A Comparison of Evolutionary Crossover Operators for Real-World Vehicle Routing Problems with Simultaneous Pickup and Delivery with Time Windows

Yelyzaveta Denysova*, Ethan Gibbons*, Beatrice Ombuki-Berman*, Alex Bailey†
*Brock University, St. Catharines, Canada
†RideCo Inc, Waterloo, Canada

***Abstract*—The vehicle routing problem with simultaneous pickup and delivery with time windows (VRPSPDTW) represents a complex challenge in logistics that combines both nuances of routing and strict time scheduling. This task has attracted growing attention in recent years with new algorithms and methodologies. This paper investigates the performance of various evolutionary crossover operators specialized for VRPSPDTW. To create an environment as close to the real-world scenario as possible, this study utilized Liu et al. [1] instances derived from real customer data of the company JD Logistics and 2 other additional datasets. Even though the results do not show a significant distinction in most of the best solutions, there is a clear leader in performance across the board amongst all the crossover. The outcomes of this evaluation highlight a dependence on the size and complexity of the datasets. This paper not only identifies the most effective crossover for practical VRPSPDTW application but also provides insights into the features contributing to their success offering guidance for future research.**

I. Introduction

The vehicle routing problem with simultaneous pickup and delivery with time windows (VRPSPDTW) is a crucial variant of a vehicle routing problem aimed to facilitate both pick-up and delivery time within a specific time frame. This problem captures the complexity of real-world logistics and the challenges of finding optimal solutions within numerous constraints. VRPSPDTW is an optimization problem aimed at minimizing the number of vehicles required to service all customers and the total distance travelled by all vehicles in a combined weighted score. The common choice for solving this type of task is an evolutionary algorithm(EA) that is inspired by natural selection. It simulates evolution with techniques such as mutation, crossover, and selection. Best-fitted individuals are selected for reproduction, creating new solutions to improve each generation to find the absolute optimal point. In this paper, a Memetic Algorithm (MA) [2] is used, which is a type of EA that utilizes both global and local search to solve optimization problems. In MA local search procedure is applied at a specified time to improve each individual. The crossover operator plays a vital role in EA. It imitates reproduction in natural evolution. The procedure aimed to create new individuals in the population by taking their parents' best characteristics for survival. The characteristics are represented through features that allow the best optimization of the objectives. Crossover is applied in EAs to improve and diversify the initial population with each generation getting closer to the optimal solution. There were 6 unique crossovers chosen to be evaluated in this paper: Best-Cost-Route-Crossover with Route Destruction(BCRCD) [2], Route-Assembly-Regret-Insertion Crossover(RARI) [3], Route-Copy-Based Crossover(RCX) [4], Two-Stage Crossover(2STAGE) [5], Exchange Route Crossover(ERX) [6], and Pheromone-Based Crossover(PB) [7]. Additionally, there are 2 variations of the original crossovers proposed in this paper: Exchange Route Crossover with Regret Insertion(ERXR) and Two-Step Pheromone-Based Crossover(TSPB). The main contribution of this paper is extensive testing of the largest collection of VRP-specific crossovers on a wide range of datasets including recent real-world data that some of the operators have never been tested on before.

II. Background and Literature Review

Many of the current works for VRPSPDTW test their algorithms and heuristics on Wang and Chen instances [8] that were derived from previous Solomon's benchmark [9]. Wang and Chen introduced their co-evolutionary algorithm as well. In their work, a hybrid of 2 simplistic crossovers was used with two populations that were focused on diversity and improvement.

As mentioned before Liu et al. [1] introduced new data for the VRPSPDTW along with a Memetic Algorithm with efficienT local search and Extended neighbourhood, dubbed MATE. MATE had 2 main advantages over other algorithms for the same problem: effective search space exploration due to its novel initialization procedure, crossover and large-step-size operators, and effective local search due to its sophisticated constant-time-complexity move evaluation mechanism. MATE included a new crossover Route-Assembly-Regret-Insertion Crossover(RARI) which followed a similar procedure as in Wang and Chen's work [8] but introduced a new regret insertion method. This algorithm was shown to outperform previous methods and found 12 new best-known solutions for Wang and Chen instances.

A follow-up study by E. Gibbons and B. Ombuki-Berman [10] proposed a new Memetic algorithm (MA) based on MATE. The new approach reduced computationally expensive constructive heuristics while still maintaining effective search. Best-Cost-Route-Crossover with Route Destruction(BCRCD) used in this algorithm is an original alteration of a previously introduced Best-Cost-Route-Crossover(BCRC) by B. Ombuki-Berman [11]. A new MA with BCRCD found 20 new best-known solutions for the Liu instances. They have also included the comparison of the BCRCD crossover with RCX [4] and RARI [3] demonstrating that BCRCD was competitive and even outperforming on some datasets. These were all included in this research to be evaluated on a wider dataset variety.

Another paper came out on a comparison of crossover operators applied to the vehicle routing problem with time windows [6]. One distinction is the problem itself slightly varies from VRPSPDTW but also it mostly had simplistic crossover operators like ordered and partially-mapped crossover. However, they did consider a vrp specific crossover called Exchange Route Crossover (ERX) that is included in this research.

The authors in [5] have proposed two evolutionary approaches with objective-specific variation operators for vehicle routing problem with time windows and quality of service objectives. The algorithm came with a new crossover as well that was able to construct offsprings in 2 stages. The algorithm was shown to outperform state-of-the-art approaches on Solomon's instances in terms of solution quality and performance time.

Crossovers mentioned so far had a lot in common as you will see in their detailed description before but one of the approaches in literature stands out due to its uniqueness. The pheromone-based crossover was designed specifically for their innovative hybrid genetic algorithm [7] that combines characteristics of both genetic and ant colony optimization algorithms.

A. VRPSPDTW Problem Formulation

This VRPSPDTW problem formulation is equivalent to the original formulation by Liu et al. [1] adopted by E. Gibbons [10].

Given $N = \{1, 2, 3, \ldots, n\}$ set of customers, depot 0 and a set of J identical vehicles. The goal of VRPSPDTW is to create a schedule and routes for each vehicle to service all customers exactly once while trying too use the least number of vehicles and cover the least total distance as well as to comply with the constraints. The problem is defined as a complete graph G with a vertex set $V = \{0\} \cup \mathbb{N}$. Each customer i in set V is associated with 5 attributes: the amount of goods to be picked up p_i, the amount of goods to be delivered d_i, the time window for the customer to receive service a_i to b_i, and the duration of service s_i. Each arc in set E of graph G has 2 attributes: distance from customer i to customer j is c_i, j and the time it takes to travel that distance is t_i, j. A set of J vehicles with capacity Q are initially located in the depot. The vehicles leave the depot to service all customers and then return to the depot. This leads to the following representation of a solution in VRPSPDTW. A solution $S = \{R_1, R_2, R_3, \ldots, R_K\}$, consists of K routes, where $K < J$. Each route $R_k = \{i_0, i_1, i_2, \ldots, i_{H_k+1}\}$ is an ordered set of H customers to be serviced by vehicle K, where $i_0 = i_{H_k+1} = 0$ is a depot. A sum of all the deliveries on route $D_R = \sum_{h=1}^{H} d_{i_h}$ must be less than vehicles capacity $D_R < Q$. The load of the vehicle after visiting customer i_h consists of remaining goods to be delivered and the picked gooods $l_{R,i_h} = \sum_{x=h+1}^{H} d_{i_x} + \sum_{x=1}^{h} p_{i_x}$ and must also be less than vehicle's capacity $l_{R,i_h} < Q$. The service of a customer i on route R_K must begin before the end of their time window $B_{R_k+1} \leq b_i$. The vehicle could arrive to the customer earlier than the beginning of their time window but must wait in that case $B_{R,i_h} = max(B_{R,i_{h-1}} + s_{i_{h-1}} + t_{i_{h-1},i_h}, a_{i_h})$ for $h = 1, ..., H$. Additionally, the vehicle must return to the depot before its closing time T implying $B_{R_k,i_{H_k+1}} < T$.

The goal of the algorithm is to come up with the optimal solution that has a minimum score. The score is calculated as a sum of 2 weighted objectives minimizing the number of vehicles and total distance travelled by all vehicles and can be represented as follows $\min\left(u_1 \cdot K + u_2 \cdot \sum_{k=1}^{K} C_{R_k}\right)$ where u_1 is a weight for objective 1 and u_2 is the weight for objective 2, while visiting each customer exactly once $\sum_{i \in V} \sum_{k=1}^{K} x_{i,j}^{k} = 1$ and complying with the constraints mentioned above.

B. Memetic Algorithm for VRPSPDTW

The algorithm used in this study is called MA and was proposed by authors in [10]. It is a class of population-based metaheuristics which imitates the evolutionary concept of natural selection, where the most fitted individuals are more likely to survive and influence the creation of a new generation making it evolve towards the most optimal fitness. The population is initialized with some heuristic, improved via local search, and the new solutions are constructed through a crossover operator. A detailed description of MA is provided in Algorithm 1.

C. Crossovers

The crossover operator plays a vital role in any evolutionary algorithm. It imitates reproduction in natural evolution. The procedure aimed to create new individuals in the population by taking their parents' best characteristics for survival. Crossover is applied in evolutionary algorithms in order to improve and diversify the initial population with each generation getting closer to the optimal solution.

There are many types of crossovers available and selecting the appropriate one is important as it will have a significant impact on the quality of the solution produced by the algorithm. Another important consideration to take into account is the efficiency of a crossover because according to another comparison study, "They [crossovers] are responsible for the greatest computational resource usage of the algorithm." [6]

This paper focuses on examining crossover operators that were developed specifically for VRPSPDTW or similar routing and scheduling optimization problems. Following is a detailed description of how each crossover operates.

Algorithm 1 Memetic Algorithm for the VRPSPDTW

Require: VRPSPDTW Instance, n_{pop}, f_{search}, p_{search}
1: $U \leftarrow$ set of n_{pop} empty solutions
2: Initialize solutions in U
3: Perform local search on $n_{pop} * p_{search}$ solutions in U
4: Maintain elitism
5: $g \leftarrow 1$
6: **while** termination condition is not met **do**
7: $U_{new} \leftarrow$ perform k-tournament selection on $U(k = 3)$
8: **for** each pair of solutions (P_1, P_2) in U_{new} **do**
9: $O_1, O_2 \leftarrow$ perform crossover using solutions (P_1, P_2)
10: O_1 and O_2 replace P_1 and P_2 in U_{new}
11: **end for**
12: $U \leftarrow U_{new}$
13: **if** $g \mod f_{search} = 0$ **then**
14: Perform local search on $n_{pop} \times p_{search}$ solutions in U
15: **end if**
16: Maintain elitism
17: $g \leftarrow g + 1$
18: **end while**
19: **return** Best solution found.

D. Best-Cost-Route-Crossover with Route Destruction

Given 2 parent solutions P_1 and P_2, the routes R_1 and R_2 are chosen at random from P_1 and P_2 respectively. The customers in R_1 are removed from P_2 and vice versa. Then, the routes R_1 and R_2 are removed from P_1 and P_2 respectively. In the end, unassigned customers are reinserted via the cheapest insertion. Figure 1 illustrates the process.

E. Route-Assembly-Regret-Insertion Crossover

Given 2 parent solutions P_1 and P_2, a new empty offspring solution O is created. Routes R_1 and R_2 are repeatedly chosen from P_1 and P_2 respectively and inserted into O until there are no more inheritable routes. Remaining unassigned customers are reinserted via regret-based insertion. The regret value represents the cost of inserting a node in the future iteration. Algorithm 2 outlines the general procedure of the RARI crossover (as in [3]) and you can refer for more details to [3].

F. Route-Copy-Based Crossover

Given 2 parent solutions P_1 and P_2, each route R in P_1 is copied over to the new empty offspring solution O with 0.5 probability. Unassigned customers are pooled in order from P_2 and inserted in solution O via the cheapest insertion. Figure 2 illustrates the process.

G. Two-Stage Crossover

Given 2 parent solutions P_1 and P_2, an empty offspring solution is created. In the first stage, parent R is chosen at random repeatedly and the most promising route R is copied to O, then the route R is discarded from R and customers

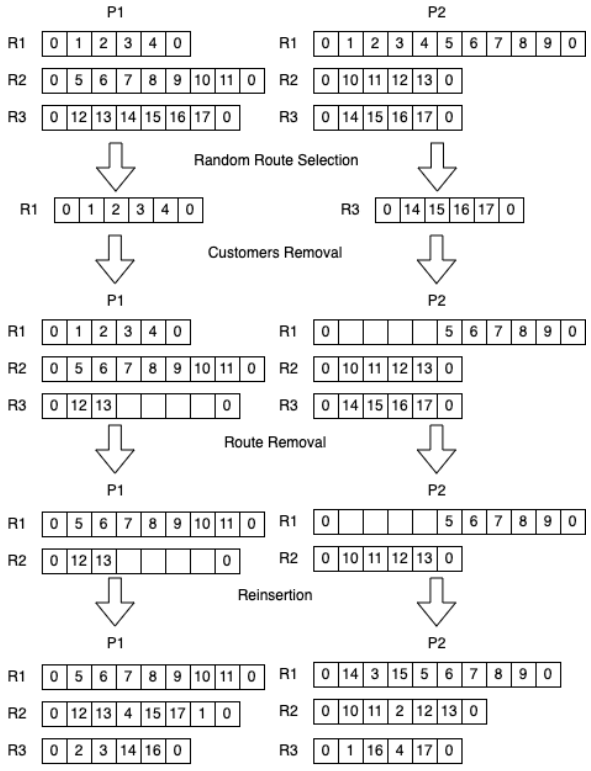

Fig. 1. BCRCD

Algorithm 2 The RARI Crossover Operator

input : parent solutions p_1, p_2
output: x_{child}
$x_{\text{child}} \leftarrow$ initialize an empty solution with no routes;
repeat
copy random route from p_1 to x_{child};
copy random route from p_2 to x_{child};
until no more inherited routes are feasible
/ regret-based insertion */*
$U \leftarrow$ all the remaining unassigned customers;
while $U \neq \emptyset$ **do**
for each node $v \in U$ **do**
calculate regret(v) according to Eq. (5);
end for
$v^* \leftarrow \arg\max_{v \in U}$ regret(v);
Insert v^* into x_{child} at its best insertion position;
$U \leftarrow U \setminus \{v^*\}$;
end while
return x_{child}

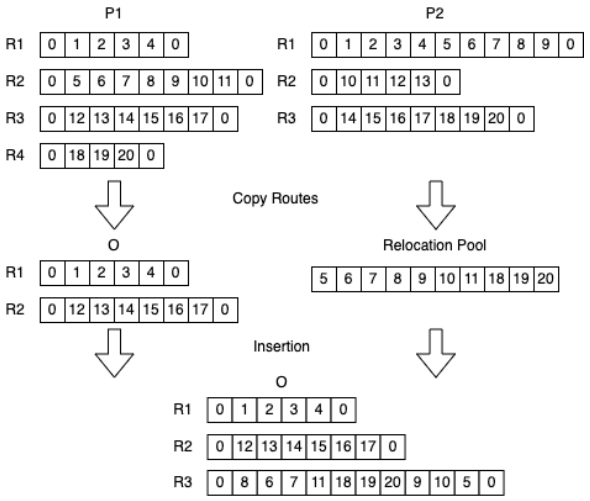

Fig. 2. RCX

in R are discarded from the other parent. In the second stage, unallocated customers are reinserted via the cheapest insertion. Figure 3 illustrates the process.

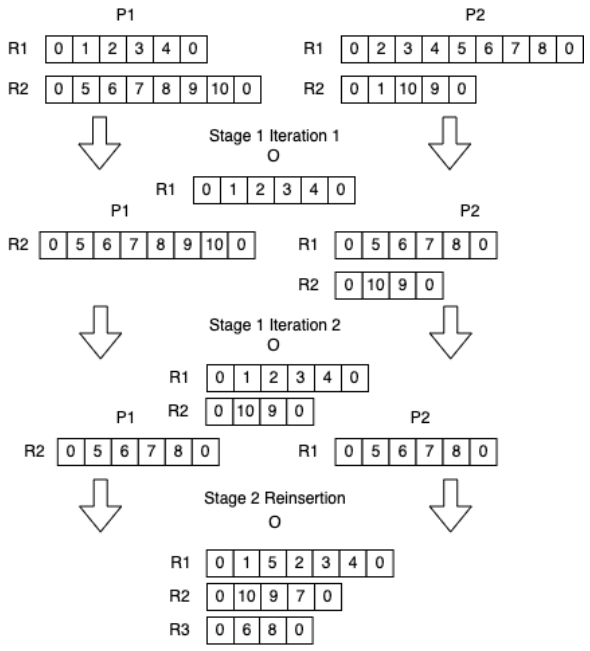

Fig. 3. 2STAGE

H. Exchange Route Crossover

Given 2 parent solutions P_1 and P_2, two offspring O_1 and O_2 created by copying P_1 and P_2 respectively. The best route in P_1 is inserted in O_2 and vice versa for P_2 and O_1. Duplicate customers are removed. Then the routes with few customers are removed as well as the route with the worst score. In the end, unassigned customers are reinserted via the cheapest insertion. Figure 4 illustrates the process.

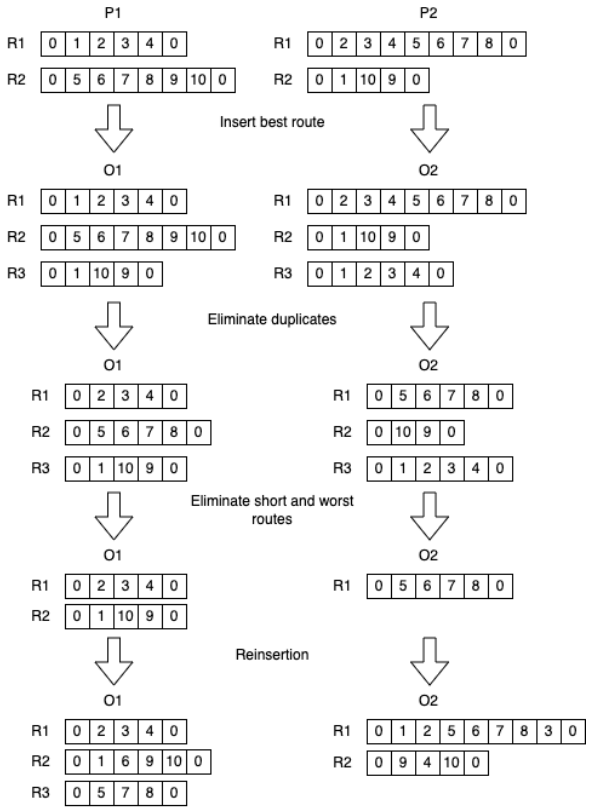

Fig. 4. ERX

I. Exchange Route Crossover Regret Insertion

Alteration of ERX crossover where regret-based insertion is used in the last step instead of the cheapest insertion. This crossover variant was tested to identify the impact of the regret-based versus cheapest algorithm in the insertion stage of the crossover operator.

J. Pheromone-Based Crossover

Given 2 parent solutions P_1 and P_2, new offspring O is constructed by utilizing local neighbourhood information in P_1 and P_2 as well as global neighbourhood information in the pheromone arcs tracking paths from all the best solutions. Algorithm 3 describes the general outline of PB crossover operator (as in [7]) and you can refer to [7] for more details.

K. Summary

Based on the process of each crossover described earlier, few distinct categories are noticeable. The first grouping is ”1 Step Constructive” VS ”2 Step Constructive”. PB crossover is in ”1 Step Constructive” and the rest is in the other category. There is a pattern in each crossover besides PB: in the first step offspring is partially constructed in some way, and in the second step the unallocated customers are inserted based on some heuristics. On the other hand, PB constructs the entire offspring using the same method.

Another category that stands out is iterative VS non-iterative and it is based on the process of constructing offspring in the

Algorithm 3 The Pheromone-Based Crossover

$i \leftarrow 1, k \leftarrow 1$
Select a random customer as initial customer, $o(i)$, of the o
while $i < n + k$ **do**
  Search $o(i)$ in parents p_1 and p_2 respectively
  if all neighbors of $o(i)$ in p_1 and p_2 have appeared in o, or all neighbors can not be feasibly added to the end of o **then**
    Generate a random decimal fraction q
    Select the next customer j according to formula 1
    if no feasible customer is found **then**
      $i \leftarrow i + 1$
      $k \leftarrow k + 1$
      $o(i) \leftarrow 0$ (depot)
      Randomly select an unvisited customer j
      $i \leftarrow i + 1$
      $o(i) \leftarrow j$
    end if
  else
    Select the nearest customer that has not appeared in o from the feasible neighbors of $o(i)$ as the next customer j
    $i \leftarrow i + 1$
    $o(i) \leftarrow j$
  end if
end while

1st step of the crossover. BCRCD and ERX are non-iterative while others are iterative because they have some sort of loop in the first step while the former ones don't. To examine the impact of this difference, an alteration to PB crossover is proposed as a Two-Step Pheromone-Based Crossover(TSPB). It follows the same process as PB crossover until 3/4 customers are allocated. Then, the rest of the nodes are placed via the cheapest insertion.

Additional grouping is possible based on the insertion technique used in the second step. Most crossovers adapted the cheapest insertion, while authors of RARI implemented a novel regret-based approach. In order to investigate the impact of this choice, a new variation of ERX crossover is proposed. Exchange Route Crossover with Regret Insertion (ERXR) is the same as ERX but with the cheapest insertion replaced by regret-based insertion.

III. Experimental Setup

All the experiments in this study were based on MA code [2] and 2STAGE, ERX, ERXR, PB, and TSPB were additionally implemented in Java. The runs were conducted on Linux machines using an Intel i7-9700 CPU at 3.00GHz. The following subsections describe the parameters settings for all tests.

A. Algorithm Parameters

These are global parameters for the algorithm set to the same reported values when testing all the crossovers.

- Generation size: 50
- Local search step: every 20 generations
- Local search rate: 0.75 (0.1 for large instances)
- Run times: 3
- Random seed: 0
- Time-out: 2 hours
- Selection: k-tournament, k=3
- Crossover probability: 1

The algorithm was set to stop if there were 300 local search steps without improvement or if the execution time was exceeding the 2-hour limit.

B. Crossover Parameters

The parameters described in this section are relevant for only specified crossovers.

- 2STAGE: m = 50%-100% of the average length of 2 parents
- ERX: cutoff length = 3
- PB: trail persistence = 0.5, random probability = 0.05
- TSPB: cheapest insertion proportion: 0.25

IV. Results

TABLE I
AVERAGE SCORE PER CUSTOMER NUMBER ACROSS DATASETS

# of Customers:	10	25	50	100	200	400	600	800	1000
BCRCD	**6348,98236**	**10551,0529**	**18994,1825**	**29720,09825**	**68362,616**	**130235,29**	**181564,806**	**209411,338**	**303619,746**
RARI	**6348,98236**	**10551,0529**	**18994,1825**	31781,96771	76900,798	153862,032	190207,2	218471,026	315802,946
RCX	**6348,98236**	**10551,0529**	19006,0735	31808,48854	77771,912	145145,17	193821,62	224805,2	321424,262
2STAGE	**6348,98236**	**10551,0529**	18997,90377	33727,73992	75732,782	144782,634	195027,162	226279,492	324312,128
ERX	**6348,98236**	**10551,0529**	18994,69958	31717,07293	71561,248	158660,242	193247,404	222141,312	316897,618
ERXR	**6348,98236**	**10551,0529**	19003,48773	31762,09306	77494,196	157555,428	190898,762	222092,2	320164,652
PB	8372,993	12606,1311	23219,61394	46402,75557	111497,472	218602,274	196540,9	225693,178	325223,48
PBE	**6348,98236**	**10551,0529**	19021,93833	33792,55687	88828,706	170361,3	195706,246	224952,648	324743,992

Table 1 depicts the best score per customer number found in 30 runs. The best score (lowest) is in bold. Figures 5-7 capture the distribution of the best scores from each of the 30 runs.

Fig. 5. All scores

Figure 5 shows the distribution of scores for all crossovers. It is clear that the pheromone-based crossover is performing significantly poorly. Due to this, figures 6 and 7 exclude the PB crossover so that the difference between other crossovers is more noticeable.

As seen from the past table and the graphs BCRCD is a clear choice to be used in the Memetic Algorithm for

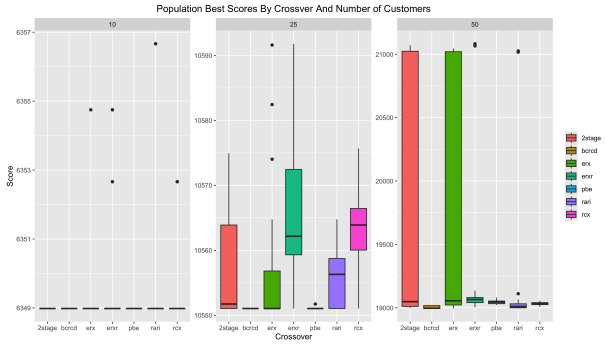

Fig. 6. Scores for crossovers without PB for smaller number of customers

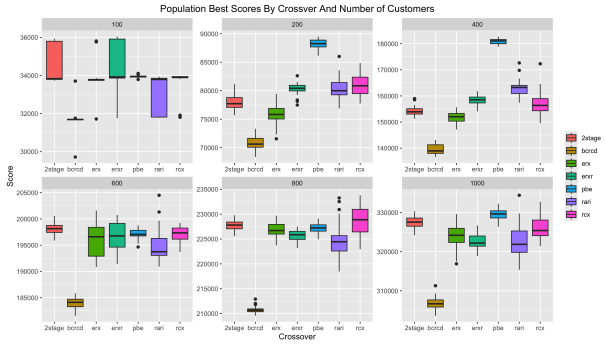

Fig. 7. Scores for crossovers without PB for bigger number of customers

solving VRPSPDTW. Not only did it outperform (or get exactly the same score) all crossovers for each customer size but it turned out to be one of the most consistent crossovers as well. As can be seen in the graphs, the small size of the box and vertical line demonstrates that most scores from 30 runs were distributed in a small range. It is worth noting that the PB crossover that performed the worst was in the separated category and each operation of this crossover can make a significant change to the genetics of 2 parents. On the other hand, BCRCD, the best-performing operator, inherited a lot of information from parents introducing little changes in the offspring. Another thing to consider is that ERX, in the same 2-step non-iterative category as BCRCD, is typically the second-best option. Non-iterative crossovers tend to inherit more information from parents without any changes, which seems to be working best for this particular problem. Comparing ERX and ERXR it seems that regret-based insertion is better suited for larger-scale instances since ERXR scores are lower than ERX for those instances. On the contrary, the cheapest insertion produced better results for the smaller datasets. As mentioned earlier PB crossover performed very poorly compared to others. However, the PBE variation that combined characteristics of both, constructive and 2-step, categories has shown a big improvement, once again highlighting the advantage of the 2-step category.

V. Conclusion

This research presents a comparison of evolutionary crossover operators applied to real-world vehicle routing problems with simultaneous pickup and delivery with time windows. Memetic Algorithm, which is a type of Genetic Algorithm with a local search procedure, was used in this study. Several crossovers specialized for problems similar to VRPSPDTW were taken from the literature and implemented as part of MA. Testing was performed for each crossover covering a variety of datasets from 10 to 1000 customers. After analyzing the results, BCRCD was shown to be the best option for crossover to be used in the Memetic Algorithm for VRPSPDTW. Several patterns were discovered indicating the advantage of 2-step type crossovers with the cheapest insertion getting better scores for smaller instances and the regret-based insertion working better with a higher number of customers.

VI. Future Work

The patterns discovered in the process of evaluating crossovers can be investigated further to introduce more improvements to the current crossovers. As shown from a comparison of ERX and ERXR, there is no clear better option between cheapest and regret-based insertion. It could be worth exploring to see if both types of insertion can be optimally used together. The cheapest insertion is considered shortsighted but it is faster, which can help to improve computational time for larger instances. On the other hand, keeping a certain portion of regret-based insertion can keep performance from deteriorating.

VII. Limitations

This paper focuses on analyzing crossover operators that are specialized for VRPSPDTW or similar problems. It is important to keep in mind that in many cases the authors developed those crossovers along with new algorithms which might be different MA but in this study for a fair comparison main algorithm was kept the same as a memetic algorithm. In this study, due to limited time and computational resources 1 dataset was chosen with each different number of customers. The original dataset collections by Wang and Chen and Liu include more than 1 instance for each size so more testing can be done. The important thing to consider when interpreting the results is that for Liu datasets of 600+ customer instances were run with a lower (0.1 compared to the usual 0.75) local search rate which might have affected the outcomes. It was done due to a 2-hour time-out so that multiple generations could be processed and local search itself does not take the entire duration.

References

[1] Shengcai Liu, Ke Tang, Xin Yao, Memetic search for vehicle routing with simultaneous pickup-delivery and time windows, Swarm and Evolutionary Computation, Volume 66, 2021, 100927, ISSN 2210-6502, https://doi.org/10.1016/j.swevo.2021.100927

[2] An Improved Memetic Algorithm for Large-Scale Real-World Vehicle Routing Problems with Simultaneous Pickup and Delivery with Time Windows, Ethan Gibbons, Beatrice Ombuki-Berman

[3] Shengcai Liu, Ke Tang, Xin Yao, Memetic search for vehicle routing with simultaneous pickup-delivery and time windows, Swarm and Evolutionary Computation, Volume 66, 2021, 100927, ISSN 2210-6502, https://doi.org/10.1016/j.swevo.2021.100927.

[4] Yongliang Lu, Una Benlic, Qinghua Wu, An effective memetic algorithm for the generalized bike-sharing rebalancing problem, Engineering Applications of Artificial Intelligence, Volume 95, 2020, 103890, ISSN 0952-1976, https://doi.org/10.1016/j.engappai.2020.103890.

[5] Gaurav Srivastava, Alok Singh, Two evolutionary approaches with objective-specific variation operators for vehicle routing problem with time windows and quality of service objectives, Applied Soft Computing, Volume 134, 2023, 109964, ISSN 1568-4946, https://doi.org/10.1016/j.asoc.2022.109964.

[6] T. M. Stehling and S. R. de Souza, "A Comparison of Crossover Operators Applied to the Vehicle Routing Problem with Time Window," 2017 Brazilian Conference on Intelligent Systems (BRACIS), Uberlandia, Brazil, 2017, pp. 300-305, doi: 10.1109/BRACIS.2017.47.

[7] Fanggeng Zhao, Dong Mei, Jiangsheng Sun and Weimin Liu, "A hybrid genetic algorithm for the vehicle routing problem with simultaneous pickup and delivery," 2009 Chinese Control and Decision Conference, Guilin, China, 2009, pp. 3928-3933, doi: 10.1109/CCDC.2009.5192035.

[8] Hsiao-Fan Wang, Ying-Yen Chen, A genetic algorithm for the simultaneous delivery and pickup problems with time window, Computers & Industrial Engineering, Volume 62, Issue 1, 2012, Pages 84-95, ISSN 0360-8352, https://doi.org/10.1016/j.cie.2011.08.018.

[9] Solomon, Marius M. "Algorithms for the Vehicle Routing and Scheduling Problems with Time Window Constraints." Operations Research, vol. 35, no. 2, 1987, pp. 254–65. JSTOR, http://www.jstor.org/stable/170697.

[10] A Memetic Algorithm for Large-Scale Real-World Vehicle Routing Problems with Simultaneous Pickup and Delivery with Time Windows. Ethan Gibbons and Beatrice M. Ombuki-Berman 15th, Metaheuristics International Conference , MIC 2024, Lorient, France, June 2024 To Appear As Springer Lecture Notes in Computer Science

[11] Ombuki, B., Ross, B.J., Hanshar, F.: Multi-objective genetic algorithms for vehicle routing problem with time windows. Applied Intelligence 24, 17–30 (2006)