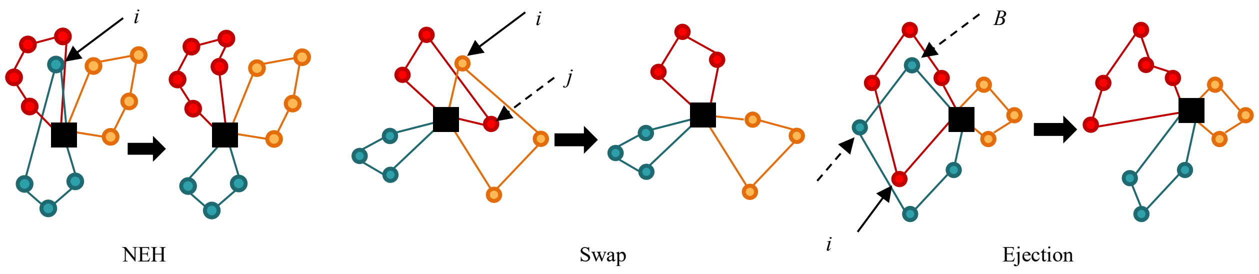

Fig. 4. Local Refinement

## *H. Mutation*

The mutation operator maintains population diversity and prevents the EA from getting stuck at the local optimum by adding (random) perturbation to the solutions. Except the best 30 solutions, 10% of the solutions in the population are selected randomly to apply the swap operator, which exchanges two random customers without violating the capacity constraint.

## *I. 2-opt*

The 2-opt operator is applied only to the best solution in the population at the last iteration. We aim to use 2-opt to remove the crosses in the route to reduce the travel distance. For each route $r$ in the solution, we start from the first link $(r_1, r_2)$ and check whether it crosses any of the following links $(r_2, r_3)$, $(r_3, r_4)$, …, until the last link. The symbol $r_i$ refers to the $i^{\text{th}}$ customer in the route $r$. If no cross is found, we move to the next link $(r_2, r_3)$ and check whether it crosses any of the following links $(r_3, r_4)$, $(r_4, r_5)$, and so on. When a cross is found between $(r_i, r_{i+1})$ and $(r_j, r_{j+1})$, we do 2-opt to reverse the sub-route $[r_i, r_{i+1}, …, r_j, r_{j+1}]$ to $[r_i, r_j, …, r_{i+1}, r_{j+1}]$. If the new solution (with the partially reversed route) is better, it replaces the original solution. In this case, we move to link $(r_{j+1}, r_{j+2})$ and check the crosses between it and its following links. If the new solution is not better, we undo the 2-opt and keep checking whether $(r_i, r_{i+1})$ crosses $(r_{j+1}, r_{j+2})$, $(r_{j+2}, r_{j+3})$, and so on.

## IV. EXPERIMENTS AND RESULTS

## *A. Benchmark Problems and Parameter Setting*

The performance of the proposed algorithm was tested by the E set of CVRPLIB [1], which was extensively used in the literature. The problem set contains 11 problem instances. The number of customers ranges from 20 to 100, and the number of vehicles ranges from 3 to 14.

In all experiments, both the population size and the generation number were set by 100. The crossover rate and mutation rate were 1.0 and 0.1, respectively. In the first experiment, six EA variants were tested by using different crossover operators. Mutation, local refinement, and 2-opt were not included so that we could focus on the difference performance caused by crossover operators. In the second experiment, all operators were included. We examined whether the proposed HLCX still helped to improve algorithm performance, and we also want to know the quality gap between solutions found by our algorithm and the best-known solutions. In the last experiment, we investigated how we can save computation time and maintain solution quality by limiting the neighborhood size in the local refinement procedure. In all experiments, each tested algorithm variant solved each problem instance for thirty times.

## *B. Experiment 1: Crossover-Only EA*

The HLCX was compared with five well-known operators: heuristic crossover (HX), partially mapped crossover (PMX), traditional cycle crossover (CX), one-point crossover (1PX), and two-point crossover (2PX). In Table III, best-known solution, BKS, presents the optimal travel distance of each instance obtained from [1]. For each EA variant, *Min* denotes the minimal total distance found over 30 runs, *%Gap* denotes $(Min - BKS)/BKS \times 100\%$, and *Avg* denotes the average total distance over 30 runs.

Table III shows that the proposed HLCX outperforms other five crossover operators in terms of both best-case and average-case performance for 10 out of 11 problem instances. The traditional CX is usually the worst performer, but our proposed longest cycle selection and nearest neighbor heuristic are able to turn it to be the best performer. When larger-scale problem instances are solved, the performance gap can be improved by 25.4% (49.3 − 23.9 = 25.4) to 35.1% (59.0 − 23.9 = 35.1). Fig. 5 shows the average convergence curves of six EA variants for solving the E-108-k8 instance. In the early stage, HX performs better than HLCX does; however, HX gets stuck and is outperformed by HLCX after 30 generations. We also examined the performance of LCX and HCX, i.e. the CX with only longest cycle selection and the CX with only nearest neighbor heuristic. The results showed that HLCX performs better than LCX and HCX, which means that both proposed ideas are effective. Since crossover is not the computational bottleneck of the whole EA, the extra computational effort required by HLCX only slightly increases the total computation time of the whole EA by no more than 5%.

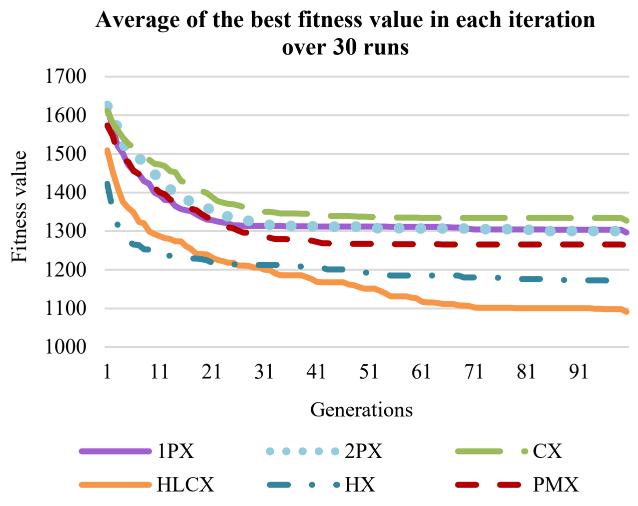

Fig. 5. The average convergence curves of EAs with six crossover operators for solving the E-108-k8 problem instance

## C. *Experiment 2: Complete EA*

This experiment is similar to the previous one, but all operators (mutation, local refinement, and 2-opt) were included. Results are summarized in Table IV. By applying all operators, all six EA variants can find the best-known solutions for the first five smaller-scale instances. For the remaining six larger-scale instances, the EA variant using HLCX found better solutions than the other five EA variants except one instance (E-N76-K14). Again, the advantage of the proposed HLCX is confirmed.

## D. *Experiment 3: Search Reduction Strategy*

Section III-G mentioned a search reduction strategy when the NEH and swap neighborhood functions are applied – only the top $\alpha$% closest customers are considered. In the last experiment, we examined the influences of the strategy. We compared the computation time and solution quality of two EA variants with $\alpha = 100$ and $\alpha = 30$. Table III summarizes the results. Time and TD refer to the average computation time and average total travel distance, respectively. Note that the EA variant with $\alpha = 30$ here is the same as the EA variant using HLCX in the previous experiment. The results show that the search reduction strategy not only saves computation time (up to around 25%) but also keeps high solution quality.

TABLE III. EFFECT OF THE SEARCH REDUCTION STRATEGY

| | α = 100 | | α = 30 | | ΔTime | ΔTD (%) |
|---|---|---|---|---|---|---|
| | Time(s) | TD | Time(s) | TD | (%) | |
| E-n22-k4 | 19.4 | 375.0 | 17.8 | 375.0 | -8.44 | 0.00 |
| E-n23-k3 | 22.0 | 569.0 | 21.6 | 569.0 | -1.70 | 0.00 |
| E-n30-k3 | 29.4 | 535.8 | 29.1 | 541.6 | -1.30 | 1.08 |
| E-n33-k4 | 32.7 | 835.8 | 31.6 | 835.0 | -3.46 | -0.10 |
| E-n51-k5 | 57.0 | 525.2 | 48.8 | 521.0 | -14.35 | -0.80 |
| E-n76-k7 | 123.1 | 702.8 | 99.9 | 697.1 | -18.85 | -0.81 |
| E-n76-k8 | 123.8 | 755.1 | 103.5 | 743.3 | -16.37 | -1.56 |
| E-n76-k10 | 107.9 | 865.7 | 87.3 | 854.2 | -19.10 | -1.33 |
| E-n76-k14 | 98.3 | 1056.2 | 83.8 | 1053.2 | -14.79 | -0.28 |
| E-n101-k8 | 287.6 | 844.5 | 216.7 | 837.0 | -24.67 | -0.89 |
| E-n101-k14 | 240.0 | 1130.9 | 198.4 | 1124.7 | -17.31 | -0.55 |

## V. CONCLUSIONS

This paper addresses the CVRP by EAs and proposes a crossover operator based on the CX operator. CX is a classical crossover operator of EAs in solving combinatorial optimization problems but did not show good performance in solving the CVRP. Two ideas, the longest cycle selection and the nearest neighbor heuristic, are proposed to improve its performance. The longest cycle selection aims to increase the modifications to the offspring, and the nearest neighbor heuristic intends to utilize domain knowledge to improve the quality of the offspring. By comparing the proposed HLCX operator with five crossover operators using 11 public problem instances, we confirmed that the HLCX could provide very good performance. This research will continue with two directions: first, we will keep improving our algorithm for solving multiobjective and large-scale CVRP instances; second, we will investigate the performance of the proposed HLCX in solving other combinatorial optimization problems.

## REFERENCES

[1] G. Laporte, “The vehicle routing problem: an overview of exact and approximate algorithms,” *European Journal of Operational Research*, vol. 59, no. 3, pp. 345-358, 1992.

[2] G. B. Dantzig and J. H. Ramser, “The truck dispatching problem,” *Management Science*, vol. 6, no. 1, pp. 80-91, 1959.

[3] J. K. Lenstra and A. Kan, “Complexity of vehicle routing and scheduling problems,” *Networks*, vol. 11, no. 2, pp. 221 – 227, 1981.

[4] B. Eksioglu, A. V. Vural, and A. Reisman, “The vehicle routing problem: A taxonomic review,” *Computers & Industrial Engineering*, vol. 57, no. 4, pp. 1472-1483, 2009.

[5] A. Hoff, H. Andersson, M. Christiansen, G. Hasle, and A. Løkketangen, “Industrial aspects and literature survey: fleet composition and routing,” *Computers & Operations Research*, vol. 37, no. 12, pp. 2041-2061, 2010.

[6] C. Prins, P. Lacomme, and C. Prodhon. “Order-first split-second methods for vehicle routing problems: a review,” *Transportation Research Part C*, vol. 40, pp. 179-200, 2014.

[7] P. Toth and D. Vigo, “The granular tabu search and its application to the vehicle routing problem,” *INFORMS Journal on Computing*, vol. 15, no. 4, pp. 333–346, 2003.

[8] J.-F. Cordeau and M. Maischberger, “A parallel iterated tabu search heuristic for vehicle routing problems,” *Computers & Operations Research*, vol. 39, no. 9, pp. 2033–2050, 2012.

[9] Y. Xiao, Q. Zhao, I. Kaku, and N. Mladenovic, “Variable neighbourhood simulated annealing algorithm for capacitated vehicle routing problems,” *Engineering Optimization*, vol. 46, no. 4, pp. 562–579, 2013.

[10] R. Baños, J. Ortega, C. Gil, A. Fernández, and F. de Toro, “A simulated annealing-based parallel multi-objective approach to vehicle routing problems with time windows,” *Expert Systems with Applications*, vol. 40, no. 5, pp. 1696-1707, 2013.

[11] Y. Xiao, Q. Zhao, I. Kaku, and Y. Xu, “Development of a fuel consumption optimization model for the capacitated vehicle routing problem,” *Computers & Operations Research*, vol. 39, no. 7, pp. 1419 – 1431, 2012.

[12] Y. Bin, Z. Z. Yang, and Y. Baozhenb, “An improved ant colony optimization for vehicle routing problem,” *European Journal of Operational Research*, vol. 196, no. 1, pp. 171–176, 2009.

[13] W. Y. Szeto, Y. Wu, and S. C. Ho, “An artificial bee colony algorithm for the capacitated vehicle routing problem,” *European Journal of Operational Research*, vol. 215, no. 1, pp. 126-135, 2011.

[14] C. Prins, “A simple and effective evolutionary algorithm for the vehicle routing problem,” *Computers & Operations Research*, vol. 31, no. 12, pp. 1985-2002, 2004.

[15] H. Nazif and L. S. Lee, “Optimised crossover genetic algorithm for capacitated vehicle routing problem,” *Applied Mathematical Modelling*, vol. 36, no. 2, pp. 2110-2117, 2012.

[16] B. M. Baker and M. A. Ayechew, “A genetic algorithm for the vehicle routing problem,” *Computers & Operations Research*, vol. 30, no. 2, pp. 787-800, 2003.

[17] K. Ganesh and T. T. Narendran, “CLOVES: A cluster-and-search heuristic to solve the vehicle routing problem with delivery and pick-up,” *European Journal of Operational Research*, vol. 178, no. 3, pp. 699-717, 2007.

[18] J. S. Wang, L. Chang, and Z. Ying, “Solving capacitated vehicle routing problem based on improved genetic algorithm,” *Proceedings of 2011 Chinese Control and Decision Conference (CCDC)*, pp. 60-64, Mianyang, China, 2009.

[19] C. H. Wang and J. Z. Lu, “A hybrid genetic algorithm that optimizes capacitated vehicle routing problems,” *Expert Systems with Applications*, vol. 36, no. 2, pp. 2921–2936, 2009.

[20] T. Tlili, F. Chicano, S. Krichen, and E. Alba, “Evolutionary algorithm based on partition,” *Proceedings of the 2015 International Conference on Intelligent Networking and Collaborative Systems*, pp. 169-175, Taipei, Taiwan, 2015.

[21] Z. Ursani, D. Essam, D. Cornforth, and R. Stocker, “Localized genetic algorithm for vehicle routing problem with time windows,” *Applied Soft Computing*, vol. 11, no. 8, pp. 5375–5390, 2011.

[22] P. C. Pop, O. Matei, and C. P. Sitar, “An improved hybrid algorithm for solving the generalized vehicle routing problem,” *Neurocomputing*, vol. 109, pp. 76–83, 2013.

[23] G. Shanmugam, P. Ganesan, and P. T. Vanathi, “Meta heuristic algorithms for vehicle routing problem with stochastic demands,” *Journal of Computer Science*, vol. 7, no. 7, pp. 533-542, 2011.

[24] A. Tiwari and P.-C. Chang, “A block recombination approach to solve green vehicle routing problem,” *International Journal of Production Economics*, vol. 164, pp.379 – 387, 2015.

[25] I. M. Oliver, D. J. Smith, and J. R. C. Holland, “A study of permutation crossover operators on the traveling salesman problem,” *Proceedings of the Second International Conference on Genetic Algorithms*, Massachusetts Institute of Technology, Cambridge, Massachusetts, USA pp. 224-230, 1987.

[26] W. Chinnasri, S. Krootjohn, and N. Sureerattanan, “Performance comparison of genetic algorithm's crossover operators on university course timetabling problem,” *Proceedings of the 2012 8th International Conference on Computing Technology and Information Management (NCM and ICNIT)*, Seoul, South Korea, pp. 781-786, April 2012.

[27] G. N. Yücenur and N. Ç. Demirel, “A new geometric shape-based genetic clustering algorithm for the multi-depot vehicle routing problem,” *Expert Systems with Applications*, vol. 38, no. 9, pp. 11859–11865, Sep. 2011.

[28] C.-C. Lu and V. F. Yu, “Data envelopment analysis for evaluating the efficiency of genetic algorithms on solving the vehicle routing problem with soft time windows,” *Computers & Industrial Engineering*, vol. 63, no. 2, pp. 520–529, 2012.

[29] M. Nawaz, J. Enscore, and I. Ham, “A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem,” *Omega-The International Journal of Management Science*, vol. 11, no. 1, pp. 91 – 95, 1983.

[30] CVRPLIB, http://vrp.atd-lab.inf.puc-rio.br/index.php/en/, accessed on 2018/12/20.

TABLE IV. PERFORMANCE COMPARISON OF SIX CROSSOVER-ONLY EAS

| | BKS | HLCX | | | HX | | | PMX | | | CX | | | 1PX | | | 2PX | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | [30] | Min | %Gap | Avg | Min | %Gap | Avg | Min | %Gap | Avg | Min | %Gap | Avg | Min | %Gap | Avg | Min | %Gap | Avg |
| E-N22-K4 | 375 | 384 | 2.4 | 389.8 | **381** | 1.6 | **381.0** | 384 | 2.4 | 398.3 | 400 | 6.7 | 417.1 | 384 | 2.4 | 413.4 | 409 | 9.1 | 429.5 |
| E-N23-K3 | 569 | **594** | 4.4 | **597.1** | **594** | 4.4 | 597.6 | 596 | 4.7 | 597.9 | 596 | 4.7 | 597.5 | 594 | 4.4 | 600 | 596 | 4.7 | 599.8 |
| E-N30-K3 | 534 | **539** | 0.9 | **543.1** | 539 | 0.9 | 544.4 | 542 | 1.5 | 558.0 | 546 | 2.2 | 565.7 | 542 | 1.5 | 558.4 | 560 | 4.9 | 582.9 |
| E-N33-K4 | 835 | **836** | 0.1 | **844.5** | 842 | 0.8 | 850.5 | 844 | 1.1 | 872.5 | 872 | 4.4 | 902.6 | 865 | 3.6 | 899.4 | 862 | 3.2 | 904.5 |
| E-N51-K5 | 521 | **549** | 5.4 | **573.8** | 587 | 12.7 | 602.0 | 578 | 10.9 | 619.8 | 647 | 24.2 | 669.8 | 588 | 12.9 | 660.4 | 616 | 18.2 | 643.9 |
| E-N76-K7 | 682 | **776** | 13.8 | **816.4** | 886 | 29.9 | 900.6 | 888 | 30.2 | 938.3 | 936 | 37.2 | 999.2 | 918 | 34.6 | 995.5 | 851 | 24.8 | 948.3 |
| E-N76-K8 | 735 | **846** | 15.1 | **881.7** | 980 | 33.3 | 999.6 | 944 | 28.4 | 1012.5 | 1084 | 47.5 | 1142.1 | 1013 | 37.8 | 1079 | 986 | 34.1 | 1061.0 |
| E-N76-K10 | 830 | **906** | 9.2 | **985.9** | 1088 | 31.1 | 1130.4 | 1142 | 37.6 | 1195.5 | 1202 | 44.8 | 1278.0 | 1246 | 50.1 | 1281 | 1142 | 37.6 | 1254.8 |
| E-N76-K14 | 1021 | **1154** | 13.0 | **1219.0** | 1325 | 29.8 | 1400.3 | 1434 | 40.5 | 1551.2 | 1495 | 46.4 | 1609.4 | 1528 | 49.7 | 1597 | 1451 | 42.1 | 1541.0 |
| E-N101-K8 | 815 | **1001** | 22.8 | **1090.9** | 1132 | 38.9 | 1164.3 | 1173 | 43.9 | 1263.4 | 1266 | 55.3 | 1327.0 | 1241 | 52.3 | 1296 | 1197 | 46.9 | 1296.2 |
| E-N101-K14 | 1067 | **1322** | 23.9 | **1437.9** | 1600 | 50.0 | 1625.0 | 1593 | 49.3 | 1709.4 | 1671 | 56.6 | 1774.1 | 1696 | 59.0 | 1833 | 1669 | 56.4 | 1765.3 |

TABLE V. PERFORMANCE COMPARISON OF SIX COMPLETE EAS

| | BKS | HLCX | | | HX | | | PMX | | | CX | | | 1PX | | | 2PX | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | [30] | Min | %Gap | Avg | Min | %Gap | Avg | Min | %Gap | Avg | Min | %Gap | Avg | Min | %Gap | Avg | Min | %Gap | Avg |
| E-N22-K4 | 375 | **375** | 0 | 375.0 | **375** | 0 | 375.0 | **375** | 0 | 375.0 | **375** | 0 | 375.7 | **375** | 0 | 375.0 | **375** | 0 | 375.0 |
| E-N23-K3 | 569 | **569** | 0 | 569.0 | **569** | 0 | 569.0 | **569** | 0 | 569.0 | **569** | 0 | 569.0 | **569** | 0 | 569.0 | **569** | 0 | 569.0 |
| E-N30-K3 | 534 | **534** | 0 | 541.6 | **534** | 0 | 535.5 | **534** | 0 | 534.9 | **534** | 0 | 536.4 | **534** | 0 | 535.5 | **534** | 0 | 536.6 |
| E-N33-K4 | 835 | **835** | 0 | 835.0 | **835** | 0 | 836.7 | **835** | 0 | 835.4 | **835** | 0 | 835.8 | **835** | 0 | 835.4 | **835** | 0 | 835.0 |
| E-N51-K5 | 521 | **521** | 0 | 521.0 | **521** | 0 | 526.6 | **521** | 0 | 523.4 | **521** | 0 | 523.6 | **521** | 0 | 525.2 | **521** | 0 | 523.7 |
| E-N76-K7 | 682 | **692** | 1.5 | 697.1 | 696 | 2.1 | 703.2 | 696 | 2.1 | 705.6 | 699 | 2.5 | 702.3 | 698 | 2.3 | 702.3 | 699 | 2.5 | 707.5 |
| E-N76-K8 | 735 | **737** | 0.3 | 743.3 | 745 | 1.4 | 750.2 | 744 | 1.2 | 751.4 | 741 | 0.8 | 753.7 | 740 | 0.7 | 752.9 | 748 | 1.8 | 751.3 |
| E-N76-K10 | 830 | **842** | 1.4 | 854.2 | 848 | 2.2 | 860.4 | 844 | 1.7 | 859.5 | 845 | 1.8 | 860.7 | 855 | 3.0 | 865.7 | 849 | 2.3 | 862.2 |
| E-N76-K14 | 1021 | 1043 | 2.2 | 1053.2 | 1055 | 3.3 | 1066.3 | 1055 | 3.3 | 1066.4 | 1059 | 3.7 | 1070.5 | 1043 | 2.2 | 1063.6 | **1039** | 1.8 | 1062.7 |
| E-N101-K8 | 815 | **827** | 1.5 | 837.0 | 831 | 2.0 | 853.0 | 831 | 2.0 | 841.0 | 840 | 3.1 | 849.4 | 829 | 1.7 | 851.0 | 835 | 2.5 | 853.0 |
| E-N101-K14 | 1067 | **1098** | 2.9 | 1124.7 | 1113 | 4.3 | 1134.4 | 1103 | 3.4 | 1153.1 | 1131 | 6.0 | 1157.6 | 1115 | 4.5 | 1148.4 | 1125 | 5.4 | 1134.4 |