Optimización de la transferencia de información en el medio marítimo

Javier Alegre Revuelta

Resumen– En el medio marítimo existen elementos e infraestructuras para la captación de información. El objeto de este proyecto es mejorar la propagación y transferencia de esta información a través de los elementos dinámicos del mar, como los barcos, y estáticos, como las boyas marítimas. Estos utilizan los protocolos oportunísticos para establecer conexión con los elementos captadores de información. Durante la realización del trabajo se modificará el comportamiento de los navíos para aprovechar los protocolos de red utilizados y mejorar la comunicación en el océano. Para conseguir el objetivo se aplicarán técnicas de aprendizaje computacional, en concreto algoritmos de aprendizaje por refuerzo y técnicas heurísticas. El proyecto también abarca como dar solución a un problema a pesar de no disponer de suficientes datos. La obtención de datos para el aprendizaje se realizará a través de un entorno de simulación que se desarrollará en el transcurso de proyecto.

Paraules clau– Aprendizaje por refuerzo, heurística, protocolos oportunísticos, aprendizaje computacional, simulador, Q-learning, datos

Abstract– In the maritime environment there are elements and infrastructures for capturing information. The purpose of this project is to improve the propagation and transfer of this information through the dynamic elements of the sea, such as ships, and static elements, such as maritime buoys. These use opportunistic protocols to establish a connection with the information capturing elements. During the work, the behavior of the ships will be modified to take advantage of the network protocols used and improve communication in the ocean. To achieve the objective, computational learning techniques will be used, specifically reinforcement learning algorithms and heuristic techniques. The project also covers how to solve a problem despite not having enough data. Obtaining data for learning will be done through a simulation environment that will be developed during the course of the project.

Keywords– Reinforcement learning, heurístics, opportunistic protocols, machine learning, simulator, Q-learning, data

1 Contexto del trabajo

Actualmente en el océano existen diferentes tipos de boyas con diferentes finalidades. Algunos de estos tipos recopilan información sobre el entorno en el que están localizadas [1]. Los barcos se nutren de esta información, pero existe un problema con la propagación de la información que almacenan las boyas, y esto es debido a la gran complejidad que existe de crear una infraestructura de red plenamente operativa en mar abierto. Las casuísticas donde agentes móviles se comunican en un entorno sin una infraestructura de red definida son comúnmente solventadas por protocolos oportunísticos [2].

Los principales actores que intervienen en la propagación de la información son los barcos. Al aproximarse a un boya se ejecutan una serie de pasos que hace posible establecer una comunicación ad-hoc entre el barco y la boya. Esta conexión es temporal y al aumentar la distancia entre ambos sujetos se perderá la comunicación. Es deseable que el número de veces que una boya establece conexión con algún barco sea elevado. De esta forma se consigue una mayor circulación de la información.

La motivación del proyecto es aumentar el número de boyas por las que se cruza un barco sin que esto suponga un gran desvío en su recorrido. Para encontrar el equilibrio entre ambos objetivos se estudiarán técnicas heurísticas y de inteligencia artificial [3].

En las siguientes secciones se detallarán los objetivos, así

• E-mail de contacto: javier.alegre@autonoma.cat
• Mención realizada: Tecnologías de la Información
• Trabajo tutorizado por: Ana Oropesa Física (DEIC)
• Curso 2021/22

como la planificación y metodología empleada en el proyecto. Posteriormente, se expondrá el desarrollo de la ejecución. Finalmente, se evaluarán los resultado y se extraerán las conclusiones derivadas de la finalización de este proyecto.

2 Objetivos

El objetivo general de este proyecto es, dada la situación de las redes oportunísticas dentro de los viajes de navíos en mar abierto, optimizar la ruta de estos para aumentar la circulación y la transferencia de datos del mayor número de boyas posibles sin desviar su ruta por completo del puerto objetivo. A su vez, este proyecto tiene los siguientes objetivos específicos:

1. Listar los tipos de redes oportunisticas mediante una investigación en medios publicados y analizar posibles casos similares a los viajes de barcos en mar abierto para recoger posibles similitudes en estrategias de resolución. Este objetivo debe realizarse durante las primeras semanas desde el inicio del proyecto.

2. Proponer diferentes técnicas heurísticas y algoritmos de aprendizaje para la resolución del problema y elegir bajo justificación el método empleado. Este objetivo debe de finalizar antes de comenzar la Fase 2.

3. Elegir el lenguaje de programación y el entorno de trabajo en función de la facilidad de uso y la experiencia previa del desarrollador. Este objetivo debe realizarse después de finalizar los dos anteriores y antes de la Fase 2 del proyecto.

4. En la fase de desarrollo se definirán cada una de las etapas de la implementación del proyecto, así como las diferentes estrategias y consideraciones que se adopten durante esta fase. Este objetivo tiene una duración de dos meses.

5. Una vez recogidos los resultados de todas las estrategias implementadas, analizar los parámetros de éxito sobre el problema y escoger mediante unas conclusiones fundamentadas la mejor estrategia para resolver el problema planteado. Este objetivo debe finalizar antes de junio de 2022.

3 Metodología

La metodología empleada en este proyecto sigue la filosofía en cascada [4]. Esta metodología consiste en la división del trabajo en fases secuenciales. Un fase no podrá ser empezada hasta que la anterior no haya sido completada. Ha sido seleccionada esta estrategia ya que encaja bien con la naturaleza del proyecto, de una duración breve y una clara definición de los objetivos del proyecto desde el inicio.

Para el control y seguimiento del proyecto se utilizan dos herramientas, un diagrama de Gantt [5] y un tablero de Trello [6]. El diagrama de Gantt es una herramienta que muestra, de manera muy visual, el avance y la planificación del proyecto, así como los hitos entregables que hay en cada momento. Esta herramienta se combina con Trello ya que esta aplicación permite un definición y un desglose de las tareas mayor y permite ver el avance del proyecto con una mayor granularidad.

4 Planificación

Este proyecto ha sido dividido en 3 fases. Estas fases están plasmadas sobre el diagrama de Gantt que se encuentra en el Anexo A.1. Este diagrama muestra también las tareas implicadas en cada fase y el intervalo de tiempo donde se llevarán a cabo. Hay tareas que contienen días marcados con una cruz blanca para indicar que son periodos de tiempo extra para evitar un desvío del proyecto en caso de aparición de imprevistos. Las marcas rojas de este diagrama muestran las diferentes fechas donde se debe liberar un entregable.

Como se explico en el apartado anterior, el control de la ejecución de todas estas tareas se realizará mediante un Trello, el cual se puede observar en la figura 1.

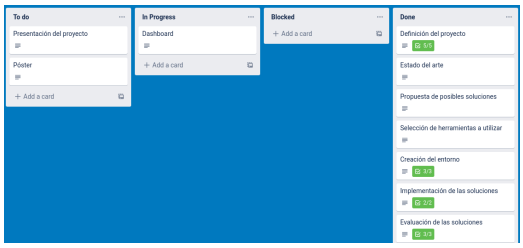


Fig. 1: Tablero de Trello

Hay 4 columnas en este tablero. La columna *To Do* es donde están las tareas pendientes de realizarse. La columna *In Progress* muestra aquellas tareas que se están llevando acabo en ese momento. El apartado *Blocked* solo se utilizará en caso de que una tarea no pueda ser completada porque existe una dependencia interna o externa que no permite avanzar. La columna *Done* se utiliza cuando una tarea ya ha sido finalizada y validada.

Cada una de las tareas contiene un descripción, unas fechas de entrega y, en algunos casos, una *checklist* con las subtareas que hay que realizar.

5 Estado del arte

Este problema puede ser dividido en dos fases diferentes. Primeramente, existe una fase de captación de información del medio acuático. Esta resuelve la problemática de recopilar información del medio a través de sensores submarinos y almacenarla en elementos que se encuentran en la superficie del agua, como son la boyas. Este tipo de redes se las conoce como *Underwater Sensor Network* (UWSN) [7].

La siguiente fase consiste transferir está información a los centros de monitorización, que pueden ser estaciones terrestres o barcos. En este proyecto se considerará que la información quiere ser transferida a navíos. Al tratarse de agentes móviles sobre un entorno sin infraestructura de red es complejo utilizar los protocolos convencionales de comunicación. Para esto se utilizan las redes *Sea Ad Hoc Network* (SANET) [8], que serán explicadas más adelante.

En la investigación del estado del arte no se han encontrado evidencia de que se estén empleando estas dos técnicas conjuntamente. Este proyecto consistirá en la aplicación de la combinación de ambas técnicas para resolver el problema

de comunicación que existe entre las estaciones de recogida de información del medio acuático y los navíos.

5.1. Underwater Sensor Network

Estas redes están compuestas de sensores submarinos que captan la información de su entorno. Estos sensores están situados a diferentes profundidades y se comunican entre si a través de transmisiones acústicas. Esto es así ya que las ondas acústicas tienen una longitud de onda larga y permite la comunicación a quilómetros. La información de los sensores es transmitida a nodos flotantes, como podrían ser las boyas, y estos a su vez la transmiten a centro de monitorización.

En la Figura 2 se muestra un esquema del funcionamiento de este tipo de redes [9]. En el Anexo A.2. se encuentra esta imagen ampliada. Existen cuatro arquitecturas básicas en las UWSN.

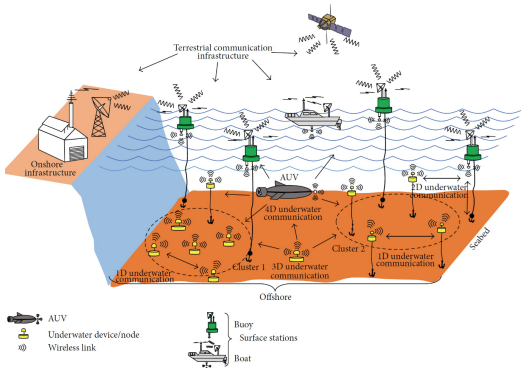


Fig. 2: Esquema de la infraestructura UWSN

La arquitectura 1D consiste en nodos autónomos que están sumergidos dentro del agua y captan información. Estos sensores son capaces de flotar a la superficie para transmitir esa información a los centros de monitorización.

La arquitectura 2D la conforman un grupo de nodos. En este grupo hay un sensor que se denomina nodo de anclaje que recibe la información del resto de nodos del grupo y, posteriormente, transmite esa información al nodo flotante.

En la arquitectura 3D hay sensores a diferentes profundidades. Hay 3 pasos en la comunicación. El primer paso es la comunicación entre sensores de diferentes grupos que se encuentran en diferentes profundidades. El segundo paso es la comunicación con el nodo de anclaje. El último paso es la comunicación del nodo de anclaje con el nodo flotante.

La arquitectura 4 se compone de la arquitectura 3D y las UWSN móviles. Estas UWSN móviles són vehiculos submarinos que recopilan información de los nodos ancla y de los sensores, si el vehiculo se encuentra suficientemente próximo de ellos. Después de captar la información estos UWSN móviles transmiten la información a las estaciones de monitorización.

Estas redes son empleadas para diferentes finalidades. Uno de sus usos es la navegación asistida, la cuál permite a guiar a las embarcaciones. También tiene usos de monitorización de diferentes aspectos marítimos, como la calidad del agua, la vida marina o explorar la topología, entre otros. Las UWSN son utilizadas para la detección de desastres como terremotos, volcanes o tsunamis. En el aspecto militar también han sido utilizados para la detección de minas submarinas ocultas. Otros ámbito donde se pueden aplicar estas redes son los deportes, donde se utilizan para saber el desempeño de un nadador o la ubicación de este.

5.2. Sea Ad Hoc Network

La redes SANET son una variante de las redes *Mobile Ad Hoc Network* (MANET) [10]. Las SANET tienen el mismo objetivo que las redes MANET pero debido al entorno donde están situadas, la velocidad, el consumo de energía y otros factores hacen que las implementaciones sean diferentes.

Las MANET son redes descentralizadas que no dependen de una infraestructura preexistente. Esta conformada por dispositivos móviles conectados de forma inalámbrica. Cada nodo puede desplazarse sin restricciones, por lo que camabiará constantemente las conexiones con el resto de nodos. Debido a esto, cada nodo debe actuar como enrutador para poder hacer llegar la información que quieren enviar el resto de nodos a su destino.

La diferencia que existe en las redes SANET es que los nodos que componen la red se encuentran en el mar, no en tierra. Esto supone que los protocolos de enrutamiento no sean exactamente los mismos que en las MANET.

Dependiendo de las características de la red se utilizarán diferentes tipos de protocolos. En una red SANET con pocos nodos es deseable utilizar un protocolo proactivo de enrutamiento. Este protocolo distribuye periódicamente en toda la red tablas de enrutamiento con listas actualizadas de destinos y sus rutas. No obstante, si la red posee muchos nodos utilizar un protocolo reactivo mejoraría el ancho de banda.Con este protocolo cada vez que un nodo quiere enviar un mensaje, este inunda la red con solicitudes para que se le devuelva una ruta hasta el destino deseado. Para grandes redes SANET sería más idóneo utilizar redes híbridas que combinan los protocolos proactivos y reactivos. Este protocolo híbrido hace una división en zonas de tal forma que las zonas internas utilizan protocolos proactivos y las zonas externas reactivos.

6 Estrategia para la solución

El enfoque de este proyecto es hacer que las rutas que toman los navíos se aproximen a los nodos flotantes para que se establezca conexión y haya un traspaso de información entre estos. La solución consiste en tomar acción sobre los barcos, y no en la configuración de los protocolos de red.

Para la tarea que se va a desempeñar se aplicarán técnicas de inteligencia artificial. Es posible abordar este problema con técnicas heurística [11] o con técnicas de aprendizaje computacional [12]. Tras definir en secciones posteriores las diferentes técnicas, así como sus ventajas e inconvenientes, se implementarán varias soluciones en base a estas.

6.1. Heurísticas

La heurística es un proceso humano de las personas, en la cuál se utilizan estrategias con el fin de llegar a la solución de un problema de manera más rápida [13]. El ser humano utiliza estas técnicas para mejorar la velocidad de respuesta,

para esforzarse menos en pensar la solución o para reducir la complejidad de un problema.

En la computación se utilizan este tipo de técnicas de forma similar. Se definen 'normas generales' para un problemas concreto. Esto lo que permite es llegar a una solución aproximada lo suficientemente correcta en un periodo más corto de tiempo que si se hubiera resuelto con otras técnicas. Las otras técnicas no heurísticas pueden resolver el problema de forma más precisa y exacta, pero generalmente son significativamente más lentas [11].

Las ventajas de estas técnicas son la rápida resolución de problemas complejos y su desventaja es la precisión de la respuesta. Es por eso que hay una serie de criterios ha reflexionar antes de decidir si es adecuado aplicar heurísticas. Las criterios son:

- Optimización: Si existen múltiples soluciones es necesario definir si se requiere obtener la solución óptima al problema o con una solución considerada correcta es suficiente.
- Completitud: En problemas con múltiples soluciones es importante definir si se va a querer obtener todas las soluciones posibles o sirve con un subconjunto de estas.
- Precisión: Algunos problemas no poseen una respuesta correcta, en dichos casos las heurísticas pueden mejorar la precisión de las soluciones.
- Tiempo de ejecución: En ciertos casos es posible que la solución encontrada sin métodos heurísticos no suponga un aumento significativo del tiempo de resolución frente a las técnicas heurísticas.

Un ejemplo de heurística es el algoritmo $A*$ [14]. Esta técnica se puede aplicar para mejorar el rendimiento del algoritmo de Dijkstra, el cuál, dado un grafo ponderado encuentra el camino más corto entre un nodo origen y un nodo destino. Sin aplicar el $A*$ lo que ocurriría es que se le daría la misma importancia a comprobar un nodo que se encuentra lejos del destino a uno que se encuentra muy cerca.

En la Figura 3 se puede apreciar un ejemplo en el que cada arista tiene un peso y además en cada nodo se le ha añadido, en naranja, un valor heurístico. Tomando como valor origen el nodo a y como nodo destino z, el valor heurístico de cada nodo disminuye cuando está próximo al destino y aumenta al encontrarse lejano.

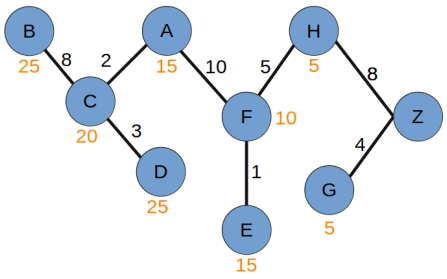


Fig. 3: Grafo ponderado

6.2. Aprendizaje computacional

El aprendizaje computacional es una rama de la inteligencia artificial que pretende imitar la forma en la que aprenden los humanos, a través de los datos y algoritmos. Generalmente, los algoritmos se clasifican en dos tipos: aprendizaje supervisado y no supervisado.

El aprendizaje supervisado consiste en, a partir de unos datos etiquetados entrenar un modelo para que cuando se obtengan datos nuevos que no estén etiquetados este modelo sea capaz de predecir su valor.

El aprendizaje no supervisado consiste en la creación de etiquetas automáticamente partiendo de un conjunto de datos no etiquetados. Estos algoritmos se basan en las características de los datos hacer agrupaciones, por ejemplo. Estas dos técnicas pueden ser utilizadas complementariamente.

Existe un subcampo dentro del aprendizaje computacional, que es el aprendizaje por refuerzo [15]. Este no sigue ninguna de las dos filosofías anteriores, pero si es cierto que esta más próximo al aprendizaje supervisado. En la aplicación de estos algoritmos el sistema inicialmente sólo conoce que acciones puede tomar. Al tomar una acción se le devuelve una recompensa o un castigo y se le pasa información del estado actual del entorno. Con esta información el sistema va aprendiendo para optimizar la toma de decisiones maximizando las recompensas y minimizando las penalizaciones. En estos sistemas se utiliza la aleatoriedad para que el algoritmo a medida que va iterando también explore nuevas combinaciones.

Un ejemplo de aprendizaje por refuerzo es un sistema que aprenda a jugar al juego arcade *Pong* [16]. Éste entorno, que simula un partida de ping pong, permite al jugador realizar dos acciones, desplazarse hacia arriba o hacia abajo. En cada momento el sistema recibe información sobre la posición exacta donde se encuentra la pelota. Si el sistema no consigue devolver la pelota éste recibe una penalización. Si por el contrario, es el rival quien no consigue devolverla recibe una recompensa. Con toda esta información el sistema irá aprendiendo para tomar la acción correcta según la posición en la que se encuentre la pelota para minimizar la penalización y maximizar las recompensas.

6.3. Herramientas de desarrollo

La estrategia de resolución de este proyecto es el aprendizaje por refuerzo. El resultado que se desea obtener debe ser óptimo. Se prioriza la optimización antes que el tiempo de ejecución. No obstante, durante la fase de implementación se aplicarán soluciones con pequeñas variaciones en las cuales se pueden aplicar técnicas heurísticas.

El lenguaje utilizado será Python [17]. Esta herramienta es comúnmente utilizada para la programación de algoritmos de inteligencia artificial. Esto supone que existe un gran soporte en la comunidad de internet para hacer frente a las dificultados que pueden surgir. Otro factor es que el desarrollador de este proyecto tiene facilidad de uso con este lenguaje de programación y le permite desenvolverse cómodamente.

7 Implementación

La implementación está dividida en dos fases diferenciadas. En una primera fase se definirá y construirá un entorno que simule barcos viajando de un punto a otro. Este simulador estará preparado para poder interactuar con él de forma

que dada una entrada que representa una acción, este devuelva una respuesta que el algoritmo que se implemente posteriormente pueda procesar correctamente.

La segunda fase de este proyecto consiste en aplicar uno de los algoritmos de aprendizaje por refuerzo que existen. Se estudiará cuaĺ es el más adecuado debido a la naturaleza de los datos. En esta fase se crearán diversas variantes de la solución, aplicando heurísticas similares a la explicada anteriormente, para comprobar posteriormente cuál de estas es más eficiente.

7.1. Entorno de simulación

La simulación de las rutas marítimas se ha centrado en viajes en el océano Atlántico. El objetivo del simulador es que sea un módulo robusto suficientemente parametrizable para que al momento de implementar la fase 2 de aprendizaje no haya que modificar el código y actúe como caja negra.

En la creación de este se ha utilizando una imagen de un mapamundi y se ha ampliado la imagen hasta solo visualizar la zona deseada. Después de obtener esa región, se a ha bajado la resolución de la imagen para trabajar sobre una matriz de píxeles más reducida. Cada píxel ha sido categorizado para representar un elemento dentro de la simulación. En la figura 4 se puede visualizar una imagen del simulador.

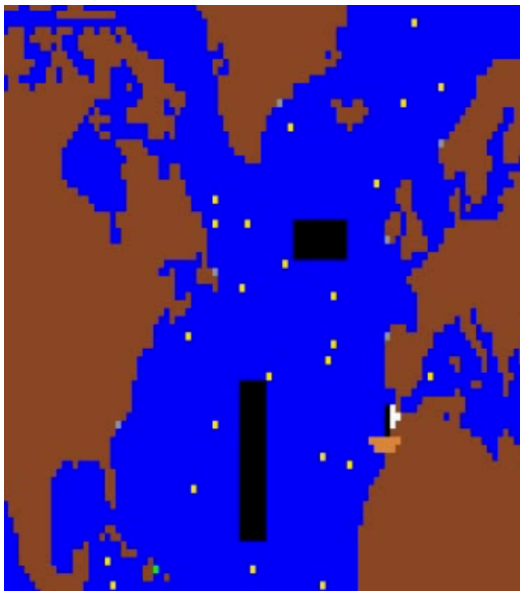


Fig. 4: Entorno de simulación

El simulador se compone de los siguientes elementos:

- Tierra: Este elemento representa la zona terrestre del mapa donde un navío jamás podrá situarse.
- Agua: Son las posiciones sobre las que el barco puede navegar.
- Barco: Este elemento, visualmente, tiene una área más amplia que el resto de elementos, pero lógicamente representa una única posición sobre la matriz.
- Boyas: Son los elementos amarillos. El objetivo del proyecto es maximizar el número de boyas por las que pasa durante su recorrido. Las boyas han sido fijadas por el desarrollador previamente.
- Puertos: Estas son las posiciones de origen o destino del barco. Se representan en color gris, exceptuando el puerto destino que es de color verde. Los puertos han sido fijados por el desarrollador previamente, pero el puerto origen y destino se escogen aleatoriamente.
- Ruta: Al desplazarse el barco, se marcará en rojo la casilla donde estaba posicionado anteriormente. Esto se utiliza para ver visualmente la ruta que toma el navío.
- Tormenta: Este elemento busca representan zonas donde los barcos pueden tener dificultades para avanzar debido a factores climatológico. El objetivo es que los barcos aprendan a desviar la ruta y alejarse de estas tormentas.

Internamente el entorno funciona en base a dos matrices. Un primera matriz bidimensional que tiene las casillas codificadas según el elemento al que pertenenzca. Esta matriz inicialmente ha sido el resultado de binarizar, ampliar y bajar la resolución a una imagen externa, con esto se ha obtenido los elementos de tierra y agua. El resto de elemento han sido introducidos manualmente. La segunda matriz es 3D para poder obtener colores RGB al visualizarla, se realiza un mapeo entre la matriz 2D y la 3D para que cada elemento sea representado por un color diferente.

El entorno posee una opción de visualización la cuál permite que al terminar la ejecución se genere un vídeo. Este vídeo se consigue agrupando secuancialmente las matrices 3D de cada momentos como si fuesen los frames de un vídeo. Por defecto, este video no se generará. En caso de llamar a la función generadora del video se grabarán todas las iteraciones realizadas. Esto será parametrizable de forma que se podrá indicar que solo grabe la ejecución de cada cierto número de iteraciones.

El sistema acción y recompensas iniciales del simulador funciona de la siguiente manera:

- Si se desea mover a un ubicación de tierra o no accesible, el entorno notificará con un código de error y no realizará nada. La ejecución no se detendrá, permanecerá a la espera de una nueva acción.
- Si el barco se mueve sobre las casillas de agua se obtendrá un castigo de -1 punto. De esta forma se pretende minimizar los movimiento para llegar al destino.
- Si el barco cruza con una boya se otorgará un premio de 10 puntos. Una boya solamente dará puntos una vez por viaje (ejecución). En caso de volver a pasar por una boya en la misma simulación se aplicará una penalización de -1,5.
- Al llegar el barco al puerto destino, este obtendrá un premio de 100 puntos. Un puerto cualquiera que no sea el de destino no obtendrá recompensa. Al llegar a esta posición la ejecución termina con éxito.
- Los movimiento sobre una zona de tormenta otorgan una penalización de -2 puntos.

7.2. Implementación del aprendizaje

Para dotar al sistema de inteligencia se implementará el algoritmo de aprendizaje por refuerzo, *Q-learning* [18]. Es-

te método contiene dos requisitos para poder llevarse a cabo. El primer requisito es que las posibles acciones a realizar sean discretas, en este caso se cumple ya que el el rango de movimientos está limitado en cuatro direcciones (arriba, abajo, derecha, izquierda). El segundo requisito es que la información que proporciona el entorno en cada momento sea discreta también, este requisito también se cumple porque el simulador devuelve la posición del barco en la matriz de dos dimensiones donde se desarrolla la acción.

Este algoritmo consiste en crear una matriz, denominada Q, donde estén representados todos los estados posibles y por cada estado haya otra dimensió que represente las posibles acciones que se pueden tomar. En este proyecto la matriz Q será de tres dimensiones y de tamaño 100x100x4. Las dos primeras dimensiones representan las posibles posiciones donde puede estar el barco. La tercera dimensión, de tamaño cuatro, representa cada una de las posibles acciones que se pueden tomar. El objetivo de esta matriz es que, una vez entrenado el modelo, indique que acción es la más adecuada dependiendo del estado.

Esta matriz tienen que ir actualizando los valores en función de la siguiente formula:

$$Q^{new}(s_t, a_t) = (1 - \alpha) * Q(s_t, a_t) + \\ \alpha * (r + \gamma * max_{a_{t+1}} Q(s_{t+1}, a_{t+1}))$$

Esta fórmula expresa que el nuevo valor para un estado determinado y un acción en concreto, $Q^{new}(s_t, a_t)$, será una ponderación entre el valor que posee actualmente y el nuevo valor determinado por la recompensa obtenida (r). Esta ponderación se produce con la constante α de la fórmula, que comprende valores entre 0 y 1. A mayor α más importancia obtendrán los nuevos valores respecto al valor que ya existía. Estos nuevos valores se calculan como, la recompensa obtenida más el máximo valor de la matrix Q en el siguiente estado, $max_{a_{t+1}} Q(s_{t+1}, a_{t+1}))$. Este valor de Q en el siguiente estado, se multiplica por un factor de descuento γ que posee valores entre 0 y 1. Este factor de descuento expresa la importancia que se la da a los futuros movimiento para calcular el valor del estado actual.

Al inicio del entrenamiento el modelo no tiene ningún tipo de información sobre cuál es la mejor acción a tomar. Es por eso que se inicia tomando decisiones aleatorias para hacer una explotación sobre el entorno y descubrir que respuesta se recibe en determinados estados y tomando determinadas acciones. Para poder reducir gradualmente el porcentaje de acciones aleatorias que se toman en cada iteración se establece un coeficiente ϵ, el cual se inicializa en uno para tomar todas las acciones de manera aleatoria, hasta llegar a un mínimo establecido en el cuál la mayoría de las acciones se toman en base a la información recopilada, por lo que el porcentaje de acciones aleatorias es muy pequeño. Es recomendable mantener un mínimo, de esta forma el sistema mantiene la exploración y es capaz de ir aprendiendo.

El entorno de simulación proporciona la información de la que se dispondría en un viaje en barco en la vida real, la posición del barco, la recompensa, las coordenadas del puerto destino y la propia localización. Con esta información se han generado tantas matrices Q como puertos diferentes existen. Esto es debido a que dependiendo del puerto de destino al que se desea llegar una misma acción puede tener repercusiones opuestas. Dependiendo de cuál sea el destino se aplicará una matriz u otra.

Las matrices han sido inicializadas a menos infinito. En este proyecto no es posible inicializar la matriz a cero, debido a que existen recompensas negativas en el simulador y esto hace que el sistema detecte como mejor opción moverse hacia un posición no explorada que a una posición negativa. Un problema asociado a esto es que el barco siempre desearía explorar zonas de tierra, debido a que son zonas que el barco no puede acceder y por lo tanto no pueden ser actualizadas.

Para introducir nuevas versiones e intentar mejorar el algoritmo se han introducido dos variantes aplicando heurísticas similares a la explicada en apartados anteriores. Se ha evaluado limitar el movimiento del navío para que no hacer movimientos en sentido contrario al destino, pero después de evaluar los posibles problemas, se ha concluido que esa solución puede llevar a casuísticas que hagan que el barco no pueda avanzar. Es por eso que las heurísticas afectan al modo en el que el sistema procesa las recompensas, ya que puede darse la situación que la mejor opción para llegar el destino sea moverse en dirección opuesta al destino.

7.3. Heurística 1: No retroceso

Esta primera versión del algoritmo ha consistido en modificar las recompensas obtenidas, para que barco obtenga una penalización doble si trata de dar un paso en el sentido opuesto a la dirección donde se encuentre al puerto destino. Todos los cálculos se han realizado sin modificar el simulador, ya que el objetivo es que el entorno sea un módulo independiente a la implementación del aprendizaje.

En la figura 5 se puede ver dibujado sobre el mapa dos líneas verdes, una que une el barco con el destino y otra perpendicular que divide el mapa en dos partes.

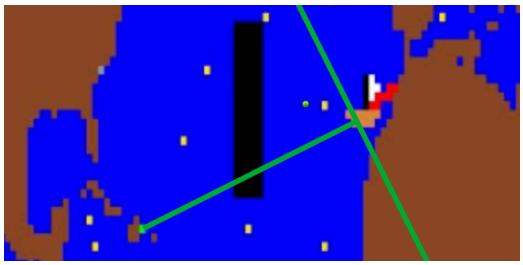


Fig. 5: Heurística 1: No retroceso

Para implementar esta heurística primero se encuentra el vector entre la posición actual del barco y el puerto destino. Una vez encontrado dicho vector, se procede a buscar la recta perpendicular que pasa por las coordenadas en las que se encuentra el barco. En función de esta perpendicular se obtiene la recompensa en base al movimiento que se ha realizado. Si la acción implica un movimiento sobre la perpendicular o en el plano opuesto al que se encuentra el destino, la recompensa es duplicada. En caso opuesto, la recompensa no es modificada y mantiene su valor original.

Estos cálculos son realizados en cada movimiento, lo que implica que la línea que divide los dos planos es modificada constantemente y a un movimiento se le puede aplicar esta penalización o no dependiendo de esto.

Esta penalización solo se ha realizado en el caso de recompensas negativas. Esta decisión se ha tomado para incentivar la búsqueda de boyas a pesar de que se encuentren, ligeramente, en otra dirección.

7.4. Heurística 2: Dirección de destino

Esta segunda heurística tiene como base el mismo principio que la anterior, avanzar en una dirección opuesta al destino conlleva mayor penalización. Igual que en la versión anterior, esto solo afecta a las recompensas negativas. La diferencia reside en que en este se calcula el ángulo entre el vector que va del barco al puerto destino y los vectores ortogonales de movimiento. Un vez obtenidos los diferentes ángulos se aplica la siguiente fórmula, con la cuál aumentará la penalización en base al valor de estos.

$$r = r * (1 + |\beta|/180)$$

La variable *r*, representa la recompensa obtenida y β es el ángulo. Lo que se obtiene con esta fórmula es que la recompensa aumente en un factor entre 1 y 2, proporcionalmente al tamaño en valor absoluto del ángulo. Lo que conlleva que, avanzar en la dirección correcta implica obtener la recompensa original o un pequeño incremento, y avanzar en dirección opuesta puede llegar ha duplicar la penalización original

En la figura 6 se puede apreciar el cálculo de los ángulos, anteriormente definidos, para cada movimiento.

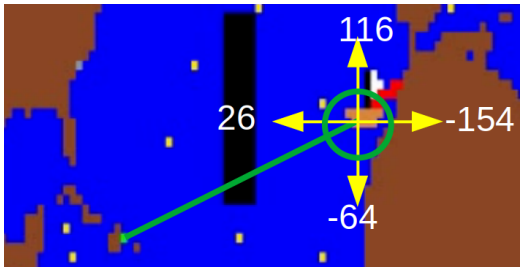


Fig. 6: Heurística de no retroceso

En la imagen se puede observar una línea verde que va desde el barco hasta el destino y con flechas amarillas están marcados cada uno de los diferentes movimientos que puede hacer el barco, así como el ángulo que conforman respecto a la línea verde. Teniendo en cuenta la fórmula anterior, un desplazamiento a la izquierda comportaría una baja penalización, mientras que un desplazamiento a la derecha sería elevada.

Un condición para que la fórmula funcione correctamente es tratar los ángulos mayores de 180º como ángulos negativos. El no considerar esto podría afectar al resultado de tal forma que, el movimiento hacia abajo sería una peor opción que el movimiento hacia arriba ya que el ángulo de -64º se interpretaría como un ángulo de 296º

7.5. Ajuste de parámetros

El algoritmo cuenta con una serie de parámetros que han sido introducidos en apartados anteriores y que se deben ajustar a la características del proyecto.

Para entrenar el modelo se deben establecer cuantas simulaciones se van hacer por cada versión y cuál es el número máximo de iteraciones que se pueden realizar por cada simulación. Cada destino se ha entrenado independientemente con 150 mil simulaciones y 400 iteraciones máximas, para cada uno. El número de simulaciones está distribuido equitativamente para que todas las posibles rutas para un destino determinado tengan el mismo número de ejecuciones. Con este valor de simulaciones se consiguen buenos resultados y no implica un tiempo de ejecución excesivo. El número máximo de iteraciones se alcanzará cuando el modelo no esté lo suficientemente entrenado y por lo tanto no consiga llegar al destino con éxito.

De tal forma que al inicio el modelo explorará el entorno en gran medida, cuando la aleatoriedad es alta. Este número de iteraciones tampoco genera un ralentización excesiva del algoritmo, y en fases finales el número de movimientos del barco suele ser menor a este valor, ya que las condiciones para dar como finalizada una simulación son llegar al máximo de iteraciones o llegar al destino con éxito.

El parámetro de aprendizaje α pondera la fórmula de actualización de la matriz. Para este proyecto se ha establecido en 0.1. Esto implica que tendrá más peso el valor actual, que no lo valores nuevos. Es conveniente que el parámetro sea bajo cuando el entorno es estático, en caso de que hubieran elementos dinámicos habría que aumentar el valor de este para que el modelo se adaptase más rápidamente al entorno cambiante.

La variable ϵ se utiliza para indicar el porcentaje de aleatoriedad de los movimientos de cada simulación. Esta variable se inicializa a 1 y se va decrementando en cada simulación hasta llegar a un mínimo, definido en 0,1. Las condiciones para actualizar épsilon son que se hayan realizado un 1,5 % de las simulaciones totales, y que la puntuación obtenida en esta simulación sea mayor a la de la anterior. Con esto se consigue asegurar un porcentaje mínimo de simulaciones en las que la toma de acciones es totalmente aleatoria y que épsilon se actualice después de realizar una simulación que es más acertada que las anteriores, y por lo tanto, se esté produciendo una mejora en el modelo. Este porcentaje también ha sido definido probando diferentes valores. La fórmula de actualización de épsilon es:

$$\epsilon = decayRate^{sim - maxSim * 0{,}015}$$

La constante $decayRate$ se define en un valor próximo a 1 y se eleva entre la diferencia de la simulación actual(sim) menos el 1,5 % de las simulaciones total que se van a realizar($maxSim$). Se han probado diferentes valores para el $decayRate$ hasta que se ha fijado en 0.99997. Con estos valores se consigue un alto número de simulaciones donde la aleatoriedad es alta y que se vaya decrementando para que en las últimas simulaciones el porcentaje de movimientos aleatorios sea mínimo.

El entorno donde se desarrolla el aprendizaje es extenso y con gran variedad de posibles movimiento. El ajuste de parámetros ha sido útil para que se realice una exploración del entorno considerable y se contemplen diferentes casuísticas. Un algoritmo poco exploratorio podría no llegar a encontrar una solución o dar una solución donde, al desconocer gran parte del mapa, no cruce con boyas próximas o no evite atravesar zonas de tormenta.

7.6. Evolución de las recompensas

Durante el proceso se han ido modificando en el entorno de simulación las recompensas de las boyas y llegar al destino. Al aumentar esos premios se ha conseguido evitar casos donde, aún estar próximo a una boya no se tome un camino para cruzarse con ella.

Con tal de evitar movimientos no deseados también se han incluido la casuística, no contemplada inicialmente, que penaliza de forma aumentada al hacer un movimiento sobre una posición ya visitada, ya sea agua o una boya. En la figura 7 se puede observar los valores más actualizados de las recompensas y penalizaciones.

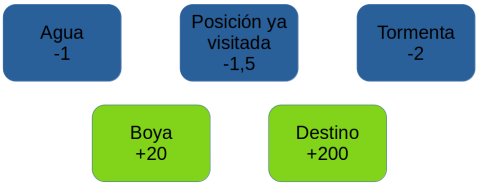


Fig. 7: Recompensas actuales del simulador

8 Análisis de Resultados

Este proyecto ha generado una solución con tres versiones diferentes. Se evaluaran las capacidades que tienen cada una de estas para cumplir el objetivo y que valor aportan cada uno de los algoritmos. Para realizar esto han sido probadas las 20 posibles combinaciones que existen entre puerto y destino por cada versión un vez terminado el entrenamiento y eliminando el factor de aleatoriedad.

En la tabla 1 se observa como la solución con la heurística de no retroceso tiene la mejor tasa de éxito, seguida de la versión sin heurística.

Solución	Tasa de éxito
No heurística	95 %
No retroceso	100 %
Dirección de destino	90 %

TABLA 1: Tasa de éxito

Tras un estudio sobre los valores de las matrices entrenadas se ha podido observar las casuísticas que dificultan a cada algoritmo el llegar al destino. En la figura 8 se observa un mapa donde las zonas más blancas son las que el algoritmo ha detectado que tienen una mayor recompensa. Esta imagen es la matriz que pertenece a la versión con heurística de dirección de destino.

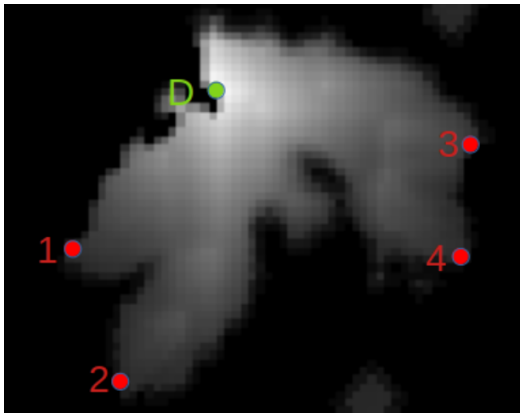


Fig. 8: Aprendizaje de la versión con heurística dirección de destino

Si se continua mirando la figura 8, un camino que produce error es llegar del puerto 2 al destino D. Los puertos que forman ángulos parecidos entre el origen y el destino o se encuentran próximo al objetivo no tienen problemas para cumplir con la tarea. Esto ocurre porque dos rutas que cumplan la primera condición acabarán convergiendo y el aprendizaje obtenido para aprender un camino será utilizado para entrenar la otra ruta y viceversa. En este caso, el puerto 2 se encuentra a una distancia considerable del destino y el resto de puertos no forman ángulos parecidos al suyo con el destino, por lo que el entrenamiento deberá conseguir hacer llegar al barco sin ayuda del aprendizaje obtenido por las otras rutas.

En la figura 9 se muestra el mapa anterior pero con otro destino y para la versión sin heurística.

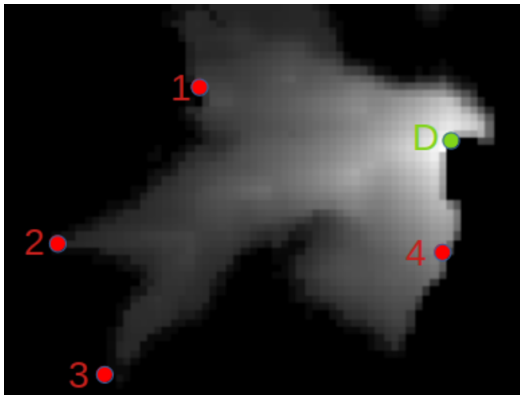


Fig. 9: Aprendizaje de la versión sin heurística

El caso de error que aparece en esta situación es llegar del puerto 3 al destino D. Esta solución desconoce completamente donde está el destino hasta que de forma aleatoria llega. Al igual que en el caso anterior el entrenamiento de otras rutas es utilizado para entrenar el resto de caminos. Debido a la aleatoriedad la condición que hace favorable que el algoritmo consiga resolver la problemática es la proximidad con el destino o que existan puertos a lo largo del camino entre origen y destino. El caso del puerto origen 3 no cumple con ninguna de las condiciones. Si bien es cierto que existe un puerto cercano, este no es de ayuda ya que no se encuentre entre el camino de origen y destino.

Los casos de éxito de todas las versiones generan en multitud de casos el mismo resultado. Han habido 7 rutas donde los tres algoritmos han conseguido el objetivo, pero han diferido en los resultados obtenidos. En la tabla 2 se puede observar los resultados obtenidos para los casos comentados.

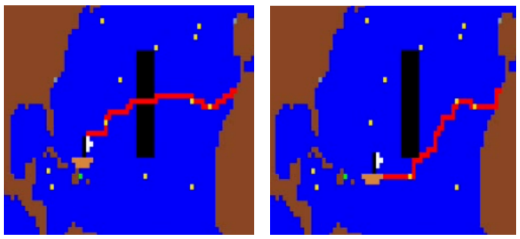
Solución	Recompensa	Pasos	Boyas
No heurística	1442	431	23
No retroceso	1470	431	24
Dirección de destino	1434	425	22

TABLA 2: Resultados

La versión sin heurística posee la segunda mejor recompensa, por detrás de la heurística de no retroceso. Esto sucede porque aunque realizan el mismo número de movimientos la versión con heurística ha conseguido pasar por un boya más. La peor recompensa es obtenida por la heurística de

dirección de destino, la cual a pesar de hacer menos pasos también se cruza con menos boyas. Esto es debido a que la heurística implementada es más restrictiva con el movimiento i por lo tanto no explora tanto el entorno como las otras soluciones.

Debido a la forma de aprendizaje, ocurren casos donde los algoritmos con heurísticas evitan cruzar las zonas tormentosas que producen más penalización y la versión sin heurística si que la atraviesa para conseguir llegar a una boya que hace que compense la penalización con la recompensa. Esto ocurre debido a que no todos los algoritmos exploran las mismas zonas del mapa y por lo tanto algunas versiones se quedarán con una solución que es buena, pero que no es la óptima. En la figura 10 se puede ver el resultado de la ejecución, donde la imagen de la izquierda pertenece a la versión sin heurística y la imagen de la derecha es el resultado de la ejecución de la heurística de no retroceso.



Versión sin heurística | Versión con heurística no retroceso

Fig. 10: Decisiones en zonas tormentosas

También ocurren casos donde la mejor ruta, aportada por las tres versiones, es atravesando la tormenta para llegar al objetivo.

Durante la ejecución del proyecto han surgido problemáticas que han dificultado el desarrollo en algunos puntos. Al momento de construir el entorno de simulación, una de las cuestiones que más complejas a resolver ha sido como representar las simulaciones de forma visual, fue un asunto donde la complejidad residía en pensar la solución y no tanto en la dificultad técnica. Finalmente, se optó por generar un vídeo a través de agrupar matrices de forma temporal.

9 Conclusiones

Este proyecto ha tenido como finalidad resolver las problemática que existe en un entorno de mar abierto para la comunicación entre boyas oceanográficas y navíos a través de protocolos oportunísticos. Aplicando técnicas de aprendizaje por refuerzo y técnicas heurísticas ha sido posible trazar las mejores rutas de un puerto origen a un destino estableciendo comunicación con el máximo número de boyas posible sin que esto suponga un gran desvío en el camino.

La ejecución del proyecto ha constado de la implementación de dos componentes dependientes entre si, el entorno de simulación y el algoritmo de aprendizaje. El primer componente es necesario para poder representar la situación donde ocurre el problema y tiene la complejidad de que debe representar el mundo de la forma más realista posible con los recursos y el tiempo disponible para la ejecución del proyecto. La implementación del aprendizaje depende en gran medida del entorno de simulación, ya que sin datos o con datos incorrectos no es posible encontrar un solución al problema. Otra dependencia que existe es entorno a la naturaleza de los datos, debido a que si las entradas y las salidas no fuesen discretas, como lo es el simulador actual, deberían cambiar aspectos importantes en el aprendizaje implementado.

Durante la realización del aprendizaje se han creado tres versiones diferentes de la solución. En primer lugar, se ha generado una solución sin heurísticas que se movía aleatoriamente para explorar el entorno y sin recibir ningún tipo de información adicional sobre las coordenadas del destino. La segunda solución implementaba una heurística que penalizaba realizar movimientos de retroceso. La última solución poseía una heurística que variaba la penalización en función de la diferencia entre el movimiento realizado y la posición del destino respecto al barco.

Después de tener el simulador y los algoritmos programados, en las primeras ejecución no se obtenían resultado coherentes. Esto se debía en casuísticas no contempladas, como que al chocar con tierra el algoritmo no producía penalizaciones y por eso en algunos casos se interpretaba que ir tierra adentro era la mejor solución al problema. Esto se solucionó inicializando las matrices a menos infinito. Otras casuísticas no contempladas fueron el hecho de que el barco hiciera movimientos repetitivos sobre sus propios pasos y también que decidiese pasar por las boyas en más de un ocasión. Para esto lo que se hizo fue generar condiciones especiales donde un vez se pase por un boya esta genera penalizaciones al volver a pasar y introducir un aumento en la penalización en los casos donde el barco hiciese un movimiento hacía un posición donde ya había estado anteriormente.

Otro problema relevante que ha habido en el proceso ha sido el ajuste de parámetros del algoritmo. Ejecutar el código realizado supone algunas horas de ejecución con lo cuál modificar un parámetro y validar si esto suponía una mejora implicaba mucho tiempo. Los parámetros más problemáticos han sido el número de simulaciones y el número de iteraciones. Escoger un número elevado de estos no solamente supone un coste computacional grande, sino que existe el riesgo de sobreentrenar el modelo. Si el modelo sufre un sobreentreno todos los algoritmos darían soluciones perfectas, ya que no se estaría realizando un aprendizaje correcto, sino que se descubriría la mejor solución probando la mayoría de los posibles casos a fuerza bruta. El problema del tiempo también ha afectado a la hora de modificar y probar cambios en la configuración del simulador, como el valor de las recompensas de los elementos.

Una vez resueltos los problemas, se ha determinado que el mejor resultado en cuanto a tasa de éxito y proporción de boyas encontradas por movimientos realizados a sido la solución con la heurística de no retroceso. En caso de necesitar un algoritmo de forma generalizada esta sería la mejor opción. No obstante, la versión con la heurística de dirección de destino permite llegar en menos movimientos al destino, sacrificando el número de boyas encontradas.

En determinadas situaciones donde se requiere llegar de forma rápida al destino y no son tan prioritarias las boyas este algoritmo es más útil. Sin embargo, hay que tener en cuenta las casuísticas, anteriormente explicadas, que pueden hacer reducir la tasa de éxito del algoritmo. La versión sin heurística no aporta características diferenciales respec-

to a las otras, es por eso que esta solución es la menos interesante de implementar.

En conclusión, este proyecto ha cumplido el objetivo general establecido. Se ha utilizado el aprendizaje por refuerzo para optimizar los viajes en barco y fomentar la comunicación en mar abierto, dando buenos resultados.

9.1. Líneas futuras

En futuras evoluciones se puede estudiar como el cambio en las recompensas y las penalizaciones afecta al aprendizaje o hacer que algunas boyas tengas más importancia que otras.

Es de interés introducir dinamismo en el entorno. Las tormentas podrían variar de posición, así como de valor de la penalización, de forma aleatoria. Con esto se conseguiría que la tormentas no fuesen estáticas y que pudiese haber tormentas más violentas que otras.

10 AGRADECIMIENTOS

Agradezco a mi tutora del TFG por ayudarme a resolver los problemas que han ido surgiendo en la realización del proyecto y a mejorar el trabajo realizado.

Agradezco también a mi familia y amigos por haberme apoyado desde el inicio del trabajo.

REFERENCIAS

[1] McMahon, M., 2022. What are the Different Types of Buoys? (with pictures). [online] wise-geek.com. Available at: <https://www.wise-geek.com/what-are-the-different-types-of-buoys.htm> [Accessed 26 February 2022].

[2] en.wikipedia.org. 2022. Opportunistic mobile social network - Wikipedia. [online] Available at: <https://en.wikipedia.org/wiki/Opportunistic mobile social network> [Accessed 26 February 2022].

[3] IBM Cloud Education, 2020. What is Artificial Intelligence (AI)?. [online] ibm.com. Available at: <https://www.ibm.com/cloud/learn/what-is-artificial-intelligence> [Accessed 26 February 2022].

[4] Workfront.com. 2022. Waterfall Methodology - A Complete Guide — Adobe Workfront. [online] Available at: <https://www.workfront.com/project-management/methodologies/waterfall> [Accessed 27 February 2022].

[5] Duke, R., 2022. Gantt.com. [online] gantt.com. Available at: <https://www.gantt.com/> [Accessed 26 February 2022].

[6] Gunnell, M., 2021. What Is Trello, and How Do You Use It?. [online] howtogeek.com. Available at: <https://www.howtogeek.com/751448/what-is-trello-and-how-do-you-use-it/> [Accessed 27 February 2022].

[7] Fattah, S., 2022. Underwater Wireless Sensor Networks. [online] Encyclopedia.pub. Available at: <https://encyclopedia.pub/3316> [Accessed 22 March 2022].

[8] Taher, H., 2018. Sea Ad hoc Network (SANET) Challenges and Development. [ebook] International Journal of Applied Engineering Research. Available at: <https://www.ripublication.com/ijaer18/ijaerv13n8_90.pdf> [Accessed 28 March 2022].

[9] Felemban, E., Shaikh, F., Qureshi, U., Sheikh, A. and Qaisar, S., 2015. Underwater Sensor Network Applications: A Comprehensive Survey. 11th ed. [ebook] Available at: <https://journals.sagepub.com/doi/full/10.1155/2015/896832#: :text=UWSN %20is %20a %20network %20of,for %20the %20benefit %20of %20humans.> [Accessed 28 March 2022].

[10] Khattri, A., 2021. Introduction of Mobile Ad hoc Network (MANET) - GeeksforGeeks. [online] GeeksforGeeks. Available at: <https://www.geeksforgeeks.org/introduction-of-mobile-ad-hoc-network-manet/> [Accessed 28 March 2022].

[11] Wagner, W., 2020. Examples of Heuristics in Computer Science - Qvault. [online] Qvault. Available at: <https://qvault.io/computer-science/examples-of-heuristics-in-computer-science/> [Accessed 29 March 2022].

[12] Education, I., 2020. What is Machine Learning?. [online] Ibm.com. Available at: <https://www.ibm.com/cloud/learn/machine-learning> [Accessed 29 March 2022].

[13] Gans, S., 2022. How Heuristics Help You Make Quick Decisions. [online] Verywell Mind. Available at: <https://www.verywellmind.com/what-is-a-heuristic-2795235> [Accessed 29 March 2022].

[14] 101 Computing. 2018. A* Search Algorithm | 101 Computing. [online] Available at: <https://www.101computing.net/a-star-search-algorithm/> [Accessed 29 March 2022].

[15] Jones, M., 2017. Train a software agent to behave rationally with reinforcement learning. [online] IBM Developer. Available at: <https://developer.ibm.com/articles/cc-reinforcement-learning-train-software-agent/> [Accessed 29 March 2022].

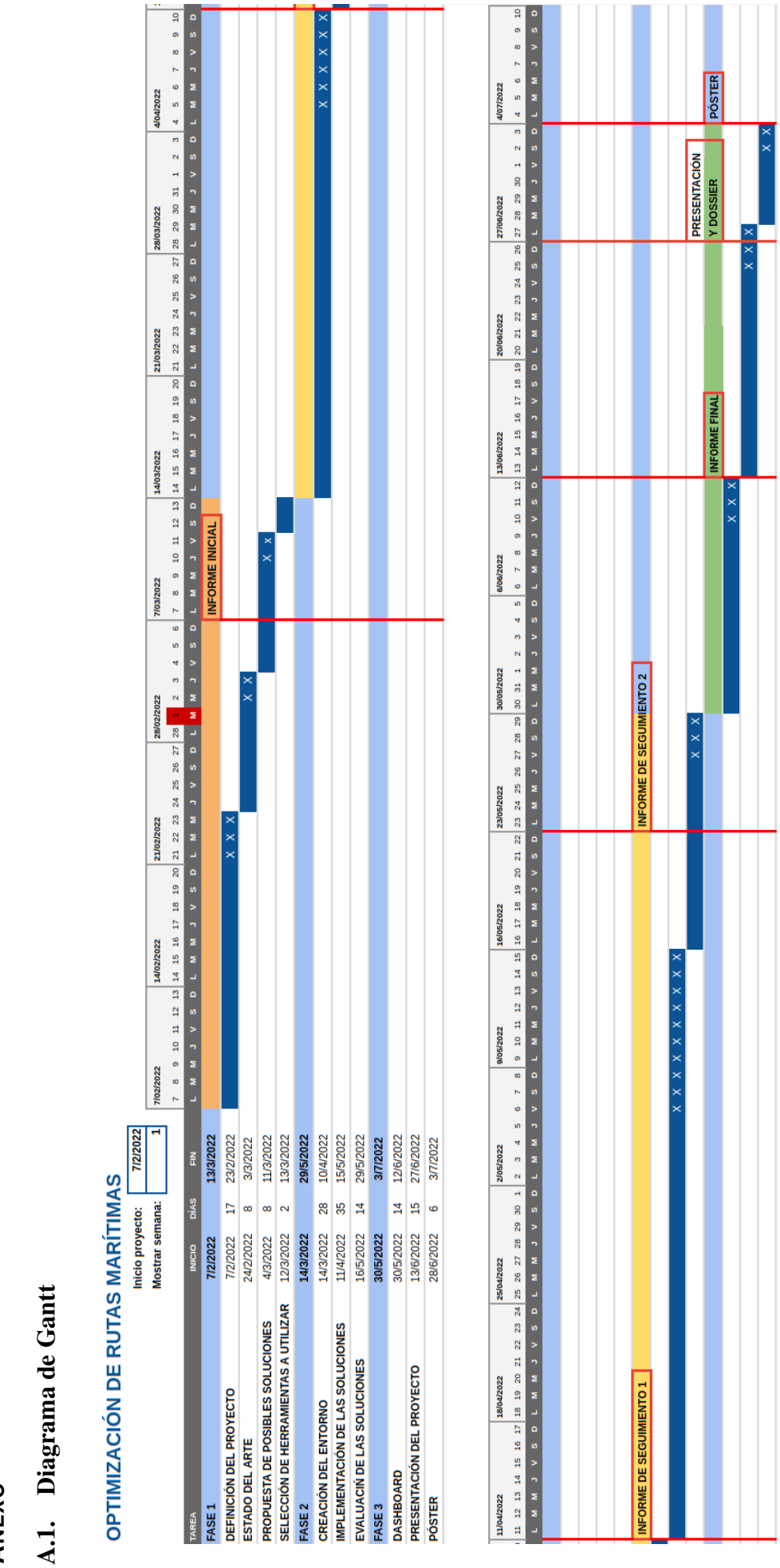
[16] Ponggame.org. 2022. Pong Game. [online] Available at: <https://www.ponggame.org/> [Accessed 6 April 2022].

[17] M, C., 2022. BeginnersGuide/Overview - Python Wiki. [online] Wiki.python.org. Available at: <https://wiki.python.org/moin/BeginnersGuide/Overview> [Accessed 6 April 2022].

[18] Violante, A., 2019. Simple Reinforcement Learning: Q-learning. [online] Towards Data Science. Available at: <https://towardsdatascience.com/simple-reinforcement-learning-q-learning-fcddc4b6fe56> [Accessed 19 April 2022].

ANEXO

A.1. Diagrama de Gantt



A.2. Underwater Sensor Network