EasyChair Preprint

№ 4611

A GRASP Hybrid Genetic Algorithm for the Capacitated Vehicle Routing Problem

Alexander F. Rego and Aurora T. R. Pozo

November 19, 2020

A GRASP Hybrid Genetic Algorithm for the Capacitated Vehicle Routing Problem

No Author Given

No Institute Given

Abstract. The vehicle routing problem is an interesting and challenging combinatorial problem, study for over more than fifty years since Dantzig and Ramser. Several researches have been conducted on this problem and its variants generating many approaches including the population-based one. In this study, we present a Grasp Hybrid Genetic Algorithm (GHGA) to solve the Capacitated Vehicle Routing Problem (CVRP). Our approach combines the efficiency of the well-known Travel Salesman Problem crossovers with a proposed Partial Intensification Mechanism (PIM), which is a combination of a modified 2-opt local movement and the Split algorithm. Additionally, we present the Neighborhood Perturbation Mechanism (NEP). Inspired in the perturbation phase of the Large Neighborhood Search, we inserted destroy-repair operators with an adaptive use of degradation ratio. Experiments were conducted on well-known Christofides et al. benchmark supporting that our approach has interesting points and it is a promising approach.

Keywords: capacitated vehicle routing problem · metaheuristic · optimization · hybrid genetic algorithm.

1 Introduction

The Vehicle Routing Problem (VRP) is a generalization of the Travel Salesman Problem (TSP), where the goals is to find the optimal set of feasible routes to be performed by a fleet of vehicles serving a set of customers. For more than 50 years, since Dantzig and Ramser [4] had introduced the problem in 1959, researchers from many fields have studied the VRP [20, 2, 22, 6].

Real-life routing problems have several constraints and aspects, which may increase the complexity, and usually have a high number of customers to be served generating large instances of problems. Scenarios like supply chain [2], fuel and battery consumption [22], emergency and disaster relief [6], have inspired extensions for VRP as VRP with Time Window (VRPTW) [12, 20], Multi-Depot VRP (MDVRP and MDVRPTW) [20] and others.

For those complex scenarios, variations, and large instances, a variety of approaches using heuristics and metaheuristics were presented, especially with the Genetic Algorithm (GA) being part of the approach [1, 23, 17, 10, 20, 12].

As noticed, most VRP problems involve capacity and time window constraints and are still a challenge. In the CVRP, a single or a fleet of identical

vehicles perform a tour through all customers, serving to their demands. The vehicle/fleet departs from the depot and returns to it once the vehicle's capacity is reached or no more customers are available.

For those reasons, to solve the CVRP problem, this study presents a hybridization of Genetic Algorithm with the following features: an initialization with a Greedy Randomized Adaptive Search Procedure (GRASP) to start the search process with good solutions; a Partial Intensification Mechanism (PIM), which combine a new way of using the 2-opt local movement with the *split algorithm* of Prins [16]; and a Neighborhood Perturbation Mechanism (NEP) inspired by the Large Neighborhood Search perturbation phase. The combination of the exploratory characteristic of GA with the intensification of local search operators and a mechanism to escape local optima shown to be very promising according to the results of our experiments.

The remainder of this research is organized as follows: Section 2 presents the notation and formally defines the problem. In the Section 3 a brief literature review is presented. The proposed GRASP Hybrid Genetic Algorithm (GHGA), the chromosome representation, the initialization method, repair method, crossovers, the Partial Intensification Mechanism (PIM), and Neighborhood Perturbation Mechanism (NEP) are presented in Section 4. Section 5 and 6, describes how experiments were conducted and the conclusion respectively.

2 Problem Definition

The CVRP is a classic problem among VRP problems. As mention before it is a problem where a vehicle or a fleet of identical vehicles serves a set of customers. This vehicle has a capacity Q, which configures a capacity constraint, and performs a tour visiting V customers. Each customer has a demand w, thereby once the vehicle's capacity is fulfilled or reach the trip maximum length L it towards the depot V_0 where it departed.

Formally the CVRP can be modeled as follow: Let $G = (V, A)$ a complete graph in which $V = \{0, 1, ..., n\}$ represents the vertices. Let n be the number of customers and V_0 denoted by the depot. A represents the sets of arcs defined as $A = \{(i, j) \mid i, j \in V\}$. The c_{ij} denotes the cost associated with each $(i, j) \in A$. $K = \{1, 2, ..., l\}$ represents a set of unlimited homogeneous vehicles, and for each $k \in K$ a Q capacity is associate. Also, for each customer a demand $w \in W$ is associated too. The objective is to minimize the total cost performed by all vehicles without exceeding each vehicle's capacity and the maximum travel distance L:

$$x_{ij}^k = \begin{cases} 1 \textit{ if arc (i,j) is taken by vehicle k} \\ 0 \qquad\qquad\qquad \textit{otherwise} \end{cases} \tag{1}$$

$$min \sum_{k=1}^{l} \sum_{(i,j)\in A} c_{ij} x_{ij}^k \tag{2}$$

$$\sum_{i=1\in W}^{n} w_i k \leq Q, \forall k \in K \tag{3}$$

$$\sum_{(i,j)\in A} c_{ij} x_{ij}^{k} \leq L \tag{4}$$

Our model presents in (2) the objective function. Constraints (3) and (4) assures that the route performed by vehicle k do not exceeds its capacity and the maximum travel distance respectively.

3 Literature Review

A brief literature review of contributions for the CVRP, but not limited only to CVRP, is provided in this section. The main goal of this review is to present the most recently approaches especially those based on population.

Prins [16] proposed a Hybrid GA or Memetic algorithm. Furthermore, was the first to use a giant TSP tour successfully. A giant TSP tour is a sequence (permutation) of n customers without route delimiters. Besides, instead of using a mutation operator, he used a local search procedure with a linear ordered crossover. In the local procedure, Prins used the *split algorithm*, detailed in Section 4, combined with insertion, swap, 2-opt (intra-route), and 2-opt* (inter-route) moves. Prins also presents a population management mechanism by removing clones (chromosomes with the same fitness) and insert a new one from his initialization method.

Nagata and Bräysy [12] presented an evolutionary algorithm using the edge assembly crossover (EAX). The chromosome is a giant TSP tour, later converted into sub-tours (routes). Once applied the EAX, infeasible offspring can be generated, then a modification mechanism is applied, which basically applies the same two local movements: 2-opt and interchange, used on the local procedure. After that, a local procedure is executed. As EAX can generate infeasible children, they present a relaxation of the capacity constraint, which is used on the modification mechanism.

Inspired by some ideas of Prins [16] and Nagata and Bräysy [12], Vidal et al. [20] proposed a Hybrid Genetic Search with an Adaptive Diversity Control (HGSADC). The HGSADC follows the principle of using a giant TSP tour, and a local procedure called *Education*. In the crossover phase, Vidal et al. use an ordered crossover for most VRPs, and for the Periodic VRP (PVRP), they proposed a new crossover called *Periodic Crossover with Insertions* (PIX). Further, they use relaxation as shown in [12]. In the *Education* procedure, they use *split algorithm* [16], and four local movements: swap, relocate, 2-opt, and 2-opt*. The *Adaptive Diversity Control* used the fitness function, called *biased fitness function*, which uses the Hamming distance and a penalty function (due to relaxation) to evaluate the chromosome. Further, promotes the *survivor selection*, which determines which *pop size* chromosomes will go to the next generation.

First, it removes the clones, which are chromosomes with the same fitness (*biased fitness*), then the worst chromosomes. Second, a *diversification* process, which keep the best *pop size*/3 chromosomes, and introduce new $4 \times$ *pop size* chromosomes. Finally, the *survivor selection* is applied again.

Li et al. [10] presented a hybridization of GA (HGA-ALS) and an Adaptive Local Search. After generate the initial population, parents are select by binary tournament and then crossed by an improved ordered crossover. The mutation operator was substituted by a Adaptive Large Neighborhood Search (ALNS), which uses ten local movements, two destroy operators, and one repair operator called *greedy insertion based on probability assignment* (GIPA). Their adaptive mechanism, rather than partition the solution into different sub-solutions, selects the best neighborhood method (local movements) to improve the current solution.

Rabbouch et al. [17] proposed a GA with a recombination step for the MDVRPTW with heterogeneous fleet. Their GA is a typical GA, using a route-based crossover (recombination) proposed by Potvin and Bengio [15], applying a repair procedure right after, and then a swap as mutation operator. Although, their GA's results are not competitive, in some instances of the tested benchmarks it achieved a GAP of 3.02% considering that Rabbouch et al. not used any local search mechanism.

Abdallah and Ennigrou [1] proposed a hybrid solution, composed by a Particle Swarm Algorithm (PSO), a GA and a Memetic Algorithm (MA), to solve the MDVRPTW with an heterogeneous fleet. In the GA they used the ordered crossover and exchange mutation operator, and in the MA, in the local procedure, a Λ-exchange method inspired by [18]. To communicate between the metaheuristics called agents, they used an acl message protocol, every time an agent improves its best result.

Zhen et al.[23] presented two solutions, a PSO and a GA both hybridized with a Local Search Variable Neighborhood Descent (LS-VND). A survival selection is applied based on a biased fitness function. On the LS-VND, three local movements are performed: reinsertion, exchange, and reverse.

This review shows that VRP is still a very interesting combinatorial problem, a relevant research topic, and the good potential of population-based algorithms to solve it. Moreover, there are still gaps to be researched on better escape mechanisms from local optimal, generation of good starting solutions, adaptive procedures, and the combination of local movements that further improve the solution.

4 Proposed GRASP Hybrid Genetic Algorithm

The GRASP Hybrid Genetic Algorithm (GHGA) (Algorithm 1) follows the structure of the genetic algorithm introduced by Holland [9]. Still, it presents interesting characteristics concerning the improvement of offspring and the exploration of the search space.

Algorithm 1: Hybrid GA

```
gen ← 0;
Generate initial population P_gen;
Repair(P_gen);
Evaluate(P_gen);
while gen < gens or stop criteria is not reached do
    Q_gen ← Tournament Selection(P_gen);
    Q_gen ← Crossover(Q_gen);
    Q_gen ← Partial Intensification(Q_gen);
    Repair(Q_gen);
    Evaluate(Q_gen);
    P_gen+1 ← Elitism(P_gen, Q_gen);
    every 100 gen do Neighborhood Perturbation(P_gen);
    gen ← gen + 1;
end
return Best(P_gen)
```

According to Algorithm 1, our approach of GA follow the basic idea of a generic GA. First, it generates a initial population, then each chromosome is evaluated. Second, for n generations or until no further improvement can be made, its selected (binary tournament selection) two parents, then its applied crossover and mutation operators. Besides, a repair method is applied if a child is infeasible. And finally, to avoid a straight elitism approach, its selected by elitism 10% of parents (P_{gen}) and 90% of the offspring (Q_{gen}) for the next generation.

In the sections 4.1, 4.2, 4.3, the GA parts are explained in detail.

4.1 Representation, Initialization, and Repair

Representation Fig. 1 shows a giant TSP tour without delimiters as a chromosome. Each allele in the chromosome is a customer. The first one to use successfully this representation was Prins [16]. This kind of chromosome allows taking advantage of several well-known TSP crossovers. However, it needs an algorithm to find a segmentation of chromosome in routes retrieving the cost and the solution. For this sake, we used the *split algorithm* from Prins [16], which is explained in detail in Section 4.2.

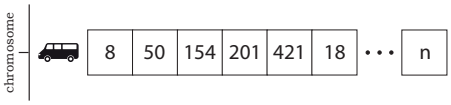


Fig. 1: Chromosome representation

Initialization To initialize the population of our GA, a random ratio of individuals was established. This ratio defines how many random individuals will be generated randomly, the other part of population is generated by our Greedy Randomized Adaptive Search Procedure (GRASP) algorithm as shown by Algorithm 2.

Algorithm 2: GRASP algorithm to initialize a chromosome

```
while chromosome is empty do
    C ← available customers;
    N ← max number of vehicles;
    n ← 1;
    while N > n and size of C > 0 do
        trip ← empty list;
        while n_capacity > 0 do
            if trip is empty then
                c ← Random Choice(C);
            end
            else
                c ← Closest Customer(trip, C);
            end
            n_capacity ← n_capacity - c_demand;
            insert c into RP;
            remove c from C;
        end
        n ← n + 1;
        if trip is not empty then
            append trip into chromosome;
        end
    end
    if chromosome is not feasible then
        chromosome ← empty_list;
    end
end
return chromosome
```

Our GRASP generates only feasible solutions, randomizing the nearest customers to the depot by an adaptive random factor (Equation 5), which is used in the *Random Choice()* method, the rest of the chromosome is constructed by a greedy approach.

$$random\_factor = total\_demand/vehicle\_max\_capacity \tag{5}$$

According to Algorithm 2, a set of available customers (C) is created with all not served customers and the maximum number (N) of available vehicles is defined. While still have available customers and vehicles to serve them, is verified if the selected vehicle still has capacity. To select the next customer to

be added in the *trip*, two methods were implemented (i) Random Choose(C) and (ii) Closest Customer($trip, C$). In the Random Choose method, sort the available customers by distance. In addition, it applies some randomness (*random factor*) into half of the sorted available customers to guarantee some diversity to the initial population. The Closest Customer method prioritizes the distance between the last customer added in the *trip* and the next one. Finally, the algorithm stops when a feasible chromosome is created.

Repair Since some chromosomes might not be selected by the Partial Intensification Mechanism (PIM) a repair method is needed. Our repair method consists of removing customers that either violates the vehicle's capacity or the maximum travel distance constraint. Once, those customers are removed the best insertion method is used to insert them in the best possible place. If the best insertion method could not find a place for a customer, a new *trip* is created to place it.

Algorithm 3: Repair

```
S ← split by capacity(chromosome);
foreach trip ∈ S do
    while trip violates L do
        r ← remove customer();
    end
end
foreach customer ∈ r do
    inserted ← insert best place(S, customer);
    if inserted is False then
        new_trip ← create new trip();
        insert customer trip(new_trip, customer);
        insert trip S(S, new_trip);
    end
end
repaired_chromosome ← concat(S);
return repaired_chromosome
```

According to Algorithm 3, the chromosome is divided by vehicle's capacity Q into a set of *trips* S. For each *trip* is verified if it violates the maximum travel distance L. If it violates, the customer with the highest cost is removed from the *trip* and placed in a set of removed r. For each customer in r, it is tried to insert it in the best place, if it is not possible a *newtrip* is created and the customer is inserted on it. Finally, the set of *trips* in S is concatenated.

4.2 Crossovers and Partial Intensification Mechanism (PIM)

Crossover The crossover operator plays a very important role in the GA. Following this principle, as TSP and VRP are related problems, many efficient TSP crossovers can be used [7, 5, 13, 19]. In our GA, five crossovers were selected:

- **Partially Mapped** [7]: it intends to pass some ordering and value information to the offspring. First, a random sub-tour is copied from parent 1 to the offspring. This sub-tour is used to create a mapping of the alleles of the sub-tour with the corresponding index in the parent 2. For example, the sub-tour is composed by alleles 4, 5, and 6, the corresponding alleles in the same index in the parent 2 are 1, 6, and 8. The mapping would be 4-1, 5-6, and 6-8. After that, it is copied the alleles of parent 2 into offspring in the same index, and if the customer is already present in the offspring it is replaced one in the mapping.
- **Edge Recombination** [21]: it is based on a list of neighbors of each allele in both parents. The neighbor list is generate by recording each allele immediate neighbors, including those that roll around the end of chromosome, for each parent. These two neighbors list are merged by a union process ignoring the duplicates. The rest of the process consisted of select randomly a allele c of any parent, append it to the offspring and remove it from the neighbor list. If c has no more neighbors, then select a new random allele that not in offspring. However, if c still has neighbors, select the one with the fewest neighbors (or a random choice between those with the same number of neighbors).
- **Order** [5]: it prioritizes the order of alleles. It builds an offspring by choosing a sub-tour in parent 1 and preserving the relative order of alleles of parent 2, by copying the alleles that are not in the selected sub-tour, starting at the last cut point position using the order of alleles of parent 2.
- **Order based** [19]: slightly different from order crossover of Davis [5], the Order based crossover selects random several indexes. It searches in parent 2 for the alleles in these indexes, then searches in the parent 1 for those alleles and replace them by the ordering of appearance in parent 2.
- **Cycle** [13]: each allele comes from one parent along with its index. For example, start with the allele in the first index of parent 1, go to parent 2 in the same index, and check the allele on it. Search in parent 1 for the allele found in parent 2 and append it to offspring.

To select a crossover we use a fixed probability of 20% for each crossover.

PIM As shown in the literature review, section 3, many hybrid approaches use local movements instead of mutation operators to intensify the search for better solutions. And the combination of local movements proved to be very effective to find good solutions [16, 12, 20].

Inspired by those works, we proposed the Partial Intensification Mechanism (PIM), which is a combination of a 2-opt movement executed k times and the *split algorithm* of Prins [16]. The *split algorithm* finds the delimiters for each

route (shortest-path problem) through an auxiliary acyclic graph. It also can be used with a heterogeneous fleet, limited or not.

The *split algorithm* [16], as mentioned before, aims to find the delimiters for the route as a shortest path problem. Define an auxiliary graph $S = (V, A, C)$, where V is the set of nodes with $n + 1$ elements indexed from 0 to n. A an arc (i, j) with $i < j$. Each arc represents a trip $(r_{i+1,j})$ from depot V_0, visiting customers V_{i+1} to V_j, and returning to the depot V_0 with a d cost associated. If a trip is feasible according to load (equation 6) and cost (equation 7) conditions, the C_{ij} is equal to the trip cost.

$$\forall (i,j) \in A : \sum_{x=i+1}^{j} q_{r_x} \leq W, \tag{6}$$

$$\forall (i,j) \in A : C_{i,j} = V_{0,r_{i+1}} + \sum_{x=i+1}^{j} (d_{r_x} + V_{V_x,V_{x+1}}) + d_{r_j} + V_{r_j,0} \leq L. \tag{7}$$

Figure 2 shows a sequence $V = (a, b, c, d, e)$ with $W = 10$ and $L = \infty$, the demand of each customer is in brackets (1). S (2) contains e.g. arcs (bold lines) $a - b$ with $C_{a,b} = 55$ for the trip $r_{a,b}$ $(0 - a - b - 0)$, c with $C_c = 60$ for the trip r_c $(0 - c - 0)$, and $d - e$ with $C_{d,e} = 90$ for the trip $r_{d,e}$ $(0 - d - e - 0)$ with a total cost of 205. The lower part (3) gives the VRP solution with three trips.

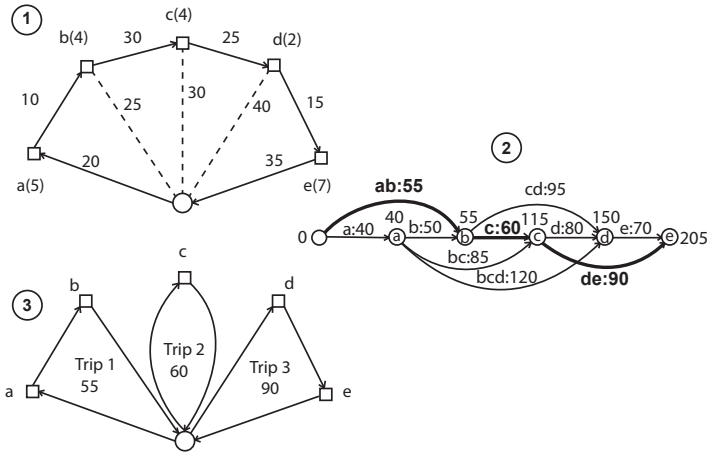


Fig. 2: Illustration of the *split algorithm* process (extracted from Prins [16])

The other part of PIM is the 2-opt local movement. Inspired by Lin–Kernighan heuristic [8], in its original version, which mention that k-opt moves are restricted to movements that can be decomposed into a 2-opt or 3-opt moves followed by a sequence of 2-opt moves. On the other hand, a k-opt move

can expend a high computational time. Therefore, in the PIM, a $k = 4$ maximum number of times was applied for the execution of the 2-opt move. This limitation not only produces improvement on the solution but also reduces the computational time if compared to an exhaustive search with 2-opt, more details in section 5.2.

As our chromosome it is a giant tour some movements performed by our 2-opt approach resemble 2-opt (intra-route) and 2-opt* (inter-route) without reversing the ends of the chromosome.

4.3 Neighborhood Perturbation Mechanism (NEP)

Any approach the solve combinatorial problems, especially NP-hard problems as VRP, often get stuck in local optimal. Local metaheuristics as Large Neighborhood Search (LNS), Adaptive LNS (ALNS), and Variable Neighborhood Search (VNS) use a perturbation procedure to change the neighborhood to explore the search space and escape from local optimal.

Inspired by these local metaheuristics, we implemented the Neighborhood Perturbation Mechanism (NEP). NEP sorted the population by fitness, selects the 50% worst chromosomes (*select worst(P_{gen})*), and applies a selected pair of destroy-repair operators. Finally, the Repair function is called for infeasible chromosomes.

Both selections methods, *select destroy operator(chromosome)* and *select repair operator(chromosome)*, used a fixed probability to select an operator. For destroying operators a fixed 33% probability was adopted for each, and for each repair operator the probability it is 50%. Inspired by some well-known ALNS operators, we selected three destroy operators. All destroy operators have a *degradation ratio*, which is based on a percentage of the total number of the customers and defines how many customers will be removed from the chromosome. It starts with 5%, and after 100 generations without improvement, it is increased by 0.5% every generation until 40%. Once an improvement is done the *degradation ratio* returns to 5%.

– **Related destroy**: remove n customers from the chromosome based in some similarity measure. This approach make the repair process easier and more likely to succeed.
  Our related destroy use a similarity metric based on Shaw similarity [14], which was design to the PVP, therefore we had to modify it to the CVRP as follow:

$$shaw = distance(customer_a, customer_b) + |w_{customer_a} - w_{customer_b}|$$

  Thereby a randomly n customers are selected (*shaw candidates*), next is picked up one customer from the *shaw candidates* list, and calculate the *shaw* for each customer that not belongs to the *shaw candidates* and inserted into the *shaw sample*. Next, the *shaw sample* is sorted by the *shaw* metric, and a customer is pick-up randomly by:

$$maximum(((size\_of(shaw\ sample)/2) - 1) \times random\_number, 0)$$

The *random_number* is a float number between 0 and 1.

- **Worst destroy**: remove n customers with the highest cost (distance) from the chromosome. For each pair $(i, i+1)$ customers is calculated the cost, and the n highest are removed from the chromosome.
- **Random destroy**: remove n randomly selected customers from the chromosome.

All destroy operators return the removed customers and the destroyed chromosome, which are passed as parameters for the selected repair operator. And for the repair process of a chromosome, two repair operator were selected:

- **Greedy repair**: inserts a customer in the least cost position in the chromosome (best insertion).
- **Noise repair**: inserts a customer in the least cost position in the chromosome given a noise parameter. In our approach, a noise factor is used as a random ratio for the cost in the best insertion method as follow:

$$noise\_factor = cost * (random\_number - 0.5)$$

$$cost = cost + noise\_factor$$

The use of a noise factor promotes the diversity in population.

5 Experiments

Sets of experiments to evaluate the GHGA were conducted. First, Section 5.1 explains the methodology adopted to set up the GA parameters. In Section 5.2 we evaluate the variation of the k parameter in the Partial Intensification Mechanism (PIM). Finally, in Section 5.3 are conducted comparisons of results of our GHGA with state-of-the-art approaches and best-known (BKS) solutions.

The GHGA was implemented in Python 3.7 and Cython 0.29. The experiments were run on an Intel Xeon CPU E5-2450 with a 2.10GHz clock and 47 GB RAM.

5.1 GHGA Configuration

To identify a good parameter set for the GHGA, we used the *Iterated Race* (IRACE) [11]. The IRACE is a framework for automating the configuration of algorithms, which consist of test new configurations according to a particular data set, select the best configuration from the recently tested samples through a race, and update the distribution of sample configurations aiming to guide the search for best configurations. It also applies the Friedman non-parametric statistical test.

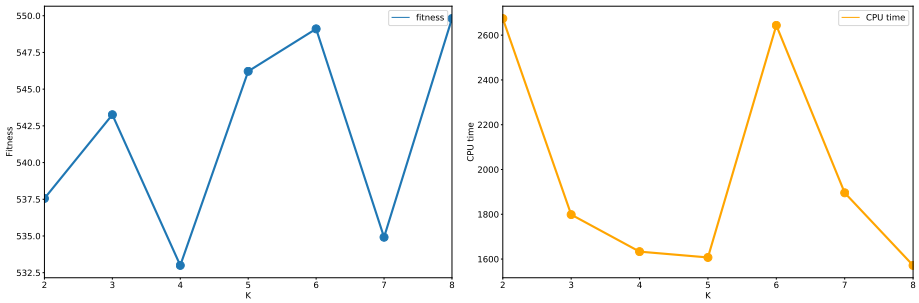
The IRACE was configured to run a thousand experiments with the set of parameters and range values shown in Table 1, under three instances of Christofides et al. benchmark [3]: CMT01, CMT03, and CMT05. Concerning a stop criterion, we adopted only the number of generations limited to 5000 generations.

Table 1: IRACE Results

Parameter		Range	IRACE Results	Final Config.
tx_pop_gen	The ratio of random chromosome generation by a randomized method, the rest of the population is generated by our GRASP	[0.2, 0.9]	0.9	0.9
cx_pb	Crossover probability	[0.5, 0.9]	[0.7, 0.9]	0.7
mut_pb	PIM probability	[0.4, 1.0]	0.9	0.9
pop_size	Size of population	[10, 16, 20, 26]	[20, 26]	20

5.2 PIM k Parameter Evaluation

Inspired by Helsgaun [8], which mentioned that high-quality solutions can be founded even though a few k-changes are performed, and following his methodology we performed two tests in the k parameter of PIM, both varying $2 \leq k \leq 8$. First, was analyzed the impact of increasing k on the quality of the solutions. Second, we analyze how the CPU time is affected. As shown in Figure 3a as the value of k changes different quality solutions are founded. And, according to Figure 3b we notice that CPU time also vary according k is increased. This might indicate that k varies throughout the search process, suggesting that a mechanism to control it would be interesting.



(a) Fitness according to k variation (b) CPU time according to k variation

Fig. 3: The k evaluation in Christofides et al. [3] CMT01 instance.

Therefore, the best value for k still is not cleared. However, Helsgaun proposed the followed equation to evaluate k:

$$Time(k) \times ((Length(k) - OPT)/OPT)$$

Where *Time(k)* is the time consumed to get the best solution (*Length(k)*), and OPT is the BKS of the current data set. The smaller the result of this equation, the better the k. Thus, using this equation, as shown in Figure 4, the best choices for k are 4 and 7.

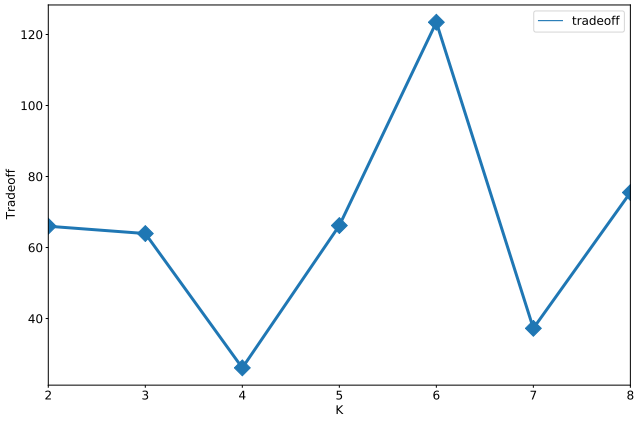


Fig. 4: Tradeoff between fitness and CPU time for k in Christofides et al. [3] CMT01 instance.

5.3 Solution Performance

Table 2: Comparison of Christofides et al. [3] benchmark with state-of-the-art approaches.

		Prins [16]		Nagata and Bräysy [12]		Vidal et al. [20]		GHGA				
Instance	n	Best	CPU (mim)	Best	CPU (mim)	Best	CPU (mim)	AVG of 5 runs	Best of 5 runs	AVG CPU of 5 runs (min)	GAP %	BKS
CMT01	50	**524.61**	0.01	**524.61**	0.07	**524.61**	0.43	537.26	524.93	25.54	0.06	524.61
CMT02	75	**835.26**	0.77	**835.26**	0.37	**835.26**	0.96	872.15	860.57	53.28	3.03	835.26
CMT03	100	**826.14**	0.46	**826.14**	0.29	**826.14**	1.27	867.24	845.36	86.32	2.33	826.14
CMT04	150	1030.46	5.50	**1028.42**	1.25	**1028.42**	2.87	1135.51	1108.75	178.82	7.81	1028.42
CMT05	199	1296.39	19.10	**1291.45**	5.04	**1291.45**	5.94	1471.63	1445.61	285.69	11.94	1291.29
CMT06	50	**555.43**	0.01	**555.43**	0.09	**555.43**	0.48	558.78	**555.43**	25.76	0.00	555.43
CMT07	75	**909.68**	1.41	**909.68**	0.65	**909.68**	1.09	973.32	962.09	54.96	5.76	909.68
CMT08	100	**865.94**	0.37	**865.94**	0.39	**865.94**	1.14	909.44	895.65	88.81	3.43	865.94
CMT09	150	**1162.55**	7.24	**1162.55**	2.26	**1162.55**	2.53	1300.46	1275.76	196.18	9.74	1162.55
CMT10	199	1402.75	26.83	**1395.85**	6.51	**1395.85**	8.22	1618.85	1586.03	302.69	13.62	1395.85
CMT11	120	**1042.11**	0.29	**1042.11**	0.35	**1042.11**	1.15	1091.60	1078.33	119.87	3.48	1042.11
CMT12	100	**819.56**	0.04	**819.56**	0.14	**819.56**	0.84	836.08	826.60	88.99	0.86	819.56
CMT13	120	1542.86	10.44	**1541.14**	1.78	**1541.14**	2.83	1622.78	1607.97	122.84	4.34	1541.14
CMT14	100	**866.37**	0.08	**866.37**	0.22	**866.37**	1.19	886.13	877.23	86.42	1.25	866.37
AVG GAP %		0.08		0.0		0.0					4.83	
AVG CPU (s)			5.20		1.39		2.21			122.58		

To compare our GHGA with BKS and some state-of-the-art approaches, we adopted the classical Christofides et al. benchmark [3] with customers ranging from 50 to 199, which is composed of 14 instances. The instances are divided into random (1-10) and clustered (11-14). Moreover, instances 6-10, 13, and 14

have duration restriction and time service. Instances 1-5, 11, and 12 do not have these restrictions.

The results for the Christofides et al. benchmark are presented in Table 2. The first two columns are the instance name and the number of customers. The 3, and 4 columns are best solutions and CPU time in minutes of Prins; 5, and 6 columns are best solutions and CPU time in minutes of Nagata and Bräysy; 7, and 8 columns are best solutions and CPU time in minutes of Vidal et al. Columns 9-12 are the average solution, best solution, average CPU time of 5 runs in minutes, and the GAP (deviation from BKS in percentage) of our solution. And the last column is the BKS. Lastly, we indicate in boldface the best result among algorithms for each instance.

According to Table 2, it is clear that our performance (CPU time) needs to be reviewed. Further investigations to optimize our implementation could improve it, but a faster language can fulfill this gap as well. On the other hand, the AVG GAP is low, and in some instances, it reaches the BKS (CMT06) or got very close to it, around 1% or less (CMT01, CMT12, CMT14), which still supports the potential of our approach.

6 Conclusion

In this research, we presented a new GRASP Hybrid Genetic Algorithm (GHGA) for the classical CVRP. Furthermore, a modification in the GRASP to generating good initial solutions, which combined with the Hybrid GA works as a guided search. Moreover, we proposed two mechanisms: PIM and NEP. Both mechanisms proved to have the potential to intensify the search and escape from local optimal respectively. Also, shows that there is still room for improvement and new ways of addressing these two fundamental processes to achieve better solutions.

Even the other approaches had better performance and solutions, the experiments showed that our approach has the potential to reach the BKS in the 14 tested instances (Christofides et al. [3]), however needs some improvements before. Among these future improvements are an adaptive mechanism for crossover operators, destroy-repair operators of the NEP, and the k parameter of the PIM. In addition to this, combinations with other local movements along with our 2-opt approach and *split algorithm*.

Concerning the performance, it is in our plans a migration from python to C++ to enhance the performance. We also intend to extend the method for other VRP problems.

References

1. Abdallah, M.B., Ennigrou, M.: Hybrid Multi-agent Approach to Solve the Multi-depot Heterogeneous Fleet Vehicle Routing Problem with Time Window (MD-HFVRPTW). Advances in Intelligent Systems and Computing **923**, 376–386 (2020)

2. Bortfeldt, A., Yi, J.: The Split Delivery Vehicle Routing Problem with three-dimensional loading constraints. European Journal of Operational Research **282**, 545–558 (2020)
3. Christofides, N., Mingozzi, A., Toth, P.: The vehicle routing problem. Combinatorial optimization p. 315–338 (1979)
4. Dantzig, G.B., Ramser, J.H.: The truck dispatching problem. Manage. Sci. **6**, 80–91 (Oct 1959)
5. Davis, L.: Applying adaptive algorithms to epistatic domains. In: Proceedings of the 9th International Joint Conference on Artificial Intelligence - Volume 1. p. 162–164. IJCAI'85, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1985)
6. Esposito Amideo, A., Scaparra, M.P., Kotiadis, K.: Optimising shelter location and evacuation routing operations: The critical issues. European Journal of Operational Research **279**(2), 279–295 (December 2019)
7. Goldberg, D.E., Lingle, R., et al.: Alleles, loci, and the traveling salesman problem. In: Proceedings of an international conference on genetic algorithms and their applications. vol. 154, pp. 154–159. Lawrence Erlbaum, Hillsdale, NJ (1985)
8. Helsgaun, K.: General k-opt submoves for the lin–kernighan tsp heuristic. Mathematical Programming Computation **1**, 119–163 (10 2009)
9. Holland, J.H.: Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence. anas (1975)
10. Li, J., Li, Y., Pardalos, P.M.: Multi-depot vehicle routing problem with time windows under shared depot resources. Journal of Combinatorial Optimization **31**, 515–532 (2014)
11. López-Ibáñez, M., Dubois-Lacoste, J., Cáceres, L.P., Birattari, M., Stützle, T.: The irace package: Iterated racing for automatic algorithm configuration. Operations Research Perspectives **3**, 43 – 58 (2016)
12. Nagata, Y., Bräysy, O.: Edge assembly-based memetic algorithm for the capacitated vehicle routing problem. Netw. **54**(4), 205–215 (Dec 2009)
13. Oliver, I., Smith, D., Holland, J.R.: Study of permutation crossover operators on the traveling salesman problem. In: Genetic algorithms and their applications: proceedings of the second International Conference on Genetic Algorithms: July 28-31, 1987 at the Massachusetts Institute of Technology, Cambridge, MA. Hillsdale, NJ: L. Erlhaum Associates, 1987. (1987)
14. Pisinger, D., Ropke, S.: A general heuristic for vehicle routing problems. Comput. Oper. Res. **34**(8), 2403–2435 (Aug 2007)
15. Potvin, J.Y., Bengio, S.: The vehicle routing problem with time windows part ii: Genetic search. INFORMS J. on Computing **8**(2), 165–172 (May 1996)
16. Prins, C.: A simple and effective evolutionary algorithm for the vehicle routing problem. Computers & Operations Research **31**(12), 1985 – 2002 (2004)
17. Rabbouch, B., Saâdaoui, F., Mraihi, R.: Efficient implementation of the genetic algorithm to solve rich vehicle routing problems. Operational Research (2019)
18. Shi, Y., Boudouh, T., Grunder, O.: A hybrid genetic algorithm for a home health care routing problem with time window and fuzzy demand. Expert Systems with Applications **72**, 160 – 176 (2017)
19. Syswerda, G.: Schedule optimization using genetic algorithms. In: The Genetic Algorithms Handbook (1991)
20. Vidal, T., Crainic, T.G., Gendreau, M., Prins, C.: A hybrid genetic algorithm with adaptive diversity management for a large class of vehicle routing problems with time-windows. Computers and Operations Research **40**, 475–489 (2013)

21. Whitley, L.D., Starkweather, T., Fuquay, D.: Scheduling problems and traveling salesmen: The genetic edge recombination operator. In: Proceedings of the 3rd International Conference on Genetic Algorithms. p. 133–140. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1989)
22. Zhang, S., Chen, M., Zhang, W., Zhuang, X.: Fuzzy optimization model for electric vehicle routing problem with time windows and recharging stations. Expert Systems with Applications **145** (2020)
23. Zhen, L., Ma, C., Wang, K., Xiao, L., Zhang, W.: Multi-depot multi-trip vehicle routing problem with time windows and release dates. Transportation Research Part E: Logistics and Transportation Review **135**, 101866 (2020)