COLREG-CBS: COLREGs-Compliant Search-based Path Planning for Autonomous Surface Vehicles

Kumar Katyayan Jaiswal, Rahul Kulkarni, Saifullah Khan, Schwitaan Iyer,Dr. Sujit P.B.

[1] Indian Institute of Science Education and Research Bhopal
Bhauri Bypass Road, Bhopal
Madhya Pradesh, India
kumarkatyayanjaiswal@gmail.com

Abstract

Autonomous surface vehicles (ASVs) are essential for several maritime applications, however, they interact with other marine vessels (non-ASVs) and ASVs in the open sea. Thus, the collision avoidance algorithms use for ASVs need to meet COLREGs rules at sea. The current collision avoidance algorithms do not scale with increasing number of agents. In this paper, we cast the collision avoidance problem as a conflict-based search problem , that is, COLREGs complaint, called as COLREG-CBS. We modify the CBS to generate path in a hexagonal grid search space meeting the COLREGs requirements and also incorporate "wait" as a possible action for determining collision free paths. The algorithm's effectiveness is demonstrated through experiments with real ASVs and computer simulations.

Introduction

An autonomous surface vehicle (ASV) is an unmanned vessel that operates on the surface of water without requiring direct human intervention. The key distinction between an ASV and an unmanned surface vehicle (USV) is that a USV might be controlled by a human operator indirectly. An ASV, on the other hand, is a vessel that can operate on its own without any human interference whatsoever (Vagale et al. 2021). Multi-Agent Pathfinding (MAPF) is an artificial intelligence problem that deals with the generation of optimal paths for agents from their respective start to goal positions such that none of the agents collide with static or dynamic obstacles along the paths. The algorithm presented over here is a centralized MAPF algorithm. , that is,, there is a common trajectory planner for all the agents, in this case, ASVs, which generates trajectories for all of them.

Conflict-based search is a centralized algorithm , that is, known for providing optimal solutions for MAPF problem instances (Sharon et al. 2015). The formulation of the algorithm can be broken down into three parts:

1. The formulation of the search space, the action space, and the single agent path finding heuristics.
2. Formulation and visualization of situations where the International Regulations for Preventing Collisions at Sea (COLREGs) are applicable. Formulation of newer conflicts, which constitute a COLREGs violation.
3. The formulation of newer constraints for prohibiting movements as minimal as possible for producing efficient and low cost paths.

It is important to note that the algorithm is non-reactive in nature. , that is,, the paths of all the vehicles are planned and re-planned together at all times and circumstances. ASVs under the influence of a reactive algorithm would always tend to take the most preferred action at each timestep; therefore such a strategy does not exhaustively search the solution space. Since the ASVs operate at sea along with other manned vessels, therefore it is imperative that these vessels obey the COLREGs. h, the regulations laid down for all the power-driven vessels operating at the high seas.

Related Work

There is extensive work done towards the COLREGs-compliant path planning of ASVs while avoiding collisions (Hu et al. 2022). In the early 2000s, rule-based approaches were used for path planning where the decision-making of the ship was dependent on analyzing its current state and then, according to some predefined rules, choosing the next possible action (Benjamin 2002),(Naeem, Irwin, and Yang 2012). However, the main limitation of such approaches is that they are dependent on the handcrafted design of ships, which is very hard to replicate in real-life vehicles. The hybrid methods of path planning involve the use of A*-variants (Wang and Xu 2020), which involve a global trajectory planner that plans trajectories considering the static obstacles and a local trajectory planner that generates instantaneous waypoints for collision avoidance whenever a dynamic obstacle (another ASV or a ship) is in close vicinity; however, path optimization is not a priority for such local planners since they are reactive in nature. Another widely popular hybrid approach is that of using COLREGs-Compliant Rapidly-exploring Random Trees (RRT), which randomly samples the search space to generate local waypoints. Such an approach might generate hazardous as well as suboptimal waypoints (Chiang and Tapia 2018). Another famous reactive approach is the use of the potential field method, which avoids collisions by generating repulsive forces; however, since this method is also reactive, it does not involve the op-

timization of trajectories (Lyu and Yin 2019). There are various optimization-based path planners; however, the major drawback of such path planners is the high computational cost (Shan et al. 2020),(Zhao et al. 2014). The most recent approach for designing path planners for ASVs is that of using Deep Reinforcement Learning (DRL). Deep reinforcement learning methods such as (Wei and Kuo 2022) and (Wen et al. 2023) demonstrate high efficiency in collision avoidance. However, using a learning algorithm raises the question of guarantees. This work attempts to enhance the ability of path planners to find efficient and low cost paths through search-based path planning in centralized multi-agent systems.

Conflict-Based Search

Conflict-based search (CBS) involves finding individual paths for all the ASVs and storing all the paths in a node. This is done using a low-level solver. Then, the high-level solver searches for any conflicts (collisions) within the trajectories of all the vehicles. If a conflict is found, child nodes are created for each agent involved within the conflict along with a constraint to prevent the agent from making the movement which led to the conflict. A constraint either prohibits an agent from walking into a *vertex* or from crossing a particular *edge* between two consecutive timesteps. Then, the low-level solver computes a new trajectory for the newly constrained agent in each of the nodes created and the newer set of trajectories are stored. The total trajectory or path cost is also stored for each node. The nodes are then stored in a set. Next, the set returns the node with the lowest path cost and the whole cycle is repeated until the high-level solver encounters a node with no conflict.

Action Space

Since MAPF is done on grids and under discretized timesteps, we divide the whole area into **hexagonal grids**. This means that in order to move from one grid to another, an ASV can move in one of these six directions: **North, North-East, North-West, South, South-East, South-West**. At any timestep, an agent can choose to be at any of the 6 neighboring grids during the next timestep, or it can decide to **Wait** at the current grid.

Path Finding for Individual agents

We require that the algorithm return a trajectory , that is, optimal for an individual agent under some specific set of constraints. A trajectory is considered optimal if it minimizes the number of discretized timesteps taken to complete the journey. For an agent, the total time taken during the journey is directly proportional to the total distance covered. This is because, between two consecutive timesteps, the agent can travel between adjacent grids; however, this direct proportionality breaks with the inclusion of the action of *wait*.

A*, heuristics and penalties

We choose A* for planning the trajectories. While planning a path from node j to node n, the agent iteratively chooses the next unvisited node to visit by finding the node that has

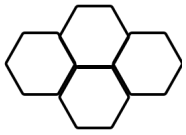

Figure 1: Hexagonal cells.

the lowest value of $f(n')$ where $f(n') = g(n') + h(n')$, $g(n')$ is the total length of the trajectory from the starting grid to the grid n'. And $h(n')$ is the total length of the trajectory returned if a path is planned from node n' to node n. The agent keeps visiting different unvisited nodes in this manner until and unless it reaches node n. To ensure that the A* algorithm returns optimal paths, we need to first ensure that the $h - heuristic$ is *admissible* and *consistent*. Therefore, we choose *Chebychev distance* as the suitable distance estimator. The *Chebychev Distance* between grids with centroid coordinates (a, b) and (c, d) is given by $max\{|a-c|, |b-d|\}$. Since we want to optimize time, our objective is to minimize the set of actions an agent takes throughout its journey. This is because the agent performs one out of the seven available actions at each timestep. However, we should take note that the units of our heuristics are *distance* and not *time*. Now, this strategy would work in optimizing *time* if we don't consider the action of *wait*. When A* uses distance as a heuristic rather than time, it might give trajectories that optimize only the former, even if the agent might have to wait for hundreds of timesteps.

Penalizing the agent for waiting

To overcome this issue, we introduce some form of penalty when the agent decides to *wait* so that the agent doesn't wait unnecessarily rather than finding a longer yet quicker path. To solve this, we introduce special kinds of nodes called *dummy nodes*. When an agent decides to *wait* at a node *n* it simply moves to a *dummy node n'* which is nothing but an exact replica of n with the exact same neighbors as those of n and $g(n') - g(n) = d_c$, and $h(n') - h(n) = 0$, where d_c is basically the distance between the centroids of any two adjacent nodes. What we are doing is basically penalizing the agent to move a distance d_c without moving any closer or farther to the target. Let us introduce a quantity called *hypothetical distance* d', which is different than *d* , that is,, the actual distance covered by the vehicle:

$$d' - d = d_c * t_w \tag{1}$$

where t_w is number of timesteps at which the agent chooses to *wait*.

Lemma 1: *For an agent traveling from grid a to grid b subject to some constraints, the path finding algorithm minimizes hypothetical distance covered during the journey.*

Proof: Since, the A* algorithm uses an *h-heuristic* which is both *admissible* and *consistent*, and to *wait* means to move a distance without actually moving an inch in reality, the *hypothetical distance* is optimized.

Theorem: *For an agent traveling from grid a to grid b subject to some constraints, the path finding algorithm returns a trajectory which covers the whole journey within the minimum possible time.*

Proof: Let d and d' be the *hypothetical distances* covered by the trajectories k (the returned trajectory) and k' (a random trajectory within the solution space) respectively. Since, we know that $d \leq d'$, and -

$$\textit{Hypothetical Distance} = d_c * t \tag{2}$$

where t is total number of timesteps required to complete the journey. Since, between timesteps t and $t + 1$, the agent covers a *hypothetical distance* of d_c no matter what action it takes. Therefore, if t_k and $t_{k'}$ are the times taken by the two paths,

$$d_c * t_k \leq d_c * t_{k'} \tag{3}$$

Therefore, $t_k \leq t_{k'}$.

COLREGs-COMPLIANT BEHAVIOR ON GRIDS

We shall briefly describe the collision avoidance rules (COLREGs). There are two situations during which the ASVs are expected to comply with the COLREGs. Please note that the *starboard* side of a vessel refers to the right side of a person standing on the vessel and facing its mast, while *port* side is exactly opposite to the *starboard* side.

- *Crossing Situation*: If two power-driven vessels are crossing each other's path such that the chances of collision seem very plausible, the vessel that has the other on her own *starboard* side shall take the responsibility of taking the necessary measures, such as altering its trajectory or waiting, and shall avoid crossing ahead of the other vessel Figure (3).
- *Head On Situation*: If two power-driven vessels are meeting on reciprocal courses, then, in order to minimize the risk of collision, both of them should turn to their *starboard* sides so that each shall pass on the *port* side of the other Figure (2).

For constraining the movement of vessels, first we develop a procedure for detecting COLREGs-violation among a set of paths. We state a few characteristics of the trajectories generated by the algorithm.

- The waypoints generated by the proposed high-level algorithm coincide with the centroids of cells. Moreover, these waypoints are time stamped. The low - level planner plans the intermediate waypoints between any two consecutively time stamped waypoints generated by the high-level algorithm.
- The gap between consecutive timestamps is uniform and hence these are called timesteps.

Each node N returned by the high level of the CBS algorithm contains trajectories for each agent involved. We require a notation for stating the cell location of an agent at a particular timestep according to the trajectory generated by the high-level algorithm. Therefore, we equip each node N with a function x. If according to the set of trajectories returned by a node N, an agent a traveling from v_1 to v_2 occupies a vertex v at time t and takes T timesteps for reaching its destination, then:

$$x_N(a, t) = v \tag{4}$$

$$x_N(a, t) = 0 \quad \forall\, t > T \tag{5}$$

$$x_N(a, 0) = v_1 \;\&\; x_N(a, T) = v_2 \tag{6}$$

It is simple to observe that the minimum number of grids that are present within any trajectory for classifying it as straight or bent is equal to three. We introduce some important symbols and definitions:

- *Representation of a trajectory using notations:* The trajectory of an agent traveling from v_n to $v_{n'}$ in T timesteps can be denoted by $v_n^0 v_x^1 v_y^{\widetilde{2}} v_z^{\widetilde{2}} v_x^2 v_z^3 .....v_{n'}^T$. The superscripts denote timesteps and the subscripts denote the identifiers for each of the cells.
- *Sub-trajectory of a trajectory:* A trajectory V of the form $v_n^{T_0} v_x^{T_1} .....v_y^{T_n}$ is called a sub-trajectory if it is completely contained within another trajectory V', $T_n - T_0 = 2$ and the vertex occupied at T_0 is different than the one occupied at T_1.
- A sub-trajectory can be classified into two parts that a sub-trajectory can be either straight or collision avoiding. If the vertex occupied at T_2 according to the sub-trajectory is different than the the one occupied at T_1 and the centroids of the vertices occupied at T_0, T_1 and T_2 are collinear. Otherwise, the sub-trajectory is collision avoiding.
- Every sub-trajectory which is not straight may have a corresponding imaginary straight sub-trajectory which can be obtained by finding a vertex v such that the centroids occupied within the sub-trajectory at T_0, T_1 and the centroid of v fall in the same line. Any vertex that is not a part of the actual trajectory but a part of an imaginary sub-trajectory traversed by the agent is called an imaginary vertex with respect to the agent. In figure (4), the vehicle's trajectory deviates by 60° towards the green grid; however, there are red waypoints leading to the red grid. If it is assumed that the movement from cell-to-cell is without any delay that is the agent does not wait at any of the intermediate timesteps, then it can be assumed that the figure showcases a sub-trajectory which is not straight however the red waypoints showcase the corresponding straight sub-trajectory. If according to the trajectory of agent a, $x(a, T) = v_{white}$ then this naturally implies that $x(a, T + 1) = v_{yellow}$ and $x(a, T + 2) = v_{green}$. Note that in this case, v_{red} is an imaginary vertex.
- If a collision avoiding sub-trajectory has a corresponding straight sub-trajectory that is involved in an edge conflict or a vertex conflict with the sub-trajectory of another vessel implies that the former's sub-trajectory demonstrates a collision avoiding behavior.
- Apart from the function x, we equip each node with function x' to refer the imaginary vertices covered by the

agent due to the angular deviation it takes while successively traversing vertices. For example, in figure (4), $x'(a, T+2) = v_{red}$.

- To represent imaginary vertices which are covered by an agent because of delay, we introduce another function called S. The function S takes a timestep and the agent as an input and simply outputs the next immediate vertex that the agent will occupy according to the trajectory of the agent. S is useful as it allows us to refer to the next immediate vertex without being aware of the exact timestep at which the agent will reach the vertex. For example, consider a sub-trajectory $v_x^0 v_y^1 v_y^2$ which is contained within the trajectory $v_x^0 v_y^1 v_y^2 v_z^3$. For this trajectory, $S(a,0) = v_y$ & $S(a,1) = v_z$. It can be observed that the sub-trajectory is certainly not a straight one. The corresponding straight imaginary sub-trajectory is $v_x^0 v_y^1 v_z^2$, given that the centroids of x, y & z are collinear. Therefore, in this case, v_z becomes an imaginary vertex for the sub-trajectory. However, under the condition that, x, y & z are not collinear, the corresponding sub trajectory to $v_x^0 v_y^1 v_y^2$ that is $v_x^0 v_y^1 v_z^2$ is not straight. Let v_k be a vertex such that v_x, v_y & v_k are in the same line. Hence, $v_x^0 v_y^1 v_k^2$ is the corresponding straight sub-trajectory. However, in this special case, both $v_x^0 v_y^1 v_k^2$ & $v_x^0 v_y^1 v_z^2$ are the corresponding imaginary sub-trajectories. The former is included because of the angular deviation, while the latter is included because of the delay.
- We introduce notations for showing the corresponding imaginary sub -trajectories of various sub-trajectories within the actual trajectory of an agent. Consider the trajectory considered within the previous case that is, $v_x^0 v_y^1 v_y^2 v_z^3$. This trajectory can be rewritten as $v_x^0 v_y^1 v_y^{\widetilde{2}} \overline{v_k^2} v_z^{\widetilde{2}} v_z^3$. All the vertices which have the same superscript are written together. Therefore, we have found a way to include both $v_x^0 v_y^1 v_k^2$ & $v_x^0 v_y^1 v_z^2$ within the actual trajectory using special symbols. The symbols $\widetilde{\ }$ & $\overline{\ }$ are introduced for incorporating the two different types of sub-trajectories.
- We introduce a symbol $\aleph$, where $k_1 \aleph k_2$ implies that k_1 is to the starboard of k_2.

Figure (11) shows a red vessel over a green grid. If the vessel travels via the red grid to another green grid shown within the figure, it can be observed that throughout this movement, the vessel deviates from its original direction by an angle of $60°$ however, if a vessel travels via red to another yellow grid, then the deviation is of $120°$.

If agent a makes no turn at t - 1 or t is greater than T , that is, the total number of timesteps taken by the agent to complete its journey, then :

$$x'_N(a,t) = 0 \tag{7}$$

For an agent a which completes its journey in T timesteps,

$$S_N(a,t) = x_N(a,t') \text{ where } t' \in \{t+1, T-1\} \tag{8}$$

$$\text{such that } x_N(a,t') \neq x_N(a,t'-1) = x_N(a,t) \tag{9}$$

$$\text{and } S_N(a,t) = 0 \text{ if } t \geq T \tag{10}$$

Note that if an agent a $waits$ from t - 1 to t .t $x_N(a,t-1)$ = $x_N(a,t)$, then

$$S_N(a,t-1) = S_N(a,t) \tag{11}$$

Let the symbol *B* denote a blue vessel and *R* denote a red vessel. In Figure (7), had *B* not *turned* and maintained its course over the green grid, it would have been in a *vertex conflict* with *R* over the red grid at timestep t. Now, this is a violation of COLREGs since it is the responsibility of *R* to take the necessary measures as mentioned within the *Crossing Situation* of COLREGs. This is because $B \aleph R$ at t - 1. Therefore, if T_a & T_b denote the trajectories of a and b respectively then a pair of vessels *a* and *b* are in a *secondary vertex conflict* denoted by *(a,b',v,t)* if :

$$v^t \subseteq T_a \text{ \& } \overline{v^t} \subseteq T_b \text{ and b } \aleph \text{ a at } t \text{ - } 1 \tag{12}$$

given that,

$$\angle x_N(b,t-2)_c x_N(b,t-1)_c S_N(b,t-1)_c = 120° \tag{13}$$

else, if $\overline{v^t} \not\subseteq T_b$:

$$\widetilde{v^t} \subseteq T_b \tag{14}$$

$$x_N(a,t) = S_N(b,t) = S_N(b,t-1) = v \text{ and b } \aleph \text{ a at } t \text{ - } 1 \tag{15}$$

Note that the subscript of c in $x_N(x,t)_c$ denotes the coordinates of the centroid of that particular vertex. Any angle mentioned in the form as mentioned in equations (13) & (18) is referring to the interior angle Figure (14). In figure (8), *R* turns over the green grid, thus creating *imaginary waypoints*. Note that *R* would have been in a head-on collision with *B* had it not *turned* or *waited*. Moreover, *B* maintains a straight course throughout the yellow grid. It seems as if *R* moves to its starboard but *B* does not. If T'_a is the trajectory of a, containing only the real vertices visited by a, then vessels *a* and *b* are in a *secondary edge conflict* denoted by *(a',b,v_1,v_2,t)*, if:

$$v_2^{t-1} {v_1}^t \subseteq T_b \tag{16}$$

$$v_1^{t-1} \overline{{v_2}^t} \subseteq T_a \text{ \& } v_1^t \not\subseteq T_a \tag{17}$$

if $x_N(b,t-2) \neq x_N(b,t-1)$, then:

$$\angle x_N(b,t-2)_c x_N(b,t-1)_c x_N(b,t)_c = 0° \tag{18}$$

Figure (13) also shows a similar *secondary edge conflict* when R takes a sharp turn. Figure (9) contains two figures showing an *edge conflict* between the imaginary waypoints of *R* and *B*. Vessels *a* & *b* are in a *secondary edge conflict* denoted by *(a',b',v_1,v_2,t)* if :

$$v_1^{t-1} \overline{v_2^t} \subseteq T_a \text{ \& } v_2^{t-1} \overline{v_1^t} \subseteq T_b \tag{19}$$

given that

$$k_1 = k_2 = 120° \tag{20}$$

except if at t - 1, $S_N(a,t-1) \aleph$ a & $S_N(b,t-1) \aleph$ b, Fig. 9 (left). or,

$$v_1^{t-1} \widetilde{v_2^t} \subseteq T_a \text{ \& } v_2^{t-1} \overline{v_1^t} \subseteq T_b \tag{21}$$

given that

$$k_1 = 120° \text{ \& } k_2 = 0° \tag{22}$$

where

$$k_1 = x_N(b,t-2)_c x_N(b,t-1)_c S_N(b,t-1)_c \tag{23}$$

$$k_2 = x_N(a,t-2)_c x_N(a,t-1)_c S_N(a,t-1)_c \tag{24}$$

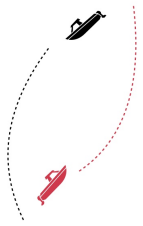

Figure 2: Head On Collision avoidance

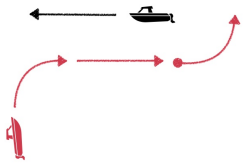

Figure 3: Crossing collision avoidance

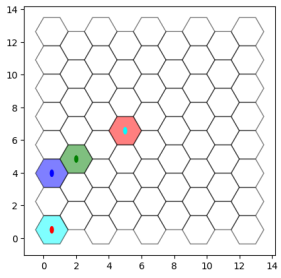

Figure 10: 4 vessels standing at their starting points with the grid which is of the same color as them being their respective destination.

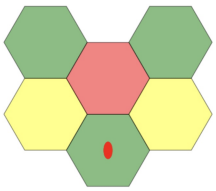

Figure 11: Colored grids for showing the kinds of turns the red vessel can take.

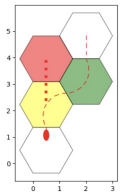

Figure 4: Imaginary path for a turning vehicle.

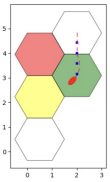

Figure 5: Imaginary path for a waiting vehicle.

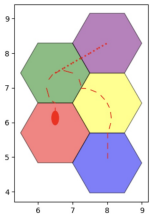

Figure 6: Vessel taking a sharp turn over the red grid.

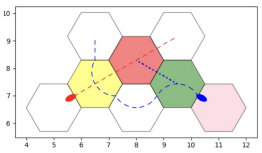

Figure 7: Violation of the Crossing Rule shown over the red grid.

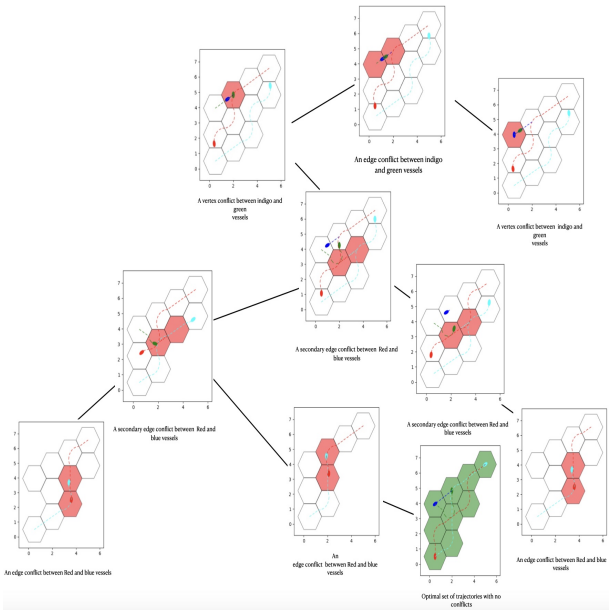

Figure 12: A COLREGs compliant Conflict-Based Search tree for finding COLREGs-compliant set of optimal trajectories for the problem presented in Fig. 10.

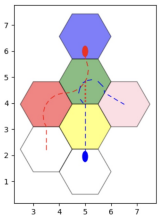

Figure 8: The imaginary trajectory of the red vessel creating an edge conflict with the actual trajectory of the blue vessel.

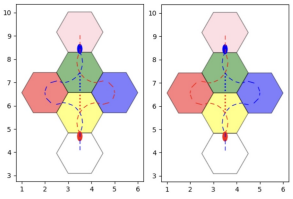

Figure 9: Violation of the Head On Rule shown in the right figure and the COLREGs-compliant version shown within the left figure.

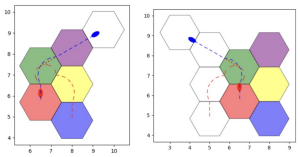

Figure 13: Secondary edge conflict when R turns sharply.

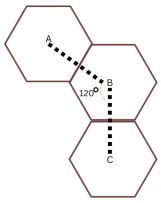

Figure 14: Vertices A,B and C forming an Interior Angle $\angle ABC = \angle CBA = 120°$

Colregs-Compliant Conflict-based search

Conflict-based search has two constraints at its disposal:

- *vertex constraint* - A *vertex constraint*, *(a,v,t)* restricts agent *a* from being at *vertex* or grid *v* at timestep *t*.
- *edge constraint* - An *edge constraint*, *(a,(v_1,v_2),t)* restricts agent *a* from moving between *vertex* v_1 to v_2 between timesteps *t* - 1 to *t*.

We introduce a few more constraints for better path efficiency:

- *turning constraint (t_c)* - Used to constrain the turns of a vessel. A *turning constraint* or t_c ,*(a,v_1,v_2,v_3,t)*, restricts agent a from ever making a transition from grid v_2 to v_3 given that it has moved from grid v_1 to v_2 between timesteps $t-1$ and t.
- *waiting constraint 1 (w_{c_1})* - The *waiting constraint 1* or w_{c_1} ,*(a,v_1,v_2,t)*, restricts agent *a* from moving between *vertex* v_1 and v_2 between timesteps *t* - 1 to *t* given that the agent had $x(a,t-2)=v_1$, that is the agent had been waiting v_1.
- *waiting constraint 2 (w_{c_2})* - The $\underset{\sim}{w_{c_2}}$,(a,t,v_1,v_2) restricts agent a from including $v_1^t \widetilde{v_2^{t+1}} v_2^{t'}$ within T_a, that is the agent is not allowed to wait at v_1 and move to v_2 given that it arrived at v_1 at timestep t.

Note that *secondary vertex conflicts* & *secondary edge conflicts* are nothing but mathematically defined violations of the *Crossing Situation* & *Head On Situation* respectively. The procedures for resolving *edge* and *vertex conflicts* remain the same as Conflict-Based Search within our strategy as shown within the algorithm. Let us assume that all the violations occur at timestep *t* within the shown figures. Figure (7) shows a *secondary vertex conflict*, that is, (R, B', v_{red}, t) where for some node N, that contains the set of trajectories containing the conflict, we have:

$$\overline{v_{red}^t} \subseteq T_b \;\&\; v_{red}^t \subseteq T_a \text{ and } R \aleph B \text{ at } t\text{ - }1 \quad (25)$$

$$x_N(R,t-1) = v_{yellow} \;\&\; x_N(B,t-1) = v_{green} \;\& \quad (26)$$

$$x_N(B,t-2) = v_{pink} \quad (27)$$

$$\angle x_N(B,t-2)_c x_N(B,t-1)_c x_N(B,t)_c = 120^\circ \quad (28)$$

Now, there are not two but three ways for preventing this conflict, either we constrain B which ensures that if,

$$x_N(B,t-2) = v_{pink} \;\&\; x_N(B,t-1) = v_{green} \text{ then,} \quad (29)$$

$$x'_N(B,t) \neq v_{red} \quad (30)$$

This is done using a *turning constraint* or $(B, x_N(B,t-2), x_N(B,t-1), x_N(B,t), t-1)$ on B. The other way is to simply constrain the edge movement of B to ensure -

$$x_N(B,t-2) = v_{pink} \implies x_N(B,t-1) \neq v_{green} \quad (31)$$

by applying the edge constraint $(B, (v_{pink}, v_{green}), t-1)$. Third way is to apply an *edge constraint*, $(R, (x_N(R,t-1), v_{red}), t)$ on R. Note that if we replace equation (31) by,

$$\widetilde{v_{red}^t} \subseteq T_b \;\&\; v_{red}^t \subseteq T_a \text{ and } R \aleph B \text{ at } t\text{ - }1 \quad (32)$$

Algorithm 1: COLREG-CBS Algorithm

Colregs-Compliant MAPF instance

Optimum Set of Non-conflicting paths

$Root.constraints \leftarrow 0$

$Root.solution \leftarrow$ Find individual paths using the low-level search

$Root.cost \leftarrow$ Sum of the costs of individual paths

Insert $Root$ into OPEN

while OPEN is not empty **do**

  $P \leftarrow$ Best node from OPEN (lowest solution cost)

  Search the paths in P for a conflict

  if No conflict found **then return** $P.solution$

  end if

  $Q \leftarrow$ First conflict in P at timestep t between agents a_i and a_j

  if $Q = (a_i, a_j, v, t)$ (Vertex Conflict) **then**

    Constraints $\leftarrow \{(a_i, v, t), (a_j, v, t)\}$

  else if $Q = (a_i, a_j, v_1, v_2, t)$ (Edge Conflict) **then**

    Constraints $\leftarrow \{(a_i, (v_1, v_2), t), (a_j, (v_2, v_1), t)\}$

  else if $Q = (a_i, a'_j, v_1, t)$ (Secondary Vertex Conflict) **then**

    if $x_P(a_j, t-1) \neq x_P(a_j, t)$ **then**

      Constraints $\leftarrow \{(a_j, x_P(a_j, t-2), x_P(a_j, t-1), x_P(a_j, t), t-1)\}$

    else

      Constraints $\leftarrow \{(a_j, x_P(a_j, t-2), x_P(a_j, t-1), S_P(a_j, t-1), t-1)\}$

    end if

    Add $(a_i, (x_P(a_i, t-1), v_1), t)$ to Constraints

  else if $Q = (a'_i, a_j, v_1, v_2, t)$ (Secondary Edge Conflict) **then**

    Constraints $\leftarrow \{(a_i, x_P(a_i, t-2), v_1, x_P(a_i, t), t-1)\}$

    if $x_P(a_j, t-2) = x_P(a_j, t-1) = v_2$ **then**

      Add (a_j, v_2, v_1, t) to Constraints

    else

      Add $(a_j, x_P(a_j, t-2), v_2, v_1, t-1)$ to Constraints

    end if

  else if $Q = (a'_i, a'_j, v_1, v_2, t)$ (Secondary Edge Conflict) **then**

    Constraints $\leftarrow \{(a_i, x_P(a_i, t-2), v_1, x_P(a_i, t), t-1)\}$

    Add $(a_j, x_P(a_j, t-2), v_2, x_P(a_j, t), t-1)$ to Constraints

  end if

  for each q in Constraints **do**

    $A \leftarrow$ A new node

    $A.constraints \leftarrow P.constraints + q$

    $A.solution \leftarrow P.solution$

    $a \leftarrow$ Agent constrained using q

    Use low-level to find path for a

    if Path Found **then**

      Update $A.solution$

      $A.cost \leftarrow$ Sum of path costs for all agents

    else

      Continue

    end if

  end for

end while=0

, that is, B $waits$ to let R cross v_{red} and then itself moves to v_{red}. Note that this is also a violation of the Colregs Crossing Rule as mentioned in equation (15) & (**??**). This conflict can be dealt in a manner exactly same as we dealt with the prior secondary vertex conflict. Note that unlike other conflicts, which require the creation of two high-level CBS nodes. A conflict whose resolution requires constraining a turn of one of the vessels would lead to the creation of three high-level CBS nodes.

Figure (8) shows a *secondary edge conflict* , that is,, $(R', B, v_{green}, v_{yellow}, t)$:

$$v_{yellow}^{t-1} {v_{green}}^{t} \subseteq T_B \tag{33}$$

$$v_{green}^{t-1} \overline{{v_{yellow}}^{t}} \subseteq T_R \;\&\; v_{green}^{t} \not\subseteq T_R \tag{34}$$

$$\angle x_N(B, t-2)_c x_N(B, t-1)_c x_N(B, t)_c = 0^\circ \tag{35}$$

A similar conflict is also shown in Figure (13) with the only difference being that over there, R makes a *sharp turn*. Since, the turn of R is what creates the conflict in the first place, we constrain this turn of R. The constraint on B depends on the fact whether $x_N(B, t-2) \neq x_N(B, t-1)$, that is, the condition in equation (18) holds or not. If the condition holds, then we will have to subject B to a *turning constraint* ,$(B, x_N(B, t-2), x_N(B, t-1), x_N(B, t), t-1)$, otherwise a *waiting constraint* , that is, $(B, x_N(B, t-1), x_N(B, t), t)$ is sufficient. In figure (9) (right side), we see another *secondary edge conflict* , that is, created by the imaginary trajectories of both the vessels with each actually, turning towards their respective port sides such that:

$$v_{yellow}^{t-1} \overline{v_{green}^{t}} \subseteq T_R \tag{36}$$

$$v_{green}^{t-1} \overline{v_{yellow}^{t}} \subseteq T_B \tag{37}$$

and both the vessels take a 60° turn which is a necessary condition according to equations (23) & (24). The turns of the vessels are constrained leading to the creation of 4 high-level nodes.

Experimental Results

Simulation Results

For simulations, we tested the algorithm over 100 problem instances for n number of agents where $n \in \{3, 5, 10, 15\}$ on a hexagonal mesh containing 99 grids. We used a single core of an Apple M1 processor as our computational resource. A time of 5 minutes was given for solving each instance. Exceeding the time limit for solving an instance implies that an algorithm failed to solve that instance. We introduce a quantity called optimality rate , O_r , where for a solvable instance m involving N agents -

$$O_r = \frac{\sum_{k=1}^{N} I_k}{S_m}, \tag{38}$$

where, I_k is the optimal path for the k^{th} agent returned by the low level solver when the k^{th} agent is under no constraint. S_m is the total path cost of the solution returned by the algorithm. It can be observed that closer the O_r is to 1, the better is the quality of the returned path. For each n, 100 simulations were carried out and the *success rate* is equal to the number of instances that were solved out of 100. The *mean time* for n is the mean of the time taken ,in seconds, to solve all the 100 problem instances. Fig. (15) & Table (1) display the results for the computer simulations. It can be observed that as n increases, the *success rate* decreases and *mean time* increases.

Table 1: Results for Computer Simulations

n	*Mean Time (s)*	*Success Rate*	*O_r Mean*	*O_r (Std. dev.)*
3	0.0432	1.00	0.9954	0.0135
5	0.0996	0.99	0.9909	0.0164
10	2.7074	0.99	0.9963	0.0056
15	8.9772	0.98	0.9965	0.0044

Physical Experimental Results

We evaluate COLREG-CBS through experimental demonstration on two ASVs. The experiment was carried out in a Lake.

ASVs Our in-house developed ASVs named "Beluga" (green-colored) and "Marlin" (red-colored) as shown in Figure 16 are identical single hull kayaks of length 2250mm made from plastic. The width of the vessels is 870 mm and the hull weight is 15kg with a payload capacity of 100kg. The vehicles are equipped with two T200 thrusters that function as a differential drive mechanism with a maximum combined thrust of 100 Newtons. They have a Pixhawk 2.4.8 autopilot with ArduRover firmware using a NEO 7M GPS module. We built an external frame using 4040 aluminum extrusions weighing about 8kg to mount the various components on the vessels. These components together add approximately 10kg to the payload. The maximum speed of each vessel is 5 knots. The resultant solution generated by the COLREG-CBS strategy is converted to GPS coordinates and communicated to respective ASV's Pixhawk autopilot from the ground station using 433 MHz radio telemetry.

Experiment . We considered an area of 40m $\times$ 40m in the lake. The area is divided into hexagonal mesh comprising 99 hexagons. The scenario of a head-on collision of two ASVs is considered in the experiment. The resultant solution generated by the COLREG-CBS strategy is shown in Figure 19. Initially, the vessels are located at their respective start location and move towards their goal location. The solution generated is converted to GPS waypoints and communicated to the vessels through the ground station. The vessels follow the set of GPS waypoints to resolve conflicts as shown in Figure 18. We have shown through simulations and real-world experiment that the COLREG-CBS technique can effectively produce conflict-free paths that satisfy COLREG standards for ASVs.

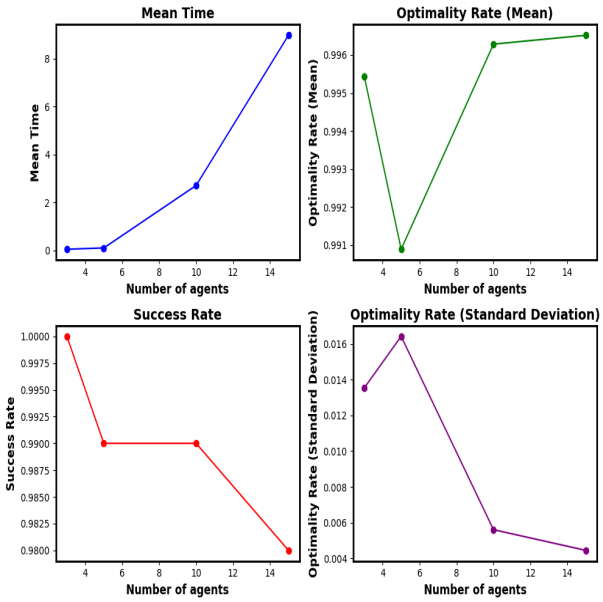

Figure 15: The effect of increasing the number of agents on Success Rate, Optimality Rate (Mean and Standard Deviation) and Mean Time (in seconds).

Figure 16: ASVs used for the experiment

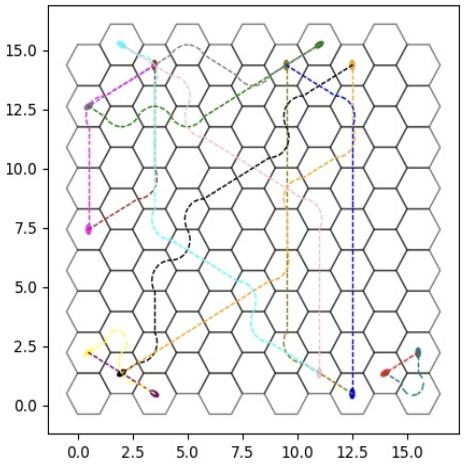

Figure 17: Trajectories generated by the Colregs-Compliant CBS algorithm for a 14 agent MAPF problem.

Figure 18: The actual GPS trajectories followed by the ASVs in the experiment.

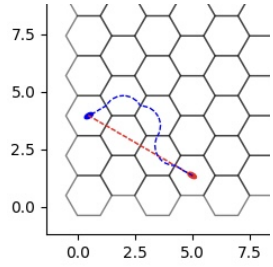

Figure 19: Computer simulation displaying the waypoints generated for a problem instance involving two agents.

Conclusions

This paper presents a CBS inspired path-planning strategy for ASVs that adhere to the COLREGs requirement for marine vehicles. We introduce the algorithms and demonstrate their application through both simulations and real-world experiments involving two ASVs. The results from these simulations and experiments confirm the effectiveness of the COLREG-CBS strategy in practical situations.

The current work can be extended by investigating the theoretical properties of the CBS strategy and analyzing the impact of communication delays on the solution process. Another extension could involve incorporating environmental factors such as currents and winds when calculating optimal paths. Additionally, the efficiency of the proposed approach should be compared with existing methods. The algorithm can be extended for decentralized implementation.

Acknowledgment

We thank the staff of Bhopal Lower Lake boating club for allowing us to use their facility for the experiments. We would also like to thank all the other students at MOONLAB, Indian Institute of Science Education and Research Bhopal for their help and guidance.

References

Benjamin, M. R. 2002. Multi-objective autonomous vehicle navigation in the presence of cooperative and adversarial moving contacts. In *OCEANS'02 MTS/IEEE*, volume 3, 1878–1885. IEEE.

Chiang, H.-T. L.; and Tapia, L. 2018. COLREG-RRT: An RRT-Based COLREGS-Compliant Motion Planner for Surface Vehicle Navigation. *IEEE Robotics and Automation Letters*, 3(3): 2024–2031.

Hu, L.; Hu, H.; Naeem, W.; and Wang, Z. 2022. A review on COLREGs-compliant navigation of autonomous surface vehicles: From traditional to learning-based approaches. *Journal of Automation and Intelligence*, 1(1): 100003.

Lyu, H.; and Yin, Y. 2019. COLREGS-Constrained Real-time Path Planning for Autonomous Ships Using Modified Artificial Potential Fields. *Journal of Navigation*, 72(3): 588–608.

Naeem, W.; Irwin, G. W.; and Yang, A. 2012. COLREGs-based collision avoidance strategies for unmanned surface vehicles. *Mechatronics*, 22(6): 669–678. Special Issue on Intelligent Mechatronics (LSMS2010 & ICSEE2010).

Shan, T.; Wang, W.; Englot, B.; Ratti, C.; and Rus, D. 2020. A Receding Horizon Multi-Objective Planner for Autonomous Surface Vehicles in Urban Waterways. In *2020 59th IEEE Conference on Decision and Control (CDC)*, 4085–4092.

Sharon, G.; Stern, R.; Felner, A.; and Sturtevant, N. R. 2015. Conflict-based search for optimal multi-agent pathfinding. *Artificial Intelligence*, 219: 40–66.

Vagale, A.; Oucheikh, R.; Bye, R.; Osen, O.; and Fossen, T. 2021. Path planning and collision avoidance for autonomous surface vehicles I: a review. *Journal of Marine Science and Technology*, 26.

Wang, N.; and Xu, H. 2020. Dynamics-Constrained Global-Local Hybrid Path Planning of an Autonomous Surface Vehicle. *IEEE Transactions on Vehicular Technology*, 69(7): 6928–6942.

Wei, G.; and Kuo, W. 2022. COLREGs-Compliant Multi-Ship Collision Avoidance Based on Multi-Agent Reinforcement Learning Technique. *Journal of Marine Science and Engineering*, 10(10).

Wen, N.; Long, Y.; Zhang, R.; Liu, G.; Wan, W.; and Jiao, D. 2023. COLREGs-Based Path Planning for USVs Using the Deep Reinforcement Learning Strategy. *Journal of Marine Science and Engineering*, 11(12).

Zhao, Y.-x.; Li, W.; Feng, S.; Ochieng, W. Y.; Schuster, W.; et al. 2014. An improved differential evolution algorithm for maritime collision avoidance route planning. In *Abstract and Applied Analysis*, volume 2014. Hindawi.