Planificación de rutas para múltiples drones usando redes neuronales profundas Transformer

Fuertes Coiras, Daniel[1,*]; del-Blanco Adán, Carlos Roberto[1]; Navarro Corcuera, Juan José[2]; Jaureguizar Núñez, Fernando[1]; García Santos, Narciso[1].

[1] Grupo de Tratamiento de Imágenes (GTI), ETSI de Telecomunicación, Universidad Politécnica de Madrid, 28040 Madrid, Spain. d.fcoiras@upm.es (DFC), carlosrob.delblanco@upm.es (CRdBA), fernando.jaureguizar@upm.es (FJN), narciso.garcia@upm.es (NGS)

[2] Airbus Defence and Space, 28906 Madrid, Spain. juan.j.navarro@airbus.com (JJNC)

***** Autor Principal y responsable del trabajo: d.fcoiras@upm.es (DFC)

Resumen: Una de las etapas más críticas del control y navegación de drones (UAV/RPAS) es la planificación de rutas. En la actualidad, dicha tarea es esencial en aplicaciones de búsqueda y rescate o en el Futuro Sistema Aéreo de Combate, donde un dron necesita planificar una ruta que minimice la distancia de vuelo entre un conjunto localizaciones para poder maximizar el número de zonas visitadas, todo ello sujeto a restricciones operativas, tales como el uso de batería o combustible. Esta tarea, ya de por sí compleja, se complica aún más en el caso de múltiples drones, donde la cooperación y coordinación de toda una flota es necesaria. En este trabajo se propone un sistema automático de planificación de rutas para múltiples drones usando técnicas de aprendizaje profundo y aprendizaje por refuerzo. El sistema divide el problema de enrutamiento en dos fases: Planificación Inicial y Ejecución de la Misión. Durante la Planificación Inicial, se realiza una agrupación de las regiones a visitar siguiendo un criterio de distancia, seguido de una asignación de dichas agrupaciones a cada dron. En la fase de ejecución de la misión, se estima la mejor ruta para cada agente mediante un *Transformer*, una arquitectura de red neuronal de última generación basada en mecanismos de atención, la cual es entrenado mediante técnicas de aprendizaje por refuerzo profundo. Esta arquitectura es capaz de obtener soluciones precisas y mucho más rápidas que los algoritmos de optimización convencionales. Para mostrar los beneficios de la solución propuesta, se han realizado varias pruebas y comparaciones con otros algoritmos de Optimización Combinatoria, incluyendo escenarios cooperativos y no cooperativos.

Palabras clave: vehículo aéreo no tripulado, aprendizaje por refuerzo profundo, modelo de atención.

1. Introducción

El creciente uso de los vehículos aéreos no tripulados (UAV/RPAS) en los últimos años se debe a su gran versatilidad y desempeño en aplicaciones, como entrega de paquetes, vigilancia, búsqueda y rescate o el Futuro Sistema Aéreo de Combate. El principal problema de estos vehículos está ligado a su manejo, ya que tradicionalmente es un operador humano el encargado de su control mediante órdenes transmitidas en tiempo real. Aparte del inconveniente que suponen las comunicaciones de baja latencia entre el operador y el UAV, existe una dificultad extra en el manejo y la coordinación de flotas de UAVs. Especialmente en este caso, se hace necesaria la automatización del control de los UAVs.

Entre las múltiples tareas que implica el control automático de UAVs, destaca la planificación de rutas o enrutado, ya que es responsable de generar caminos óptimos entre diversas localizaciones con el fin de minimizar la distancia recorrida. Esto permite tanto la maximización del número de zonas visitadas, como la optimización de la batería/combustible de los UAVs. Para múltiples UAVs, las rutas deben también considerar la cooperación entre los miembros de la flota para maximizar los resultados de la misión. Todo ello conduce a plantear la planificación de rutas como un problema de Optimización Combinatoria (OC) [1]. Más concretamente, este problema se ajusta al Problema de Orientación en Equipo (POE) [2], una variante multi agente del Problema de Orientación (PO). En el POE, se considera un equipo de agentes (UAVs) que deben visitar un conjunto de regiones y regresar a la base (la inicial u otra distinta) en un tiempo límite. Además, cada vez que un agente llega a una nueva región, recibe una recompensa. Debido la restricción temporal, el objetivo en el POE no es visitar todas las zonas (no siempre es posible) sino maximizar la suma total de recompensas recibidas.

Aunque abordar la planificación de rutas como un POE ha sido la opción mayoritaria en la literatura científica, su formulación tiene algunas restricciones que no se ajustan adecuadamente a aplicaciones cooperativas más realistas. Un ejemplo es la restricción de que cada región sea visitada, como máximo, por un único agente. En aplicaciones como vigilancia o búsqueda y rescate, donde la aparición de objetivos es estocástica; o en el Futuro Sistema Aéreo de Combate, en el que un único dron puede no ser suficiente para escenarios complejos de combate, dicha restricción es muy limitante y puede implicar una pérdida notable de eficiencia en la misión. Además, los algoritmos dedicados al POE se caracterizan por una gran complejidad de diseño y un tiempo de computación elevado, lo que ha propiciado que el número de trabajos que lo abordan sea significativamente menor que los dedicados al PO. Por este motivo, en [3] se propone una alternativa que divide el escenario en zonas asignadas a cada vehículo, de manera que se pueda abordar el PO de forma independiente en cada zona. Está solución simplifica el POE, pero sigue siendo ineficiente porque impide que las regiones sean visitadas por más de un UAV y porque el espacio de soluciones está limitado por la división en regiones.

Pese a que la propuesta anterior consigue simplificar el problema de enrutamiento de múltiples agentes, predecir rutas óptimas para un único agente sigue siendo una tarea difícil. El PO es un problema *NP-hard* (*Non-deterministic Polynomial-time hard*), siendo no posible hallar una solución exacta en tiempo polinómico. Esta razón ha llevado a muchos investigadores a decantarse por soluciones heurísticas con las que encontrar rutas que se aproximen a la ruta óptima en tiempos relativamente bajos. Algunos ejemplos son: Tsiligirides [4], que añade a la búsqueda heurística una etapa adicional para mejorar las rutas encontradas; Optimización por Enjambre de Partículas [5], que mejora a los métodos de búsqueda exhaustiva desplazando múltiples partículas (soluciones candidatas) por el espacio de búsqueda; y los Algoritmos Genéticos (AG) [6], inspirados en el proceso de selección natural (biológica) para evolucionar y mejorar conjuntos de soluciones. Uno de los AG más relevantes es Compass [7] porque está diseñado específicamente para problemas de enrutamiento. Como

alternativa a los métodos heurísticos, cabe destacar las soluciones de programación lineal. Este tipo de soluciones sacrifican velocidad de computación a cambio de mejores soluciones. Algunos ejemplos incluyen: OR-Tools [8], la herramienta de Google para resolver problemas de enrutamiento; y Gurobi [9], que obtiene resultados muy buenos a costa de un tiempo de ejecución excesivamente alto.

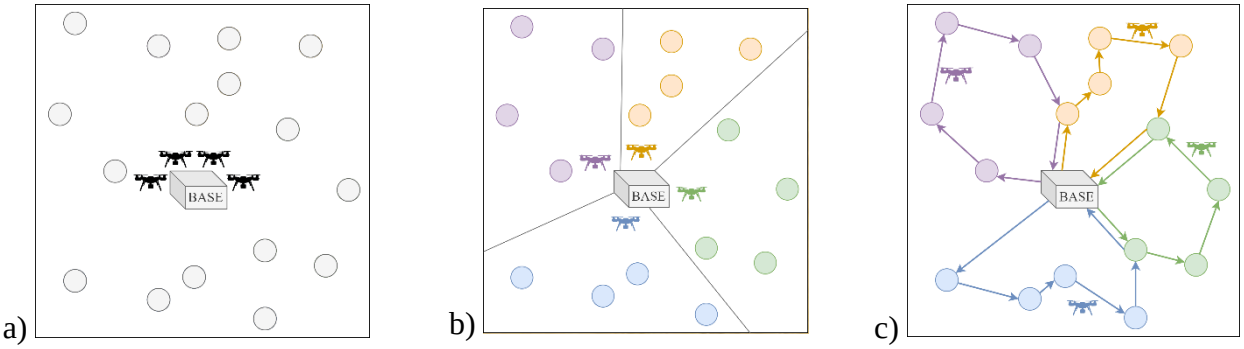

Figura 1. Ejemplo de misión con una flota de 4 UAVs: (a) Un escenario compuesto por una base (recuadro) con 4 UAVs y un conjunto de regiones (círculos). (b) La fase de Planificación Inicial agrupa las regiones y asigna cada una a un UAV. (c) La fase de Ejecución de Misión visita tantas regiones como sea posible de las asignadas (prioritarias) y algunas regiones asignadas a otros UAVs.

Las soluciones más recientes y prometedoras están más relacionadas con el área de Aprendizaje Profundo por Refuerzo, que hacen uso de redes neuronales para aprender a maximizar la recompensa total recogida, siendo capaces de hallar soluciones aproximadas muy rápidamente. En [10] nacen las *Pointer Networks* (PN), basadas en redes recurrentes y un mecanismo de Atención. El uso de las PN se expandió a otros trabajos como [11], donde se añade una Red Neuronal de Grafos (*Graph Neural Network* - GNN) a la PN para mejorar los resultados. El principal inconveniente de los algoritmos basados en redes recurrentes es que estas tienden a "olvidar" la información pasada de secuencias muy largas de datos, por lo que no son aplicables a grandes grafos. Como alternativa, en [12] se propuso una nueva estructura de red neuronal llamada *Transformer*. Estas redes tratan de imitar a las redes recurrentes, codificando los nodos para que se relacionen todos entre sí. La gran diferencia es que son capaces de procesar los datos en paralelo (en vez de secuencialmente), mejorando así el rendimiento en todos los sentidos. Pronto, estas redes fueron aplicadas a problemas de enrutado con gran éxito [13].

En este artículo, se propone un algoritmo de enrutado para múltiples UAVs cooperativos basado en una red *Transformer* que soluciona varios de los problemas anteriores. Consta de dos etapas, como se puede ver en la Figura 1. La primera, denominada Planificación Inicial e inspirada en [3][14], realiza una partición de las regiones y asigna los grupos/clústeres de regiones a cada UAV. Se emplea el algoritmo de K-Means con restricción de tamaño [15] para que todos los grupos tengan un tamaño similar. Esta estrategia logra reducir significativamente el coste computacional, aunque con el inconveniente de obtener soluciones subóptimas. En la segunda etapa, Ejecución de Misión, se proponen, para cada UAV, una ruta que no solo se ciñe al grupo de regiones asignadas, sino que también involucra regiones de otros UAVs para favorecer la cooperación. Con ello, se logra compensar el efecto negativo derivado de la primera etapa, ya que se amplía el espacio posible de soluciones inicialmente restringido. Para tal propósito, se ha diseñado un algoritmo basado en la arquitectura *Transformer*, que destaca por estimar soluciones precisas en tiempos de ejecución significativamente menores que la mayoría de los algoritmos clásicos de optimización. Por último, cabe destacar dos ventajas adicionales del sistema propuesto. Primero, se elimina la restricción del POE de impedir revisitar regiones. Y segundo, se da la opción de compartir regiones durante la ejecución de la misión, favoreciendo la cooperación entre agentes e incrementado la recompensa total obtenida en la misión.

2. Desarrollo

El sistema propuesto para el enrutamiento cooperativo multi-UAV se compone de las fases de Planificación Inicial, basada en un algoritmo de K-Means con restricción de tamaño, y Ejecución de Misión, modelada como una variante del PO (Sección 2.1) y resuelta mediante una red neuronal *Transformer* (Sección 2.2) entrenada con un algoritmo de Aprendizaje por Refuerzo (Sección 2.3).

2.1. Formulación del problema

El problema es una variante del PO llamada Problema de la Orientación con Múltiples Premios y Tipos de Nodo (PO-MP-TN), que tiene distintas recompensas dependiendo del tipo de nodo visitado (inicial o compartido). En este problema, se deben considerar m agentes que deben visitar un conjunto $\{0,\dots,n+1\}$ de nodos/regiones tal que $m \leq n$, donde los nodos 0 y $n+1$ son la base inicial y final. También debe tenerse en cuenta que los nodos $\{1,\dots,n\}$ se agrupan en m subconjuntos/clústeres $\{C_a | a = 1,\dots,m\}$. Es recomendable equilibrar el número de regiones para optimizar el rendimiento global, de manera que $(n \bmod m)$ subconjuntos contengan $\left\lfloor \frac{n}{m} \right\rfloor + 1$ nodos, y el resto contenga $\left\lfloor \frac{n}{m} \right\rfloor$ nodos, siendo mod el operador módulo. Un agente $a = 1,\dots,m$ es recompensado con $r_i^a = r_{ini},\ i = 0,\dots,n+1$ cada vez que visita una región de su propio clúster C_a (región inicial). Y es recompensado con $r_i^a = r_{com}$ cada vez que visita una región de otro clúster (región compartida). Además, se asume que $r_{ini} > r_{com}$ para que los agentes prioricen sus propias regiones. Considerando estas recompensas, el objetivo es obtener la mejor ruta posible mediante la maximización de la función objetivo

$$max \sum_{a=1}^{m} \sum_{i=1}^{n} \sum_{j=1}^{n+1} r_i^a \, x_{ij}^a, \tag{1}$$

donde $x_{ij}^a = 1$ si el nodo j es visitado por el agente a inmediatamente después del nodo i, y 0 en caso contrario. La Ecuación 1 debe estar sujeta las siguientes restricciones para garantizar que haya una solución factible al problema:

$$\sum_{j=1}^{n+1} x_{0j}^a = \sum_{i=0}^{n} x_{i(n+1)}^a = 1; \quad a = 1,\dots,m \tag{2}$$

$$\sum_{i=0}^{n} x_{ik}^a = \sum_{j=1}^{n+1} x_{kj}^a \leq 1; \quad a = 1,\dots,m; \quad k = 1,\dots,n \tag{3}$$

$$\sum_{i=0}^{n} \sum_{j=1}^{n+1} t_{ij}^a \, x_{ij}^a \leq t_{max}; \quad a = 1,\dots,m \tag{4}$$

$$u_i^a - u_j^a + n x_{ij}^a \leq n - 1; \quad a = 1,\dots,m; \quad i,j = 1,\dots,n \tag{5}$$

La restricción de la Ecuación 2 garantiza que el recorrido comience y finalice en la base. La Ecuación 3 evita que un nodo sea visitado más de una vez por el mismo agente (las regiones pueden ser visitadas por múltiples agentes, pero un agente no puede revisitar una región), y asegura que el camino sea continuo. La Ecuación 4 impone t_{max} como tiempo de misión, siendo t_{ij}^a el tiempo o la distancia recorrida por el agente a desde el nodo i hasta el nodo j. También asegura que haya tiempo suficiente para volver a la base. Por simplicidad, se asumen distancias euclidianas y velocidades equivalentes entre todos los UAVs (aunque se podrían considerar otros casos). Finalmente, la Ecuación 5 evita que la ruta final sea una combinación de subrutas, siendo $u_i^a, u_j^a = 1,\dots,n$ el orden posicional de los nodos i y j en la ruta del agente a.

2.2. Red neuronal Transformer

El funcionamiento de la red *Transformer* se representa en la Figura 2. Esta red posee una arquitectura del tipo codificador-decodificador. La idea es codificar el grafo de nodos a la entrada para obtener una representación que resalte las relaciones entre aquellos nodos que puedan conformar una mejor ruta (secuencia de nodos). El módulo de decodificación se encarga, por tanto, de analizar esta representación codificada y predecir la mejor ruta posible.

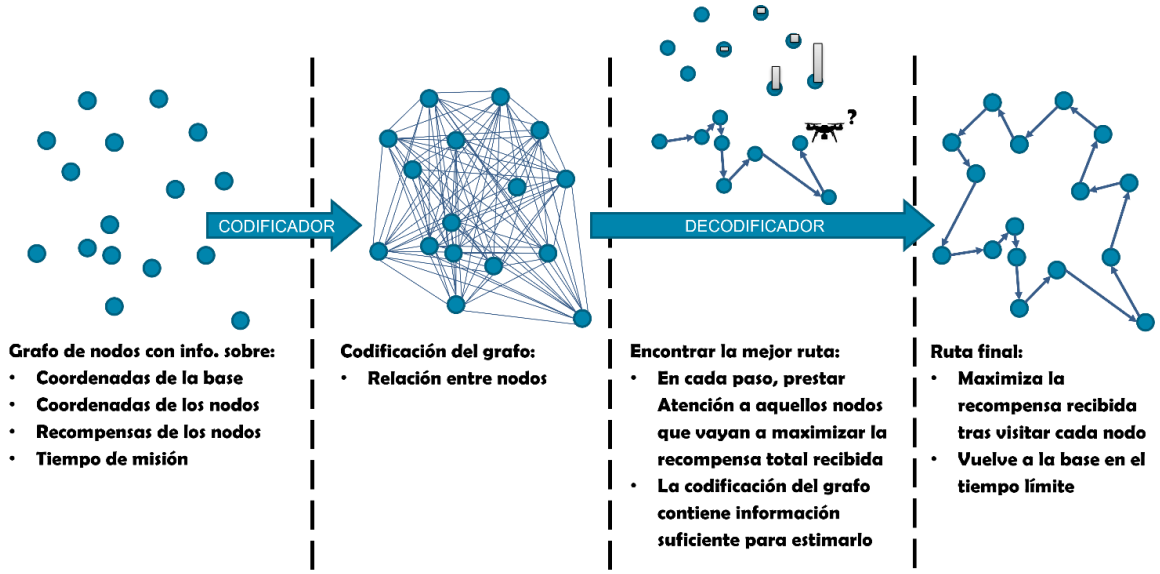

Figura 2. Estructura de la red neuronal *Transformer*. Esta red codifica la información de un grafo de nodos con el objetivo de obtener una representación que ayude al módulo de decodificación a estimar la mejor ruta (aquella que maximiza el valor de las recompensas obtenidas y vuelve a la base dentro del límite de tiempo).

El codificador consta de una proyección lineal de los datos de entrada y tres bloques de codificación similares a los de [12] y basados en el mecanismo de Atención. Los datos de entrada constan de toda la información del grafo, esto es, las coordenadas de la base y los nodos, las recompensas de los nodos y el tiempo de misión. La proyección del grafo representa una codificación lineal que facilita a los bloques de Atención posteriores la comprensión e interrelación de los datos. Los bloques de Atención generan una representación vectorial del grafo que, tras un proceso de entrenamiento, permitirá al decodificador deducir la relación entre los distintos nodos para hallar el mejor camino posible.

El decodificador toma la información del codificador para calcular secuencialmente el recorrido que maximice la Ecuación 1. Esto es, dada la codificación de los nodos y la situación actual del agente (en cada instante), el decodificador estima el mejor próximo nodo a visitar, construyendo una ruta completa tras varias iteraciones. La situación del agente se obtiene de la proyección lineal de la posición y el tiempo restante del agente. Esta proyección se combina posteriormente con los datos que provienen del codificador mediante un nuevo bloque de Atención cruzada (nodos y agente) para obtener una representación que relacione la situación del agente con el resto de nodos visitables. Se utiliza un mecanismo de enmascaramiento para filtrar los nodos que se pueden visitar según las restricciones de la Sección 2.1. Un último bloque de Atención recibe esta representación para calcular las probabilidades normalizadas de cada nodo. Estas probabilidades indican la confianza de la red de que cada nodo sea la mejor elección para encontrar la mejor ruta. En la fase de inferencia se seleccionan los nodos de mayor probabilidad para obtener la mejor ruta. En cambio, durante el entrenamiento conviene facilitar el aprendizaje de la red mediante una estrategia de “exploración vs explotación”.

2.3. Entrenamiento mediante Aprendizaje por Refuerzo

Se entrena una sola red *Transformer* ya que durante la inferencia se comparten los pesos entre los agentes. El algoritmo de entrenamiento es *Reinforce* [16], cuya función de coste a minimizar es la función objetivo negativa del PO-MP-TN definida en la Ecuación 1, es decir, el objetivo es maximizar la recompensa total recibida. Por lo tanto, la función de coste se puede definir como:

$$L(v) = -\sum_{a=1}^{m}\sum_{i=1}^{n}\sum_{j=1}^{n+1} r_i^a \, x_{ij}^a, \tag{6}$$

donde v es una ruta que resuelve una instancia α del PO-MP-TN. Una instancia contiene un escenario PO-MP-TN particular compuesto por un límite de tiempo dado y un número determinado de nodos, definidos por sus coordenadas y recompensas. Para minimizar la función de coste $L(v)$, se utiliza *Gradient Descent*, donde el gradiente, según *Reinforce*, en este caso se define como:

$$\nabla \mathcal{L}(\theta|\alpha) = E_{p_\theta(v|\alpha)}\left[\left(L(v) - b(\alpha)\right)\nabla \log p_\theta(v|\alpha)\right] \tag{7}$$

$$b(\alpha) = \underset{v}{argmax}\, p_\theta(v|\alpha)$$

En la Ecuación 7, $b(\alpha)$ es una referencia similar a la que se usa en [13] y representa el coste (recompensa negativa) de la mejor ruta predicha por el modelo para cada instancia α. El motivo de incluir $b(\alpha)$ es reducir la alta varianza del gradiente, ya que este se estima muestreando una sola trayectoria de la instancia α y, por lo tanto, varía significativamente de una trayectoria a otra.

3. Resultados y discusión

Se han generado automáticamente un conjunto de instancias/muestras del PO-MP-TN para comparar el sistema propuesto con otros algoritmos del estado del arte. Además, se ha evaluado todo el sistema de enrutamiento multi-agente utilizando escenarios cooperativos y no cooperativos. Cada conjunto de las instancias generadas contiene un número diferente de nodos ($n \in [20,50,100,200]$) y agentes ($m \in [2,4]$). Para escenarios cooperativos y no cooperativos, la recompensa de las regiones compartidas es $r_{com} = 0{,}5$ y $r_{com} = 0$, respectivamente. En ambos casos, la recompensa por las regiones iniciales es $r_{ini} = 1$. Las coordenadas del escenario están normalizadas en el rango [0,1] para facilitar el aprendizaje de la red. El límite de tiempo t_{max} se expresa como un valor de distancia normalizado (se asume una velocidad constante equivalente a una unidad de distancia por unidad temporal), y su valor se fija para permitir que cada agente llegue aproximadamente a la mitad de los nodos: $t_{max}(n = 20) = 2$, $t_{max}(n = 50) = 3$, $t_{max}(n = 100) = 4$ y $t_{max}(n = 200) = 5$. La razón es que estos casos son más desafiantes que aquellos en los que t_{max} permite visitar casi todos o ninguno de los nodos. Por último, para cada configuración del PO-MP-TN, se han generado tres conjuntos para entrenar (10^6 instancias), validar (10^4 instancias) y evaluar (10^4 instancias) el sistema.

Teniendo en cuenta estos conjuntos de datos, se ha realizado una comparación entre diferentes algoritmos para valorar el rendimiento del sistema propuesto para un solo agente. Estos algoritmos son: Compass [7], Tsiligirides [4], OR-Tools [8], *Pointer Network* (PN) [10] y *Graph Pointer Network* (GPN) [11]. Los resultados de la comparación se exponen en la Figura 3. Se puede observar que Compass alcanza el valor de recompensa más alto, seguido de cerca por el *Transformer* propuesto. Sin embargo, el coste computacional de Compass es mucho más alto que el de *Transformer*, lo que limita su uso en misiones cooperativas que requieren decisiones rápidas. Compass comparte este problema con OR-Tools, que no solo es menos preciso que la red *Transformer*, sino que también es 2 órdenes de magnitud mayor en tiempo de inferencia. Los demás algoritmos logran tiempos comparables al

rendimiento de *Transformer* para tamaños de grafos pequeños ($n \in [20, 50]$). Para grafos de gran tamaño ($n \in [100, 200]$), *Transformer* es considerablemente más rápido que PN y GPN, pero un poco más lento que Tsiligirides. Por lo tanto, se puede deducir que el tiempo de cálculo de Tsiligirides escala mejor para una gran cantidad de nodos. De hecho, Tsiligirides y Compass alcanzan tan buenos resultados (de tiempo y recompensa, respectivamente) porque están diseñados específicamente para resolver el PO, pero su principal inconveniente es que no se pueden usar para resolver otros problemas de Optimización Combinatoria, a diferencia del resto de métodos. En cualquier caso, entre los algoritmos más rápidos, *Transformer* alcanza las recompensas más altas. Por tanto, se puede confirmar que *Transformer* es la solución que ofrece el mejor equilibrio entre calidad y velocidad.

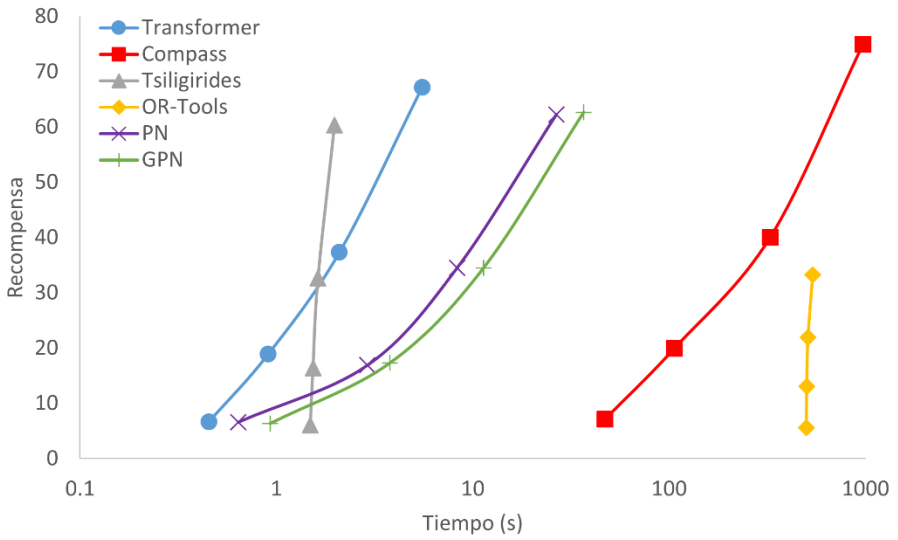

Figura 3. Comparación entre el Transformer y otros algoritmos del estado del arte en términos de tiempo de ejecución (en escala logarítmica) y recompensas recogidas en escenarios de 20, 50, 100 y 200 nodos.

Una vez evaluado el rendimiento mono-agente, se proporciona el rendimiento global del sistema multi-UAV en la Tabla 1, en términos de recompensa y número de regiones visitadas por la flota. Se consideran casos con 2 y 4 agentes en escenarios de $n = 100$ nodos para estudiar el impacto de la cooperación en misiones con diferente número de clústeres. Observando los resultados, la cooperación implica obtener un mayor valor de recompensa y visitar más nodos, ya que los UAVs no solo llegan a sus regiones iniciales, sino también a regiones de otros clústeres. La limitación de tiempo se aprovecha mejor con cooperación porque el tiempo restante se puede utilizar para mejorar la recompensa final. Sin cooperación, los beneficios de desplegar 4 UAVs en lugar de 2 son relativamente pequeños, ya que 2 UAVs son suficientes para visitar casi todas las regiones. Sin embargo, siguiendo la estrategia de compartir nodos se consigue aumentar la cantidad de nodos visitados a medida que aumenta la cantidad de agentes utilizados. Un beneficio adicional de la estrategia cooperativa se da en situaciones en las que faltan uno o más UAVs (destruidos, sin batería/combustible, etc.). Gracias a la cooperación, este problema se apacigua con UAVs aliados visitando las regiones asignadas al UAV desaparecido.

Agentes	Escenarios cooperativos		Escenarios no cooperativos	
	Recompensas	Nodos	Recompensas	Nodos
2	93,304	98,746	90,544	92,170
4	149,508	206,664	99,996	112,580

Tabla 1. Recompensas y número medio de nodos visitados por el *Transformer* entrenado con $n = 100$ nodos para flotas de 2 y 4 agentes en escenarios cooperativos y no cooperativos.

4. Conclusiones

En este trabajo se ha propuesto y evaluado una solución para la gestión de rutas de múltiples UAVs cooperativos. La solución consiste en dividir el problema en una primera fase (Planificación Inicial), que asigna un grupo de regiones a cada agente mediante K-Means con restricción de tamaño, y una etapa posterior (Ejecución de Misión), donde una variante del PO (denominada PO-MP-TN) se resuelve con una red neuronal *Transformer* para obtener las rutas de múltiples UAVs. Los UAVs funcionan de forma cooperativa gracias a una estrategia de intercambio de regiones que mejora el rendimiento del sistema durante la misión. Se ha demostrado que la red *Transformer* ofrece un mejor equilibrio entre calidad y velocidad de computación que otros algoritmos del estado del arte. Además, el sistema global puede funcionar muy rápido, lo que es necesario para muchas aplicaciones basadas en UAVs, como en vigilancia, en entrega de paquetes, en el Futuro Sistema Aéreo de Combate, etc.

Agradecimientos

Este trabajo ha sido parcialmente financiado por el proyecto PID2020-115132RB (SARAOS) financiado por MCIN/AEI/10.1303501100011033 del Gobierno de España. Además, los autores agradecen el apoyo de Airbus Defence and Space S.A.U.

Referencias

1. Vesselinova N, Steinert R, Perez-Ramirez DF, Boman M. Learning Combinatorial Optimization on Graphs: A Survey With Applications to Networking. IEEE Access. 2020;8:120388–416.
2. Gunawan A, Lau HC, Vansteenwegen P. Orienteering Problem: A survey of recent variants, solution approaches and applications. Eur J Oper Res. 2016;255(2):315–32.
3. Sung I, Nielsen P. Zoning a Service Area of Unmanned Aerial Vehicles for Package Delivery Services. J Intell Robot Syst. 2020;97(3):719–31.
4. Tsiligirides T. Heuristic Methods Applied to Orienteering. J Oper Res Soc. 1984;35(9):797–809.
5. Jie KW, Liu SY, Sun XJ. A hybrid algorithm for time-dependent vehicle routing problem with soft time windows and stochastic factors. Eng Appl Artif Intell. 2022;109:104606.
6. Lambora A, Gupta K, Chopra K. Genetic Algorithm - A Literature Review. In: International Conference on Machine Learning, Big Data, Cloud and Parallel Computing. 2019. p. 380–4.
7. Kobeaga G, Merino M, Lozano JA. An efficient evolutionary algorithm for the orienteering problem. Comput Oper Res. 2018;90:42–59.
8. Perron L, Furnon V. OR-Tools [Internet]. Available from: https://developers.google.com/optimization/
9. Gurobi Optimization L. Gurobi Optimizer Reference Manual [Internet]. 2022. Available from: https://www.gurobi.com
10. Vinyals O, Fortunato M, Jaitly N. Pointer Networks. In: Advances in Neural Information Processing Systems. 2015.
11. Ma Q, Ge S, He D, Thaker D, Drori I. Combinatorial Optimization by Graph Pointer Networks and Hierarchical Reinforcement Learning. CoRR. 2019;abs/1911.0.
12. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, et al. Attention is All you Need. In: Advances in Neural Information Processing Systems. 2017.
13. Kool W, van Hoof H, Welling M. Attention, Learn to Solve Routing Problems! In: International Conference on Learning Representations. 2019.
14. Fuertes D, Del-Blanco CR, Jaureguizar F, Navarro JJ, García N. Solving Routing Problems for Multiple Cooperative Unmanned Aerial Vehicles using Transformer Networks. Submitted.
15. Bennett KP, Bradley PS, Demiriz A. Constrained K-Means Clustering. 2000.
16. Williams RJ. Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning. Mach Learn. 1992;8(3):229–56.