Towards a Better Understanding of Genetic Operators for Ordering Optimization: Application to the Capacitated Vehicle Routing Problem

S. Ben Hamida[1], R. Gorsane[2] and K. Gorsan Mestiri[2]
[1]*LAMSADE CNRS UMR 7243, Paris Dauphine University, PSL Research University, France*
[2]*InstaDeep, Tunisia*

Keywords: Genetic Algorithms, Ordering Optimization, CVRP, Hybridization, Exploitation/Exploration.

Abstract: Genetic Algorithms (GA) have long been used for ordering optimization problems with some considerable efforts to improve their exploration and exploitation abilities. A great number of GA implementations have been proposed varying from GAs applying simple or advanced variation operators to hybrid GAs combined with different heuristics. In this work, we propose a short review of genetic operators for ordering optimization with a classification according to the information used in the reproduction step. Crossover operators could be position ("blind") operators or heuristic operators. Mutation operators could be applied randomly or using local optimization. After studying the contribution of each class on solving two benchmark instances of the Capacitated Vehicle Routing Problem (CVRP), we explain how to combine the variation operators to allow simultaneously a better exploration of the search space with higher exploitation. We then propose the random and the balanced hybridization of the operators' classes. The hybridization strategies are applied to solve 24 CVRP benchmark instances. Results are analyzed and compared to demonstrate the role of each class of operators in the evolution process.

1 INTRODUCTION

Ordering Optimization is a sub-field of combinatorial optimization consisting of searching for an arrangement solution of an ordering problem such as ordering tasks in a scheduling problem or ordering cities for the traveling salesman problem. Solutions are often encoded with an order based scheme using a relative order of the symbols in the genotype. A solution is then a sequence of n ordered objects. A genotype might be a string of numbers that represent a position (number) in the sequence. Given n unique objects, $n!$ permutations of the objects exist, which make the problem NP-Complete and hard to solve. The interest in using GA for ordering application has been going on since the 1990s. Throughout these decades, much effort has been invested to propose powerful variation operators as well as hybrid methods replacing operators with heuristic or meta-heuristic techniques. The present study focus on genetic operators for ordering optimization.

We propose in this paper a short review of variation operators for ordering optimization with GA. We then propose to classify these operators according to the information from the parent chromosomes implied in the reproduction step. Thus, for crossover operators, we distinguish between position operators, using only the position of the genes (symbols) in the chromosomes, and advanced or heuristic operators using additional information such as the neighborhood and the performance of the parents. Similarly, we classify mutation operators into two categories: Position mutation applying a random variation, and local mutation applying local optimization on the mutated solution.

Each class of operators is applied to solve some selected benchmark instances of the Capacitated Vehicle Routing Problem (CVRP). The performance of each class is analyzed according to the distance of the best solution to the optimum and the population convergence speed. We then study the combination of the different operators' classes according to the exploration/exploitation ability of each class. Otherwise, to handle constraints of the CVRP application, we propose a simple procedure based on a segregational rank that is introduced in the selection operators. This procedure gives advantage randomly to feasible or unfeasible solutions to preserve diversity in the population.

The paper is organized as follows. Section 2 reviews the different categories of variation operators

Ben Hamida, S., Gorsane, R. and Mestiri, K.
Towards a Better Understanding of Genetic Operators for Ordering Optimization: Application to the Capacitated Vehicle Routing Problem.
DOI: 10.5220/0009832704610469
In *Proceedings of the 15th International Conference on Software Technologies (ICSOFT 2020)*, pages 461-469
ISBN: 978-989-758-443-5

for order based optimization and explains the operational mode of the operators implemented for the experimental study. Section 3 formulate the CVRP problem, introduce the segregational selection for handling constraints, and summarize the general loop of the implemented GA and its parameters. The different classes of operators are studied in the section 4 through the first series of tests. Then, the hybridization strategies and their corresponding experimental study are detailed in section 5.

2 REVIEW AND CLASSIFICATION OF THE VARIATION OPERATORS

Standard order-based variation operators for GAs are essentially "blind" operators acting regardless of the fitness of the solutions to cross or mutate. For ordering optimization, "blind" crossover operators are inspired from standard binary crossovers, such as the uniform crossover (Syswerda, 1989) and the order-based crossover (OX) (Davis, 1985; M. et al., 1987; Deep and Adane, 2011) described below. We design this class of operators as "Position Crossovers" since they need only the sequences of symbols and their positions to be applied. Similarly, several mutation operators for ordering optimization change randomly one or several symbols in different positions in the chromosome. They are designed as "Position Mutation" operators. Four operators in this category are described and illustrated in section 2.3.

To increase the GA exploitation ability, a great number of advanced and heuristic crossover operators have been proposed. An advanced crossover operator considers an individual as the origin of its search and tries to produce a better solution. They are designed in this paper as "Heuristic Crossovers". They use some heuristic information from the parents to hopefully produce better offspring. The heuristic information could be the adjacency describing gene neighbors in the parents, the objective function, the neighborhood map, distance between genes, etc. Three crossovers are selected from this class for the experimental study: the Distance Preserving crossover (DPX) (Freisleben and Merz, 1996), the Alternating Edges Crossover (AEX) (Grefenstette et al., 1985) and the Edge Recombination crossover (ERX) (Whitley et al., 1989).

As for Heuristic crossover, advanced mutation operators use a local search algorithm that starts from a solution and ends up in a local minimum where no further improvement is possible. It aims at intensifying the search by exploiting search paths determined by the neighborhood of the corresponding solution (Neri et al., 2012). The classical local search λ-opt is studied in this work with λ=2 or 3. For further intensification of the search around the best area, a stochastic Hill-climbing technique is implemented as a local search operator.

The following subsection describes the variation operators implemented for the experimental study. The set of these operators and their corresponding classes are summarized in table 1.

Table 1: Variation operators' classes.

Operator Class	Selected Operators
Position Crossover (PosCross):	OX1, OX4, PMX
Heuristic Crossover (HeurCross):	DPX, AEX, ERX
Position Mutation (PosMut):	ISM, SWM, SHM, DM
Local Mutation (LocalMut):	λ-Opt(λ=2/3), Hill-Climbing

2.1 Position or "Blind" Crossovers

Position or "blind" crossovers recombine the genetic material of the parents into a new configuration without considering their initial performance in the purpose to explore new research directions. Thus, they can be seen as explorative more than exploitative operators. The oldest operator in this category is the uniform order-based crossover (Syswerda, 1989) that generates a random binary template to decide from which parent the gene is selected at each position. Several operators were proposed in the same category which main purpose is to produce feasible solutions without considering their performance, such as the order based crossover and the partially matched crossover implemented for this work.

The Order-based Crossover (OX): OX, proposed by Davis (Davis, 1985), is a variation of the uniform crossover introduced in the purpose of preserving the relative order of symbols in the sequences to be combined. The first implementation of the OX operator (OX_1) generates two cut points for both parents. The symbols between the two cut points are copied to the children. Then, starting from the second cut point of one parent, the symbols from the other parent are copied to one offspring in the same order, omitting those which already exist.

Deep and Mebrahtu (Deep and Adane, 2011) proposed three other variations of OX. In the first variant, the cut points in the two parents are at different positions, but the size of the substring between the cut points is the same for both parents. The symbols are copied to the offspring respecting the same rules defined for the original OX. This variant might also be applied with different substring sizes (called OX_4).

Partially Matched (or *Mapped*) Crossover (PMX): The PMX crossover is a well-known operator proposed by Goldberg and Lingle (Goldberg and Lin-

gle, 1985). It has two tasks: cross two parents and prevent the repetition of symbols in the offspring. It operates in two main steps: *swap* and *map*. The first step is quite similar to the OX. Two crossover points are selected randomly. The sub-strings between these two points are exchanged to create two incomplete offspring. In the second step, each offspring is completed using the remaining elements in the corresponding parent. If the element is already present, then a correction is performed according to the mapping defined with the matching section.

Other Position Crossovers: Several other crossover operators for ordering optimization could be classified as Position crossovers, such as the Maximal Preservative operator (PMX) (Mühlenbein et al., 1988), the voting recombination crossover (Mühlenbein, 1989) and the Cycle Crossover (M. et al., 1987). A description of these operators could be found in (Pétrowski and Ben-Hamida, 2017).

Otherwise, some variants for most of the position operators were proposed either for comparative purposes or to be adjusted to the handled ordering problem. For example, the Non-Wrapping Order Crossover (Cicirello, 2006) is a variant of OX and the Extend PMX (Tao, 2008) is a variant of PMX. A short review with a comparative sutdy could be found in (Karakatič and Podgorelec, 2015).

2.2 Heuristic Crossovers

Distance Preserving Crossover (DPX): DPX (Freisleben and Merz, 1996) generates an offspring close to its parents by preserving some common sub-strings. It starts by detecting common sub-paths of two parents using the Hamming distance. Then, it uses a greedy method to reconnect them and produce a child.

Alternating Edges Crossover (AEX): AEX (Grefenstette et al., 1985) considers a genotype as a directed cycle of arcs, where an arc is defined by two successive symbols. To create a child from two parents, arcs are chosen in alternation from the two parents, with some additional random choices in case of unfeasibility. AEX starts by choosing the arc $(g_1 \rightarrow g_2)$ from the first parent. Next, the arc having as first gene g_2 in the second parent is selected which the second gene is added to the child. AEX proceeds in the same way until the child is complete.

Genetic Edge Recombination Crossover (ERX): ERX (Whitley et al., 1989) considers a genotype as an edge and it is based on the gene adjacency. The aim is to keep as many neighbor as possible in the offspring. ERX starts by building the "edge-map" that gives for each vertex the set of edges that start or finish on it.

To produce an offspring, ERX starts from a random symbol designed as current symbol x_c. It finds the element in the neighbor set of x_c having the lower neighbor set cardinality. This element is added to the child, removed from the "edge-map" and becomes the new current element. The process continues until all entries are visited.

Other Heuristic Crossovers: Grefenstette proposed in (Grefenstette, 1987) a set of Heuristic crossover for the TSP: HGreX, HRndX, HpPoX. As AEX, HGreX considers the genotype as a directed cycle of arcs. The Inver-Over operator (Tao and Michalewicz, 1998) can be classified in the same category as it considers a solution as a directed graph. It invert some substring from a given parent in the hope to improve the fitness.

2.3 Position or "Blind" Mutation

Random or "blind" mutation is widely used for combinatorial order-based optimization for explorative purpose and to maintain the population diversity (Pétrowski and Ben-Hamida, 2017). Four operators are selected and implemented for this work: exchange mutation or swap mutation (SWM) (M. et al., 1987), displacement mutation (Michalewicz, 1992) (DM), shifting mutation (SHM) (Fogel, 1988) and insertion mutation (ISM). SWM selects two random positions along the parent string and exchange the corresponding symbols. SHM chooses randomly two positions on the parent genotype. Then, the symbol in the first position is shifted from its current position to the second position. DM displaces a randomly selected subsequence in the parent genotype.

Note that several other mutation operators having the same goal have been proposed for ordering optimization such as the scramble operator (Syswerda, 1991) that selects a random sub-sequence in the parent genotype and scrambles its symbols, or the inversion mutation operator (Fogel, 1988) that inverts a randomly selected sub-string in the parent genotype.

2.4 Local Mutation

The idea of introducing a local search technique in the GA engine for combinatorial problems is supported by several researchers since many decades ago. Several works in the literature demonstrated that such hybridization improves the exploitation of best solutions and might speed convergence to the optimal or to a near-optimal solution (Neri et al., 2012). We present below two simple techniques to implement a local search in the mutation step, the λ-Opt local search and the stochastic Hill-climbing.

λ-Opt Mutation: λ-Opt is a simple local search algorithm. The main idea behind it is to take a route that crosses over itself and reorder it so that it does not. It works as follows. λ edges are removed from the tour of the corresponding solution. The remaining segments are reconnected in all possible ways. If any improvement is identified, then the corresponding movements are applied to the solution.

Stochastic Hill-Climbing: The simplest way to perform a local search in a combinatorial space is to perturb a trial solution S and to replace it by the new solution if it performs better. This procedure is known as Hill-Climbing (HC). It tries to explore the neighborhood of a given solution and to move it to the eventually best direction. HC can be applied using the random genetic mutation operators to generate solutions in the neighborhood of the trial solution. However, for an ordering problem, the neighborhood is very large and a full exploration is impossible. The stochastic Hill-Climbing is widely applied in this case. It picks a single random neighbor and accepts it if it is better than S. The algorithm terminates upon a computational limit.

3 APPLICATION TO THE CAPACITATED VEHICLE ROUTING PROBLEM

Ordering optimization problems in the real world are generally submitted to a set of constraints. For this reason, we chose for our study the Capacitated Vehicle Routing Problem (CVRP) described in the following subsection. The rest of the section describes the GA implemented for the experimental study.

3.1 CVRP Formulation

The vehicle routing problem (VRP) is an important problem class in the field of operations research and transport logistics optimization. Its original formulation has been defined over 60 years ago by Dantzig and Ramser(Dantzig and Ramser, 1959) and consists of a fleet of identical vehicles serving a set of customers with a certain demand from a single depot and having a certain capacity. This formulation is referred to as the Capacitated Vehicle Routing Problem (CVRP).

The mathematical formulation of the CVRP can be defined as follows: let $G = (N,A)$ be an undirected graph, where N is the set of nodes $N = \{0,1,\cdots,n\}$ and $E = \{(i,j) : i,j \in N, i \neq j\}$ is the set of edges joining the nodes. Node 0 refers to the depot from which the vehicles start from and comeback to. The other nodes represent the customers having each a known non-negative demand q_i for customer i. A set of K vehicles $N = \{V_1,V_2,\cdots,V_k\}$ having each a maximum capacity Q is provided. The travel distance between node i and j is defined by $d_{ij} > 0$.

The CVRP can be stated as follows:

$$min \sum_{i=0}^{N} \sum_{j=0}^{N} \sum_{k=1}^{K} d_{ij} X_{i,j}^{k} \tag{1}$$

Where

$$X_{i,j}^{k} = \begin{cases} 1 & \text{if } V_k \text{ travels from node i to node j} \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

Subject to:

$$\sum_{k=1}^{K} \sum_{i=0}^{N} X_{i,j}^{k} = 1, j \in \{1 \cdots N\}, i \neq j \tag{3}$$

$$\sum_{k=1}^{K} \sum_{j=0}^{N} X_{i,j}^{k} = 1, i \in \{1 \cdots N\}, i \neq j \tag{4}$$

$$\sum_{i=0}^{N} \sum_{j=0}^{N} X_{i,j}^{k} R_i \leq Q_k, k \in \{1 \cdots K\} \tag{5}$$

$$\sum_{j=1}^{N} X_{0,j}^{k} - \sum_{j=1}^{N} X_{j,0}^{k} = 0, k \in \{1 \cdots K\} \tag{6}$$

$$\sum_{k=1}^{K} X_{0,j}^{k} \leq K \tag{7}$$

The objective function defined in (1) minimizes the total travelled distances by all the vehicles. Constraint sets (3) and (4) guarantee that each customer is visited only once. Constraint set (5) ensures that the total request of the customers being served by a vehicle does not exceed the vehicle capacity. Constraint set (6) guarantees that all routes start and finish at the depot, and constraint (7) requires that there are a maximum of K routes for serving the customers.

To encode a VRP solution, several representations are possible: vector, matrix, graph, etc (Pétrowski and Ben-Hamida, 2017). The most common encoding technique is the intuitive path-encoding in which each vehicle route is designed by a sequence of numbers corresponding to the requests indexes in the request vector R. In the present work, to encode a CVRP solution S, two strings of positive numbers are used: $S = (P,L)$, where P designs the global path of the solution defined by the connection of the different vehicle routes without including the depot, and L is a string vector designing the routes lengths of all the vehicles.

3.2 Handling Constraints

To handle constraints without implementing costly tools or adding new parameters, we propose the segregational selection for the parental selection. The main idea of the segregational selection is to sort the population (or a set of individuals for tournament selection) either according first to fitness (total distance for CVRP) and then violation rate (capacity overflow for CVRP), or according first to violation rate and then total distance. Segregational selection aims to balance between best feasible solution and unfeasible solutions having the lowest total distance.

3.3 The Algorithm

For the experimental study, the implemented generational genetic loop respects the Steady State GA (SSGA). Indeed, with the SSGA, at each iteration, only two parents are selected to generate a new offspring that replaces the worst offspring in the population. In our case, the new offspring is generated with a reproduction procedure using the different variation operators described in section 2.

Algorithm 1: SSGA.

1: Initialize population with μ feasible solutions
2: Evaluate initial population
3: **while** non STOP condition **do**
4: select two parents (P_1, P_2) using the segregational tournament selection
5: $C \leftarrow$ Reproduction(P_i, P_j)
6: set W the worst solution according to either fitness or violation rate
7: Replace W with C in the parent population
8: **end while**

3.4 Benchmark Data and Parameters Setting

SSGA with the variation operators is implemented in Python language. Its performance is then validated on some selected benchmark instances from the set of CVRPs proposed by (Augerat et al., 1995) available on the "VRP Web" site [1]. The set A contains 27 instances with a number of requests n varying from 32 to 80 and a number of vehicles k varying from 5 to 10. The vehicle capacity is set to 100 for all the problems. For a comparison purpose, the optimal or the best-known solution (BKS) is given with the SSGA results. A set of preliminary tests have been applied to tune the SSGA parameters and the retained values are summarized in table 2. As shown in the table, some parameters may have different values from one experimental study to another.

Table 2: Parameters used for SSGA.

Hyperparameter	Value
Crossover probability	0.8
Mutation probability	0.4
Population size	200
Tournament size	8
Maximum nb of it	50000 (1^{st} study)
	300000 (2^{d} study)
It without improvement	10000
Runs per Instance	20 (1^{st} study), 30 (2^{d} study)

4 COMPARING OPERATORS CLASSES

To study the SSGA behavior when applying one of the operator classes introduced above, two benchmarks instances are selected: the first has 34 requests and 5 vehicles (n34-K5) and the second has 39 requests and 5 vehicles (n39-K5). The corresponding BKS are 778 and 822 respectively. The GA parameters applied for this study are set as given in table 2. A first series of tests are run using a single operator class at a time. In the second series of tests, each crossover class is applied with one operator selected randomly from the two mutation classes. Four combinations are then studied: (PosCross+PosMut), (PosCross+LocalMut), (HeurCross+PosMut), (HeurCross+LocalMut).

To compare the different combinations, we extract some synthetic information from the obtained results. First, we compute the minimum and average relative distance (loss) to the optimum: $(\%loss = 100 * (Best - optimum)/optimum)$. Results are illustrated in figure 1. Second, we record the relative convergence speed according to the maximum number of iterations defined in the parameters' settings. The aim is to visualize if the evolution has stopped due to a stagnation problem or because of the maximum number of iterations is reached. A high value generally reflects rapid, often premature convergence. Results are illustrated in the figure 2. Apart from that, to visualize the exploration/exploitation ability of the operators classes along the evolution, the best fitness spot of the best run of each test case is illustrated with a curve in figure 3.

Figures 1 (a) and (b) show clearly that the performance of each class of operators depends on the problem to solve. For the n34-K5 case, mutation operators, either in PosMut or LocalMut class, were able to bring the population near the optimal solution. It wasn't the case with n39-k5, which is more difficult

[1] http://www.bernabe.dorronsoro.es/vrp/

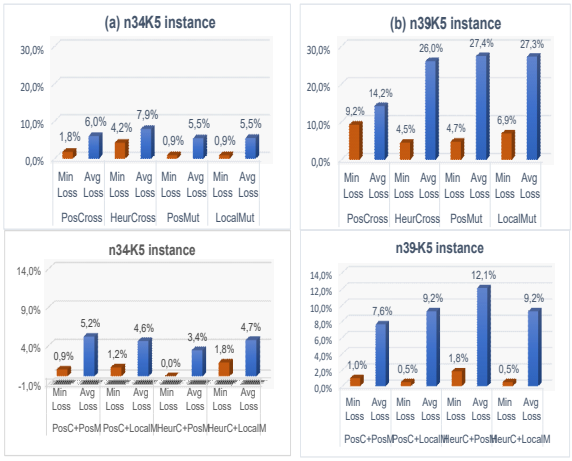

Figure 1: Minimum and average loss according to BKS obtained by each class or combination of classes of operators for the CVRP instances n34-K5 and n39-K5.

to solve than n34-k5. For the difficult cases, the use of both crossover and mutation operators is needed in evolutionary search. Indeed, Figures 1(c) and (d) show that the deviation to the optimum is greatly improved with the combined classes. For example, with the combination (PosCross+LocalMut), the deviation of best solution is about only 0.5% to the optimum (Fig 1(d)), while it is about 9.2% for PosCross and 6.2% for LocalMut (Fig 1(b)) when applied with combination. Similarly, combining operators classes for the n34-K5 case allows a better exploration of the search space, especially for the HeurCross class which its combination with PosMut class has allowed it to easily find the optimum.

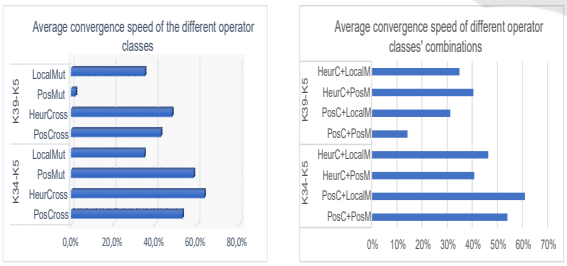

Figure 2: Convergence Speed according to the maximum number of iterations for each class or combination of classes of operators for the CVRP instance n34-K5 and n39-K5.

Otherwise, it can be noted through the figures 1 and2 that some classes of operators have a great ability to perform local search around the best solutions. This great exploitation capacity, applied with a high probability since the first steps of the evolution, bring the population around local optima. The SSGA diversification process helps in some cases to explore other regions but ends up falling back into the trap of premature convergence. This is the case for HeurCross class operators.

Other classes of operators have the opposite problem. Their exploration capacity allows the population to continuously explore new regions in the research space and avoid local optima. It is the case of the PosMut class. However, the convergence is too slow and does not necessarily lead to the optimum region. This is due to the lack of effective exploitation of the best solutions. The question now is how to take advantage of the strengths of each class and create a perfect balance between exploitation and exploration along the evolution.

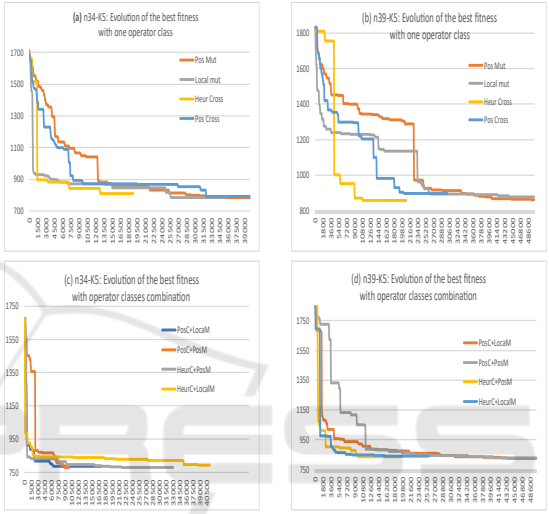

Figure 3: Evolution of the best fitness of the best run in the series of tests for solving the cases n34-k5 and n39-k5.

Combining Genetic Algorithms with local search techniques in the mutation step is widely used and proved to be efficient in speeding up the evolution process for ordering optimization. However, when applied with heuristic crossover, this advantage can become a difficulty. Indeed, if the two steps of reproduction have as common objective to exploit the best solutions to produce more competitive ones, the exploration of the research space is reduced and the population is attracted to local optima. This is the famous dilemma between exploitation and exploration for genetic algorithms. We believe that it is important, for ordering optimization, to control the simultaneous use of different classes of operators. Thus, we propose and test in the following section two strategies to allow a simultaneous use of all classes of operators. The second strategy introduces in addition a restrictive combination between classes to control the population convergence.

5 COMBINING VARIATION OPERATORS

5.1 Hybridization Strategies

As concluded in the first comparative study, the exploitation/exploration abilities of the variation operators differ from a class to another. Each operator can help in the evolution of the population towards better areas in the search space but in a different way. We then make the first hypothesis: "Using variation operators from different classes improve the GA efficiency for ordering optimization." However, heuristic crossover when applied with local mutation can lead to a premature convergence. To prevent this phenomenon, we propose to combine heuristic crossover with random mutation, and similarly, position crossover with local mutation. The objective is that random operators control the convergence speed of heuristic/local operators and vise versa . Thus, we make the second hypothesis: "An operator class with high exploitation ability could be controlled with an operator class with a high exploration ability".

To test the two hypotheses, we propose below two hybridization strategies. The first one is the random hybridization where the crossover and mutation operators are chosen randomly from the set of all the operators' classes. The second strategy is the balancing hybridization which is a conditional hybridization. If one class of operators is used for crossover, it will not be used again for mutation. Note that the operator within the given class is chosen randomly.

5.1.1 Random Hybridization (RH)

Random hybridization is quite equivalent to the multi-operator mechanism proposed in (Puljić and Manger, 2013) with an additional condition on the set of variation operators which must includes some operators from each class. To produce an offspring with the RH strategy, a crossover operator is selected randomly from the PosCross or HeurCross classes. The produced offspring is then mutated using a random operator from PosMut or LocalMut classes (Algo. 2). There is no additional restriction on the choice of operators either for crossover or mutation.

Algorithm 2: RH Reproduction(P_1, P_2).

1: select randomly opX in PosCross ∪ HeurCross
2: With a probability p_c, apply opX to (P_1, P_2) and get the new child C
3: SELECT randomly opM in LocalMut ∪ PosMut
4: With a probability p_m, apply opM to C

5.1.2 Balancing Hybridization (BH)

Balancing hybridization aims to balance the GA exploration and exploitation abilities by mixing heuristic operators with position operators. Thus, if an exploitative offspring is generated with an heuristic crossover, it can be mutated only with a position mutation operator to add an exploratory dimension. Similarly, if an exploratory child is generated with a position crossover, it can be mutated only with a local search based mutation for an exploitative purpose (Algo. 3).

Algorithm 3: BH Reproduction(P_1, P_2).

1: select randomly opX in PosCross ∪ HeurCross
2: With a probability p_c, apply opX to (P_1, P_2) and get the new child C
3: **if** opX ∈ PosCross **then**
4: SELECT opM from LocalMut class
5: **else**
6: SELECT opM from PosMut class
7: **end if**
8: With a probability p_m, apply opM to C

5.2 Results and Discussion

For each series of 30 runs, we record: the cost of the best obtained solution (Best), the average cost over the thirty runs (Average) and the percentage of the relative deviation (loss) from the best-known solution (BKS): $\%Loss = 100 * (Best - BKS)/BKS$ (%Deviation). Results for the second experimental study are given in table 3. The relative deviation values (loss) are also illustrated in figure4 to make easier the comparison between the RH and BH procedures.

The different values recorded for each benchmark instance demonstrate that the SSGA efficiency is greatly improved with the hybridization strategies, either random or balanced hybridization. Indeed, the deviation or loss to the optimum is less than 1% for 11 instances with the "RH reproduction" and for 10 instances with the "BH Reproduction" as illustrated in figure 4. These results confirm our first hypothesis that the application of the multi-operator mechanism including operators from the four classes enhances the efficiency of the Genetic Algorithm. However, except for some cases, "BH Reproduction" was not able to outperform the "RH Reproduction" as we supposed in the second hypothesis. As illustrated in figure 4, with the RH Reproduction", the GA is able to get closer to the optimum than with "BH Reproduction", especially for the cases with a number of requests greater than 42 (except for the n65-k5 case). Through these

Table 3: Results with CVRP A.

		Random Hybridization			Balancing Hybridization		
Instance	BKS	Best	Average	% Deviation	Best	Average	% Deviation
A-n32-k5.vrp	784	784	806	0	784	803	0
A-n33-k5.vrp	661	661	689	0	661	694	0
A-n33-k6.vrp	742	742	757	0	743	761	0.13
A-n34-k5.vrp	778	778	797	0	778	800	0
A-n36-k5.vrp	799	805	831	0.75	814	840	1.88
A-n37-k5.vrp	669	670	706	0.15	670	703	0.15
A-n37-k6.vrp	949	953	1009	0.42	953	1036	0.42
A-n38-k5.vrp	730	731	763	0.14	733	759	0.41
A-n39-k5.vrp	822	834	863	1.46	825	864	0.36
A-n39-k6.vrp	831	839	866	0.96	839	875	0.96
A-n44-k7.vrp	937	947	984	1.07	957	981	2.13
A-n45-k7.vrp	1146	1166	1203	1.75	1176	1206	2.62
A-n46-k7.vrp	914	953	986	4.27	923	978	0.98
A-n48-k7.vrp	1073	1115	1164	3.91	1107	1150	3.17
A-n53-k7.vrp	1010	1031	1100	2.08	1038	1189	2.77
A-n54-k7.vrp	1167	1193	1362	2.23	1222	1548	4.71
A-n55-k9.vrp	1073	1075	1168	0.19	1104	1507	2.89
A-n60-k9.vrp	1408	1414	1596	0.43	1405	1619	3.77
A-n63-k10.vrp	1315	1378	1846	4.79	1349	1456	4.57
A-n63-k9.vrp	1634	1685	2441	3.12	1717	2960	5.08
A-n64-k9.vrp	1402	1472	1903	4.99	1459	2136	4.07
A-n65-k9.vrp	1177	1289	2698	9.52	1219	1754	3.37
A-n69-k9.vrp	1168	1186	1491	1.54	1219	1549	4.37

results, we can understand that there is a synergy between the different classes of operators. So it is useful to combine the different classes without restriction. However, HeurCross and LocMut classes are helpful to refine the best solutions in the second step of the evolution but could penalize the search process at the beginning of the evolution. Thus, we think that is advantageous to control the application of the different combination between operators classes with a dynamic probability.

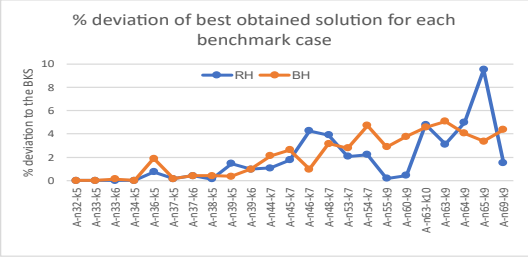

Figure 4: Relative Deviation to the optimum given by the best solution obtained with RH and BH reproduction.

6 CONCLUSION

This paper proposes a classification of the variation operators used for evolutionary ordering optimization into four classes: Position crossover, Heuristic crossover, Position mutation and Local mutation. After studying the exploitation/exploration ability of each class on solving two benchmark instances of the Capacitated Vehicle Routing Problem, we proposed two hybridization strategies between operators classes: Random Hybridization and Balancing Hybridization. Results obtained for 24 CVRP benchmark instances showed clearly that combining variation operators from different classes increases the GA robustness when solving constrained ordering optimization problems. However, the balancing hybridization needs further control on the application of the different classes combinations. Future works aim to introduce this control with a dynamic or adaptive probability and apply the implemented GA for solving other CVRP benchmark series.

REFERENCES

Augerat, P., Belenguer, J. M., Benavent, E., Corberán, A., Naddef, D., and Rinaldi, G. (1995). *Computational results with a branch and cut code for the capacitated vehicle routing problem*. IMAG.

Cicirello, V. A. (2006). Non-wrapping order crossover: An order preserving crossover operator that respects absolute position. In *Proceedings of the 8th annual conference on Genetic and evolutionary computation*, pages 1125–1132. ACM.

Dantzig, G. B. and Ramser, J. H. (1959). The truck dispatching problem. *Management science*, 6(1):80–91.

Davis, L. (1985). Job shop scheduling with genetic algorithms. In *Proceedings of the 1st International Conference on Genetic Algorithms*, pages 136–140, Hillsdale, NJ, USA. L. Erlbaum Associates Inc.

Deep, K. and Adane, H. M. (2011). New variations of order crossover for travelling salesman problem. *International Journal of Combinatorial Optimization Problems and Informatics*, 2(1):2.

Fogel, D. B. (1988). An evolutionary approach to the traveling salesman problem. *Biological Cybernetics*, 60(2):139–144.

Freisleben, B. and Merz, P. (1996). A genetic local search algorithm for solving symmetric and asymmetric traveling salesman problems. In *Evolutionary Computation, 1996., Proceedings of IEEE International Conference on*, pages 616–621. IEEE.

Goldberg, D. E. and Lingle, R. (1985). Alleles, loci, and the traveling salesman problem. In *Proceedings of an International Conference on Genetic Algorithms and Their Applications*, volume 154, pages 154–159. Lawrence Erlbaum, Hillsdale, NJ.

Grefenstette, J., Gopal, R., Rosmaita, B., and Van Gucht, D. (1985). Genetic algorithms for the traveling salesman problem. In *Proceedings of the first International Conference on Genetic Algorithms and their Applications*, pages 160–168. Lawrence Erlbaum, New Jersey (160-168).

Grefenstette, J. J. (1987). Incorporating problem specific knowledge into genetic algorithms. *Genetic algorithms and simulated annealing*, 4:42–60.

Karakatič, S. and Podgorelec, V. (2015). A survey of genetic algorithms for solving multi depot vehicle routing problem. *Applied Soft Computing*, 27:519 – 532.

M., I., Smith, D. J., and Holland, J. R. C. (1987). A study of permutation crossover operators on the traveling salesman problem. In *Proceedings of the Second International Conference on Genetic Algorithms and Their Application*, pages 224–230, Hillsdale, USA.

Michalewicz, Z. (1992). *Genetic algorithms+ data structures= evolution programs*. Springer Verlag.

Mühlenbein, H. (1989). Parallel genetic algorithms, population genetics and combinatorial optimization. In *Workshop on Parallel Processing: Logic, Organization, and Technology*, pages 398–406. Springer.

Mühlenbein, H., Gorges-Schleuter, M., and Krämer, O. (1988). Evolution algorithms in combinatorial optimization. *Parallel Computing*, 7(1):65–85.

Neri, F., Cotta, C., and Moscato, P. (2012). *Handbook of memetic algorithms*, volume 379. Springer.

Pétrowski, A. and Ben-Hamida, S. (2017). *Evolutionary Algorithms*. John Wiley & Sons.

Puljić, K. and Manger, R. (2013). Comparison of eight evolutionary crossover operators for the vehicle routing problem. *Mathematical Communications*, 18(2):359–375.

Syswerda, G. (1989). Uniform crossover in genetic algorithms. In *Proceedings of the 3rd International Conference on Genetic Algorithms*, pages 2–9, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Syswerda, G. (1991). Schedule optimization using genetic algorithms. In Davis, L., editor, *Handbook of Genetic Algorithms*. Van Nostrand Reinhold, New York, NY.

Tao, G. and Michalewicz, Z. (1998). Inver-over operator for the tsp. In *5th International Conference Parallel Problem Solving from Nature Amsterdam, The Netherlands*, pages 803–812. Springer.

Tao, Z. (2008). Tsp problem solution based on improved genetic algorithm. In *Natural Computation, 2008. ICNC'08. Fourth International Conference on*, volume 1, pages 686–690. IEEE.

Whitley, L. D., Starkweather, T., and Fuquay, D. (1989). Scheduling problems and traveling salesmen: The genetic edge recombination operator. In *ICGA*, volume 89, pages 133–40.