TRANSNAV
http://www.transnav.eu
the International Journal
on Marine Navigation
and Safety of Sea Transportation
Volume 19
Number 4
December 2025
DOI: 10.12716/1001.19.04.03

Determination of the Optimal Route in a Defined Environment for a Vessel

N. Popowniak
Gdynia Maritime University. Gdynia, Poland

ABSTRACT: The desire to minimise costs and increase reliability are driving change in the industry. The most optimal solutions exclude the human factor and instead introduce process automation. The first autonomous units are already sailing around the world, but this is a relatively new and very broad topic, so there is still a considerable area for research. The known control systems for ships are already well developed and proven in practice. The author of this paper aims to find an algorithm for determining the trajectory of a ship's course, so that an autonomous point-to-point voyage would be possible in conjunction with control systems. For this purpose, they chose four path planning methods: Hibrid A*, Rapidly-exploring Random Tree, Bidirectional RRT and Motion Planning Networks. The selected methods were pre-tested in a familiar environment. Simulation results for different settings were compared with each other to find the optimal path.

1 INTRODUCTION

In an era of rapid development in autonomous technologies, unmanned maritime vessels, which can navigate independently through the waters, minimizing human involvement in the navigation process, are gaining increasing interest. These vessels can be divided into three groups. The first, practically eliminating human involvement, are fully autonomous vessels, where all decisions are made by artificial intelligence based on data provided, including maps and sensors. Another group also belongs to unmanned vessels, but they are remotely controlled by a human on land. The final group consists of semi-autonomous vessels, where a crew is on board, and decisions are made by humans with the assistance of artificial intelligence.

Fully autonomous vessels promise a revolution in maritime transport. They offer the potential for greater safety by eliminating human error, the main cause of maritime accidents, and by eliminating the risk of crew kidnapping for ransom. Furthermore, artificial intelligence algorithms design routes, optimizing fuel consumption. They also require no rest periods, allowing them to operate 24/7. This, with many other factors, translates into lower operating costs[1,2].

Safe travel from point A to point B is possible thanks to dynamic positioning (DP), collision avoidance algorithms(CAS), or others systems used to control the ship[3]. This topic is being intensively developed by various researchers. An example is algorithms designed to control ship propulsors for precise point-to-point navigation like prediction control systems.[4,5] These ideas are still being developed and optimized[6,7]. For these algorithms to work properly, it is important to determine the appropriate trajectory that will enable the ship to move. The trajectory, or the path taken by the vessel, is one of the most important elements ensuring safe and efficient movement on the water. Its correct determination is

crucial for minimizing the risk of collision, optimizing fuel consumption, and adapting to changing environmental conditions such as currents, wind, and depth.

The route planning methods used in autonomous vessels are developing in parallel with advances in robotics, artificial intelligence, and satellite navigation. In practice, both classic graph algorithms, such as Dijkstra and A*, and more advanced approaches based on machine learning, evolutionary algorithms, fuzzy systems, and methods inspired by swarm behavior are used. It is a demanding field of research and ongoing experimentation that combines the fields of computer science, robotics and marine engineering.

This work will examine the following algorithms: Hybrid A*, Bidirectional RRT, MPNet, Rapidly-exploring Random Tree (RRT). Using Matlab, preliminary simulations will be carried out in an environment that reflects the real object—Lake Kamionka. The results will be compared, and then the algorithm that best determines the optimal trajectory will be used to determine the trajectory on the real model of the LNG Carrier Dorchester Lady.

2 ALOGRITHMS

The MATLAB environment is a powerful tool offering a wide range of utilities for determining the optimal path between a starting point and an endpoint. Among the available methods, key place are occupied by approaches based on graphs, probabilistic methods, optimization techniques, and complex hybrid methods. This very diversity necessitated a selection: the article's author initially reviewed several dozen different trajectory determination techniques, only for selected four for the final research, which will be discussed in detail.[8] Three of the selected algorithms utilize RRT to a greater or lesser extent. This chapter will describe them to highlight their differences and similarities. The fourth option, the Hybrid A* algorithm, is very popular and often chosen for comparison with RRT[9].

2.1 *Rapidly-exploring Random Tree (RRT)*

RRT is an algorithm for exploring high-dimensional spaces by randomly building a tree with branches that fill the space. The algorithm grows toward large, unexplored areas of the problem using samples taken randomly from the search space. One of the disadvantages of this solution is the lack of guarantee to find the shortest and optimal path[10]. Despite this inherent limitation, the method gained significant popularity after its introduction in the late 1990s, especially in robotics for quickly planning paths around complex obstacles. Researchers frequently address the sub-optimality issue by employing variants like RRT* (RRT-star), which incorporates a rewiring step to converge asymptotically to an optimal solution[11].

To define the algorithm we need:

- Configuration space C, being a set of all possible states
- Obstacle space C_{obs}, is a subset of C aimed at representing collision places
- Free space C_{free}, which is a subset of C without collisions. It can be defined as: $C_{free}=C \setminus C_{obs}$
- Starting point q_{init}, the start point of the algorithm, where $q_{init} \in C_{free}$
- Goal G, the region which the algorithm tries to reach
- Tree T, being a data structure consisting of a set of vertices V and a set of collision-free paths between vertices E
  $T=(V,E)$
- Metric ρ, meaning a function defining the distance between two points
- Expansion step ϵ, a constant or maximum length of a new branch in one iteration
- Number of iterations K
- SAMPLE (C), a function returning a random configuration q_{rand} from the space C
- NEAREST (T, q_{rand}), a function searching through the set of vertices from tree T to find the one closest to q_{rand}, returning it as q_{near}
- STEER (q_{near}, q_{rand}, ϵ) a function aimed at generating a new vertex q_{new}
- COLLISION_FREE (q_{near}, q_{new}) a function returning true if the trajectory connecting q_{near} and q_{new} does not pass through C_{obs}. Otherwise, it returns false and the algorithm performs next iteration.

The description of how it works can be explained in a few steps:

1. $V \leftarrow \{q_{init}\}$;
2. $E \leftarrow \varnothing$;
3. For k=1 to K:
   - $q_{rand} \leftarrow$ SAMPLE (C);
   - $q_{near} \leftarrow$ NEAREST (T, q_{rand});
   - $q_{new} \leftarrow$ STEER (q_{near}, q_{rand}, ϵ);
   - COLLISION_FREE (q_{near}, q_{new});
4. $V \leftarrow V \cup \{q_{new}\}$;
5. $E \leftarrow E \cup \{(q_{near}, q_{new})\}$;
6. Return T;

2.2 *Bidirectional RRT (BiRRT)*

BiRRT is an extension of the classic RRT. It simultaneously builds two trees—one from the starting point (T_{start}) and the other from the goal point (T_{goal}). The main purpose is to speed up the solution finding, although it requires additional computations. In this way, using BiRRT reduces the expected distance each tree must explore. Leading to faster solution finding in complex spaces compared to the unidirectional RRT. In practice, this efficiency stems from a simple principle: two small search spaces are always preferable to one large one[10].

Since the building process is similar to that of the standard RRT, only the attempt to connect the two trees will be described here, as this is the key operation in each iteration and drives the algorithm's work. After expanding tree starting at the start node ,the algorithm tries to use the newly added vertex q_{new}, as a target for the tree starting at the goal node, using the CONNECT operation. This connection step attempts to iteratively grow the target tree (T_{target})towards the q_{new}. The operation succeeds if T_{target} manages to reach q_{new} within a specified distance, creating the path link. If the operation fails, T_{target} is expanded as far as possible towards the new vertex. This loop can either return a new vertex, or end the algorithm with one of two outcomes: *PathFound* or *Trapped*. The algorithm ends when the two trees are successfully connected.[12]

2.3 *MPNet Planner*

Trajectory planning for autonomous systems is based on an advnced learning algorithm that effectively combines the speed of machine learning with the reliability of classical methods- such as RRT. The main advantage of this approach lies in the fact that the system is not forced to re-explore the space in a time-consuming manner every time a new task arises. The algorithm learns a generalizable heuristic, significantly accelerateing the process of finding the optimal path. The architecture of this solution, often implemented in variants of the MPNet network, is two-part[13]. The first module is responsible for information condensation, encoding the input map environment into a compact representation using the base point set encoding method. Then, this encoded information goes to the second module, which is the Feedback Network. This network, further supplied with the current initial and target states, is designed to predict the next state that will bring the vessel closer to its goal. The model usually consists of a module encoding the environment and a Planning Network, which is a deep neural network that takes as input:

- Encoded environment E
- Starting point q_{start}
- Goal point q_{goal}, generates a set of proposed intermediate states $(q_1,q_2,\ldots,q_k)$. Which in simplification can be presented as:

$$f\left(g_{start}, g_{goal}, E\right) \rightarrow \left(q_1, q_2, \ldots, q_k\right)$$

Then, the algorithm checks if each trajectory between the pair q_i, q_{i+1} is free from collisions. If the segment is safe, it is kept, otherwise a new pair of points is determined. In such case, the algorithm can in first option reuse the neural network to find a collision-free connection, but it has a limited number of tries. If it cannot find a path, it switches to a classical algorithm to find the connection[14].

2.4 *HybridA**

The Hybrid A* algorithm is a combination of global grid-based path searching with a realistic vehicle movement model. In its essence, it acts like classic A, because it expands nodes in a discretized state space, but using heuristic. It uses advanced metrics, such as the Dubins distance, to effectively guide the search towards the goal. Thanks to this, the algorithm does not wander, and its work is target-oriented. Hybrid A* is ideal for path planning for nonholonomic vehicles. This is its biggest advantage. Instead of simply move to an adjacent grid cell, it generates subsequent states based on a simulation of vehicle motion. This means that from each start node, it simulates feasible vehicle maneuvers, like driving straight, turning left, and turning right[15,16].

Combines the A* algorithm (discrete search) with the possibility to consider kinematic constraints. It searches through the three-dimensional state space SE(2):

$$s = \left(x, y, \theta\right)$$

where:
x,y are positions in the coordinate system
θ is the vehicle's orientation angle

To enable graph search (like in A*), the SE(2) space is partially discretized by dividing x and y into a grid of cells, and θ is divided into a finite number of segments.

To evaluate each state, we use:

- the actual cost $g(s)$ of traveling from the starting state s_{start} to the current state s
- the heuristic $h(s)$, which is an estimated cost from s to s_{goal}
- the estimated total cost $f(s)$, described by the formula:

$$f\left(s\right) = g\left(s\right) + h\left(s\right)$$

In the case of the Hybrid A* algorithm, when the current node is close to s_{goal}, the algorithm attempts to use a Reeds-Shepp connection to check if it can directly connect these two points. This process is called an analytical expansion mechanism.

3 RESULTS

Each of the algorithms was tested four times with different settings. For RRT, BiRRT, and MPNet, the parameter "Maximum length of motion" was changed, while for HybridA, it was "Length of motion primitives." While trying to find the optimal path, I assumed that the minimum distance between turning points should not be less than three times the length of the vessel model to ensure smooth control. It was also assumed that trajectories should not come closer to obstacles than 2 meters, so this distance was added to each obstacle boundary using inflate. For the first three algorithms, setting the value to 50 caused a very high number of course changes, which did not meet the assumptions, so a reduce function was applied to decrease the number of points.

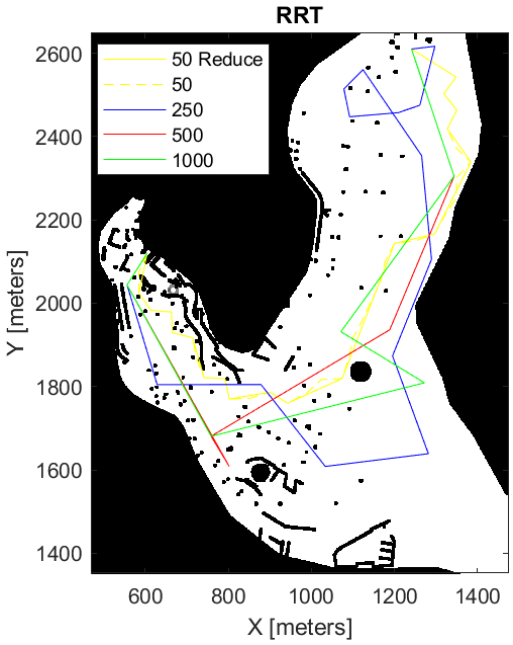

Figure 1. Routes determined by the RRT

The visualization above illustrates the solutions obtained for the RRT algorithm. It can be observed that the generated trajectories are non-optimal and differ significantly from routs a human driver would choose. For a Maximum length of motion set to 50, unnecessary maneuvers between obstacles are visible, which could be navigated around the side. For the remaining values, the algorithm significantly extended the routes; for instance, at a value of 250, it creates a large loop around obstacles, and at a value of 500, it performs a reverse motion, traversing the same segment twice.

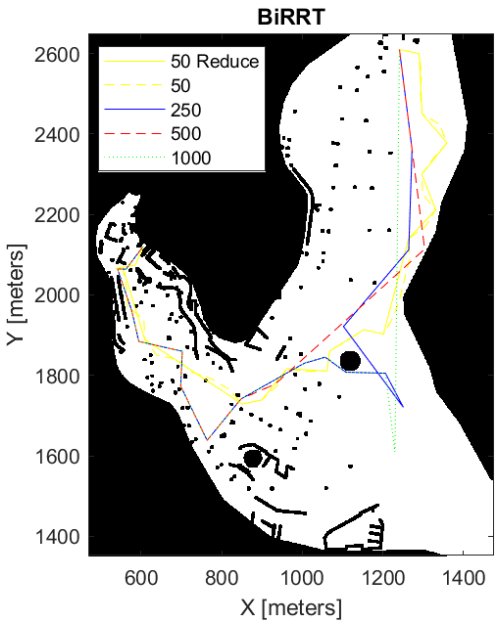

Figure 2. Routes determined by the BiRRT

Trajectories generated by the Bi-RRT algorithm are far from optimal, but show a noticeable improvement compared to the standard RRT. For the three largest values of Maximum length of motion, beginning of T_{start} are identical. To make sure we could tell the difference between the paths, we marked them with dashed lines as well as colors. The critical moment was the merging point of the algorithm's trees, where an unnecessary route extension is noticeable for the values of 1000 and 250.

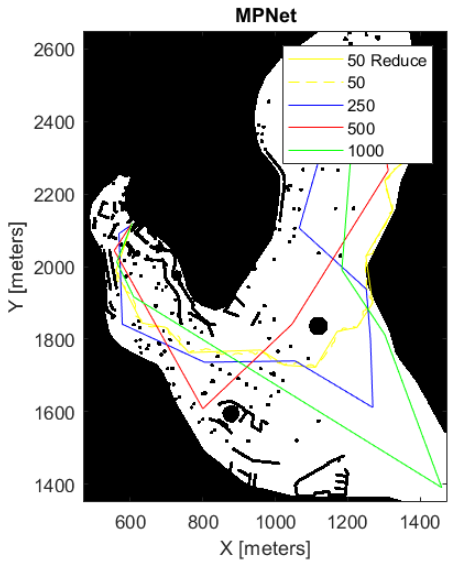

Figure 3 Routes determined by the MPNet

Analyzing the results for MPNet, it can be stated that the obtained paths also do not belong to the optimal set, primarily due to their very long trajectories. This is particularly visible for the Maximum length of motion value of 1000, where one of the nodes is located at a significant distance from the other routes.

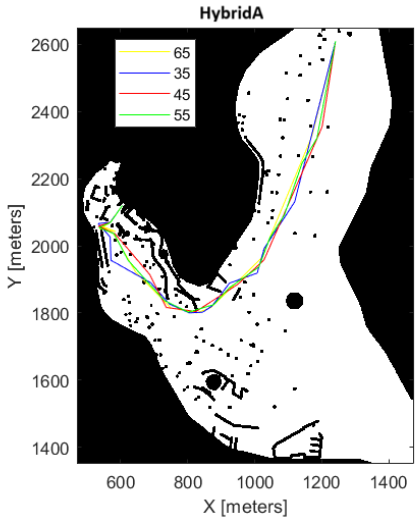

Figure 4. Routes determined by HybridA*

The trajectories generated by Hybrid A* are very similar to each other, often intersecting. The largest differences can be observed at the beginning of the routes, where obstacles are navigated around in various ways depending on the setting of the Length of motion primitives. Then, the algorithm generated the trajectories so similarly, that they pass perfectly between the same obstacles.

Looking at the results of the first three algorithms, it can be noticed that the trajectories determined are far from optimal, mainly because they are very long. However, for HybridA*, the results seem to be very good. But, when looking at the beginnings of the trajectories, unnecessary maneuvers during the port exit can be observed, which was shown in Figure 5

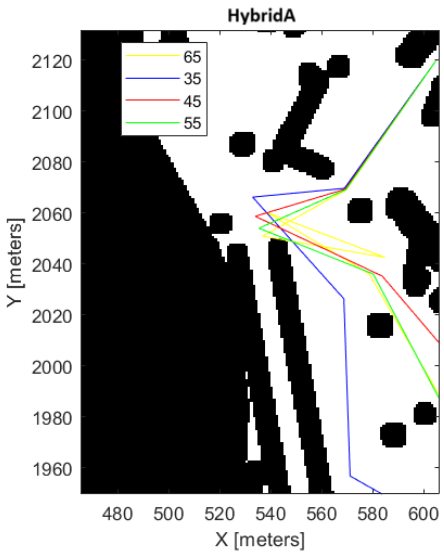

Figure 5. Close-up of the beginning of the HibridA routes

For better comparison of the results and maintaining readability, the best route was chosen for

each algorithm and then placed on a comparative graphic:

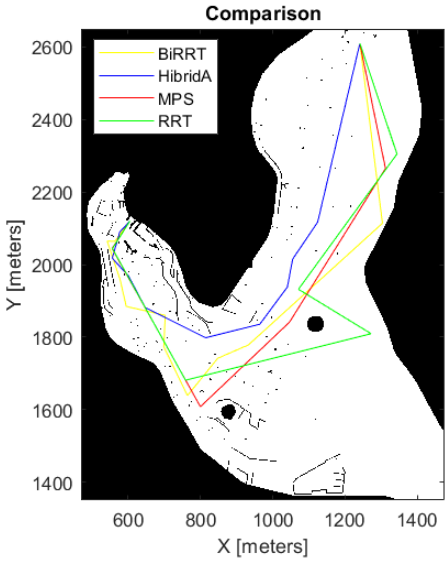

Figure 6. Comparison

To ensure a better comparison of the generated routes, parameters such as the route length in [m] and the number of turning maneuvers were calculated for each of them. Thanks to this, the best trajectory will be selected not only based on the author's subjective opinion but will be supported by data.

The parameters of each route were presented in the table below:

Table 1 Comparison of route parameters

Name	Length [m]	Turn Points
RRT 1000	2054.23	6
RRT 500	1896.21	6
RRT 250	2598.82	15
RRT 50	1826.38	37
BiRRT 1000	2351.76	14
BiRRT 500	1851.57	11
BiRRT 250	2250.95	17
BiRRT 50	1918.68	39
MPNet 1000	2509.16	7
MPNet 500	1782.15	5
MPNet 250	2158.78	12
MPNet 50	1875.81	17
Hybrid 55	1451.42	14
Hybrid 45	1458.97	10
Hybrid 35	1490.58	14
Hybrid 65	1546.48	12

The shortest trajectory was generated by the Hybrid A* algorithm, while the longest were produced by RRT 250 and MPNet 1000, which differ by a mere 90 meters. As could be suspected based on the visualizations, the routes closest to each other in terms of distance were generated by the Hybrid A* algorithm. However, the fewest turning maneuvers were obtained for MPNet 500. However, in the case of sea navigation, fewer turns are less important than the length of the route traveled.

Considering all these parameters, the HybridA* algorithm was selected for Length of motion primitives of 55 as the optimal solution.

This trajectory was supposed to be tested on the real object, which is Dorchester Lady located on Lake Kamionka. However, due to training happening at that time, one of the ports was not available, so a new starting and ending point was determined. The algorithm easily calculated the new trajectory, after which the vessel passed without collision[17,18]

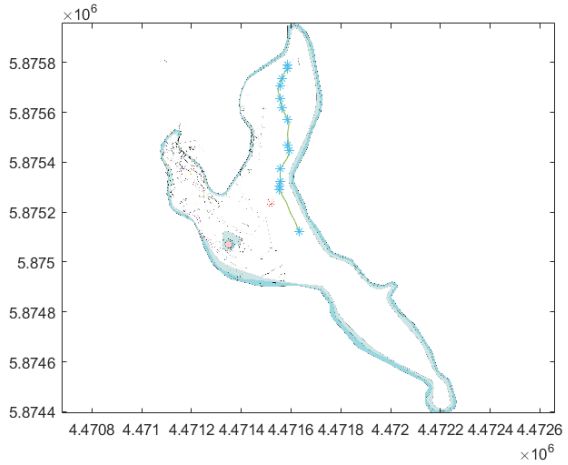

Figure 7. Traveled route

4 SUMMARY

The work carried out showed the advantages and disadvantages of each algorithm. Although the determined routes were collision-free, they contained too many turns, were too long, or both. This work was the first step in familiarizing with the operation of the available algorithms, to understand their principles and to help the author get involved in the topic of trajectory planning for watercraft.

REFERENCES

[1] Ziajka-Poznańska, E., & Montewka, J.Costs and benefits of autonomous shipping—a literature review. Applied Sciences,2021, 11(10), 4553.

[2] Chaal, M., Ren, X., BahooToroody, A., Basnet, S., Bolbot, V., Banda, O. A. V., & Van Gelder, P. Research on risk, safety, and reliability of autonomous ships: A bibliometric review. Safety science,2023, 167, 106256.

[3] Witkowska, A., Śmierzchalski, R. Adaptive dynamic control allocation for dynamic positioning of marine vessel based on backstepping method and sequential quadratic programming. Ocean Engineering, 2018, 163, 570-582.

[4] Koznowski, W.; Kula, K.; Lazarowska, A.; Lisowski, J.; Miller, A.; Rak, A.; Rybczak, M.; Mohamed-Seghir, M.; Tomera, M. Research on Synthesis of Multi-Layer Intelligent System for Optimal and Safe Control of Marine Autonomous Object. Electronics 2023, 12, 3299

[5] Miller, A. Model predictive ship trajectory tracking system based on line of sight method. Bulletin of the polish academy of sciences, 2023,71(5)

[6] Miller, A.; Rak, A. Measurement System for the Environmental Load Assessment of the Scale Ship Model. Sensors 2023, 23, 306

[7] Gierusz, W.; Miller, A. . Prediction control systems in marine applications. TransNav: International Journal on Marine Navigation and Safety of Sea Transportation, 2020, 14(2)

[8] Brzózka,J.,Dorobczyński L., Wydawnictwo Naukowe PWN. MATLAB : środowisko obliczeń naukowo-technicznych. Wyd. 1, dodr. 1. Warszawa: Wydawnictwo Naukowe PWN, 2008

[9] Raibail, M., & Rahman, A. H. A.. Path Planning of Multi-robots in Confined Spaces. In International Conference on

Robot Intelligence Technology and Applications, 2023, (pp. 200-208). Cham: Springer Nature Switzerland.
[10] Aliyu, D. A new potential field-based algorithm for path planning. Journal of Intelligent & Robotic Systems, 1996.
[11] Rodriguez, S., Tang, X., Lien, J. M., & Amato, N. M. An obstacle-based rapidly-exploring random tree. In Proceedings 2006 IEEE international conference on robotics and automation, 2006. ICRA 2006. (pp. 895-900). IEEE.
[12] Jordan, M., & Perez, A. Optimal bidirectional rapidly-exploring random trees,2013
[13] Qureshi, Hussain, A. ,Miao, Y., Simeonov, A., C. Yip. M."Motion Planning Networks: Bridging the Gap Between Learning-Based and Classical Motion Planners." IEEE Transactions on Robotics 37, no. 1, 2021, 48–66.
[14] Qureshi, A. H., Simeonov, A., Bency, M. J., & Yip, M. C. Motion planning networks. In 2019 International Conference on Robotics and Automation (ICRA),2019, (pp. 2118-2124). IEEE.
[15] Dmitri, D., Thrun, S., Montemerlo, M., Diebel, J. Practical Search Techniques in Path Planning for Autonomous Driving. American Association for Artificial Intelligence, 2008.
[16] Petereit, Janko, Emter, T., Frey, W.C., Thomas Kopfstedt, and Andreas Beutel. Application of Hybrid A* to an Autonomous Mobile Robot for Path Planning in Unstructured Outdoor Environments, ROBOTIK 2012, 2012, pp. 1-6.
[17] Miller, A., & Walczak, S. Maritime autonomous surface ship's path approximation using Bézier curves. Symmetry,2020, 12(10), 1704.
[18] Miller, A., Rybczak, M., & Rak, A. Towards the autonomy: Control systems for the ship in confined and open waters. Sensors,2021 21(7), 2286.