Cellular Genetic Algorithm for Solving a Routing On-Demand Transit Problem

Olfa Chebbi
Higher Institute Of Management Of Tunis 41, Rue de la Liberté 2000 Bardo, Tunisia
olfaa.chebbi@gmail.com

Ezzeddine Fatnassi
Higher Institute Of Management Of Tunis 41, Rue de la Liberté 2000 Bardo, Tunisia
ezzeddine.fatnassi@gmail.com

Jouhaina Chaouachi
IHEC Carthage Présidence-2016 Tunis, Tunisia
siala.jouhaina@gmail.com

Nouha Nouri
Higher Institute Of Management Of Tunis, Tunisia
nouri.nouha@yahoo.fr

ABSTRACT

To provide sustainable and efficient urban logistics and transportation services, urban mobility tools are facing challenges on reducing carbon emission, waiting time for passengers and transit time. The emergence of many new intelligent and electric transportation system offers many new possible solutions to achieve urban sustainability. This paper proposes to treat the Personal Rapid Transit System (PRT) as an efficient sustainable transportation tool for urban areas. This paper proposes to deal with static problem of routing PRT'vehicles to minimize total energy consumption while considering the battery capacity of vehicles. For that purpose, we describe a multiple crossover Cellular Genetic Algorithm combined with a local search. Numerical experiments on 1320 instances show that our hybrid algorithm is efficient in which the average percent deviations relative to the lower bound over 1320 instances is about 1.632%, and the average running time is about 26.3 seconds.

Categories and Subject Descriptors

F.2.2 [**Artificial intelligence**]: Problem Solving, Control Methods, and Search —*Heuristic methods*; I.2.9 [**Artificial intelligence**]: Robotics—*Autonomous vehicles*

Keywords

Personal Rapid Transit, Evolutionary Computation, Vehicle Routing Problem, Cellular Genetic Algorithm

1. INTRODUCTION

Due to the increasingly greenhouse gas emissions and serious environmental deterioration, sustainable development is becoming a vital issue facing urban areas. Both environmental regulations and customer pressures have driven cities to adopt a more eco-friendly transportation systems. Among them, one could consider the Personal Rapid Transit (PRT or podcars).

PRT is a new public transportation system that has the main features of private cars as it uses small, electrical, and fully automated vehicles for moving people. PRT provides a direct nonstop transportation service from departure to destination for the users of this system. This is due to the stations located off the main line avoiding the pods to stop at intermediate stations. As it is fast and comfortable, PRT has many advantages compared with other forms of public transportation. Unlike buses, trains, or subway trains, PRT can serve individual passengers, and each vehicle can be shared only by a group of people who know each other. This feature offers a high level of privacy and security for the users. PRT is also an on-demand service, which, unlike traditional public transportation modes, runs on the specific demands of the passenger: the passenger arrives at a PRT station and then chooses his destination station. The PRT central control system will automatically assign an empty free vehicle to him.

Unfortunately, this kind of system that uses the on-demand service can result in a high level of empty vehicle displacement because of empty vehicles waiting or moving to take passengers. This feature can result in a high level of energy wastage.

This is further compounded by the additional empty moves to charge the vehicle's battery in the depot.

In the literature, different optimization problems related to PRT exist. One could note works related to the minimization of waiting time of passengers [15], traveled distance [7], fleet size [6], simulation [10], operational planning [8], and so on. However, we should note that there is few works in the literature that focused on real energy minimization for such an intelligent transportation system as PRT [11], [16].

In this paper, we consider the static problem of minimizing the total energy consumption of the PRT system such that each passenger's demand must be satisfied, each passenger is served by exactly one vehicle, and each vehicle's route begins

GECCO '16, July 20-24, 2016, Denver, CO, USA

DOI: http://dx.doi.org/10.1145/2908812.2908921

and ends at the depot. This static problem can be useful for giving lower bounds to evaluate dynamic strategies used for routing PRT vehicles. This static problem could also be used within stochastic dynamic strategies for routing PRT vehicles. In fact, it exists in the literature different strategies that use the solution giving by a static problem. Finally, this static problem could also be used for routing PRT vehicle in the schedule mode of operation as in the Morgan town PRT system. This schedule mode generally sends vehicles based on a predefined pairs of origin destination'passenger demand. The schedule mode of operation is used mainly on the rush hours where the passengers demand is highly predictable.

The considered problem represents an interesting, novel and emerging issue for PRT. The PRT problem treated in this paper was studied only in the work made by Mrad and Hidri [17] in which they solved the PRT problem using an exact method and by Mrad et al. [16] in which they used a constructive heuristic to solve this problem. However, we should note that the exact method introduced in [17] consumes a huge amount of computational time and is able to only solve small size instances up to 100 trips. The method introduced in [16] could solve large size instances but provide low quality solutions. Therefore, **the objective of this paper is to provide a solution approach capable of finding good quality solutions for large size instances in a small computational time**.

In this paper, we focus on implementing a specific metaheuristic Parallel Genetic Algorithm (PGA)[4] namely a cellular genetic algorithm (CGA). PGA is a class of evolutionary algorithms in which the different individuals in the population evolve in parallel. Although, the use of metaheuristics helps to diminish the computational time during the optimization process, the exploration of the search space remains time-consuming for many real life problems. That is why the use of a parallel structure is considered in this work. Using parallel algorithms has the advantage to reduce the resolution time and to improve the quality of final obtained solutions since the search process is conducted in multiple areas in the search space at the same time. The development of a CGA has a goal to provide a tool for finding high quality solutions for the PRT routing problem to minimize its total energy consumption.

Since the introduction of CGA algorithm, it has gained much popularity because of its robust mechanism and easy implementation [3]. However, until now there are few studies of applying CGA algorithm into green logistics and public transportation management areas. Consequently, this research is a pioneering attempt for the integration of PGA, green logistics and public transportation. Differing from the original and basic CGA, we introduce a specific multiple crossover CGA algorithm by incorporating specific PRT related operators such as an intensification strategy, a linear programming module and multiple crossover operator within the basic CGA for the PRT problem. The computational performance of the proposed multiple crossover CGA (MCCGA) is measured and proved to be very effective.

The contributions of this paper are several:

- This paper proposes a PGA namely a MCCGA which is proven to be effective and efficient in solving the PRT problem. It is a pilot attempt of applying PGA into green logistics and public transportation.
- The paper also studies the close relation between the presented problem and an important variant of the vehicle routing problem (VRP) that is present in many real world applications but not well studied in the literature called the asymmetric distance constrained vehicle routing problem (ADCVRP).
- This paper also analyzes the performance of the proposed algorithm through a rigorous statistical study and sensitivity analysis tools.

The remainder of this paper is organized as follows: in Section 2, we define and model the PRT problem. In Section 3, we present our MCCGA optimization framework. The computational results are presented and discussed in Section 4. Section 5 concludes the paper.

2. PRT PROBLEM DEFINITION

In this section, we present the problem definition as defined in the works made by Mrad and Hidri [17], Chebbi and Chaouachi [7] and Mrad et al. [16]. We suppose that all the decisions related to the construction and the organization of a PRT are already made. We focus exclusively on the operational level of decisions related to PRT system as we intend to minimize the total energy consumption of the whole system. A PRT system has a guideway network (GN) and an unlimited number of homogenous PRT vehicles that have a battery capacity (B). B is expressed as the maximum allowable running time for a PRT vehicle. The GN has M stations, one depot D and enough guideways to make the moves between each couple of stations possible. We suppose that when needed, the vehicles should return to the depot to charge the battery. We suppose that initially an unlimited number of PRT vehicles are present in D. The exact number of vehicles that the PRT system needs would be expressed as a decision variable. We note $COST$ as the matrix representing the energy consumption cost of the direct link between each couple of stations. We note SP as the matrix representing the total traveled time between each pair of stations.

The problem treated in this paper is based on the assumption that we have a list of a pre-scheduled trips T [7]. Each trip t_i is defined by its original station OS_i, destination station DS_i, departure time DT_i, arrival time AT_i. In order to ensure a high quality of transportation service, we suppose that no waiting time is allowed for PRT'users. Consequently, a PRT vehicle has to be present in the departure station of the passengers' trip before their arrival or at worst at the same time as they arrive to their departure station.

The PRT problem is defined on an asymmetric graph $G = \{\vec{V}, \vec{E}\}$ where $\vec{V} = \{v_0, v_1, v_2, ..., v_n, s, t\}$ is a set of nodes on which $v_0, v_1, v_1, v_2, ..., v_n$ define the different trips that the PRT system should cover and s and t are two dummy nodes that define the depot. We also define $\vec{V}^* = \vec{V} \backslash \{s, t\}$. As a set of arcs, we define $\vec{E} = \{(v_i, v_j) : v_i, v_j \in \vec{V}\}$ as follows:

- If $v_i, v_j \in V^*$ with $AT_i + SP_{(DS_i, OS_j)} \leq DT_j$ then the arc (i, j) exists and has c_{ij} as a cost, which represents the energy consumed from the arrival station of the trip i to the departure station of the trip j in addition to the cost of serving trip j.
- For each node $i \in V^*$, we add an arc (s, i) that has the cost of the consumed energy from the depot to the

departure station of the trip i in addition to the cost of serving trip i.

- For each node $i \in V^*$, we add an arc (i,t) that has as cost the consumed energy from the arrival station of the trip i to the depot.

The graph G on which the problem is defined is directed and non-complete: if the arc (i,j) exists, the opposite arc (j,i) does not exist. Thus, the adjacency matrix related to the graph G will be sparse with a low sparsity rate. A sparse matrix is a matrix populated mostly with zeros. The sparsity rate represents the percentage of the non zero elements on that matrix. We should note also that our problem can be assimilated to the asymmetric distance-constrained vehicle routing problem (ADCVRP).

The distance constrained vehicle routing problem is a variant of the capacitated vehicle routing problem (CVRP). On the DCVRP, we replace the capacity constraint that exists on the CVRP by a maximum distance, time, cost constraint imposed for each route (the battery capacity in our case). Our problem is asymmetric as the distance from any node i to any node j is different than the opposite distance from j to i. The ADCVRP is proven to be NP-Hard [5].

In the literature, only few works studied the ADCVRP. We could cite for instance the work of Laporte et al. [14] and Almoustafa et al. [5] where a ADCVRP formulation and exact solution approach to solve it are proposed. The main difference between our problem and the ADCVRP is the low sparsity rate of our graph. In fact for two nodes i and j if the arc (i,j) exists the arc (j,i) doesn't exists.

In the next, we define a specific meta-heuristic approach based on the CGA that combines linear programming techniques, local search and various crossover operators for solving our PRT problem. Our decision to choose a meta-heuristic approach was based on the fact that it is the most suitable algorithm for solving NP-Hard problems.

3. THE MULTIPLE CROSSOVER CELLULAR GENETIC ALGORITHM

The cellular evolutionary algorithm defines a set of evolutionary algorithms where an individual mates with its close neighbors in a structured population. Among cellular algorithms, one can pay special attention to the CGA. The CGA was applied successfully for solving different problems[2],[3]. The specific version of the CGA developed namely MCCGA in this paper (see Algorithm 1) can be considered as a contribution to the literature related to CGA. In fact, this algorithm is a first attempt to use and apply specific CGA for solving green logistics and on-demand transportation related routing problems.

In this section, we present the main features of the MCCGA to solve the PRT problem presented in this paper.

3.1 Solution Representation and evaluation function

One of the most important issue related to the design of genetic algorithm is the representation of each individual (chromosome) in the population. In the vehicle routing problem (VRP) literature, there exists two main principles for solving such a problem namely the cluster-first route-second principle and the route-first cluster-second principle

Algorithm 1 Multiple Crossover Cellular Genetic Algorithm (Network Topology of PRT: GN , List of Trips to serve: T)

Construct the graph G based on GN and T
Initialize-parameter()
while Not reach termination criterion **do**
  for $x = 1 \rightarrow WIDTH$ **do**
    for $y = 1 \rightarrow HEIGHT$ **do**
      p1 = Individual-At(x,y).
      p2 = Select-From-Neighborhood(x,y).
      offspring1 =One-point crossover form1(p1,p2).
      offspring2 =One-point crossover form2(p1,p2).
      offspring3 =Two-points crossover form1(p1,p2).
      offspring4 = Two-points crossover form2 (p1,p2).
      offspring5 = Three-points crossover (pt1,p2).
      Apply the mutation operator on the five generated offpsrings
      Recombining The generated offsprings
      Evaluate The five generated offsprings
      if one of the five new generated offsprings is better than Individual-At(x,y) **then**
        Insert The best new offspring in the new population popaux at position (x,y).
      else
        Insert $Individual - At(x,y)$ in the new population popaux at position (x,y).
      end if
    end for
  end for
  pop = popaux
  Improvement-procedure(pop)
end while
individual = Best-individual(pop)

[20]. We based our work on the second principle of route-first cluster-second as it offers many advantages compared to other classical approaches for solving VRP [20].

Consequently and in our proposed algorithm, a solution (individual) will be represented as a permutation of trips.

As for the evaluation of different individuals in our algorithm, we used and adapted the Split function of Prins [19] used initially for the CVRP. The use of the Split function as an evaluation method for the permutation based algorithms to solve VRP is known to be an effective and efficient approach [20].

This function has the ability to get an optimal split of a given permutation into a set of feasible routes subject to some constraints. In our algorithm, we submit each individuals when needed to the adapted Split function in order to extract the least cost split option subject to the battery constraint in our PRT system.

The Split function founds the optimal splitting of a giant tour by computing a shortest path in an auxiliary graph H.

More specifically, the Split function constructs a graph $H = (Y, U)$ where Y is the set of nodes and U is the set of edges. Starting from a giant tour S composed by n trips, the Y set contains $n + 1$ nodes numbered from 0 to n. The 0 node represents a dummy node and the other nodes starting from 1 to n represent the n trips for the PRT system. Each feasible subsequence in S $(y_i, y_{i+1}, y_{i+2}, ..., y_k)$ is modeled by an edge in the set U. The cost of this edge is the cost of the road starting from the depot covering the nodes $i, i + 1, i + 2, ..., k$ and returning back to the depot. After constructing this auxiliary graph, the Split function computes the shortest path in H starting from node 0 to node n to find the optimal split option of S. More details could be found in [19].

3.2 Initial population

The initial population consists of a set of randomly generated individuals in addition to a structured individual. In fact, only one individual in the first population will result from solving the following linear algorithm: Let us first introduce the following integer variables:

$$x_{ij} = \begin{cases} 1 & \text{if node } j \text{ is visited after node } i \\ 0 & \text{Otherwise} \end{cases}$$

c_{ij} is the cost in the graph G of moving from node i to node j. $\delta^+(i)$ is the set of edges that have i as a root.

$\delta^-(i)$ is the set of edges that have i as a sink. Let define also c_{ij} that represents the cost of moving from node i to node j in the graph G that problem is defined on.

$$\textbf{PRT}: \qquad \text{Min} \sum_{(i,j)\in E} c_{ij} x_{ij}$$

$$\sum_{j\in\delta^+(i)} x_{ij} = 1 \forall i \in V^*$$

$$\sum_{j\in\delta^-(i)} x_{ji} = 1 \forall i \in V^*$$

$$x_{ij} \in \{0,1\} \forall \quad (i,j) \in E$$

This linear algorithm will generate a set of roads starting and ending at the depot that will not for sure respect the battery capacity. However, the summing up of all the obtained roads together while using the Split function of Prins [19] can represent a good starting point for our algorithm. We call the solution obtained from this linear program SLP. As an illustrative example, let suppose that the relaxed linear program generates a set of roads as shown in Figure 1. The relaxed linear program has resulted of 4 roads among them 2 infeasible roads. To generate the SLP solution, we concatenate the different roads to form a one single permutation that cover al the different trips. Then, we used the Split function on the obtained permutation. This results on a new feasible solution and ensures to generate the right and feasible evaluation from it.

3.3 Neighborhood

The neighborhood of each solution will always contain four individuals: the North, the East, the West, and the South (WEST)[3]. Thus for each evolutionary step, each individual will be recombined with one of its neighbors chosen randomly.

3.4 Crossover and mutation

In the proposed MCCGA, the crossover and mutation operators are applied with respective probabilities P_C and P_M.

Unlike traditional CGA, we decided to extend the cellular memetic algorithm by implementing a multiple crossover cellular genetic algorithm (MCCGA). In our algorithm and when it comes to performing a crossover operator, the MC-CGA, will apply simultaneously five different crossover operators.

Generally, one could note that the crossover is a variation operator that imitates natural reproduction by exchanging genes from two parents to generate new, fresh, and more

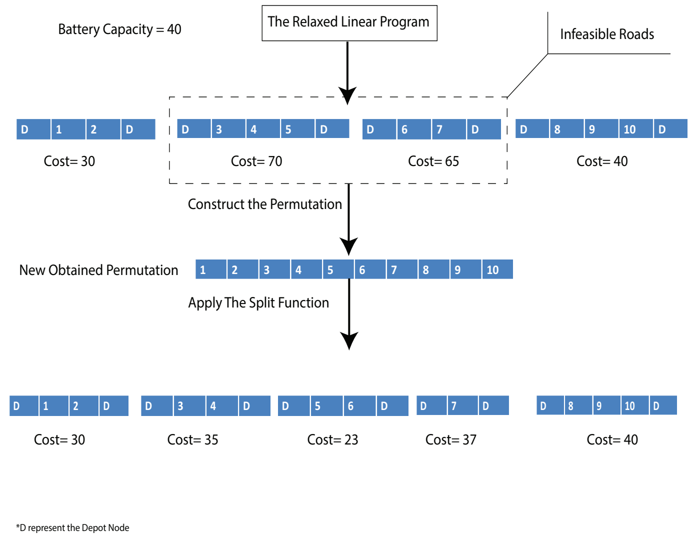


Figure 1: Example of the generation of our SLP Solution

enhanced offsprings. In permutation problems, the fact of simply copying genes from parents to offsprings could result on infeasible solutions that have redundant genes. Therefore, a modification has to be made to guarantee the feasibility of our offsprings. The advantage of using a multiple crossover operator scheme is its possibility to perform more searches than the single crossover scheme. In fact, in the multiple crossover algorithm, each crossover operator will guide the search process into a different area in the neighborhood of the two selected parents. Thus, the multiple crossover scheme applied on the cellular population could yield a more effective search process than the basic CGA. The crossover operators used in this paper are as follows:

- One Point crossover: The one-point crossover works by cutting one of the two selected individuals once at a specific point. The child receives the genes before the cutting point, and the other missing genes are copied from the second parent according to their order of appearance in the second parent. This is the form 1 of our one-point crossover. The form 2 works by copying to the offspring the genes after the cutting point and the other missing genes are copied from the second parent according to their order of appearance.

- Two Points crossover: A part from the one-point crossover, the two-point crossover involves having two cutting points. Thus, we get three different subsequences. The form 1 of this crossover works by copying the first and third sequence to the offspring. The missing trips will be obtained from the second parent while respecting their order of appearance. The form 2 of the two-point crossover works by copying the trips between the two cutting points in the first parent to the offspring and copying the missing trips from the other parent.

- Three Points crossover: This operator works almost as the same as the two-point crossover. Three cutting points are chosen randomly. Then, we obtain a parent divided into four different subsequences. The offspring will get the first, and the third subsequence and the missing genes are copied from the second parent according to their order of appearance.

Thus, we get five different offsprings. Later, we apply the mutation operator on those five offsprings with a probability P_M. Among the five new offsprings, only the best one will be a candidate for the insertion in the population.

For the mutation operator, we use the insertion operator. Based on a permutation, this operator takes at random a gene and insert it at a random position.

3.5 Improvement procedure

To enhance the performance of our MCCGA (Algorithm 1), we integrate an improvement technique at the end of the evolutionary process (the end of each generation). This improvement technique consists on applying a local search procedure (see Algorithm 2) at the end of the evolutionary process.

Algorithm 2 Local search Procedure

$y = RAND(0, 1)$
if $y < 0.5$ **then**
  for all Individuals in the population **do**
    Generate 100 neighbors using Exchange-Mutation or Displacement-Mutation
    Insert the best neighbor in the population
  end for
else
  for all Individual X in the population **do**
    for $i = 1 \rightarrow$ Size-Problem **do**
      Select Operator Exchange-Mutation or Displacement-Mutation Randomly
      $X_{new} = Operator(X)$
      Intensification-Strategy(X_{new})
      if X_{new} is better than X **then**
        $X = X_{new}$
      end if
    end for
  end for
end if

The exchange mutation operator chooses two genes at random at a given permutation and exchanges their position. The displacement mutation operator chooses a subsequence of genes in a given permutation and changes its location in the solution.

3.6 Enhancing procedure

As our graph has a low connection rate, the quality of the solution declines. This specific feature results for some permutations in many roads with a low frequency of using the battery because of successive trips i and j in the permutation are without an arc between them. To overcome this constraint, we use a self-recombination method within the evaluation function.
Let us suppose that each generated road from the Split function has a head trip H that represents the starting node of the road and a queue trip Q that represents the ending node in the road and let us suppose also that D is the depot. This method will try to put the generated roads that fit together to enhance the solution's quality by minimizing the interconnection links between the different road's.

The used self recombination method is explained in the Algorithm 3.

Algorithm 3 self recombination method (Solution S)

Apply the Split function on S
$nb{-}Road$ = number of the generated roads from S using the Split function
for $i = 1 \rightarrow$ nb-Road **do**
  for $j = 1 \rightarrow$ nb-Road **do**
    R_1 = Choose A Road At Random
    R_2 = Choose A Road At Random
    if $cost(Q_{R_1}; H_{R_2}) < cost(Q_{R_1}; D) + cost(D; H_{R_2})$ **then**
      $Append(R_1, R_2)$
    end if
  end for
end for

4. EXPERIMENTAL TEST

The different algorithms described in this paper were coded in C++ language. The experiments are performed on a laptop with Intel i5 2.3 GHZ processor[1], Windows 7, and 6 GB memory.

4.1 Research Questions

In this section, we present our research questions. Let us recall that our objective is to develop an PGA to solve large instances of PRT' energy minimization problem in a small computational time. Our research questions (RQ) are as follow:

1. RQ1: Is the MCCGA able to make an efficient search to find good quality solution for our problem?

2. RQ2: What is the impact of each component of the MCCGA on the overall performance of the algorithm?

3. RQ3: Does the MCCGA present a robust approach to solve our problem?

4. RQ4: How the MCCGA perform in comparison with other traditional solution approaches such as genetic algorithm and other methods from the literature to solve the same treated problem?

4.2 Test Instances

We tested our algorithm on 1320 instances of Mrad et al.[17]. These instances are based on the PRT' instances generator of the literature taken from the work of Mrad and Hidri [17]. The size of these instances ($nb - trip$) varies from 10 to 400 trips. For each class's size, 40 instances are generated.

4.3 Quality Metrics

To define the quality of the proposed heuristics, we based our analysis on the following measures:

[1]The used processor is equipped with four independent actual processing units which make it perfect for testing our proposed parallel MCCGA. In fact, the code was developed in C++ while taking into account the parallel aspect of the MCCGA in order increase its speed.

- the GAP is defined as follows:

$$GAP = (\frac{(SOL_{heuristic} - LB)}{LB}) \times 100$$

For each instance, we take the LB as the linear relaxation of the mathematical model of Kara given in [5] for the resolution of the ADCVRP problem coded using Cplex12.1.

- Time: The time that the algorithm used to reach its results.

4.4 Parameter Tuning

Parameter tuning is a very important issue for heuristics in general. In fact, a good parameter tuning procedure could enhance the performance of our algorithm [9]. In this paper, we used a parameter tuning method inspired from the work of Nguyen et al. [18]. The parameter tuning was made on the set of instances of size 200 trips. We used this relatively hard testing instances' bed as it represents an interesting challenge for our algorithm. Therefore, the obtained parameters set is as follow:

- population size: 16 (4×4).
- $max_{generations}$: 1000.
- $max_{generationsWithoutImprovement}$:200.
- crossover rate : 0.9.
- mutation rate: 0.3.

4.5 Computational Results

In this section, we present the computational results of the MCCGA presented in this paper. For each class we present a value representing the average among the instances within each class. Results are shown in Table 1.

We should note the good performance of our MCCGA as it was able to find an average GAP of 1.632 in 26.437 seconds.

4.6 Statistical Analysis of The Results

We decided to compare the results of our proposed MCCGA against the ones of the basic version of a CGA. Therefore, we developed a basic version of CGA where we use only the one point crossover instead of the multiple crossover operator proposed by the MCCGA. We tested the basic CGA on the same set of instances as the MCCGA. We used also the same parameters of the MCCGA for the basic CGA. Our tests for the basic version of CGA on the same set of 1320 PRT instances reach an average Gap of 2.106 % in 24.287 seconds. Because we tested our algorithms on the same instances, we decided to do further comparative tests by doing further statistical analysis. We first need to test the normality of the input data. For this purpose, we used the Shapiro-Wilk normality test (S-W test), Kolmogorov-Smirnov normality test (K-S test), and D'Agostino-Pearson test. All these tests were not successful ($P - value < 0.0001$). Hence, we conclude that the input data are not normally distributed. Thus, we use the Wilcoxon matched-pairs signed-rank test to enrich our comparative study [21]. The Wilcoxon matched-pairs signed-rank test is a non-parametrical procedure employed in a hypothesis testing to compare two paired groups. In this test, H0 implies that the percent deviation of the results of the two algorithms are the same, whereas the H1 hypothesis implies that the average deviation of the first algorithm is lower (better) than the second algorithm.

Table 1: Results of the MCCGA

$nb - trip$	Average Gap %	Average Time(sec)
10	1.105	0.651
15	0.799	0.891
20	0.687	1.209
25	1.110	1.564
30	1.063	2.009
35	1.470	2.366
40	1.207	2.766
45	1.099	3.289
50	1.142	3.821
55	1.563	4.594
60	1.029	4.941
65	1.357	6.117
70	1.796	6.412
75	1.419	7.454
80	1.618	7.900
85	1.576	9.011
90	1.512	9.879
95	1.751	10.789
100	1.791	11.695
110	1.653	13.807
120	1.860	16.341
130	2.266	18.562
140	1.472	20.994
150	1.563	23.743
160	2.088	27.919
170	1.928	30.666
180	1.777	34.519
190	2.067	39.649
200	2.287	43.382
250	1.906	68.090
300	2.346	100.451
350	2.594	142.811
400	2.954	194.120
Average	1.632	26.437

From the results of the Wilcoxon matched-pairs signed-ranks test[2], we can conclude that the MCCGA outperforms the basic CGA because the P-value for the different comparisons is less than 0.05. Therefore, we clearly see that, in terms of the results, the MCCGA algorithm is more efficient than the basic CGA. The basic CGA suffers from the fact that it focuses more on the local search procedure which would not propose a good balance between exploration and exploitation procedures during the search process. The MCCGA, by generating five parallel offsprings at each evolutionary step, performs the global search more intensively than the basic version of CGA.

4.7 The impact of the population seeding

In this section, we want to discover the impact of the population seeding on the performance of the algorithms. To that aim, a lesion study of the initial population procedure has to be conducted to make a direct comparison between the MCCGA with and without the integration of the SLP solution. A lesion study is that in which one component of the algorithm is disabled to know its contribution on the final results. Figure 2 reports the effect of the integration of the SLP solution in the initial population. We should note that in Figure 2 simple MCCGA refers to the MCCGA version without the SLP solution and the MCCGA refers to our proposed algorithm.

[2]Based on this test, we found a $P-value$ equals to < 0.0001 which means that the two algorithms found statistically different results.

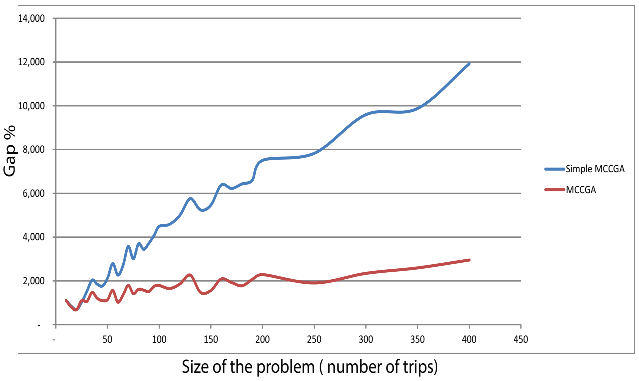


Figure 2: Results of the Lesion Study.

The lesion study suggests that our population-seeding procedure plays an important role in the achieved performance. As predicted, the incorporation of the SLP solution has the effect of accelerating the convergence rate and moving the algorithm faster to get high-quality solutions.

4.8 Run Time Analysis

While analyzing the efficiency of our heuristic, one should focus also on the running time of the basic CGA and the MCCGA. We decided to focus on the speed of the two algorithms as a function of the size of the problem. For that purpose, we conducted a correlation test to see if a statistical relation between the problem's size and the running time exists. From the obtained results of the non parametric Spearman correlation test, we could clearly state that a relation between the running time and the size of the problem exists. For more extensive analysis, we used a cubic fits equation to estimate the running time precisely as a function of the size of the problem (Figure 3).

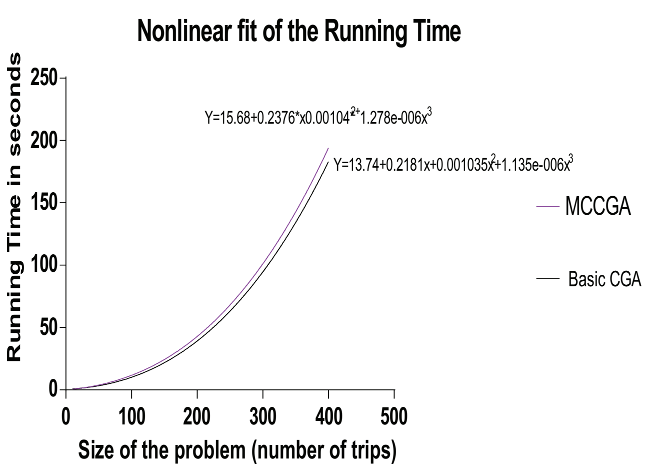
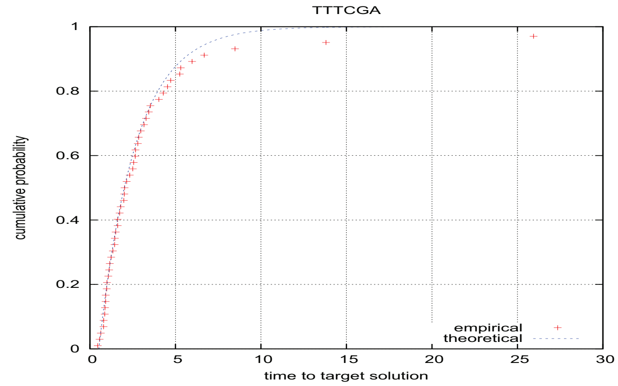
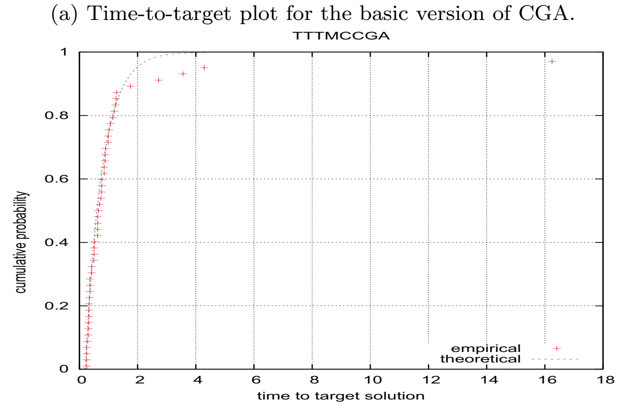


Figure 3: Non Linear Regression of the running time

As a post analysis for the non linear regression, we performed a Runs test. From this final analysis, we could note that the cubic-fit estimates of the running time are more precise for the MCCGA. As an explanation for the complexity of our algorithm, we could focus on the split function that has a complexity of $O(n^2)$[19].



(a) Time-to-target plot for the basic version of CGA.



(b) Time-to-target plot for the MCCGA.

Figure 4: Time-to-target plot for the CGA and the MCCGA.

For final analysis of the CPU run time, we used a methodology for the graphical analysis to compute the theoretical and empirical distribution of the random variable time to target value for our algorithms. We used in this paper the Time-to-target plots methodology. It was first introduced by Feo et al. [12] as a runtime distributions analysis for combinatorial algorithms. This method have been advocated in the combinatorial optimization literature by Hoos and Stützle [13] as an effective way to study the running times of heuristics algorithms. For that purpose, we run each algorithm 50 times on a fixed instance while using a random seed and record the time used to reach a fixed target solution.[3]. Figure 4 presents the empirical and theoretical distribution of the random variable time to target solution.

It clearly shows the efficiency of the MCCGA in comparison to the basic CGA. In fact, the MCCGA has a probability of 0.75 of finding the target value in 1 second, whereas the same probability is 0.2 for the basic CGA.

4.9 Comparison with other methods from the Literature

Finally, we wanted to compare our MCCGA against other methods from the literature. Consequently, we compare our algorithm against a simple genetic algorithm (GA) and against the linear programming heuristics of Mrad et al. [16] used for solving this problem.

[3]We used in this study the Perl program developed by Aiex et al. [1] to analyze the generated data.

We used for this comparison the $ARPD$ metric which is computed as
$ARPD = (\frac{(SOL_{heuristic} - SOL^*_{heuristic})}{SOL^*_{heuristic}}) \times 100.$

We should note that the $SOL^*_{heuristic}$ is the best obtained value from the MCCGA, the method of Mrad et al. and the GA. Results of this comparison shows that the MCCGA outperforms the GA and the method of Mrad et al. [16] as it founds a small average ARPD of 0.0258% against 0.83% for the GA and 4.282% for the method of Mrad et al. [16].

5. CONCLUSIONS

This article deals with minimizing energy consumption for PRT. We first modeled our problem as an ADCVRP where each trip is represented as a node and the energy consumption as the cost of the arcs between them. With that model as our basis, we developed a MCCGA. Our approach is based on the route first cluster second principle used essentially for VRP problems. For that purpose, we developed and adapted the Split function as an evaluation method for different solutions in our algorithm. Numerical experiments made on 1320 instances highlight the effectiveness of our algorithm and show that the MCCGA performs better than the basic version of the CGA as it found an average gap of 1.632%. The PRT problem could be enriched by the different additional constraints that exist in the VRP literature such as heterogeneous fleet of vehicles, more tough time window and multiple PRT depots. We could also focus a future research on integrating more specific constraints related to the PRT system such as limited parking places in each PRT station or the stochastic and dynamic passenger demand.

6. REFERENCES

[1] R. Aiex, M. Resende, and C. Ribeiro. Ttt plots: a perl program to create time-to-target plots. *Optimization Letters*, 1(4):355–366, 2007.

[2] A. Al-Naqi, A. T. Erdogan, and T. Arslan. Dynamic fault-tolerant three-dimensional cellular genetic algorithms. *Journal of Parallel and Distributed Computing*, (0):–, 2012.

[3] E. Alba and B. Dorronsoro. Computing nine new best-so-far solutions for capacitated vrp with a cellular genetic algorithm. *Information Processing Letters*, 98(6):225 – 230, 2006.

[4] E. Alba and M. Tomassini. Parallelism and evolutionary algorithms. *Evolutionary Computation, IEEE Transactions on*, 6(5):443–462, 2002.

[5] S. Almoustafa, S. Hanafi, and N. Mladenović. New exact method for large asymmetric distance-constrained vehicle routing problem. *European Journal of Operational Research*, 226(3):386–394, 2013.

[6] O. Chebbi and J. Chaouachi. Optimal fleet sizing of personal rapid transit system. In *Computer Information Systems and Industrial Management*, pages 327–338. Springer, 2015.

[7] O. Chebbi and J. Chaouachi. Reducing the wasted transportation capacity of personal rapid transit systems: An integrated model and multi-objective optimization approach. *Transportation Research Part E: Logistics and Transportation Review*, In Press, 2015.

[8] E. Fatnassi, J. Chaouachi, and W. Klibi. Planning and operating a shared goods and passengers on-demand rapid transit system for sustainable city-logistics. *Transportation Research Part B: Methodological*, 81, Part 2:440 – 460, 2015. Optimization of Urban Transportation Service NetworksOptimization of Urban Transportation Service Networks.

[9] E. Fatnassi, O. Chebbi, and J. Chaouachi. Discrete honeybee mating optimization algorithm for the routing of battery-operated automated guidance electric vehicles in personal rapid transit systems. *Swarm and Evolutionary Computation*, 26:35 – 49, 2016.

[10] E. Fatnassi, O. Chebbi, and J. C. Siala. Evaluation of different vehicle management strategies for the personal rapid transit system. In *Modeling, Simulation and Applied Optimization (ICMSAO), 2013 5th International Conference on*, pages 1–5. IEEE, 2013.

[11] E. Fatnassi, O. Chebbi, and J. C. Siala. Comparison of two mathematical formulations for the offline routing of personal rapid transit system vehicles. In *The International Conference on Methods and Models in Automation and Robotics*, 2014.

[12] T. A. Feo, M. G. Resende, and S. H. Smith. A greedy randomized adaptive search procedure for maximum independent set. *Operations Research*, 42(5):860–878, 1994.

[13] H. Hoos and T. Stützle. On the empirical evaluation of las vegas algorithms-position paper. Technical report, Citeseer, 1998.

[14] G. Laporte, Y. Nobert, and S. Taillefer. A branch-and-bound algorithm for the asymmetrical distance-constrained vehicle routing problem. *Mathematical Modelling*, 9(12):857 – 868, 1987.

[15] J. D. Lees-Miller. Minimising average passenger waiting time in personal rapid transit systems. *Annals of Operations Research*, pages 1–20, 2013.

[16] M. Mrad, O. Chebbi, M. Labidi, and M. A. Louly. Synchronous routing for personal rapid transit pods. *Journal of Applied Mathematics*, 2014(1):1–8, 2014.

[17] M. Mrad and L. Hidri. Optimal consumed electric energy while sequencing vehicle trips in a personal rapid transit transportation system. *Computers & Industrial Engineering*, 79:1–9, 2015.

[18] V.-P. Nguyen, C. Prins, and C. Prodhon. A multi-start iterated local search with tabu list and path relinking for the two-echelon location-routing problem. *Engineering Applications of Artificial Intelligence*, 25(1):56–71, 2012.

[19] C. Prins. A simple and effective evolutionary algorithm for the vehicle routing problem. *Computers & Operations Research*, 31(12):1985 – 2002, 2004.

[20] C. Prins, P. Lacomme, and C. Prodhon. Order-first split-second methods for vehicle routing problems: A review. *Transportation Research Part C: Emerging Technologies*, 40(0):179 – 200, 2014.

[21] F. Wilcoxon. Individual comparisons by ranking methods. *Biometrics bulletin*, pages 80–83, 1945.