International Journal of Artificial Intelligence,
ISSN 0974-0635; Int. J. Artif. Intell.
2025, Volume 23, Issue Number 1
Copyright © 2025 [International Journal of Artificial Intelligence]

Metaheuristic solution for vehicle transportations to improve Moroccan logistics

Chakour Ilias, Otman Abdoun

ISISA Team, Faculty of Science,
Abdelmalek Essaadi University
Tetouan, Morocco
Emails: ilias2000.chakour@gmail.com, o.abdoun@uae.ac.ma

ABSTRACT

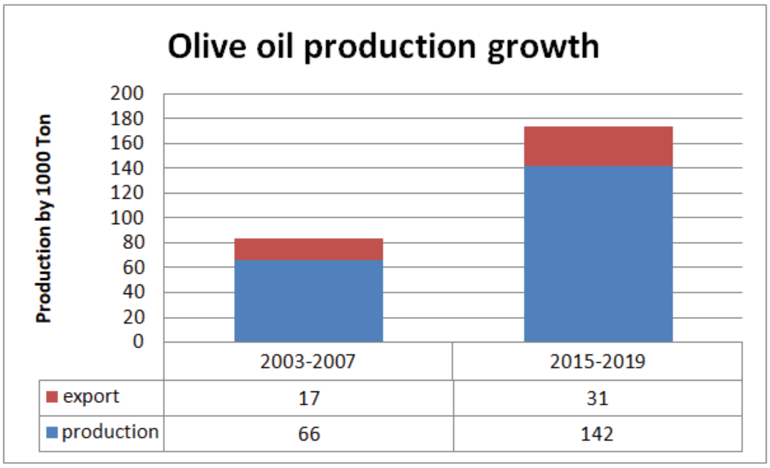
The objective of this study is to minimize transportation costs of the Moroccan supply chain and enhance the overall logistics performance of its different sectors. To achieve that we studied the Vehicle Routing Problem (VRP), which is a well-known problem, where multiple vehicles has to deliver goods to different customers efficiently, meanwhile the Pickup and Delivery Problem (PDP) is one of its variants where the task of each vehicle is to pickup goods and take them to their delivery location, which could be a representation of the supply chain transporting goods, especially since the founding of a new logistic zone in Morocco that offers transportation services. To calculate an optimized solution that would reduce the costs of the transportations, we used the cluster first, route second approach, we start by assigning a set of orders to each vehicle, which we consider a cluster, the assignment is done by using a hybrid of the genetic algorithm and a centroid based local optimization process, to further enhance the effectiveness of this stage, we used an immigration operator to delay the convergence of the solutions, we called the algorithm the Centroid Optimized Genetic Immigration Clustering (COGIC) after finishing this stage, we are left with multiple Travelling Salesman Problem (TSP) instances, we route each solution by using an insertion algorithm, this algorithm is very optimal computational time wise, we exploit its speed further by not only routing the solutions passed from the first stage, but also other solutions derived from them.

Keywords: Moroccan Logistics; Pickup and Delivery Problem; Optimization; Genetic Algorithm; Cluster First Route Second; Vehicle Routing Problem

Computing Classification System: •Mathematics of computing~Mathematical analysis~Mathematical optimization~Discrete optimization~Optimization with randomized search heuristics~Evolutionary algorithms

1. INTRODUCTION

The Moroccan transportation ministry has implemented a strategy of constructing logistic zones to enhance supply chain efficiency (Moroccan Agency for the Development of Logistics, 2024). This initiative has significantly benefited various industries, particularly the olive oil sector. Olive oil production saw a substantial increase from 66,000 tons in 2007 to 142,000 tons in 2019. However, this production growth also led to higher logistics costs. To mitigate these expenses, it is essential to minimize the travel distance for vehicles transporting goods, both from olive farms to factories and from factories to distributors. Additionally, the exportation of olive oil has experienced similar cost increases, with export volumes rising from 17,000 tons annually in 2007 to 31,000 tons in 2019 (Ministry of Agriculture, 2024).



	2003-2007	2015-2019
🟥 export	17	31
🟦 production	66	142

Figure 1. Olive oil production growth

A representation of the transportations of the supply chain could be a logistic zone offering the transportation services as a depot, and production locations as pickup points that must have their goods transported to distribution locations as delivery points, which is the same premise of the Pickup and Delivery Problem (PDP) a variant of the Vehicle Routing Problem (VRP), An NP-hard problem, where a fleet of homogeneous vehicles delivers goods from a depot to the customers (Toth et al., 2002), in the PDP variant (Savelsbergh et al., 1995), each order represent a pair of locations, the pickup and the delivery, the vehicle must carry the goods from the pickup to the delivery to complete the order, the request can be either goods or persons transportation, the latter is called dial a ride, other constraints could be added to the problem like capacity of vehicles, multi depot, and time window, among others.

Some of the popular PDP variants are the PDPTW (Dumas et al., 1995) that consider the time window constraint, where every pickup and delivery point can only be visited during a certain period, another popular variant is the DPDP that consider the dynamic assignment of orders after the vehicles departs, and they need to adjust their route while minimizing the travel costs (Li et al., 2021), there is also the PDPT that consider vehicles transfers, where the load that was picked up by a certain vehicle can be transferred to another one and be delivered by it (Sobotka et al., 2023).

Several studies treated the PDP problematic (Koç et al., 2020), some of them used reinforcement learning models (Wong et al., 2022), while other used deterministic methods (Küçük et al., 2019), and others used Metaheuristic hybrids of the genetic algorithm (GA) (Kasuma et al., 2022;Wang et al., 2021), in our study we did adopt the capacity constraint, and we considered one of the recently built logistic zones as the depot that sends its vehicles to do transportation of goods, the capacity of the vehicle is a critical criteria in the transportation of the supply chain. The solution that we propose to the problematic adopts the cluster first, route second approach, where we use in the first step the GA (Baker et al., 2003) with a centroid local optimization and an immigration operator (Tajani et al., 2017) to define the clusters to be visited by each vehicle, the objective function of this stage uses the centroid of the clusters to estimate their cost, and we try to optimize it. For the routing stage we used the insertion algorithm (Fargiana et al., 2022) for its efficacy, considering its execution time is very fast, we further exploited its speed by generating several solutions from to the ones in the population to have a higher chance of finding a more optimal solution.

The rest of this research is organized as follows: Section 2, we review the state of the art, examining several studies that have addressed the PDP and its variants, proposing various solution approaches. In section 3 we explained the studied problematic and its mathematical model. In Section 4, we proposed our solution approach, COGIC, along with the algorithms utilized. Section 5 demonstrates the effectiveness of COGIC on the Breedam benchmark (Dorronsoro Bernabé., 2024), comparing it to the solutions of Google OR-Tools (Google, 2024). Finally, we conclude with a summary and references.

2. STATE OF ART

The VRP aims to minimize the distance travelled by multiple vehicles that deliver goods to customers, and they all depart from the depot and return to it (Toth et al., 2002), one of it variant is the PDP, where every order is a pair of locations, and every vehicle must take goods from the pickups to their corresponding delivery, while respecting a capacity constraint (Savelsbergh et al., 1995).

Several studies studied the PDP (Koç et al., 2023), especially ones with the time windows constraint (Dumas et al., 1991), where every location must be visited during a period of time, and every location got a time service that must be summed up when visiting it. (Küçük et al., 2019) proposed an exact solution using a constraint programming model for the PDPTW, it was compared with algorithms in related literature on known instances of (Li & Lim, 2024) , among them (Ropke et al., 2006) that proposed an adaptive Large Neighborhood Search (LNS) for the PDP, and (Männel et al., 2016) that studied PDP with 3D loading constraints, using the LNS for the routing and a tree search for the packing, their results (Küçük et al., 2019) shows that their algorithm finds satisfying solutions, however the execution times vary drastically from a solution to another.

The transshipment variant of the PDP is widely studied as well, where a fleet of heterogeneous vehicles can do transshipment between distribution centers to optimize the distance traveled and ultimately reduce the costs, (Wang et al., 2023) proposed an algorithm that uses the neighborhood search to find a solution for it, where a priority constraint was considered on some orders, the experimental results showed a big improvement compared to CPLEX ones, and their computational time were very reasonable.

The study by (Zong et al., 2022) addressed the standard PDP using a reinforcement learning method, which involves three steps. First, they measured the dependency of different nodes. Next, they leveraged the decision dependence among various vehicles by using cooperative multi-agent decoders. Finally, they trained the integrated model using a cooperative A2C algorithm. Their experimental results outperformed other baselines, including (Nazari et al., 2019), who employed a framework for solving the VRP using reinforcement learning, and (Kool et al., 2019), who employed attention layers on a model that solves various graph-based problems, including the TSP and VRP. Additionally, (Zong et al., 2022) demonstrated significant computational speed during solution inference.

The study of (Al Chami et al., 2017) treated the PDPTW, considering a selective constraint where the visit of all locations isn't obligatory, the solution proposed was a hybrid of the GA and a local search based on swaps, their results were very good in small instances comparing them with CPLEX results, while bigger instances the results weren't compared but they were promising, and improvement ideas were proposed for it.

The research (Wang et al., 2021) explored the Multi-Depot PDP with Resource Sharing (MDPDPRS), involving distribution and pickup depots. These depots interchange resources, enabling vehicles to handle both pickup and delivery orders, also being able to conclude at a different depot from their original one. Additionally, they accounted for time windows by imposing penalties if not adhered to. To address these challenges, they developed a two-stage algorithm. Initially, they applied customer clustering using the k-means algorithm, followed by the routing. This hybrid approach combined k-means, Clark-Wright (CW), and NSGA-II algorithms. CW was utilized to construct the starting solution for NSGA-II in the routing stage. Benchmarking on C-mdvrptw datasets demonstrated the superiority of the proposed KCW-NSGA-II algorithm over standard NSGA-II and other algorithms.

The study by (Kusuma et al., 2022) addressed the standard PDP using a cluster-first, route-second approach. They employed a GA for the clustering and used the nearest distance for the routing. Their results indicated that their proposed model outperformed the comparison models, including those by (Wang et al., 2021), and (Andriansyah et al., 2019) who utilized a simulated annealing approach for the PDP with Last-In-First-Out (LIFO) constraints (Carrabs et al., 2007), time duration, and vehicle limits, where orders have time windows and the last pickup orders must be delivered first.

3. STUDIED PROBLEM

In our PDP problematic we have a map with a singular depot, and multiple orders, each order has two points on the map, pickups and a deliveries, each pair must be visited by the same vehicle, and the pickups must be visited prior to their corresponding delivery, every point in the map must be visited only once, at the visit of a pickup location, the vehicle must have the capacity to carry the goods on it, and when it arrive to the delivery point, the capacity used by its corresponding pickup point is freed.

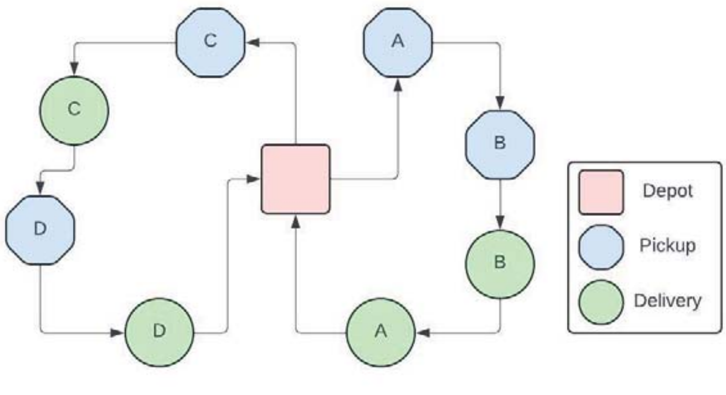


Figure 2. Pickup and delivery example

Every vehicle must depart and end on the depot location, and vehicles are homogeneous, they all have the same cost for the same travelled distance and the same capacity. In the figure 2 we can see that the pickup delivery pairs not necessarily need to be visited consecutively like in the A pair, but the capacity of the vehicle must be enough to carry goods from both the A and B pickups, otherwise he must visit them like in the C and D pairs, delivering the goods before picking new ones.

In our problematic we have a set of 2N orders composed of N pickup and N delivery nodes given in the instances, the initial node v_0 is the depot from where all vehicles depart and end, in our format we assume that every odd v_i is a pickup node, and the next consecutive v_{i+1} is its corresponding delivery node. We represent the Euclidian distances between different node as E = $\{e_{ij}\}$, where $0 \ll i, j \ll 2N$. For the capacity constraint we assume that every pickup node has a load volume d_i, and its corresponding delivery node has a load volume of $-d_i$. All the orders are assigned to vehicles with individual capacity c_k, and since we are working with a homogenous fleet of vehicles, the capacity is the same for all the vehicles. To indicate if a vehicle k goes directly from the node v_i to the node v_j we use $x_{ijk} \epsilon$ {0,1}, and to denote the arrival time we use T_i to the node v_i. To denote a consecutive routing sequence starting and ending at the node v_0, and does not include it in the middle we use S ϵ V. We can mathematically formulate it as follow:

$$\min \sum_{k=1}^{K} \sum_{i=0}^{2N} \sum_{j=0}^{2N} e_{ij} x_{ijk} \quad (1)$$

$$\text{s.t.} \sum_{k=1}^{K} \sum_{j=0}^{2N} x_{ijk} = 1, \forall i \in [0,2N] \quad (2)$$

$$\sum_{k=1}^{K} \sum_{i=0}^{2N} x_{ijk} = 1, \forall j \in [0,2N] \quad (3)$$

$$\sum_{i \in S'}^{a} d_i \le C_k, \forall S' \epsilon\, S, \forall k \epsilon [1, K] \quad (4)$$

$$\sum_{j=0}^{2N} x_{ijk} = \sum_{j=0}^{2N} x_{i+1,jk}, \forall k \epsilon [1, K], \forall i \epsilon [1,2N] \cap \{2x+1\} \quad (5)$$

$$T_i \le \ T_{i+1}, \forall i \epsilon [1,2N] \cap \{2x+1\} \quad (6)$$

The objective function (1) aims to minimize the total traveling distance of all the vehicles. Constraints (2) and (3) ensure that each location is visited, and exactly visited once. Constraint (4) ensures that no vehicle exceeds its capacity limit. Constraint (5) ensures that each pickup delivery pair is visited by the same vehicle, and constraint (6) ensures that pickups always are visited prior to their corresponding delivery.

4. PROPOSED APPROACH COGIC

A PDP solution consist of a set of locations for every vehicle available, we decomposed the solving process to two stages, in the first one we create clusters of orders for each vehicle, since the orders consist of pair of locations, the clusters are less evident, that why we are going to use the genetic algorithm and the centroid local optimization to find suitable clusters, the immigration operator is used to avoid early convergence, then we proceeded to the second stage, where we generated more solutions from each one in the population by merging some clusters, then we routed the clusters of each solution using an insertion algorithm and we took the solution with the best fitness.

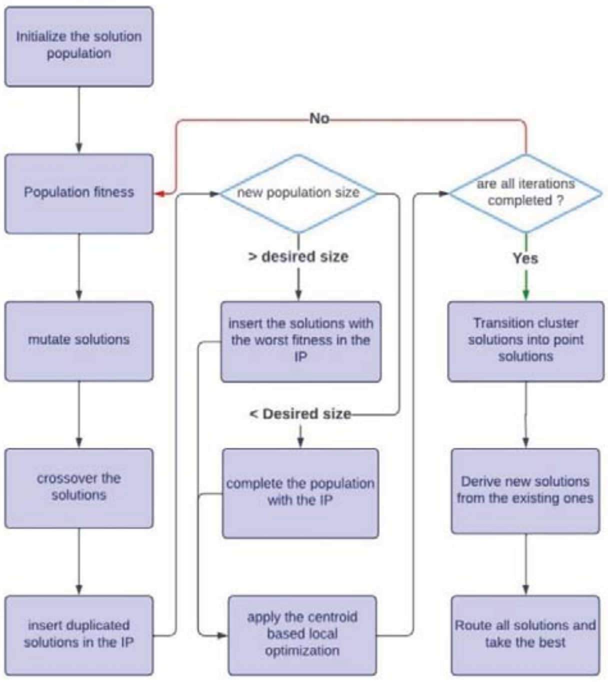

Initialize the solution population
Population fitness
mutate solutions
crossover the solutions
insert duplicated solutions in the IP
No
new population size
> desired size
insert the solutions with the worst fitness in the IP
< Desired size
complete the population with the IP
apply the centroid based local optimization
are all iterations completed ?
Yes
Transition cluster solutions into point solutions
Derive new solutions from the existing ones
Route all solutions and take the best

Figure 3. Solution process

4.1. Clustering with COGIC

In this stage we used the genetic algorithm and the centroid local optimization to find the best cluster, we also used the immigration operator to further delay the convergence of the solution population, since in this stage we don't have the routes, we need an objective function that can estimate the value of the clusters, for that we calculated the centroid of the clusters, and we minimized the sums of distances between each point and its corresponding centroid, since the points are in pairs forming orders that must be in the same route, each order can have a point representing it, and it is the one used in the local optimization step, we chose that the center point isn't the best representative point of the order, that may lead to wrong results in some edge cases where there is two orders mirroring each other's, and rarely would be optimal to have them in the same cluster, since pickups must be visited strictly before their corresponding delivery point, we decided that the point representing the order would be in the vector of the two points, but 20% closer to the pickup point, and this is the pseudo code of the genetic clustering with immigration:

```
Algorithm 1: Genetic clustering with immigration
    Initialize population of cluster solution
    Initialize an empty immigration population IP
    For number of iterations
        Evaluate the fitness of the population
        Mutate every solution
        Initialize an empty new population
        For solution in population
            If solution is the best solution of the population
                Add the solution to the new population
                Crossover with n random solutions
                Add the new solutions to the new population
            Else
                Crossover with m random solutions //n>m
                Add the new solutions to the new population
            End if
        End for
        Add solutions with duplicated fitness to the IP
        Delete the solutions with duplicated fitness in the new
        If new population > desired population size
            Add solutions with the worst fitness to the IP
            Delete solutions with the worst fitness
        Else if new population < desired population size
            Add solutions from the IP to the new population
        End if
        Solution population = new population
        Optimize each solution locally
    End for
End
```

We start the algorithm 1 by initializing a population of random solutions and an empty immigration population, then we loop for the number of iterations, beginning by the fitness evaluation of each solution in the population by using the objective function, then we mutate every solution, the mutation is done by selecting a random pair of pickup and its delivery point, and assigning it to a different cluster, the number of times this is done depends on a parameter that we named degree of mutation.
We followed by doing the selection and crossover, those are done by creating an empty new population, and then looping through the solutions, if it’s not the current best solution of the population, the solution get to crossover with random n solutions, and the generated solutions passes to the next generation, but if it is the current best solution of the population, it passes directly to the next generation, and get to crossover more times than the rest of solutions, the goal of this selection is to spread its influence of the best solution more in the new generation.
In the next process, the solutions with duplicated fitness get transferred to the immigration population until every solution is unique, this process is done to slow the convergence in the population, after this process, if the size of the new population is still higher than the desired size, the solutions with worst fitness gets transferred to the immigration population until the desired size is reached, in the other case when its lower than the desired size, we use the solutions of the immigration population to achieve the desired size, if we use all of our saved solutions from the immigration population we

duplicate the best solutions until the desired size is reached. And the last process of this clustering step is the centroid local optimization:

```
Algorithm 2: Centroid local optimization
  For number of iterations
    Select a random order
    For clusters in a solution
      If random order is in this cluster
        Save the original cluster
      End if
        Calculate the centroid of the cluster
        Save the closest cluster
      End if
    End for
    Transfer the order to the closest cluster
  End for
End
```

The algorithm 2 start by selecting a random order, then we loop through the clusters of the solutions to find the original cluster and the closest cluster to it, the latter is determined by the distance between the order and the centroid of the cluster, after finding both we transfer the order to the closest cluster, we do this process for every solution a number of times determined by a parameter that we named degree of local optimization. After finishing all the iterations, we go to the routing phase.

4.2. Routing with the insertion algorithm

In the second phase of our solution approach, we route all the solutions of the population, to do that we use algorithm 3, for every solution in the population we loop through their clusters, and for every one of them we create an empty cluster, and we insert the depot at the start and end of it, and then we insert the pickup and delivery pairs one by one by finding their valid and most optimal index in the new cluster, the validity of the index is checked by verifying that the capacity constraint is respected and the pickup points are visited before their corresponding delivery points, after finishing it we replaced the original cluster with the routed one.

```
Algorithm 3: Insertion routing algorithm

  For solution in population
    For cluster in solution
      Initialize an empty cluster
      Insert depot at the start and end of the new cluster
      For location in cluster
        For index in new cluster
          Save the index with the best fitness
        End for
        Insert the location in the optimal index
      End for
      Replace the cluster with the new cluster
    End for
  End for
  Evaluate the fitness of the population
  End
```

```
Algorithm 4: Solutions derivation algorithm

  For solution in population
    Save original fitness
    For cluster X in solution
      For cluster Y in solution
        If X != Y
          Merge X and Y
          Route the new cluster
          If the fitness improved
            Repeat this process with the new
            cluster and the other ones
          End if
        End if
      End for
      Save the resulting solution
    End for
  End for
  Fitness evaluation
  Best solution is saved
  End
```

Taking into consideration that the fitness function for the clusters in the first phase is an estimation of the quality of the routes of the solutions, the solution with the best clusters may not be the best solution after making the routing, to mitigate the drawback of this approach, after routing all the solutions we produce new solutions from the ones in the population, to do so we apply algorithm 4.
We start algorithm 4 by looping through the solutions, and for every one of them, we start merging clusters and routing them with the insertion algorithm, then we check if the fitness of the solution improves, if so we repeat the process with the new merged cluster in the solution, this step is only possible with the speed of the insertion algorithm, otherwise the computational time of the algorithm would skyrocket.

5. NUMERICAL RESULTS

Now that we have seen the approach, we are going to test our COGIC algorithm with Breedam benchmark instances, those instances are problems for the Vehicle Routing Problem with Pickup and Delivery (VRPPD). Since the pickup and delivery locations aren't paired in the VRPPD, we modified it, so the consecutive pickup and delivery locations form a pair. The benchmark consists of 60 instances and the maximum number of vehicles is 10, and all of them have the same vehicle capacity of 100, and all the weight of the picked-up goods is 10.
The tests were executed in a machine Intel Core i5-10300H CPU with 16 GB Ram, in the table 2 we compared our results with the ones from Google OR-Tools ran in Google Colab in an environment of 13 GB Ram. It provides a set of algorithms and tools to solve various types of optimization problems efficiently we used the heuristic option of parallel cheapest insertion, we tested other heuristics and metaheuristics, but the one that we chose gave us the best results.

Table 1: COGIC algorithm parameters

Parameter	Value
Iterations	5000
Population	30
Centroid bias	0.2
Best Crossover	2
Normal Crossover	1
Degree of local	10
Degree of mutation	1

The parameters used in the COGIC algorithm are 5000 iterations with a population of 30 solutions, and to calculate the average fitness we took a sample of 10 results, the centroid bias determines how closer the middle point of each order pair is to the pickup, in this case is 20% closer to the pickup point, this information is used in the local optimization step (Algorithm 2). The best solution gets to crossover twice, while the rest of solutions get to crossover once, the degree of mutation determines how many mutations is applied to each solution in every solution, and the degree of local optimization determines the number of times the centroid local optimization will be applied to each solution.

Table 2: Results for Breedam benchmark

Problem	COGIC			OR-Tools	
	Best	Average	Time	Best	Time
1P1	**1164**	1200	35	1232	574
2 P1	**1166**	1204	35	1168	374
3 P1	**1168**	1212	35	1498	972
4 P1	**667**	682	35	869	825
5 P1	**635**	646	53	824	670
6 P1	**655**	662	35	1121	556
7 P1	**1046**	1111	35	1200	714
8 P1	**1108**	1138	35	1331	771
9 P1	**1073**	1150	35	1659	725
10 P1	**1008**	1066	35	1320	687
11 P1	**995**	1036	35	1492	738
12 P1	**995**	1043	35	1416	522
13 P1	**1133**	1158	35	1349	679
14 P1	**1126**	1166	35	1398	707
15 P1	**1116**	1152	35	1357	783
16 P1	**801**	847	35	1211	812
17 P1	**758**	789	35	1133	1293
18 P1	**727**	741	35	1042	2057
19 P1	**510**	516	35	522	449
20 P1	**468**	473	35	1036	503
21 P1	**474**	483	35	797	539
22 P1	**644**	689	35	855	755
23 P1	**588**	622	35	979	1099
24 P1	**597**	620	35	936	750
25 P1	**668**	700	35	982	455
26 P1	**650**	692	35	1202	392
27 P1	**739**	818	35	1174	397
28 P1	**733**	792	35	867	230
29 P1	**636**	746	35	1003	1115
30 P1	**691**	870	35	1075	1799
31 P1	**680**	728	35	812	740
32 P1	**675**	749	35	1124	659
33 P1	**721**	764	35	1142	1570
34 P1	**542**	560	35	626	560
35 P1	**551**	559	35	817	648
36 P1	**536**	554	35	1026	626
37 P1	**625**	653	35	841	570
38 P1	**666**	711	35	920	855
39 P1	**672**	737	35	1144	1576
40 P1	**671**	711	35	917	300
41 P1	**703**	784	35	950	1104
42 P1	**803**	857	35	1216	645
43 P1	**681**	775	35	1061	489
44 P1	**825**	893	35	1069	1338
45 P1	**1078**	1285	35	1093	742
46 P1	**776**	800	35	949	312
47 P1	**749**	815	35	898	825
48 P1	**883**	905	35	1054	381
49 P1	**548**	555	35	576	509
50 P1	**545**	553	35	802	729
51 P1	**566**	571	35	1007	835
52 P1	**852**	899	35	1013	619
53 P1	**832**	893	35	1085	660
54 P1	**884**	918	35	977	581
55 P1	**794**	851	35	1109	1031
56 P1	**807**	870	35	1018	634
57 P1	**856**	899	35	1165	805
58 P1	**883**	968	35	978	1110
59 P1	**932**	963	35	1092	681
60 P1	**923**	990	53	1302	777

Looking at the results of Table 2, the best fitness of the COGIC algorithm is always more optimal than the ones of OR-Tools. In some problems like 26P1 we see a bigger gap between our results and OR-Tools, the reason for that is that the two algorithms take different approaches on the number of vehicles used, COGIC tends to use fewer vehicles in problems where the depot isn't in the center, and it indeed proves being more efficient.

In instances 45P1, not only the result of the two algorithms is close but also the gap of the COGIC best solution and its average is significant, the main reason for that is the different amount of vehicles between the solutions, and using multiple of them gives a more optimal fitness. And in problems like 53P1 treat edge cases where multiple pickup and delivery pairs are opposite each other's in the map, this is the reason why we didn't consider the center of the pair in the local optimization step (algorithm 2), but we used instead a center bias of 20% closer to the pickup point, this way we avoided joining inefficient pairs together in the same cluster.

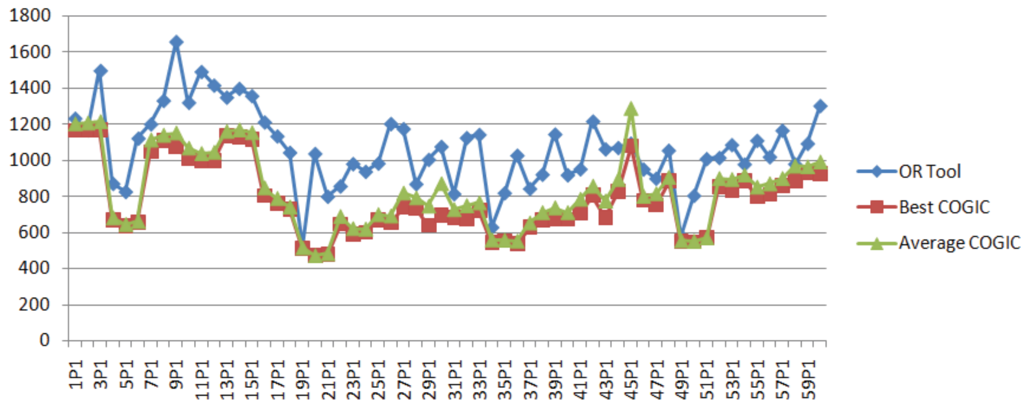


Figure 4. Comparison of COGIC and OR-Tools

In the graph of figure 4, we can see that the only instance where the average solution wasn't better than the one from OR-Tools is 45P1, however the best solution of COGIC was still better, the reason is that the depot is in the southern edge of the map, and the algorithm tend to use a lower amount of vehicles in cases where the depot isn't in the center, but in the clusters, the sums of the pickup loads was superior to the vehicles capacity, that what led to inefficient routing in some solutions, but some mutations from the genetic algorithm did drive the population to use multiple vehicles, and therefore route more efficiently, that was the reason that led some of the solutions to surpass the one from OR-Tools.

6. CONCLUSION

The productivity of different sectors in the Moroccan economy rely in the logistics of the supply chain, and improving it, results in lowering the costs in all of them, that was the reason we studied the Pickup and Delivery Problem (PDP), we used a cluster first, route second approach, and we featured the novel Centroid Optimized Genetic Immigration Clustering (COGIC), were we created clusters using the genetic algorithm and a centroid based local search method, assisted by an immigration operator, then we proceeded to the routing stage by using a fast insertion algorithm, we exploited its speed by merging the cluster and routing them recursively. In the benchmarking phase we treated the Breedan instances that contain different edge cases problems, and we compared the results with the ones from OR-Tools, they proved being more optimal and faster in execution time.

7. REFERENCES

Savelsbergh, Martin WP, and Marc Sol. 1995. "The General Pickup and Delivery Problem." Transportation Science 29, no. 1: 17-29.

Dumas, Yvan, Jacques Desrosiers, and Francois Soumis. 1991. "The Pickup and Delivery Problem with Time Windows." European Journal of Operational Research 54, no. 1: 7-22.

Koç, Çağrı, Gilbert Laporte, and İlknur Tükenmez. 2020. "A Review of Vehicle Routing with Simultaneous Pickup and Delivery." Computers & Operations Research 122: 104987.

Ropke, Stefan, and David Pisinger. 2006. "An Adaptive Large Neighborhood Search Heuristic for the Pickup and Delivery Problem with Time Windows." Transportation Science 40, no. 4: 455-472.

Männel, Daniel, and Andreas Bortfeldt. 2016. "A Hybrid Algorithm for the Vehicle Routing Problem with Pickup and Delivery and Three-Dimensional Loading Constraints." European Journal of Operational Research 254, no. 3: 840-858.

Küçük, Mustafa, and Şeyda Topaloğlu Yıldız. 2019. "A Constraint Programming Approach for the Pickup and Delivery Problem with Time Windows." Pamukkale Üniversitesi Mühendislik Bilimleri Dergisi 25, no. 9: 1041-1049.

Kusuma, Purba Daru, and Meta Kallista. 2022. "Pickup and Delivery Problem in the Collaborative City Courier Service by Using Genetic Algorithm and Nearest Distance." Bulletin of Electrical Engineering and Informatics 11, no. 2: 1026-1036.

Wang, Yong, Lingyu Ran, Xiangyang Guan, and Yajie Zou. 2021. "Multi- Depot Pickup and Delivery Problem with Resource Sharing." Journal of Advanced Transportation 2021, no. 1: 5182989.

Baker, Barrie M., and MA1951066 Ayechew. 2003. "A Genetic Algorithm for the Vehicle Routing Problem." Computers & Operations Research 30, no. 5: 787-800.

Tajani, Chakir, Otman Abdoun, and Ahmed Idrissi Lahjouji. 2017. "Genetic Algorithm Adopting Immigration Operator to Solve the Asymmetric Traveling Salesman Problem." International Journal of Pure and Applied Mathematics 115, no. 4: 801-812.

Fargiana, Farid, Respitawulan Respitawulan, Yusuf Fajar, Didi Suhaedi, and Erwin Harahap. 2022. "Implementation of Cheapest Insertion Heuristic Algorithm in Determining Shortest Delivery Route." International Journal of Global Operations Research 3, no. 2: 37-45.

Toth, Paolo, and Daniele Vigo, eds. 2002. The Vehicle Routing Problem. Philadelphia: Society for Industrial and Applied Mathematics.

Sobotka, Václav, and Hana Rudová. 2023. "Real-World Pickup and Delivery Problem with Transfers." In Proceedings of the International Symposium on Combinatorial Search, vol. 16, no. 1: 83-91.

Li, Xijun, Weilin Luo, Mingxuan Yuan, Jun Wang, Jiawen Lu, Jie Wang, Jinhu Lü, and Jia Zeng. 2021. "Learning to Optimize Industry-Scale Dynamic Pickup and Delivery Problems." In 2021 IEEE 37th International Conference on Data Engineering (ICDE), 2511-2522. IEEE.

Zong, Zefang, Meng Zheng, Yong Li, and Depeng Jin. 2022. "MAPDP: Cooperative Multi-Agent Reinforcement Learning to Solve Pickup and Delivery Problems." In Proceedings of the AAAI Conference on Artificial Intelligence, vol. 36, no. 9: 9980-9988.

Al Chami, Z., H. Manier, M-A. Manier, and C. Fitouri. 2017. "A Hybrid Genetic Algorithm to Solve a Multi-Objective Pickup and Delivery Problem." IFAC-PapersOnLine 50, no. 1: 14656-14661.
Andriansyah, A., N. Prasanti, and H. Y. Sastra. 2019. "Simulated Annealing Algorithm for Solving Pickup and Delivery Problem with LIFO, Time Duration, and Limited Vehicle Number." In IOP Conference Series: Materials Science and Engineering, vol. 697, no. 1: 012021. IOP Publishing.

Carrabs, F., R. Cerulli, and J.-F. Cordeau. 2007. "An Additive Branch-and-Bound Algorithm for the Pickup and Delivery Traveling Salesman Problem with LIFO or FIFO Loading." INFOR: Information Systems and Operational Research 45: 223–238.

Wang, Yu, Renrong Zheng, Yan Zhao, and Chengji Liang. 2023. "Pickup and Delivery Problem of Automobile Outbound Logistics Considering Trans-Shipment among Distribution Centers." Systems 11, no. 9: 457.

Nazari, Mohammad, Afshin Oroojlooy, Lawrence V. Snyder, and Martin Takác. 2018. "Reinforcement Learning for Solving the Vehicle Routing Problem." In Proceedings of the 32nd Conference on Neural Information Processing Systems (NeurIPS 2018).

Kool, Wouter, Herke van Hoof, and Max Welling. 2019. "Attention, Learn to Solve Routing Problems!" In 7th International Conference on Learning Representations (ICLR 2019), New Orleans, LA, USA, May 6-9. OpenReview.net.

Moroccan Agency for the Development of Logistics. "Active Logistics Areas." AMDL. Accessed May 27, 2024. https://www.amdl.gov.ma/amdl/en/active-logistics-areas/.

Ministry of Agriculture, Fisheries, Rural Development, Water and Forests. "سلسلة الزيتون." Accessed May 24, 2024. https://www.agriculture.gov.ma/ar/filiere/olivier.

Dorronsoro, Bernabé. "Vehicle Routing Problem: Problem Instances." Accessed May 27, 2024. https://www.bernabe.dorronsoro.es/vrp/index.html?/Problem_Instances/instances.html.

Google. "OR-Tools." Accessed May 27, 2024. https://developers.google.com/optimization.

SINTEF. "Li & Lim Benchmark." Accessed May 27, 2024. https://www.sintef.no/projectweb/top/pdptw/li-lim-benchmark/.