

Diseño Avanzado de Algoritmos

**Actividad 2. Planificación
multiobjetivo de rutas de
drones en entornos urbanos**

Ainoa Palomares Ruiz

Datos del estudiante

Nombre y apellidos Ainoa Palomares Ruiz

Fecha de entrega 04/01/2026

Actividad 2. Planificación multiobjetivo de rutas de drones en entornos urbanos

Introducción

En esta práctica se aborda un problema de planificación de rutas para un dron de reparto en un entorno urbano con restricciones. El objetivo es encontrar un circuito Hamiltoniano que visite todos los destinos partiendo y regresando a un hub central, minimizando simultáneamente la distancia recorrida, el riesgo acumulado y el consumo de batería, y evitando zonas restringidas de vuelo (no-fly zones).

Dada la complejidad combinatoria y multiobjetivo del problema, se emplean tanto técnicas exactas como heurísticas y metaheurísticas, con el fin de comparar calidad de soluciones, tiempos de ejecución y escalabilidad.

Modelado del problema

El problema se modela como un grafo completo implícito cuyos vértices representan puntos de entrega y estaciones de recarga. Cada arista posee un peso vectorial $\langle \text{distancia}, \text{riesgo}, \text{consumo de batería} \rangle$.

Las zonas no-fly se representan mediante polígonos, y cualquier arista que interseque alguno de ellos se considera inválida. Para ello se utilizan tests de intersección de segmentos vistos en el Tema 7.

Algoritmos implementados

○ Branch & Bound (BB)

Se ha implementado un algoritmo exacto de Branch & Bound que explora el espacio de soluciones mediante backtracking y poda basada en cotas inferiores de distancia. El algoritmo devuelve un conjunto de soluciones no dominadas que aproximan la frontera de Pareto.

Debido a su elevado coste computacional, se impone un límite temporal de 2 minutos por instancia.

```
def backtrack(path, visited):
    nonlocal best_distance

    # ⌚ Límite de tiempo
    if time.time() - start_time > time_limit:
        raise TimeLimitReached

    # ✂ Poda
    if lower_bound_distance(path) >= best_distance:
        return

    # ✅ Caso base
    if len(path) == N:
        if valid_edge(path[-1], hub):
            full_path = path + [hub]
            d, r, b = path_cost(full_path)

            if d <= best_distance:
                best_distance = d
                best_solutions.append((full_path, (d, r, b)))
        return
```

En el fragmento anterior se observa la estructura principal del algoritmo Branch & Bound, incluyendo la poda por cota inferior y el control explícito del tiempo de ejecución mediante una excepción.

○ Heurística geométrica (GEO)

La heurística geométrica construye una única solución factible basándose en criterios geométricos. Es extremadamente rápida, pero no genera frontera de Pareto

○ Simulated Annealing (SA)

Simulated Annealing parte de una solución inicial y la mejora mediante perturbaciones controladas, aceptando soluciones peores con cierta probabilidad decreciente. Ofrece un compromiso entre calidad de solución y tiempo de ejecución.

Diseño Experimental

Se han generado cuatro instancias con $N = 10, 15, 20$ y 25 nodos, todas con al menos tres zonas no-fly.

Branch & Bound se ejecuta una sola vez por instancia debido a su alto coste computacional, mientras que GEO y SA se ejecutan cinco veces, utilizando el tiempo medio como métrica representativa.

Se miden tiempo de ejecución, memoria pico, distancia, riesgo y consumo de batería.

```
def measure(func, *args):
    tracemalloc.start()
    start = time.time()

    result = func(*args)

    elapsed = time.time() - start
    current, peak = tracemalloc.get_traced_memory()
    tracemalloc.stop()

    return result, elapsed, peak / 1024 # KB
```

Este procedimiento se utiliza de forma uniforme para medir el tiempo de ejecución y el consumo máximo de memoria de cada algoritmo.

Resultados y Análisis

Tabla 1. Resultados experimentales por instancia y algoritmo

N	Algoritmo	Tiempo (s)	Distancia	Riesgo	Batería	Observaciones
10	BB	6.04	32.03	1.60	6.41	Frontera completa (25 soluciones)
10	GEO	0.0002	27.45	1.37	5.49	Mejor solución
10	SA	0.17	35.77	1.79	7.15	Menor calidad
15	BB	120.0	—	—	—	Timeout
15	GEO	0.0003	39.61	1.98	7.92	Estable
15	SA	0.17	42.09	2.10	8.42	Peor que GEO
20	BB	120.0	69.87	3.49	13.97	Frontera parcial
20	GEO	0.0004	41.66	2.08	8.33	Muy competitiva
20	SA	0.19	41.87	2.09	8.37	Similar a GEO
25	BB	120.0	96.70	4.83	19.34	Frontera parcial
25	GEO	0.0005	48.32	2.42	9.66	Buena escalabilidad
25	SA	0.20	48.14	2.41	9.63	Mejor solución

Resultados numéricos

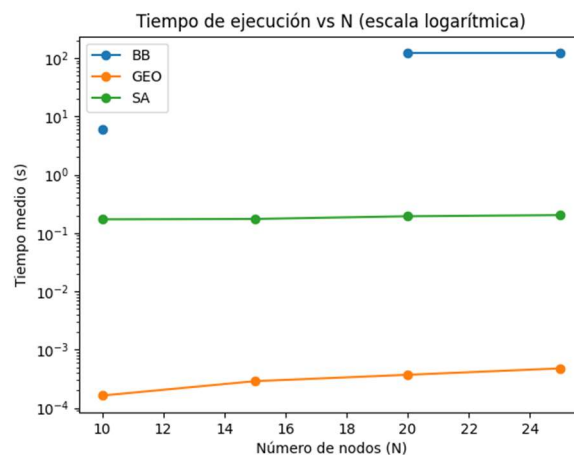
En la tabla 1 se resumen los resultados obtenidos para cada algoritmo e instancia, incluyendo tiempo de ejecución, distancia, riesgo y consumo de batería.

Cabe destacar que el comportamiento de Branch & Bound no es estrictamente monótono respecto al tamaño de la instancia, ya que depende fuertemente de la geometría concreta del problema y de la calidad de las cotas obtenidas durante la exploración del árbol de búsqueda. Por este motivo, en algunas instancias de mayor tamaño se obtienen soluciones parciales, mientras que en otras más pequeñas el algoritmo no alcanza a producir resultados dentro del límite temporal. Este efecto se analiza visualmente en las gráficas presentadas a continuación.

A continuación, se analizan los resultados obtenidos mediante representaciones gráficas.

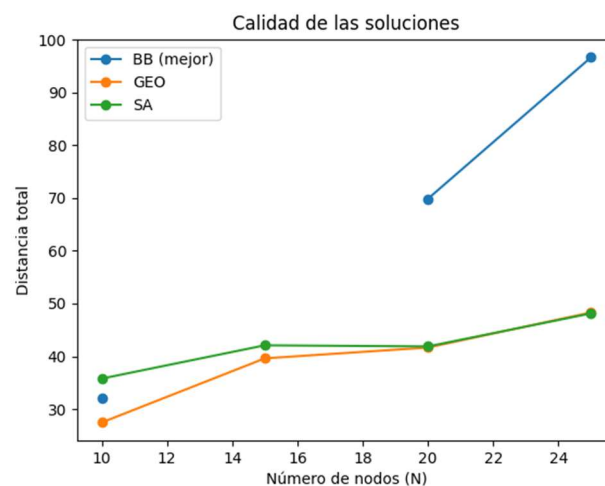
Análisis gráfico

Figura 1: Tiempo de ejecución vs N (escala logarítmica)



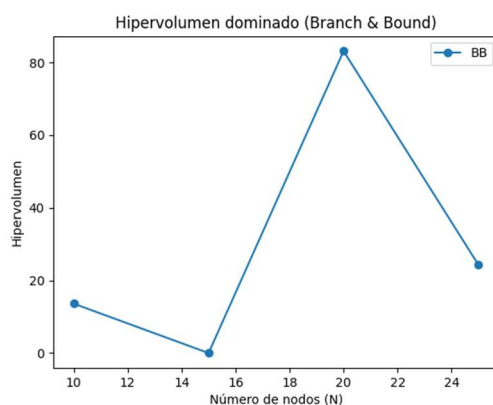
Como se observa en la figura 1 se aprecia claramente el crecimiento exponencial del tiempo de ejecución de Branch & Bound, frente al comportamiento prácticamente constante de la heurística geométrica y el crecimiento suave de Simulated Annealing. Esto evidencia la limitada escalabilidad de los métodos exactos frente a las heurísticas.

Figura 2: Calidad de las soluciones (distancia)



Distancia total obtenida por cada algoritmo para distintas instancias. Aunque Branch & Bound proporciona soluciones óptimas cuando finaliza, las heurísticas GEO y SA alcanzan soluciones de calidad similar o incluso superior en instancias grandes, donde el método exacto no puede explorar completamente el espacio de búsqueda.

Figura 3: Hipervolumen dominado (BB)

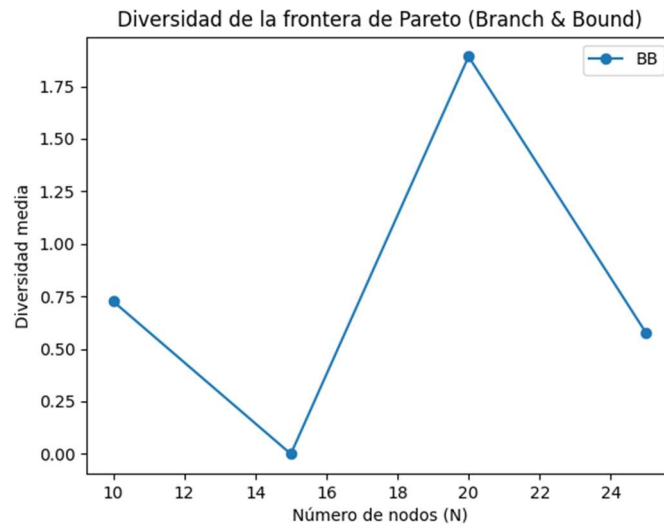


Hipervolumen dominado por la frontera de Pareto obtenida mediante Branch & Bound.

El hipervolumen cuantifica simultáneamente la calidad y extensión de la frontera de Pareto. A medida que aumenta el tamaño del problema, el valor del hipervolumen se vuelve más irregular debido a la obtención de fronteras parciales bajo restricciones temporales

El hipervolumen se ha calculado únicamente para el algoritmo Branch & Bound, ya que es el único que genera explícitamente una frontera de Pareto. Las heurísticas geométricas y de Simulated Annealing producen una única solución final y, por tanto, no permiten definir una frontera sobre la que calcular métricas de hipervolumen o diversidad.

Figura 4: Diversidad de la frontera



Diversidad media de la frontera de Pareto obtenida por Branch & Bound. La diversidad se ha medido como la distancia media entre soluciones consecutivas de la frontera. Valores elevados indican una mejor dispersión de soluciones, lo que resulta deseable en problemas multiobjetivo.

La diversidad de la frontera se ha calculado únicamente para el algoritmo Branch & Bound, dado que es el único que genera un conjunto de soluciones no dominadas que define una frontera de Pareto. Las heurísticas geométricas y de Simulated Annealing producen una única solución final, por lo que no resulta posible definir ni evaluar métricas de diversidad para estas técnicas.

Conclusiones

Los resultados muestran que Branch & Bound es adecuado para instancias pequeñas, donde permite obtener fronteras de Pareto exactas, pero presenta serias limitaciones de escalabilidad. Por el contrario, las heurísticas geométricas y de Simulated Annealing ofrecen soluciones de alta calidad en tiempos muy reducidos, siendo más apropiadas para instancias grandes.

El uso combinado de métodos exactos como referencia y heurísticas para la resolución práctica resulta una estrategia eficaz para problemas de planificación multiobjetivo con restricciones geométricas.

Enlace a github:

<https://github.com/Noa12-uni/Actividad-2.-Dise-o-an-lisis-y-comparaci-n-de-algoritmos.git>