Algoritmos Avanzados de Búsqueda y Optimización

Informe Obligatorio

Franco Filardi,

Joaquín Repetto

Índice

Índice **1**
Descripción del problema **2**
Motivación y relevancia **3**
Modelo formal **3**
Datos: 3
Variables: 3
Función objetivo: 3
Restricciones: 4
Criterio de aceptación: 4
Algoritmos **5**
Branch and Bound: 5
A*: 5
Experimentos **7**
Resultados: 7
Implementación en la vida real **9**
Conclusiones **9**
Futuras mejoras **10**
Referencias **11**

Descripción del problema

Un estudiante universitario debe planificar su jornada diaria considerando tiempo libre disponible, actividades con horarios específicos (por ejemplo, el gimnasio o la biblioteca solo abren dentro de ciertos intervalos), y tiempos de traslado entre ubicaciones (ya sea a pie, en transporte público o en auto).

Cada actividad tiene un valor de utilidad, un tiempo de duración, y un horario en el que puede realizarse.

El estudiante dispone de un tiempo total limitado (por ejemplo, 8 horas) y desea planificar la ruta óptima que le permita maximizar el aprovechamiento del día, seleccionando qué lugares visitar y en qué orden, sin exceder su tiempo disponible y teniendo en cuenta la disponibilidad de los lugares que desea visitar. A su vez, el estudiante podrá elegir además de estar maximizando la utilidad del dia, minimizar el tiempo diario que pasó durante la jornada en un medio de transporte

Este problema puede modelarse como un problema de planificación de tareas con ventanas de tiempo, donde cada actividad tiene un intervalo horario en el que puede realizarse y un valor asociado.

A su vez, los traslados entre actividades introducen un costo adicional (tiempo o distancia), haciendo necesario decidir qué conjunto de actividades realizar y en qué orden, para maximizar el valor total obtenido dentro del tiempo disponible del estudiante.

Motivación y relevancia

El problema representa una situación cotidiana de planificación eficiente: cómo un estudiante puede aprovechar al máximo su día dentro de limitaciones de tiempo, distancias y prioridades teniendo en cuenta ciertas restricciones..

Desde el punto de vista académico, el problema permite combinar técnicas exactas y heurísticas vistas en el curso, evaluando su rendimiento comparativo en términos de tiempo y calidad de las soluciones.

Modelo formal

Datos:

- V={1,2,...,n}: conjunto de posibles lugares a visitar (clases, biblioteca, comedor, gimnasio, supermercado, etc.).
- dij: distancia o tiempo de viaje entre el lugar i y el lugar j.
- vi: valor o utilidad asociada a visitar el lugar i.
- ti: tiempo necesario para realizar la actividad
- Tmax: tiempo total disponible del estudiante (por ejemplo, 8 horas).

Variables:

xij = 1 si el estudiante se desplaza directamente del lugar

yi = 1 si el estudiante decide visitar el lugar i, 0 en caso contrario.

Función objetivo:

Maximizar la utilidad de las recorriendo x ubicaciones en el tiempo disponible del alumno

$$Z = \sum_{i} v_i y_i - \alpha \sum_{i,j} d_{ij} x_{ij}$$

α: peso que penaliza el tiempo de transporte (por ejemplo, si se viaja en auto o en bus).

Restricciones:

1. Cada lugar visitado debe tener una salida y una llegada única.

$$\sum_j x_{ij} = y_i, \quad \sum_j x_{ji} = y_i$$

2. El tiempo total invertido no puede exceder el disponible:

$$\sum_i t_i y_i + \sum_{i,j} d_{ij} x_{ij} \leq T_{max}$$

3. Se deben eliminar subrutas para garantizar un recorrido válido.
4. El recorrido no tiene porque empezar y terminar en el mismo lugar
5. No se pueden recorrer ubicaciones que no estén disponibles al momento del recorrido
6. Horario disponible: si <= inicio(i) <= ei, donde si y ei son la apertura y cierre de la actividad. Si el estudiante llega fuera de ese rango, la actividad no puede realizarse.

Criterio de aceptación:

- El modelo es aceptable si las rutas generadas son factibles en la vida real.
- Debe existir una consistencia temporal: Las actividades deben ordenarse cronológicamente según su horario real
- Una solución heurística es aceptable si su valor total es cercano al óptimo.
- El modelo es aceptable si permite obtener resultados útiles en tiempos de ejecución adecuados.

Algoritmos

En nuestro caso, utilizaremos Branch and Bound y A*, adaptados al contexto del problema de planificación de tareas con ventanas de tiempo (Task Scheduling) y costos de traslado entre actividades.

Branch and Bound:

La idea de usar este algoritmo es explorar todas las posibles combinaciones y secuencias de actividades, pero descartando aquellas que no puedan superar la mejor solución encontrada hasta el momento.

A medida que se agregan actividades a la planificación parcial del estudiante, estas se incorporan al árbol de búsqueda, actualizando el valor acumulado y el tiempo total utilizado.

Mediante una cota superior basada en el valor máximo potencial restante, se podan las ramas que no puedan mejorar la solución actual, reduciendo significativamente el espacio de búsqueda.

A*:

El algoritmo A* nos permitirá encontrar una solución casi óptima de forma más rápida que Branch and Bound, utilizando una heurística que estima el valor máximo alcanzable desde el estado actual.

Como A* busca minimizar una función de evaluación y en nuestro caso queremos maximizar el valor total de las actividades realizadas, adaptamos la función de evaluación de la siguiente forma:

$$f(n) = -(v) + h(n)$$

Heurística utilizada:

Para estimar el valor máximo que aún es posible obtener desde un estado parcial, empleamos una heurística basada en el enfoque del Knapsack Fraccional, que permite construir una cota superior realista y admisible. Dado un estado n, calculamos el tiempo restante *Trem = Tmax - t(n)* y seleccionamos las actividades no visitadas ordenándolas por su densidad de valor (*va/da*) Luego se "llena" dicho tiempo agregando actividades completas mientras entren, y cuando se alcanza la primera actividad que no entra entera, se toma únicamente la fracción que cabe en el tiempo restante. Esto produce la siguiente heurística:

$$h(n) = -\left(\sum_{i=1}^{k-1} v_{a_i} \; + \; v_{a_k} \cdot \frac{T_{\text{rem}} - \sum_{j=1}^{k-1} d_{a_j}}{d_{a_k}}\right)$$

donde:

- *"vai" e*s valor de la actividad "*ai"*
- *"dai"* es su duración
- *"Trem"* el tiempo disponible desde el estado actual
- *"a1, … , am"* actividades no visitadas ordenadas por "*va/da*"
- *"k"* es la primera actividad cuya duración excede el tiempo restante

Cumpliendo esta heurística con las siguientes propiedades:

- Admisible: la heurística nunca sobreestima el valor real que puede alcanzarse, ya que únicamente permite fraccionar la primera actividad que no entra, tal como en el óptimo del knapsack fraccional.
- Consistente: el valor heurístico siempre disminuye de manera coherente cuando se avanza a un sucesor, manteniendo la propiedad "*h(n) <= c(n, n') + h(n')*"
- Dominancia: la propiedad de dominancia se evalúa únicamente comparando dos heurísticas distintas. Una heurística h2 domina a otra h1 si h2(n) >= h1(n) para todo estado n , siendo ambas admisibles. En nuestro caso, la heurística implementada domina a heurísticas más simples, como la heurística trivial h(n) = 0 o la heurística basada únicamente en contar actividades restantes. Al proporcionar una cota

superior más ajustada y específica del valor potencial futuro, nuestra heurística es más informada y típicamente expande menos nodos que heurísticas más débiles.

Experimentos

Para comparar ambos algoritmos se evaluaron 2 conjuntos de datos:

- Una instancia chica de 5 actividades, usada más que nada como prueba inicial para ver que todo funcione bien.
- Una instancia más grande de 10 actividades, donde se incluyeron casos más realistas con actividades como Desayuno, Gimnasio, Tiempo libre, etc.

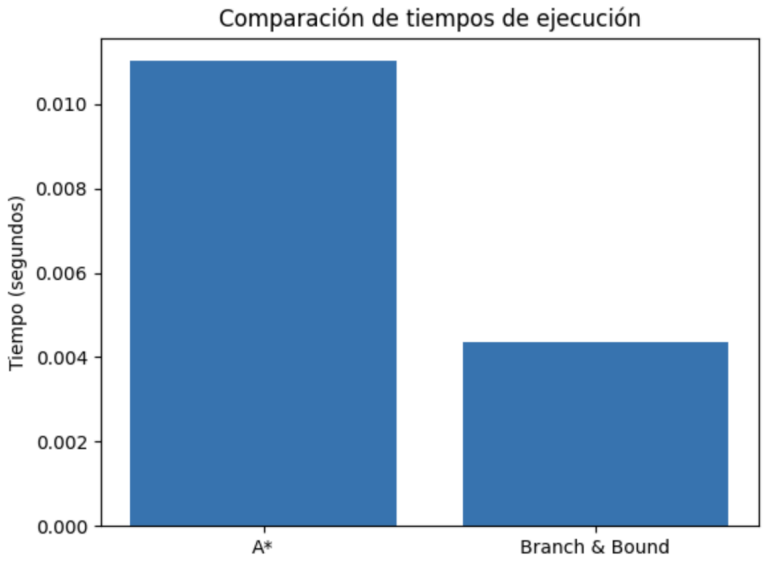
Estas instancias incluyen restricciones como ventanas horarias apretadas, tiempos variables de traslado y actividades mutuamente excluyentes por disponibilidad.

Resultados:

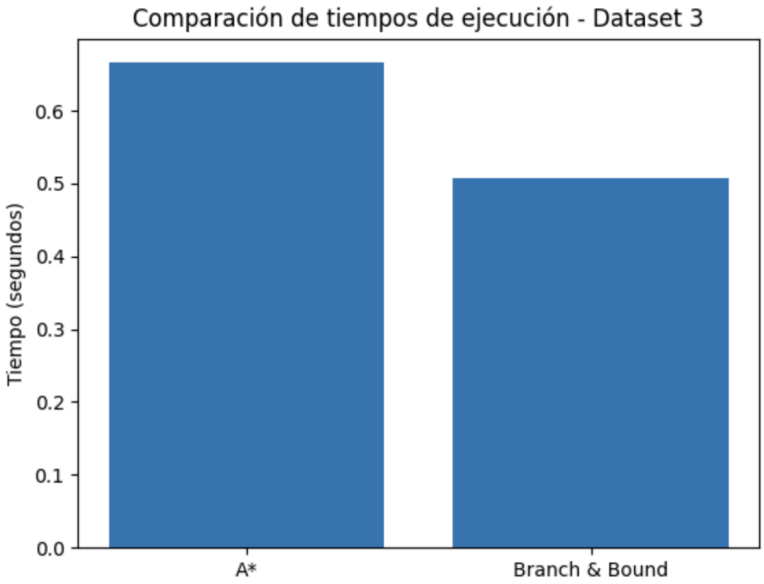
Al ejecutar ambos algoritmos utilizando la instancia grande de actividades, obtuvimos los siguientes resultados:

Algoritmo	Tiempo final	Valor total	Transporte	Objetivo
A*	20 hs	61	2.6 hs	60.35
Branch & Bound	20 hs	61	2.6 hs	60.35

Esto demuestra que ambos algoritmos lograron llegar a una misma solución óptima. La diferencia se vió reflejada en el tiempo que les llevó llegar a este resultado:



Como podemos ver, Branch & Bound logró un tiempo muchísimo mejor que A*, cuando nosotros pensamos que esto no iba a ser así. Tras analizar los resultados, nos dimos cuenta que las actividades definidas tienen horarios consecutivos y bastante rígidos, por lo que muy pocas combinaciones son realmente factibles. Esto favorece a Branch & Bound, por lo que implementamos un tercer dataset con muchas más posibles combinaciones para ver qué pasaba, y obtuvimos estos resultados:



En este caso, sigue ganando Branch & Bound, pero por mucha menos diferencia. Por ende, podemos concluir que, a mayor cantidad de combinaciones de actividades posibles, Branch & Bound se vuelve cada vez peor, mientras que A* comienza a aparecer como una mejor solución.

Implementación en la vida real

El modelo desarrollado y los algoritmos utilizados (Branch & Bound y A*) pueden aplicarse directamente a diversos escenarios del mundo real donde es necesario planificar actividades bajo restricciones de tiempo, disponibilidad y costos de desplazamiento. Entre las aplicaciones más relevantes se encuentran los siguientes.

- Aplicaciones de agenda inteligentes: Sistemas como Google Calendar, Microsoft Outlook o aplicaciones de productividad personal pueden beneficiarse de un motor de planificación que sugiera automáticamente la mejor distribución del tiempo diario.
- Aplicaciones de movilidad personal: Apps como Moovit, Citymapper o Waze podrían integrar este tipo de planificación para proponer itinerarios óptimos que no solo minimicen el tiempo de viaje y costos, sino que también maximicen el “valor” del día para el usuario

Conclusiones

Gracias a los experimentos, llegamos a varias conclusiones. La primera de ellas es que ambos algoritmos logran llegar a una solución óptima, lo que valida la correctitud del modelo.

Además, observamos que Branch & Bound resulta mejor cuando las actividades deben realizarse una a continuación de la otra, o cuando el horario para realizarlas es muy rígido, ya que esto provoca que existan pocas posibles combinaciones.

Sin embargo, cuando estas posibles combinaciones comienzan a aumentar, la diferencia entre ambos algoritmos comienza a ser cada vez menor, por lo que concluimos que, a la larga, A* será una mejor solución debido a su menor costo computacional cuando hay muchas posibles combinaciones.

Futuras mejoras

- A futuro, sería bueno implementar algunas nuevas funcionalidades, tales como:
  Incorporar el costo económico del transporte como parte de la función objetivo, permitiendo optimizar no solo tiempo sino también gasto monetario.
- Modelar el cansancio o nivel de energía del estudiante como una restricción adicional, limitando la cantidad o el tipo de actividades que pueden realizarse consecutivamente.
- Introducir incertidumbre en el modelo, integrando factores como demoras, variabilidad en los tiempos de viaje o condiciones de tráfico reales, mediante técnicas estocásticas o simulación.
- Extender el modelo hacia una planificación semanal, permitiendo que la optimización contemple dependencias entre días, descansos y actividades recurrentes.

Referencias

- Wikipedia contributors. (2025). Iterative deepening A*. In Wikipedia. Retrieved August 27, 2025, from https://en.wikipedia.org/wiki/Iterative_deepening_A%2A
- Boyd, S. (2018). Branch and Bound — Lecture Slides. Stanford EE364b: Convex Optimization II. Retrieved September 17, 2025, from: https://web.stanford.edu/class/ee364b/lectures/bb_slides.pdf
- Boyd, S. (2018). Branch and Bound — Lecture Slides. Stanford EE364b: Convex Optimization II. Retrieved September 17, 2025, from: https://web.stanford.edu/class/ee364b/lectures/bb_slides.pdf