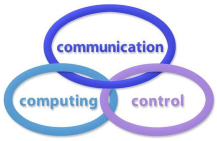
INTERNATIONAL JOURNAL OF COMPUTERS COMMUNICATIONS & CONTROL
Online ISSN 1841-9844, ISSN-L 1841-9836, Volume: 19, Issue: 6, Month: December, Year: 2024
Article Number: 6865, https://doi.org/10.15837/ijccc.2024.6.6865



CCC Publications

Development of a Model for Generating Trajectories for an Autonomous Naval Vehicle Using Genetic Algorithms in MATLAB

V. Olivares, A. Oddershede, L. Quezada, M. Vargas, C. Montt

Víctor E. Olivares*
Department of Industrial Engineering
University of Santiago of Chile, Chile
Av. Víctor Jara 3769, Santiago, Chile
*Corresponding author: victor.olivares@usach.cl

Astrid Oddershede
Department of Industrial Engineering
University of Santiago of Chile, Chile
Av. Víctor Jara 3769, Santiago, Chile
astrid.oddershede@usach.cl

Luis Quezada
Department of Industrial Engineering
University of Santiago of Chile, Chile
Av. Víctor Jara 3769, Santiago, Chile
luis.quezada@usach.cl

Manuel Vargas
Department of Industrial Engineering
University of Santiago of Chile, Chile
Av. Víctor Jara 3769, Santiago, Chile
manuel.vargas@usach.cl

Cecilia Montt
Department of Industrial Engineering
University of Santiago of Chile, Chile
Av. Víctor Jara 3769, Santiago, Chile
cecilia.montt@usach.cl

Abstract

This work presents the development of a model for generating trajectories for an autonomous naval vehicle using genetic algorithms implemented in MATLAB. The primary objective is to optimize the routes the vehicle must follow, minimizing the traveled distance and ensuring efficient navigation. Various scenarios were tested by varying model parameters such as the number of environmental control points, the number of generations, and the number of individuals to evaluate the genetic algorithm's performance. In each scenario, results were analyzed in terms of minimum

traveled distance and the optimal sequence of trajectory points (FITNESS). The results show that the genetic algorithm can find efficient solutions, adapting to different configurations of points and generations. Specific examples illustrate the optimal generated trajectories, accompanied by graphical representations visualizing the sequence of points. This study demonstrates the effectiveness of genetic algorithms in route planning for autonomous naval vehicles and provides a solid foundation for future research and applications in autonomous navigation.

Keywords: Matlab, TSP, Genetic Algorithm, Cluster, Autonomous Naval Vehicle

1 Introduction

Quintero Bay, located in the Valparaíso Region of Chile, has been drastically transformed by intensive industrial activity, making it a concerning focus of pollution [1]. Numerous industrial facilities, including refineries and chemical plants, as well as port facilities, have left a profound environmental impact, making it one of the most polluted areas in the country [2]. Quintero Bay has numerous records of environmental emergencies due to the increase in pollutants such as PM2.5, PM10, PM6, SO2, NOx, VOCs [3], among others, resulting in frequent health and environmental alerts. In this context, there is a need to design an unmanned naval vehicle for Quintero Bay that allows reading environmental indicators, adapting to different situations, and ensuring future interactions with other vessels and port infrastructure. An essential part of this design is developing a navigation model based on Genetic Algorithm and cluster formation, which generates an efficient route from various points to collect samples within Quintero Bay avoiding different obstacles such as docks, ships, rocky areas, ocean currents, tides, among others.

There is an urgent need to address the environmental and operational challenges present in Quintero Bay and to leverage technological advancements that the use of unmanned naval vehicles offers in exploring and monitoring maritime environments. This justification is detailed in the following points:

- Environmental Impact and Monitoring: Continuous monitoring of pollutants in its waters is essential to evaluate and mitigate environmental impacts. Autonomous vehicles offer a versatile and accurate platform for performing these tasks without exposing crew members to risks and minimizing disturbance to the environment [4, 5].
- Technological Innovation: Designing a vehicle specifically adapted to the needs of Quintero Bay represents a step forward in applying advanced technology to solve marine problems. Integrating control systems, sensors, solar energy, and guidance and navigation devices into a single system provides a unique opportunity to improve monitoring and exploration efficiency [6].
- Operational Efficiency and Costs: Using this autonomous naval vehicle in the bay reduces the need for manned vessels, leading to significant savings in operational and personnel costs. Additionally, automating routine and hazardous tasks increases safety and continuity of operations [7].
- Resource Optimization: Mission planning for autonomous vehicles can be adjusted to specific schedules and needs, optimizing resource use. Moreover, by allowing continuous data collection, these vehicles can provide detailed real-time information, improving decision-making and environmental management [8, 9].
- Knowledge Generation: Implementing these vehicles will generate valuable data and knowledge about the marine ecosystem in that area, useful for scientific research, policy formulation, and environmental education [10].

The design of an Unmanned Naval Vehicle for Quintero Bay and developing an efficient navigation algorithm will generate a unique capability to address critical environmental problems, the opportunity to apply cutting-edge technology to solve them, benefits in terms of efficiency and costs, and contribute to scientific knowledge and sustainable management of this valuable marine environment [11].

2 Problem

Designing a Navigation System for an autonomous naval vehicle is crucial when considering its operation in complex environments like Quintero Bay. This maritime area, known for hosting multiple industrial facilities and considerable vessel traffic, presents significant challenges that the autonomous vehicle must address. The navigation system must effectively detect and avoid various obstacles, such as moving ships, docks, buoys, and other static and dynamic elements (currents and tides) present in the bay.

Route planning also plays a crucial role. Planning algorithms must consider the positions of detected obstacles and the maritime environmental conditions, such as currents, tides, and weather conditions, which may affect the vehicle's navigation [12]. This ensures that the optimal route is chosen to minimize risks and maximize the vehicle's operational efficiency.

Designing the Navigation System for an autonomous naval vehicle in Quintero Bay requires an innovative approach that integrates advanced technology and efficient strategies.

3 Problem Formulation

Figure 1 shows the problem posed. The Autonomous Naval Vehicle must travel a set of points marked with a red circle within Quintero Bay, collecting data or samples at each point to detect possible pollutants in the water or air. Therefore, the shortest route must be found to allow the vehicle to travel the set of points only once and return to the origin point. The route generation must consider the bay's obstacles, such as docks, ships, buoys, rocks, small fishing boat anchorage areas, ocean currents, tides, etc.

This context leads us to use the Traveling Salesman Problem (TSP) methodology, using the Genetic Algorithm heuristic implemented in MATLAB [13] and a cluster generation method [14].

Figure 1: Quintero Bay and sampling points

4 Solution Proposed

The Traveling Salesman Problem (TSP) is a classic combinatorial optimization problem. This problem aims to find the shortest route that allows a salesman to visit a set of cities only once and return to the origin city. It is relevant in fields such as logistics, route planning, and computational biology [13]. The TSP can be mathematically formulated as a minimization problem. Given a set of cities (points) and the distances between each pair of cities, the goal is to find a permutation of the cities that minimizes the total distance traveled. Mathematically, if C is the set of cities and $d(i, j)$

the distance between cities i and j, the objective is to find a sequence (permutation) of the cities $(c_1, c_2, \ldots, c_n)$ such that the total sum of the distances is minimal.

$$\sum_{i=1}^{n-1} d(c_i, c_{i+1})$$

There are multiple approaches to solving the TSP, classified as exact and heuristic. This work selected the application of Genetic Algorithms, implemented through MATLAB, optimizing route planning and decision-making in dynamic and complex environments. These algorithms use principles inspired by biological evolution to generate optimal, adaptive, and robust solutions to unforeseen environmental variations, such as the presence of ships, docks, and other static and dynamic obstacles.

The genetic algorithm's ability to explore and exploit different possible solutions ensures that the autonomous naval vehicle can navigate safely and efficiently, minimizing risks and maximizing operational effectiveness. This innovative approach not only improves the accuracy and reliability of the navigation system but also facilitates the vehicle's adaptability to changing conditions, ensuring optimal performance in Quintero Bay.

To incorporate restrictions such as docks, rocks, anchorages, and other obstacles, the TSP-GA model is complemented with a heuristic to generate clusters [14] and connect these clusters through common points. The model is calibrated with Data Adjustment to match the model's outputs with the defined restrictions.

The following two solution strategies are used in the search for these heuristics:

- Route first and cluster second
- Cluster first and route second

The "Cluster First and Route Second" strategy offers a significant advantage when addressing the TSP in Quintero Bay, an area characterized by some fixed and dynamic obstacles. This strategy facilitates and provides flexibility to incorporate these obstacles as restrictions in the model using genetic algorithms and clusters. In MATLAB, these restrictions can be quickly implemented by adjusting the "distance matrix." By clustering first, optimal segments for the route are identified, facilitating the integration of natural barriers and obstacles in the routing process. Then, by routing after, the route within each cluster is optimized, ensuring that obstacles are considered in the distance calculations. This methodology not only improves the model's accuracy but also simplifies its implementation, as modifications to the distance matrix in MATLAB are straightforward and manageable. Thus, a balance is achieved between computational efficiency and fidelity to the real environment of Quintero Bay, ensuring practical, applicable, and flexible solutions in complex scenarios.

5 Cluster First and Route Second

As previously described, clusters or groups of points must first be generated considering the restrictions, using the "Sweep" algorithm [15]. The proposed method comprises three phases. In the first phase, groups (sub-clusters) of points are generated, with each group of points being on the same route of the naval vehicle in the final solution. In the second phase, common points between two clusters are identified; these points will serve as bridges connecting two neighboring clusters. Therefore, constructing the routes for each sub-cluster is a TSP that, depending on the number of points in the sub-cluster, can be solved exactly or approximately. In the third phase, data refinement is performed on the "Distance Matrix" in MATLAB, eliminating connections between points of different clusters and forcing the connection of common points belonging to two clusters.

For the first phase, i.e., forming the sub-clusters, the "Sweep" algorithm is used. In this sweep heuristic, clusters are formed by rotating a semi-line clockwise, originating from one end of the obstacles (see Fig. 2) and incorporating the points "swept" by the semi-line until some restriction is exceeded, such as an angle, battery charge, number of points, among others. Once a cluster is formed, another cluster is formed starting from the last point reached until the same restriction is met. The process is repeated until all points are in some cluster.

In the second phase, common points between clusters are identified to ensure connectivity between the clusters. These common points allow for a smoother transition between clusters, facilitating the construction of a continuous and efficient route for the naval vehicle.

In the third phase, data refinement is carried out on the "Distance Matrix" in MATLAB. This step involves updating the distance matrix to reflect the constraints imposed by the clusters and the identified common points. This refinement ensures that the genetic algorithm accurately considers the obstacles and restrictions present in Quintero Bay, leading to an optimal route for the autonomous naval vehicle.

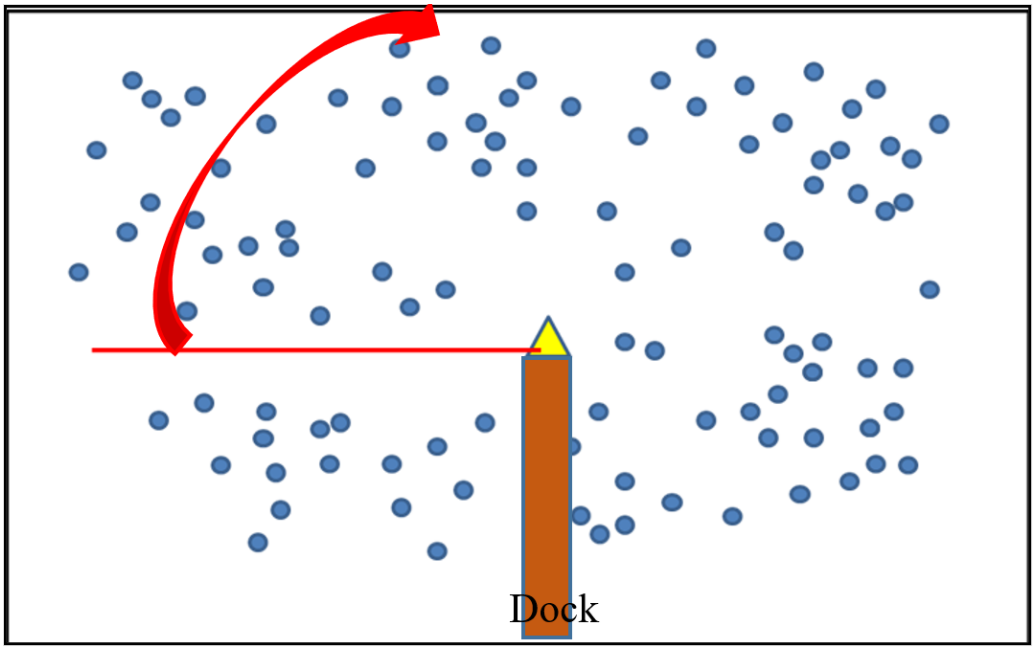


Figure 2: Sweeping points with a ray

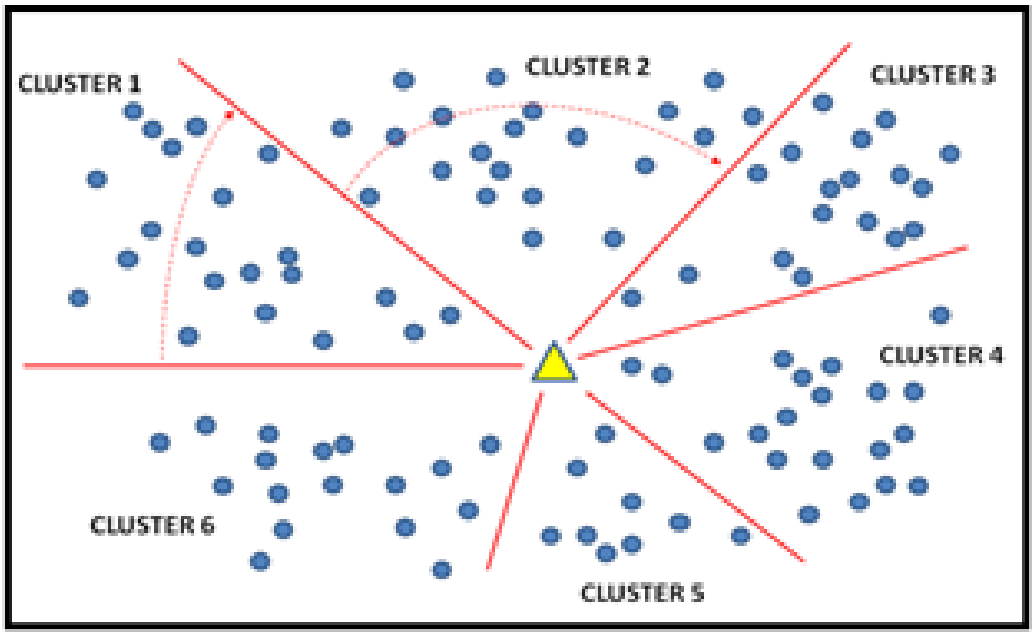


Figure 3: Forming Clusters with Sweep

In Fig. 3, we illustrate how to generate clusters using the sweep method. This method starts by selecting the edge of an obstacle as the center of a semicircle. A semi-straight line then sweeps through a set of points in a 270-degree path, grouping all the points covered by the line into a cluster.

Next, a new obstacle is chosen, and a new semicircle is formed, creating another cluster with the swept points. The 270-degree sweep ensures that there are common points between clusters, which act as bridges connecting adjacent clusters when forming the TSP route. This technique ensures complete area coverage and facilitates the integration of obstacles into the route optimization model, allowing efficient connections between different segments of the path.

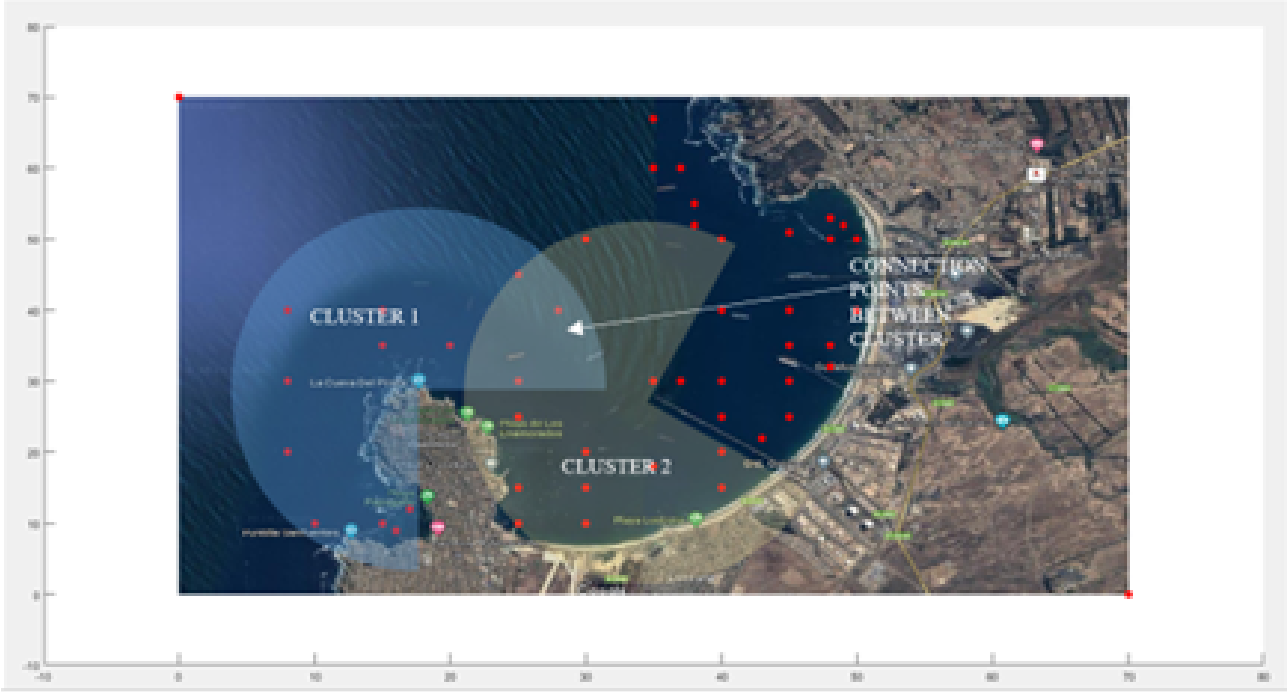


Figure 4: Example of forming clusters of points with Sweep in Quinteros Bay

6 Refining a Distance Matrix in Matlab

Once the points belonging to each cluster and the common points between neighboring clusters, which act as connection bridges, are identified, the "distances" matrix in MATLAB is refined. Also, the obstacles are incorporated into the matrix; these obstacles are modeled by constraints on the distance matrix D.

Let O be the set of obstacles, each represented as a forbidden region $o_k \subset \mathbb{R}^2$.

The distance is modified depending on the presence of obstacles:

$$d_{ij}\begin{cases}\infty & if\ the\ trajectory\ between\ p_i\ and\ p_j\ intersect\ a\ forbidden\ region\ o_k \epsilon O \\ euclidian\ distances\ between\ p_i\ and\ p_j & otherwise\end{cases}$$

Figure 5 shows the general structure of the "distances" matrix in MATLAB [17], containing sub-matrices representing clusters and zones of common points that act as bridges between neighboring clusters. For example, the sub-matrix of distances between points of CLUSTER 1 is shown in yellow, the sub-matrix containing distances between points of CLUSTER 2 is shown in orange, and the sub-matrix containing distances between points of CLUSTER 3 is shown in light blue. The areas framed in red represent sub-matrices of distances of points at the intersection of two neighboring clusters, functioning as bridges between these clusters. These points can be unidirectional, omnidirectional, or have other constraints, such as specific directions, branching points, or sink points. This modified distance matrix is the basis for running the TSP. Finally, areas of sub-matrices with problem constraints, indicating infeasible connections, are shown in white.

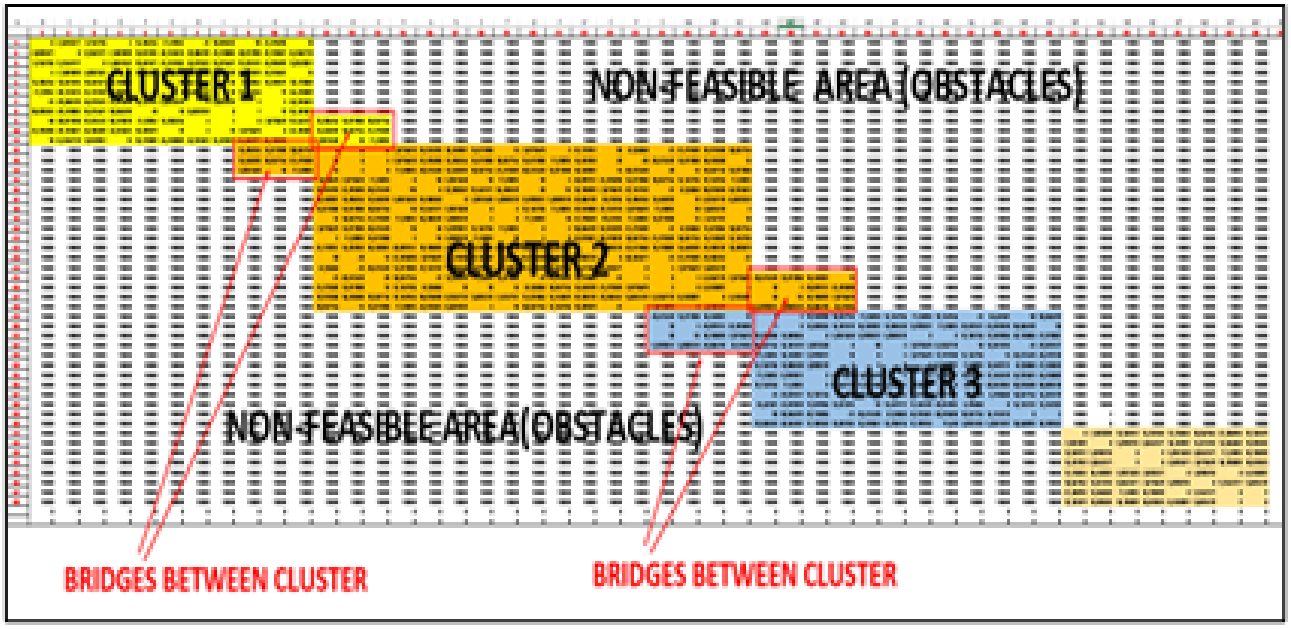


Figure 5: Distance matrix refinement

7 Genetic Algorithm Results

To evaluate the performance of the genetic algorithm used to generate the Naval Vehicle's trajectories, various scenarios with different variations in the number of points, population, and generations were tested. In each scenario, the algorithm's effectiveness was determined in terms of the number of points and the resulting distance of the best trajectory sequence obtained (FITNESS). Figure 6 shows the result of applying the TSP method with constraints using a Genetic Algorithm with 50 points, a population of 200 individuals, and 800 generations, resulting in the BEST FIT of 467.8809. This figure displays the optimal route sequence that the Naval Vehicle should follow, accompanied by a graphical representation of this sequence. These examples illustrate how the algorithm optimizes trajectories and demonstrate its ability to find efficient solutions in different configurations.

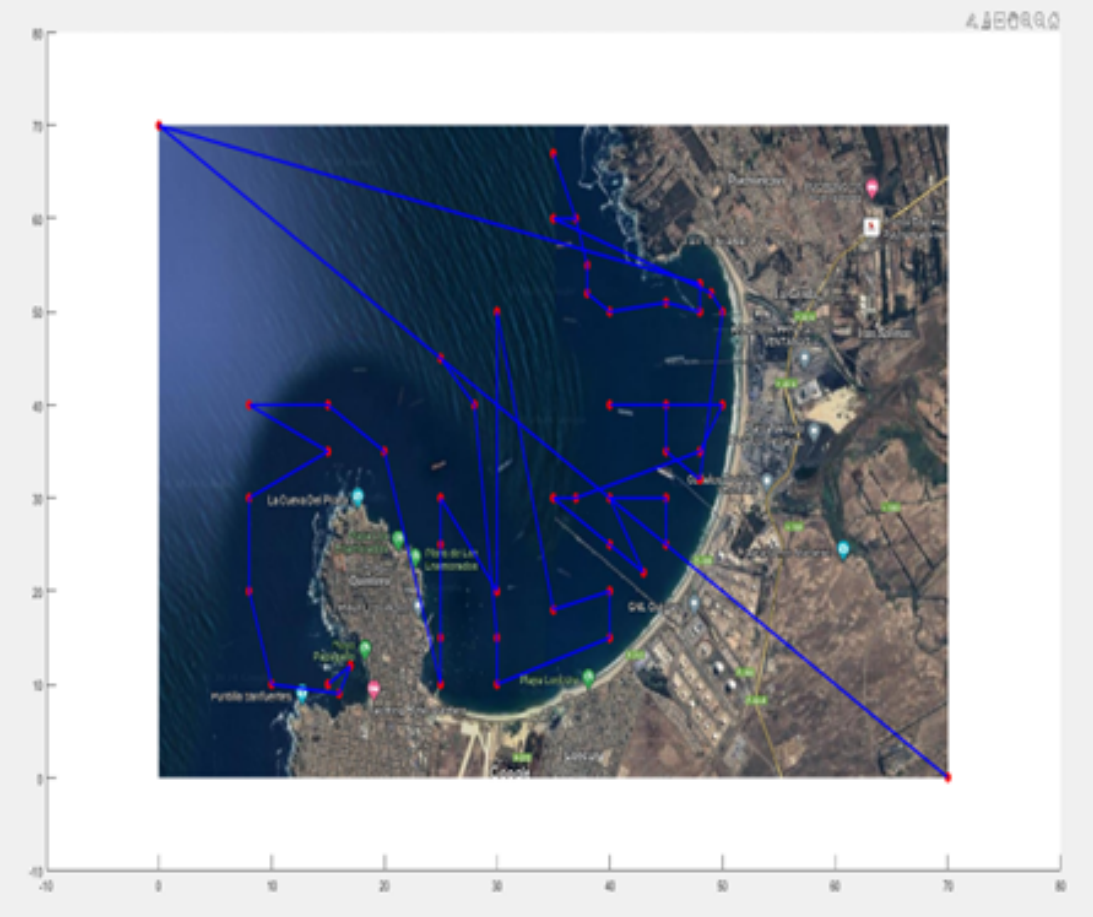


Figure 6: Path resulting of a GA run

Next, Table 1 shows the results (FIT) of 10 runs of the Genetic Algorithm for the TSP with 200 generations, for different numbers of points, ranging from 10 points to 50 points.

Table 1: Results of 10 runs of the GA with 200 generations

N° of Points	N° run of the GA									
	1	2	3	4	5	6	7	8	9	10
10	247,47	247,47	247,47	247,47	247,47	247,47	247,47	247,47	247,47	247,47
15	273,90	287,96	273,90	273,90	273,90	273,90	273,90	273,90	273,90	273,90
20	315,96	316,95	316,95	315,96	344,18	315,96	315,96	342,99	315,96	342,99
30	423,63	405,70	408,48	421,58	413,00	422,89	401,86	420,94	445,68	430,37
40	527,20	494,46	502,80	464,94	524,75	477,29	557,75	500,64	509,29	508,31
50	583,60	539,44	589,29	616,02	615,17	631,17	629,58	629,85	638,73	595,56

Here, it can be observed that for a layout of up to 10 points, the GA produces optimal routes immediately and in very suitable times. Therefore, for clusters with 10 or fewer points, one can rely on obtaining the optimal result by selecting the minimum with just a few GA runs. Similarly, for 10 points, all GA runs delivered the same fitness result, meaning there is no difference between the average and the best fitness.

Between 10 and 15 points, the GA's accuracy is still quite good. For example, in Table 7, it can be seen that for 15 points, only in one out of 10 runs did the algorithm not deliver the optimal result.

Between 15 and 20 points, the minimum fitness result is repeated with some frequency, and with a few runs, a good result can be achieved to provide an appropriate route for the Naval Vehicle.

In Table 2, the average results of ten GA runs for each group of points and their standard deviation (SD) and relative standard deviation (RSD) are shown.

Table 2: Summary of GA results

N° Points	Best Fitness	Mean	SD	RSD %
10	247,47	247,47	0	0
15	273,90	275,30	4,45	1,62
20	315,96	324,39	13,12	4,04
30	401,86	419,41	12,93	3,08
40	464,94	506,74	26,19	5,17
50	539,44	606,84	30,42	5,01

Figure 7 shows that as the number of points in the cluster increases, the difference between the average and the best fitness also increases. This means there is a greater dispersion in the GA results with 200 generations.

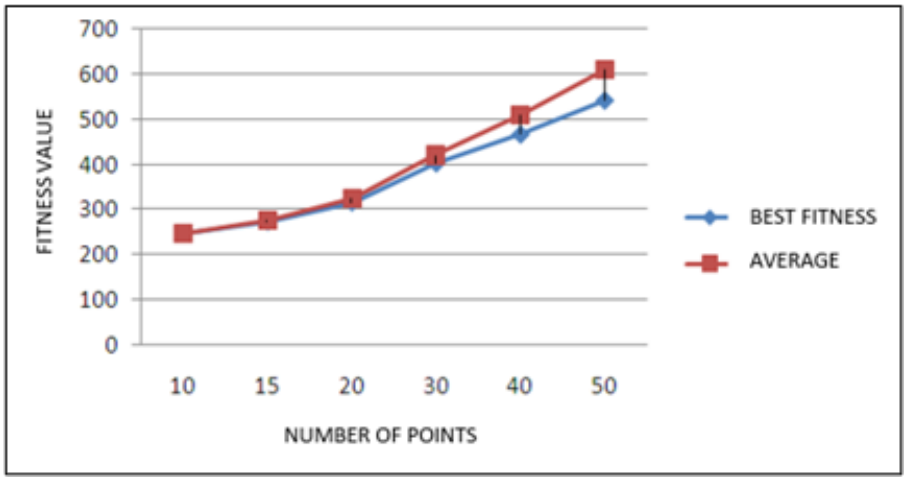


Figure 7: Difference between the mean and the best fitness for each group of points

Regarding the time taken by the GA [4] to produce a result, Table 3 shows ten runs of the GA for each grouping of points (from 10 to 50).

Table 3: Time of 10 runs of the GA with 200 generations

N° of points	GA Run Number									
	1	2	3	4	5	6	7	8	9	10
10	3,72	3,88	3,83	3,86	3,83	3,83	3,86	3,82	3,86	3,82
15	3,85	3,73	3,41	4,01	3,86	3,45	3,85	3,45	3,83	3,84
20	3,99	3,72	3,86	3,81	3,78	3,85	3,75	3,83	3,85	3,85
30	3,81	3,99	4,01	3,98	4,03	3,85	3,85	3,88	3,85	3,85
40	4,99	4,13	4,79	3,64	3,72	3,85	3,85	3,87	3,91	3,94
50	3,99	4,19	4,00	4,06	3,84	4,01	3,89	3,87	3,91	3,94

Table 4: Mean and SD of 10 runs of the GA with 200 generations

N° of points	Mean	SD
10	3,83	0,04
15	3,85	0,07
20	3,83	0,05
30	3,89	0,09
40	3,94	0,14
50	3,97	0,10

Based on the data from Table 3, the mean and standard deviation shown in Table 4 indicate that both the average and SD of the times for the ten runs of the GA for each grouping of points (from

10 to 50) do not show significant differences. This is because in genetic algorithms, the runtime is proportional to the number of defined generations, which in this case is fixed at 200 generations for all cases. However, slight increases in processing times can be observed as the number of points considered in each run increases. This trend is also reflected in Figure 8, where each runtime is plotted for each GA run per point grouping.

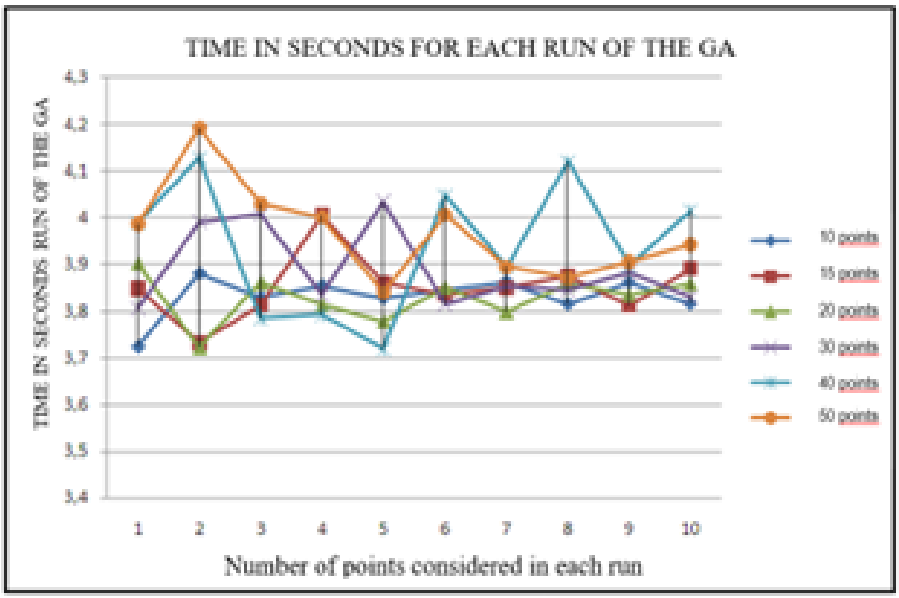


Figure 8: GA run process time

Based on the results obtained in Table 1, it is recommended that for a cluster containing 30 or more points, the number of generations and/or population size in the genetic algorithm code should be increased. However, this will result in longer execution times for the GA.

Considering the requirements for the design of the system in managing the Naval Vehicle, given the system's dynamics, the entire process of clustering creation, trajectory generation, and dispatch assignment should ideally take no more than one minute. Therefore, it is crucial for the GA to provide a suitable response as quickly as possible.

Next, Table 5 shows the results of GA runs for 50 points, varying the number of generations. The aim is to determine the impact on the times (in seconds) that the GA takes to deliver results by varying the number of generations in the model.

Table 5: GA result for 50 points varying number of generations

N° Generations	GA FITNESS			TIME GA RUN MEAN	
	BEST	MEAN	SD	MEAN	SD
300	530.6143	555.8617	13.0781	5.6279	0.047
400	501.5421	535.3700	20.6006	7.3773	0.073
490	486.8844	517.6826	20.2412	8.9190	0.065
600	479.3115	505.0828	13.1281	10.8182	0.089
800	467.8809	495.2497	17.6451	15.6451	0.826

Next, Figure 9 shows a graph illustrating that as the number of generations increases, the GA finds better fitness values and equivalently improves the average fitness values as shown in Figure 10.

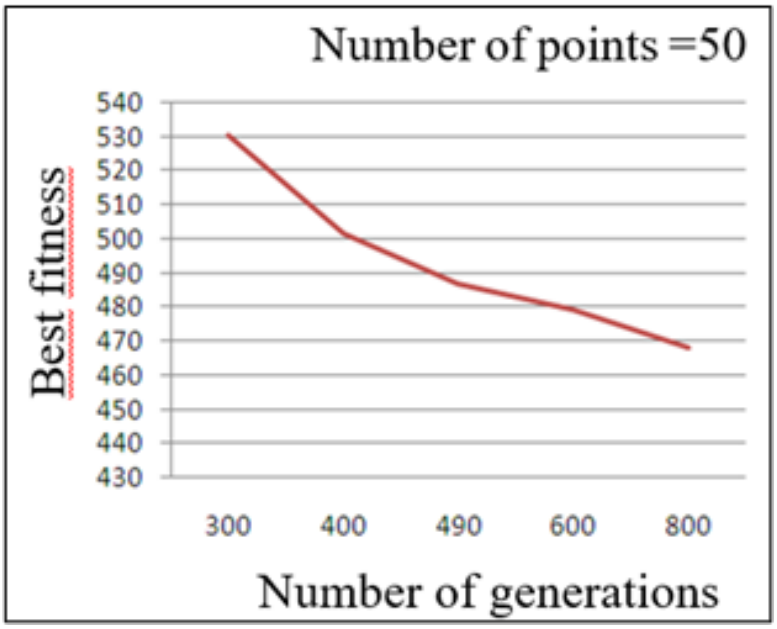


Figure 9: Variation of the best fitness value modifying the number of generations

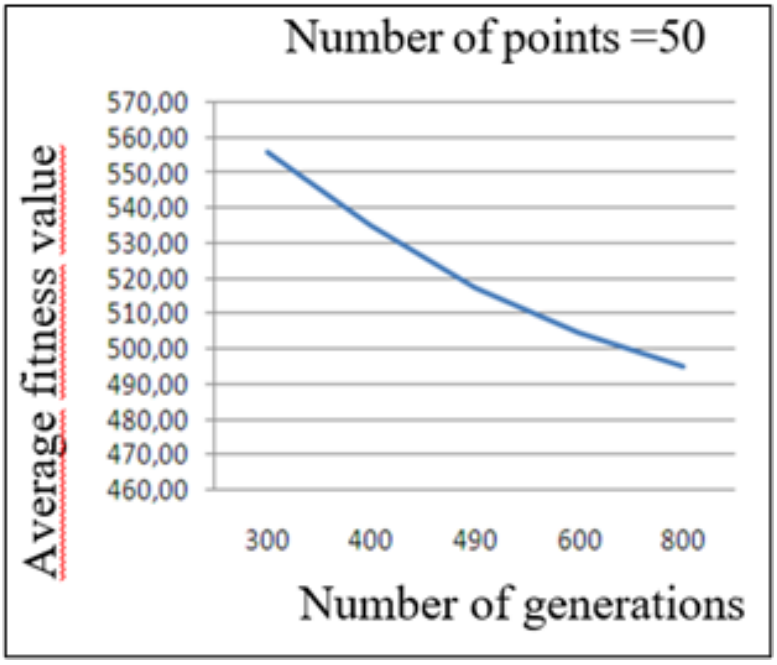


Figure 10: Variation of the mean fitness value with No. of generations

Finally, Figure 11 shows how the average time for the GA to deliver a result (run) increases as the number of generations also increases.

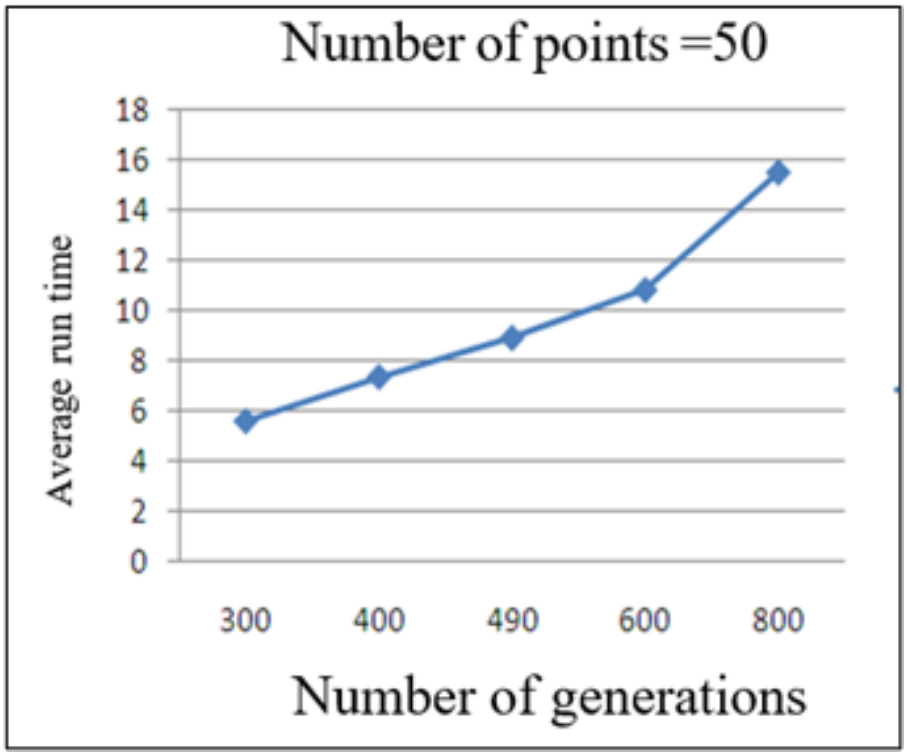


Figure 11: Variation of the average run time by modifying the number of generations

Next, we observe the behavior of the GA by changing only the population size of individuals. Table 6 shows results from several runs of the GA with variations in the population size ranging from 60 to 300 individuals. The values displayed represent the route distance obtained by the GA in each run (out of 10) with 50 points and 800 generations.

Table 6: Results of 10 GA runs varying the population number

N° of POP	N° run of the GA									
	1	2	3	4	5	6	7	8	9	10
60	506,17	485,23	516,57	527,14	486,94	471,37	505,47	493,74	485,08	488,02
100	470,72	476,97	479,20	506,01	477,49	482,27	495,24	488,95	478,84	486,37
150	480,25	478,40	484,01	479,90	489,10	458,97	487,17	494,44	491,00	481,14
200	502,17	468,84	482,19	477,69	485,97	493,25	495,48	507,88	486,92	498,95
250	477,09	483,11	458,97	482,60	515,47	496,54	508,13	471,85	489,81	474,30
300	466,81	458,42	482,11	490,62	461,59	492,07	495,62	469,80	500,19	477,48

The results shown in Table 6 indicate that there is no significant difference in the average value of GA runs for each population size when varying the number of individuals in the GA population. However, better results could be achieved since increasing the population size would explore other areas of the solution space due to mutations, but this does not guarantee finding a better solution. The number of GA runs would need to be increased.

Table 8 shows variations in population size and the impact on CPU-Timing in seconds.

Table 8: CPU-Timing (sec) of the GA in 10 run with 50 points and 800 generations

N° of POP	N° run of the GA									
	1	2	3	4	5	6	7	8	9	10
60	15,11	15,10	15,10	15,15	15,18	15,18	15,16	15,17	15,14	15,12
100	15,53	15,50	15,57	15,64	15,54	15,62	15,68	15,65	15,63	15,60
150	16,15	16,18	16,14	16,10	16,01	16,11	16,04	16,09	16,06	16,04
200	16,64	16,69	16,63	16,70	16,63	16,66	16,66	16,66	16,66	16,68
250	17,25	17,23	17,27	17,22	17,16	17,45	17,23	17,18	17,23	17,20
300	17,77	17,77	17,82	17,74	17,83	17,74	17,80	17,76	17,81	17,80

In this table, it can be observed that the CPU-Timing increases by an insignificant amount in terms of seconds. Therefore, this parameter is not crucial for finding quick and good solutions to the TSP using GA. However, it is recommended to keep a high value for this parameter. Based on the results shown in this analysis, the following values for parameters such as the number of generations and population are suggested according to the number of points. These values are shown in Table 9.

Table 9: Suggested GA parameter values according to the number of points

N° of Points	N° of Generations	N° of Population
10	200	60
20	200	60
30	950	60
40	950	60
50	600	200
60	800	200
70	900	200
80	900	200
90	950	200
100	1500	300
110	1500	300
120	1500	400
130	1500	400
140	1500	400
150	1500	400

8 Conclusions

This study presents a comprehensive analysis of the implementation of a Navigation System for autonomous naval vehicles, utilizing cluster formation and Genetic Algorithms (GA) for route generation. Throughout the work, various GA parameters, such as the number of generations and population size, have been evaluated for their impact on the performance and efficiency of the generated routes. The results indicated that for clusters with up to 10 points, the GA can find optimal solutions quickly. However, as the number of points increases, the accuracy and processing time of the GA also increase, suggesting the need to adjust the algorithm's parameters for larger clusters. It is recommended to increase the number of generations and population for clusters with 30 or more points, although this will increase execution time. Additionally, it has been shown that the GA can explore different areas of the solution space with a larger population size, though this does not guarantee better solutions in all cases. The integration of these elements, along with advanced sensing and data processing technologies, is crucial to ensure safety, efficiency, and regulatory compliance in autonomous navigation within challenging port environments such as Quintero Bay. In summary, a model has been designed and tested that determines the appropriate parameters to guarantee efficient routes, thus ensuring the feasibility of autonomous navigation in these environments.

9 Acknowledgment

Special thanks to the Department of Industrial Engineering from the University of Santiago de Chile for the support during the development of the study.

References

[1] Gayo, E., Smith, N., & Vásquez, P. (2022). Environmental Injustice and Sacrifice Zones: A Case Study of Quintero Bay, Chile. *Environmental Research Letters*, 17(4), 045001. doi:10.1088/1748-9326/ac5a2f.

[2] Khalil, L., & Alarcon, L. (2021). Atmospheric pollutants in Quintero Bay and their impacts. *Environmental Science & Pollution Research*, 28(18), 22867–22881.

[3] Muñoz, D., & Martínez, A. (2020). Industrial emissions and air quality monitoring in Quintero Bay. *Journal of Environmental Management*, 276, 111241.

[4] Mertens, M., & Gawel, E. (2019). Environmental monitoring by autonomous vehicles: Challenges and opportunities. *Environmental Science & Technology*, 53(12), 6663-6671. https://doi.org/10.1021/acs.est.9b00900.

[5] Rodrigues, J. P., & Coelho, H. (2018). Autonomous marine vehicles for environmental monitoring: Applications and technologies. *Journal of Marine Systems*, 180, 31-40. https://doi.org/10.1016/j.jmarsys.2018.01.003.

[6] Silva, M., & Costa, P. (2020). Integrating advanced technologies for autonomous marine vehicles in environmental monitoring. *Ocean Engineering*, 200, 107101. https://doi.org/10.1016/j.oceaneng.2020.107101.

[7] Walker, B. (2018). Reducing operational costs with autonomous marine vehicles. *Maritime Economics & Logistics*, 20(2), 245-258. https://doi.org/10.1057/s41278-018-0012-3.

[8] Thompson, L., & Brown, S. (2019). Optimizing mission planning for autonomous marine vehicles. *Journal of Field Robotics*, 36(4), 621-634. https://doi.org/10.1002/rob.21901.

[9] Green, R., & Palmer, T. (2020). Resource-efficient deployment of autonomous vehicles for environmental monitoring. *IEEE Transactions on Automation Science and Engineering*, 17(3), 1542-1554. https://doi.org/10.1109/TASE.2020.2986547.

[10] Johnson, M., & Andrews, K. (2021). Enhancing scientific research with autonomous marine vehicles. *Environmental Research Letters*, 16(11), 114020. https://doi.org/10.1088/1748-9326/ac1f1f.

[11] Chen, X., & Wang, Y. (2022). Autonomous vehicles as tools for marine ecosystem research. *Marine Ecology Progress Series*, 667, 1-15. https://doi.org/10.3354/meps13687.

[12] Yao, J., & Zhang, W. (2020). Path planning for autonomous underwater vehicles considering ocean currents and obstacles. *IEEE Transactions on Industrial Electronics*, 67(6), 5181-5190. https://doi.org/10.1109/TIE.2019.2938870.

[13] MathWorks. (2023). Solving the Traveling Salesman Problem using Genetic Algorithm. Retrieved from https://www.mathworks.com/help/gads/genetic-algorithm.html.

[14] Jain, A. K., & Dubes, R. C. (1988). Algorithms for Clustering Data. Prentice Hall. https://doi.org/10.5555/59311.

[15] Densham, P. J., & Rushton, G. (1992). A more efficient heuristic for solving large p-median problems. *Geographical Analysis*, 24(1), 36-46. https://doi.org/10.1111/j.1538-4632.1992.tb00250.x.

[16] MathWorks. (n.d.). Create distance matrix. In MATLAB documentation. Retrieved from https://www.mathworks.com/help/matlab/ref/squareform.html.

[17] MathWorks. (n.d.). Create distance matrix. In MATLAB documentation. Retrieved from https://www.mathworks.com/help/matlab/ref/squareform.html.

Journal's webpage: http://univagora.ro/jour/index.php/ijccc/

This journal is a member of, and subscribes to the principles of,
the Committee on Publication Ethics (COPE).
https://publicationethics.org/members/international-journal-computers-communications-and-control

Cite this paper as:

Olivares, V.; Oddershede, A.; Quezada, L.; Vargas, M.; Montt, C. (2024). Development of a Model for Generating Trajectories for an
Autonomous Naval Vehicle Using Genetic Algorithms in MATLAB, *International Journal of Computers Communications & Control*, 19(6), 6865, 2024.
https://doi.org/10.15837/ijccc.2024.6.6865