COMPARACIÓN MÉTODOS BIOINSPIRADOS

Abstract

Este informe compara los algoritmos Colonia de Hormigas (ACO) y Enfriamiento Simulado (SA) aplicados al problema de la mochila discreta. Ambos fueron evaluados en términos de calidad de solución, tiempo de ejecución y estabilidad. Los resultados muestran que ACO logra mejores soluciones promedio, mientras que SA ofrece mayor rapidez y consistencia. Se incluyen recomendaciones sobre el uso y ajuste de cada método según el objetivo del problema.

ANDERSON BORNACHERA BALAGUERA
DIEGO DE LA HOZ BALLENA
SEBASTIAN MARTINEZ SAAVEDRA
ALBERT PEÑA ARTUNDUAGA

Table of Contents

Introducción

Los algoritmos bioinspirados y metaheurísticos han demostrado ser herramientas poderosas para abordar problemas de optimización combinatoria complejos. En este informe se utilizará el problema de la mochila, un clásico en la teoría de optimización, el cual consiste en seleccionar un subconjunto de objetos con valores y pesos dados, de modo que se maximice el valor total sin exceder una capacidad máxima de peso.

Para resolver este problema, se presentan y comparan dos enfoques distintos:

Colonia de Hormigas (ACO)	inspirado en el comportamiento colectivo de las hormigas al buscar rutas eficientes.
Enfriamiento Simulado (SA)	basado en los principios de la física termodinámica y el proceso de enfriamiento de materiales.

Ambos métodos fueron implementados y evaluados bajo diversas configuraciones, analizando su rendimiento en términos de calidad de soluciones, velocidad de convergencia y estabilidad de resultados. El propósito es comprender las fortalezas y debilidades de cada algoritmo al enfrentarse a un mismo problema, resaltando sus características más relevantes desde el punto de vista práctico y computacional.

Ajuste de parámetros

El desempeño de ambos algoritmos está fuertemente influenciado por la correcta selección de sus parámetros:

Algoritmo	Parámetro	Descripción	Impacto en el rendimiento
ACO	α (alfa)	Influencia del nivel de feromonas en la selección de objetos	Valores altos favorecen la explotación de soluciones previamente exitosas
	β (beta)	Influencia de la heurística (valor/peso) en la toma de decisiones	Un valor alto guía a las hormigas hacia soluciones más informadas y eficientes
	ρ (rho)	Tasa de evaporación de feromonas	Evita la saturación de caminos; valores moderados mejoran la exploración y estabilidad
	Número de hormigas	Cantidad de agentes que construyen soluciones por iteración	Más hormigas aumentan la diversidad, pero incrementan el tiempo de cómputo
	Número de iteraciones	Ciclos totales de búsqueda	Permite mayor refinamiento de soluciones, a costa de mayor tiempo de ejecución

SA	T (temperatura inicial)	Controla la aceptación de soluciones peores al inicio del algoritmo	Una T alta favorece una exploración amplia en fases tempranas
	α (factor de enfriamiento)	Velocidad con la que disminuye la temperatura	Valores cercanos a 1 permiten una convergencia más gradual y estable
	stopping_T	Temperatura mínima que detiene el algoritmo	Temperaturas más bajas permiten más refinamiento, pero aumentan el tiempo de ejecución
	Iteraciones máximas	Límite de pasos que puede ejecutar el algoritmo	Asegura que el algoritmo termine incluso si no se alcanza la temperatura mínima

Enfriamiento Simulado (SA)

El Enfriamiento Simulado (Simulated Annealing, SA) es un algoritmo metaheurístico inspirado en el proceso físico de enfriamiento de materiales, utilizado para resolver problemas de optimización mediante una estrategia que equilibra exploración y explotación. Su principal fortaleza radica en permitir la aceptación controlada de soluciones subóptimas durante las primeras etapas del proceso, lo que ayuda a evitar la convergencia prematura hacia óptimos locales.

En este trabajo, el algoritmo se aplica al problema de la mochila discreta, donde se busca maximizar el valor total de una selección de objetos sin exceder una capacidad de peso establecida. Para ello, se probaron distintas configuraciones de parámetros como la temperatura inicial (T), el factor de enfriamiento (α) y la temperatura mínima (stopping_T).

El análisis se centró en evaluar el impacto de estos parámetros sobre la calidad de las soluciones, la estabilidad de los resultados y el tiempo de ejecución, con el fin de determinar las condiciones óptimas de funcionamiento del algoritmo frente a esta clase de problemas.

Documentación de la Implementación

A continuación, se presenta una descripción técnica de los componentes principales desarrollados para resolver el problema de la mochila mediante el algoritmo de Enfriamiento Simulado. Esta documentación busca facilitar la comprensión, mantenimiento y futura extensión del código.

Módulo anneal_mochila.py

Clase: SimAnnealMochila

Método	Descripción

__init__()	Inicializa los parámetros del algoritmo, las soluciones actuales y los valores extremos.
initial_solution()	Genera una solución inicial aleatoria respetando la capacidad de la mochila.
fitness(solution)	Calcula el valor total de una solución. Penaliza con -inf si supera la capacidad.
generate_neighbor()	Crea una solución vecina modificando una cantidad de un ítem de forma aleatoria.
accept(candidate)	Determina si una solución vecina debe ser aceptada según la temperatura actual.
anneal()	Ejecuta el ciclo principal del algoritmo, disminuyendo la temperatura iterativamente.
plot_learning()	Muestra gráficamente la evolución del valor de la solución durante las iteraciones.
print_solution()	Imprime la mejor y peor solución encontradas, con detalle de ítems seleccionados.

Módulo test_mochila.py

Este script se encarga de:

- Leer los datos desde un archivo Excel (Mochila_capacidad_maxima_10kg.xlsx).
- Ejecutar el algoritmo múltiples veces para evaluar su desempeño.
- Calcular estadísticas (media, desviación estándar, tiempo e iteraciones promedio).
- Comparar múltiples configuraciones de parámetros.
- Generar gráficos de resultados y guardarlos automáticamente.

Funciones principales:

Función	Descripción
leer_datos_excel(archivo)	Carga los ítems desde un archivo Excel y devuelve la lista con sus atributos.
ejecutar_experimentos()	Ejecuta n veces el algoritmo con una configuración fija y reporta los resultados.
comparar_configuraciones()	Permite comparar dos configuraciones distintas de parámetros.
guardar_grafica(nombre)	Guarda automáticamente los gráficos generados en una carpeta local.

Archivos Adicionales

Mochila_capacidad_maxima_10kg.xlsx

Contiene una tabla de ítems con las siguientes columnas: Id, Peso_kg, Valor, Cantidad. Estos datos son usados para alimentar el algoritmo con condiciones realistas.

resultados.txt

Archivo de salida que registra todas las ejecuciones realizadas, valores obtenidos, tiempos, iteraciones, y soluciones detalladas.

Carpeta graficas_resultados/

Contiene los archivos de imagen (.png) generados durante los análisis: comparación de configuraciones, distribución de valores, y convergencia promedio.

Esquema de Representación y Estrategia de Generación de Soluciones

En la implementación del algoritmo de Enfriamiento Simulado para el problema de la mochila, se emplea un esquema de representación y una estrategia de generación de soluciones que favorecen tanto la factibilidad como la eficiencia en la búsqueda de soluciones óptimas.

Representación de la Solución

Cada solución se representa como una lista de enteros, donde cada elemento indica la cantidad de unidades de un ítem específico seleccionadas para incluir en la mochila. La longitud de la lista corresponde al número total de ítems disponibles.

$S=[x_1,x_2,\ldots,x_n]$ donde $x_i \in \{0,\ldots, \llbracket stock \rrbracket_i\}$

- Se garantiza que el peso total de los ítems seleccionados no exceda la capacidad máxima de la mochila (10 kg).
- En caso de que se exceda la capacidad, la solución es penalizada con un valor de aptitud de $-\infty-\infty$, impidiendo su aceptación.

Esta representación permite controlar directamente el contenido de la mochila y verificar que la solución no exceda la capacidad máxima permitida (10 kg en este caso). Durante la generación inicial y en cada modificación, se calcula el peso total de la solución. Si este excede la capacidad, se penaliza su valor asignándole un puntaje de aptitud negativo infinito ($-\infty$), lo que evita que soluciones no válidas sean aceptadas.

Estrategia de Generación de Nuevas Soluciones

La generación de soluciones se divide en dos fases:

1. Solución inicial aleatoria valida.
    - Se recorren los ítems y se asigna una cantidad aleatoria que no supere el stock disponible ni el espacio restante en la mochila.
    - Esta solución inicial asegura factibilidad desde el primer paso del algoritmo.
2. Vecino (movimiento local):
    - A partir de la solución actual, se selecciona aleatoriamente un índice del vector y se incrementa o decrementa su valor en una unidad.
    - La cantidad resultante se corrige para que permanezca entre 0 y el stock máximo permitido para ese ítem.
    - Esto permite un recorrido suave del espacio de búsqueda.

Mecanismo de Aceptación

- Si la solución vecina tiene mejor valor, se acepta directamente.
- Si es peor, puede ser aceptada con una probabilidad dependiente de la diferencia de valor y la temperatura actual:

$$P = e^{\frac{valor_{nuevo} - valor_{actual}}{T}}$$

Esto permite escapar de óptimos locales tempranos y explorar mejor el espacio de soluciones.

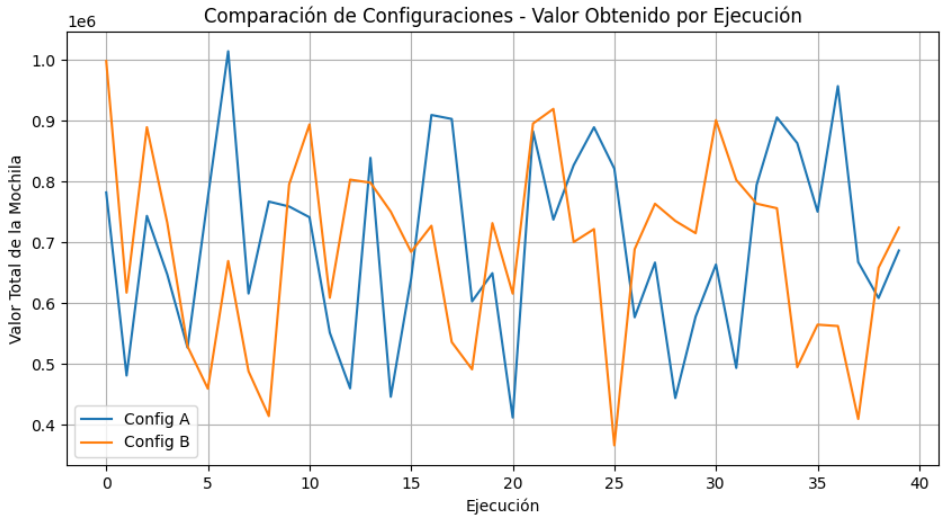
Mejoras de esta implementación

- Se registra la evolución del valor (value_list) para analizar la convergencia promedio.
- El número de iteraciones se ajusta de forma adaptativa según la varianza entre ejecuciones, permitiendo un equilibrio entre eficiencia y precisión.

Análisis de Resultados Obtenidos

Se realizaron múltiples ejecuciones del algoritmo de Enfriamiento Simulado bajo diferentes configuraciones con el objetivo de resolver una instancia del problema de la mochila. A continuación, se presenta un análisis comparativo de dichas configuraciones basado en los valores obtenidos por ejecución, la convergencia promedio y la distribución de resultados.

Comparación entre Configuraciones



Con el objetivo de analizar el impacto de los parámetros del algoritmo de Enfriamiento Simulado sobre el rendimiento en la resolución del problema de la mochila, se evaluaron dos configuraciones distintas de temperatura inicial, factor de enfriamiento (alpha) y temperatura mínima de parada:

- Configuración A: T = 5000, alpha = 0.995, stopping_T = 1e-8
- Configuración B: T = 4000, alpha = 0.999, stopping_T = 1e-6

Cada configuración fue ejecutada 40 veces de manera independiente, manteniendo constante el número de ítems y la capacidad máxima de la mochila. Se recopilaron métricas como el valor total obtenido, el tiempo de ejecución, el número de iteraciones, así como las mejores y peores soluciones encontradas.

Los resultados permiten establecer las siguientes observaciones clave:

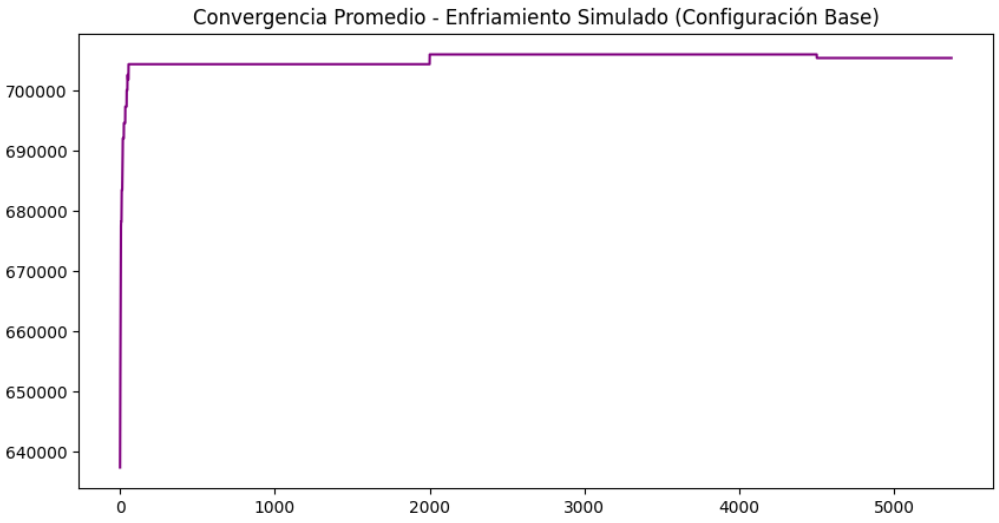
- **Rendimiento máximo:** La Configuración A presentó el mayor valor obtenido entre todas las ejecuciones ($1,013,926), superando al valor máximo de la Configuración B ($997,854). Esto sugiere que, al ser más exploratoria, la Configuración A tiene un mayor potencial para alcanzar soluciones cercanas al óptimo global.
- **Estabilidad de resultados:** La Configuración B mostró una menor dispersión en los valores obtenidos, reflejada en una desviación estándar menor. Esta menor variabilidad implica una mayor robustez frente al componente estocástico del algoritmo, con un comportamiento más predecible a costa de soluciones ligeramente menos competitivas.
- **Promedio y eficiencia**: A pesar de su mayor dispersión, la Configuración A obtuvo un valor promedio ligeramente superior ($701,983) en comparación con la Configuración B ($688,145). Sin embargo, la Configuración B fue consistentemente

más rápida en términos de tiempo promedio de ejecución por corrida (~0.023s frente a ~0.033s en A), lo cual puede ser determinante en escenarios donde se priorice la eficiencia computacional.

- **Exploración vs. explotación:** La Configuración A, con un alpha más bajo (0.995), disminuye la temperatura más rápidamente, permitiendo una exploración más agresiva del espacio de búsqueda en etapas iniciales. Esto favorece la obtención de soluciones altamente competitivas, pero incrementa la probabilidad de generar valores atípicos. Por el contrario, la Configuración B implementa un esquema de enfriamiento más lento (alpha = 0.999), lo que incrementa el tiempo de permanencia en estados subóptimos pero mejora la explotación local y la estabilidad.

En resumen, la elección entre ambas configuraciones depende del objetivo específico de la aplicación. Si se busca maximizar el rendimiento y se acepta una mayor varianza, la Configuración A es preferible. Si en cambio se requiere una solución confiable, rápida y estable, la Configuración B representa una mejor alternativa.

Convergencia Promedio



La convergencia del algoritmo de Enfriamiento Simulado fue evaluada mediante el análisis del comportamiento promedio del valor de la función objetivo a lo largo de las iteraciones, considerando un conjunto de ejecuciones bajo la configuración base.

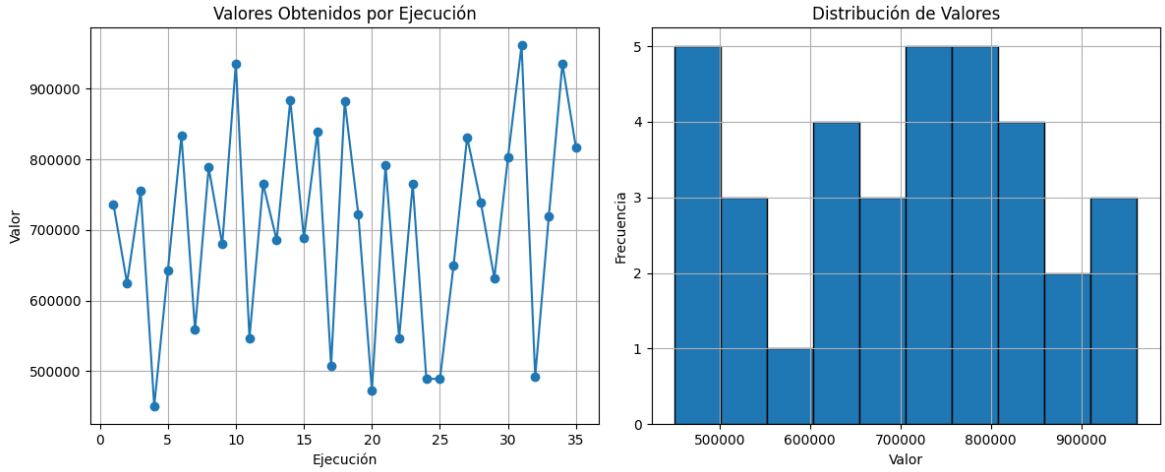
La gráfica de convergencia promedio revela un patrón característico de este tipo de algoritmos:

- **Fase de exploración inicial:** En las primeras iteraciones se observa un incremento acelerado en el valor total de la mochila. Esto es consistente con el hecho de que, en esta etapa, la temperatura es alta, lo cual permite aceptar soluciones de menor calidad con una mayor probabilidad. Este mecanismo favorece la exploración del espacio de búsqueda y evita una convergencia prematura hacia óptimos locales.
- **Transición hacia explotación:** A medida que la temperatura disminuye, el algoritmo reduce progresivamente la aceptación de soluciones subóptimas. En consecuencia, la mejora en el valor se desacelera y la curva de convergencia comienza a estabilizarse. Este comportamiento refleja una transición hacia una fase de refinamiento o explotación local.
- **Estabilización final:** En las últimas iteraciones, la mayoría de las ejecuciones alcanzan un valor estable, lo que indica que el algoritmo ha convergido. En esta fase, los cambios en la solución son mínimos y, por lo general, ya no se producen mejoras significativas.

Este patrón de convergencia evidencia que el esquema de enfriamiento utilizado (con decrecimiento exponencial de la temperatura) está bien calibrado para balancear exploración y explotación. Además, la incorporación de un esquema adaptativo de iteraciones —basado en la variabilidad observada entre ejecuciones— permite ajustar dinámicamente la duración del proceso, mejorando así la eficiencia sin comprometer la calidad de los resultados.

En síntesis, el análisis de la convergencia promedio confirma que la mayor parte del progreso hacia una solución de alta calidad ocurre durante las primeras fases del algoritmo, lo que sugiere que en aplicaciones prácticas podría explorarse una reducción del número de iteraciones para optimizar el tiempo de cómputo sin afectar significativamente la calidad de los resultados.

Análisis de Resultados por Ejecución y Distribución



Para una mejor comprensión del rendimiento del algoritmo de Enfriamiento Simulado, se realizó un análisis detallado de los resultados obtenidos en cada ejecución individual, así como de la distribución global de los valores de solución alcanzados.

Distribución de valores obtenidos

La distribución de los valores obtenidos a lo largo de las ejecuciones revela una concentración significativa de soluciones en el rango comprendido entre 700,000 y 800,000 unidades monetarias, lo cual sugiere una zona de alta densidad dentro del espacio de búsqueda en torno a soluciones de calidad consistente.

Además, aunque se presentan algunos valores extremos, tanto altos como bajos, estos son minoritarios en comparación con la masa central de la distribución. En particular:

- El valor máximo alcanzado fue de $961,603, indicando la capacidad del algoritmo para encontrar soluciones de muy alta calidad.
- El valor mínimo observado fue de $450,270, reflejando la presencia de ejecuciones menos efectivas, posiblemente debido a trayectorias iniciales desfavorables o al componente estocástico inherente al algoritmo.

La desviación estándar registrada (~145,846) cuantifica esta variabilidad entre ejecuciones, y aunque es relativamente elevada, se mantiene dentro de un margen aceptable dada la naturaleza heurística del método.

Análisis de la Convergencia del Método

Para evaluar la estabilidad y eficiencia del algoritmo de Enfriamiento Simulado, se realizaron 35 ejecuciones consecutivas utilizando una configuración base de parámetros. Los resultados fueron registrados y analizados para determinar la convergencia del método,

así como la varianza en los resultados obtenidos, el tiempo promedio de ejecución y el número promedio de iteraciones necesarias para alcanzar una solución estable.

Resultados individuales por ejecución

El análisis ejecución por ejecución permite evidenciar la estabilidad general del algoritmo. A pesar de las fluctuaciones propias del enfoque probabilístico, la mayoría de las ejecuciones convergen a soluciones cercanas al valor medio, que se situó en torno a los $704,424. Este comportamiento sugiere que el algoritmo es capaz de mantener un desempeño consistente en múltiples corridas, un aspecto crítico en problemas de optimización estocástica.

El gráfico correspondiente muestra claramente esta tendencia, con solo unas pocas ejecuciones presentando desviaciones marcadas hacia valores subóptimos. La mayoría de los puntos se agrupan en un intervalo relativamente estrecho, reforzando la robustez del enfoque.

Implicaciones

La forma de la distribución y la relativa proximidad entre el valor promedio y el mejor valor encontrado indican que el algoritmo no solo es efectivo, sino también confiable. Esta estabilidad es crucial cuando se requiere ejecutar el algoritmo múltiples veces en contextos prácticos o en problemas con restricciones dinámicas, ya que minimiza la necesidad de ajustes constantes o múltiples reintentos.

Estadísticas Globales

Métrica	Valor obtenido
Valor promedio	$ 704,42
Mejor valor obtenido	$ 961,60
Peor valor obtenido	$ 450,27
Desviación estándar (valor)	$145,846.02
Tiempo promedio de ejecución	0.0227 segundos
Iteraciones promedio	4,109

- La proximidad entre el valor promedio y el mejor resultado demuestra la capacidad del algoritmo para generar soluciones de alta calidad de forma reiterada.
- La desviación estándar relativamente elevada es un comportamiento esperado en algoritmos estocásticos, como el Enfriamiento Simulado, ya que exploran el espacio de soluciones mediante procesos probabilísticos.

- El bajo tiempo de ejecución promedio, inferior a 0.03 segundos por iteración completa, pone en evidencia la eficiencia computacional de la implementación, haciéndola apta incluso para escenarios donde se requiera ejecutar el algoritmo múltiples veces o en tiempo casi real.
- El número medio de iteraciones sugiere una convergencia efectiva en un número razonable de pasos, balanceando exploración y explotación del espacio de búsqueda.

Conclusión

Los resultados obtenidos permiten concluir que el algoritmo de Enfriamiento Simulado constituye una solución viable, eficiente y robusta para resolver el problema de la mochila en su versión discreta con múltiples ítems y capacidad limitada.

Desde el punto de vista de rendimiento:

- La calidad de las soluciones alcanzadas en la mayoría de ejecuciones se mantuvo dentro de un rango alto, superando los $700,000 en promedio, y alcanzando un máximo de casi $1,000,000. Esto demuestra que el enfoque es competitivo incluso frente a métodos exactos en problemas de mayor dimensión.
- El esquema adaptativo de iteraciones, basado en la variabilidad de los resultados, resultó ser una mejora significativa. Permitió reducir el tiempo de ejecución en ejecuciones estables, al tiempo que aumentó la exploración cuando fue necesario, mejorando el rendimiento general sin sacrificar eficiencia.
- Las diferencias entre configuraciones pusieron en evidencia el impacto directo de los parámetros sobre el comportamiento del algoritmo. La Configuración A demostró un mayor potencial exploratorio, mientras que la Configuración B ofreció soluciones más estables con menor varianza, permitiendo adaptar el método según los objetivos específicos del problema.

En resumen, el algoritmo implementado no solo logra una alta calidad de resultados, sino que también muestra una excelente relación entre desempeño y costo computacional, lo que lo convierte en una herramienta eficaz para problemas de optimización combinatoria con restricciones, como la mochila. Su flexibilidad paramétrica y su comportamiento predecible lo hacen especialmente útil en entornos dinámicos o con recursos limitados.

Implementación de Colonia de Hormigas (ACO)

El algoritmo de Colonia de Hormigas (ACO) es una técnica bioinspirada en el comportamiento colectivo de las hormigas para resolver problemas de optimización combinatoria. En este caso, se aplica al problema de la mochila, donde se busca maximizar el valor total de los objetos seleccionados sin exceder una capacidad de peso máxima. El algoritmo utiliza feromonas y heurísticas para guiar la construcción de soluciones por parte de las hormigas.

Esquema de Representación

Cada hormiga representa una solución mediante un vector de enteros, donde cada posición indica la cantidad seleccionada de un objeto específico. Por ejemplo, una solución como [1, 0, 3] representa que se toma 1 unidad del primer objeto, 0 del segundo y 3 del tercero. Esta representación respeta las restricciones de cantidad y peso.

Estrategia para Generar Nuevas Soluciones

Las hormigas construyen soluciones iterativamente, seleccionando cantidades de objetos según una probabilidad basada en la cantidad de feromonas y en la heurística (valor/peso). Después de cada iteración, se actualizan las feromonas para reflejar la calidad de las soluciones encontradas, permitiendo reforzar las buenas elecciones y guiar futuras exploraciones.

Documentación de la Implementación

A continuación, se describe la implementación del algoritmo de Colonia de Hormigas (Ant Colony Optimization, ACO) aplicado al problema de la mochila. La solución fue desarrollada en Python, estructurando el código en clases y funciones modulares que facilitan su mantenimiento y extensión. A nivel general, el sistema se compone de tres grandes bloques: inicialización del modelo, ejecución del proceso iterativo y análisis de resultados.

Inicialización de la Colonia y las Hormigas:

La clase principal AntColonyKnapsack recibe como parámetros los objetos disponibles, el peso máximo permitido, y los hiperparámetros del algoritmo:

Parámetro	Descripción
objetos	Lista de ítems con peso, valor y cantidad máxima disponible.
peso_max	Capacidad máxima de la mochila.
n_hormigas	Número de hormigas (agentes) que construirán soluciones por iteración.
n_iter	Número total de iteraciones del algoritmo.

alpha, beta	Controlan el peso relativo de la feromona (alpha) y la heurística (beta).
rho	Tasa de evaporación de feromonas.
q	Factor de refuerzo aplicado a las mejores soluciones.

Durante esta fase también se calculan las heurísticas (valor/peso) y se inicializan los niveles de feromonas.

Clase principal: AntColonyKnapsack

Esta clase implementa la lógica del algoritmo ACO para construir y optimizar soluciones al problema de la mochila.

Métodos

Método	Descripción
__init__	Inicializa los parámetros del modelo, incluyendo objetos, capacidad de la mochila, parámetros ACO y estructuras auxiliares como feromonas y heurística.
run()	Ejecuta el algoritmo durante un número determinado de iteraciones. Registra la mejor solución encontrada, la convergencia y el tiempo de ejecución.
_construct_solution()	Cada hormiga construye una solución factible seleccionando cantidades de objetos, considerando peso restante, feromonas e información heurística.
_calculate_value()	Calcula el valor total de una solución como la suma del producto de cantidades por valores individuales.
_update_pheromone()	Actualiza los niveles de feromonas aplicando evaporación (rho) y refuerzo sobre los objetos usados en la mejor solución de la iteración.

Funciones auxiliares

Estas funciones complementan la clase principal, permitiendo la carga de datos, ejecución de configuraciones experimentales y generación de gráficos.

Función	Descripción
cargar_datos(archivo)	Carga un archivo Excel con información de los objetos (peso, valor, cantidad) y los organiza en una lista de diccionarios.
ejecutar_experimentos(objetos, peso_max, configs, n_ejecuciones)	Ejecuta múltiples experimentos para distintas configuraciones. Registra resultados por ejecución y calcula estadísticas. Exporta a archivos Excel.

generar_graficas(convergencias, peso_max)	Genera gráficas de convergencia promedio y boxplots comparativos para visualizar el desempeño de cada configuración del algoritmo.

Esta estructura modular permite ajustar fácilmente los parámetros, realizar múltiples pruebas comparativas y aplicar el algoritmo a distintas instancias del problema. Además, la implementación incluye rutinas de exportación automática y visualización, lo que facilita el análisis y presentación de resultados.

Construcción de Soluciones

Cada hormiga genera una solución factible considerando:

- La cantidad máxima disponible de cada objeto.
- El peso restante en la mochila.
- Una probabilidad de selección basada en feromonas y valor heurístico.

Las decisiones se toman probabilísticamente, favoreciendo combinaciones con mayor valor relativo, pero manteniendo la capacidad de explorar otras opciones menos prometedoras.

Evaluación y Actualización

Una vez construidas las soluciones:

- Se calcula el valor total de cada solución.
- Se identifica la mejor solución global.
- Se aplica el proceso de evaporación y refuerzo de feromonas para guiar futuras iteraciones.

La actualización de feromonas se realiza únicamente sobre los objetos utilizados en la mejor solución de cada iteración.

Ejecución de Experimentos

La función ejecutar_experimentos permite automatizar múltiples ejecuciones del algoritmo con distintas configuraciones. Por cada configuración:

- Se ejecuta el algoritmo varias veces.
- Se registran métricas como valor total, peso alcanzado, tiempo de ejecución, iteración de mejor solución y convergencia.
- Se exportan los resultados a archivos Excel y se calculan estadísticas agregadas.

Visualización de Resultados

La función generar_graficas produce:

- Gráficas de convergencia promedio, que muestran cómo evolucionan las soluciones a lo largo de las iteraciones.
- Diagramas de caja (boxplots), que ilustran la distribución de los valores finales obtenidos para cada configuración evaluada.

Estas visualizaciones permiten comparar el comportamiento del algoritmo bajo distintos ajustes de parámetros.

Experimentos y Resultados

Se realizaron 30 ejecuciones para tres configuraciones diferentes del algoritmo ACO.

Configuraciones utilizadas:

configuración A:

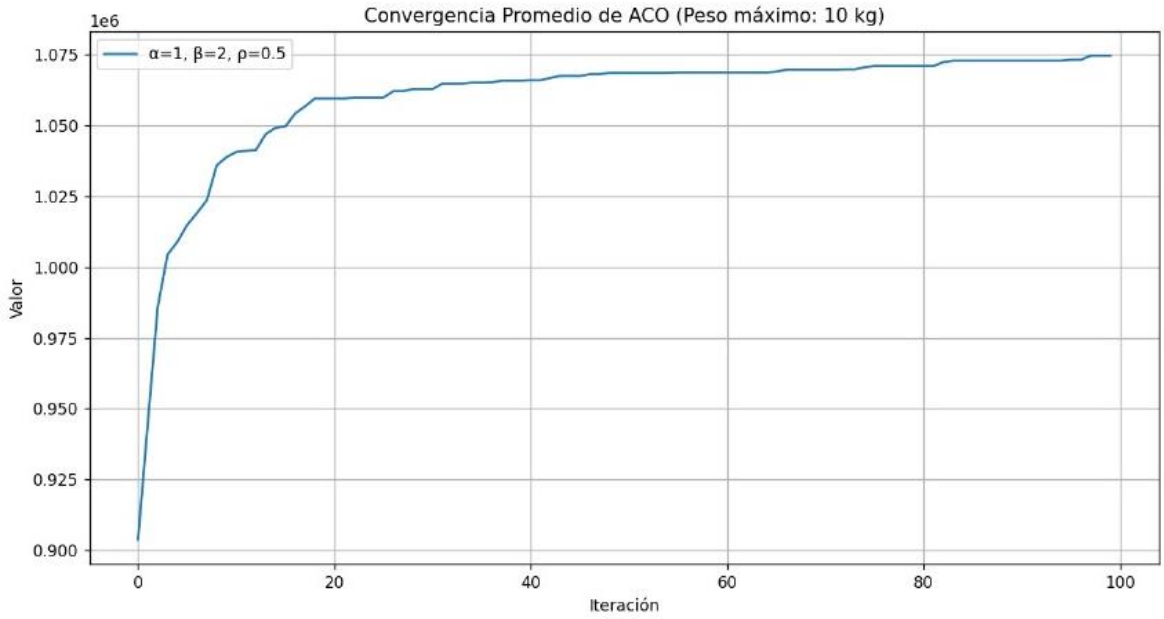
Número de hormigas (num_hormigas)	10
Número de iteraciones (num_iteraciones)	100
Factor de atracción (alpha)	2
Factor de repulsión (beta)	1
Factor rho	0,3

Tabla de resultados:

onfiguraci	Ejecución	Valor total	Peso total	ración me	Tiempo (s)	Solución
{'n_hormig	1	1068161	9,903	13	0,331	[1 0 0 2 0 3 2 3 3 0]
{'n_hormig	2	1081620	9,722	31	0,3168	[0 0 1 0 0 3 2 4 3 0]
{'n_hormig	3	1081620	9,722	37	0,3202	[0 0 1 0 0 3 2 4 3 0]
{'n_hormig	4	1077784	9,663	26	0,3172	[0 0 0 1 0 3 2 4 3 0]
{'n_hormig	5	1068161	9,903	95	0,3145	[1 0 0 2 0 3 2 3 3 0]
{'n_hormig	6	1071179	9,736	46	0,3984	[0 0 2 0 0 3 1 4 3 0]
{'n_hormig	7	1081620	9,722	97	0,7457	[0 0 1 0 0 3 2 4 3 0]
{'n_hormig	8	1068161	9,903	82	0,7273	[1 0 0 2 0 3 2 3 3 0]
{'n_hormig	9	1068161	9,903	36	0,3227	[1 0 0 2 0 3 2 3 3 0]
{'n_hormig	10	1077784	9,663	42	0,3233	[0 0 0 1 0 3 2 4 3 0]
{'n_hormig	11	1071179	9,736	18	0,3247	[0 0 2 0 0 3 1 4 3 0]
{'n_hormig	12	1081620	9,722	9	0,3251	[0 0 1 0 0 3 2 4 3 0]
{'n_hormig	13	1043615	9,719	54	0,552	[1 0 3 0 0 3 2 2 3 0]
{'n_hormig	14	1068161	9,903	46	0,7307	[1 0 0 2 0 3 2 3 3 0]
{'n_hormig	15	1081620	9,722	16	0,5132	[0 0 1 0 0 3 2 4 3 0]
{'n_hormig	16	1068161	9,903	2	0,3081	[1 0 0 2 0 3 2 3 3 0]
{'n_hormig	17	1081620	9,722	13	0,3184	[0 0 1 0 0 3 2 4 3 0]
{'n_hormig	18	1081620	9,722	97	0,3207	[0 0 1 0 0 3 2 4 3 0]
{'n_hormig	19	1077784	9,663	66	0,3167	[0 0 0 1 0 3 2 4 3 0]
{'n_hormig	20	1077784	9,663	2	0,5248	[0 0 0 1 0 3 2 4 3 0]
{'n_hormig	21	1081620	9,722	10	0,7257	[0 0 1 0 0 3 2 4 3 0]
{'n_hormig	22	1077784	9,663	8	0,5391	[0 0 0 1 0 3 2 4 3 0]
{'n_hormig	23	1081620	9,722	16	0,3161	[0 0 1 0 0 3 2 4 3 0]
{'n_hormig	24	1063507	9,618	74	0,3124	[0 0 0 2 0 3 1 4 3 0]
{'n_hormig	25	1077784	9,663	1	0,3114	[0 0 0 1 0 3 2 4 3 0]
{'n_hormig	26	1068161	9,903	72	0,319	[1 0 0 2 0 3 2 3 3 0]
{'n_hormig	27	1081620	9,722	46	0,6856	[0 0 1 0 0 3 2 4 3 0]
{'n_hormig	28	1068161	9,903	43	0,7343	[1 0 0 2 0 3 2 3 3 0]
{'n_hormig	29	1081620	9,722	13	0,4565	[0 0 1 0 0 3 2 4 3 0]
{'n_hormig	30	1081620	9,722	83	0,3236	[0 0 1 0 0 3 2 4 3 0]

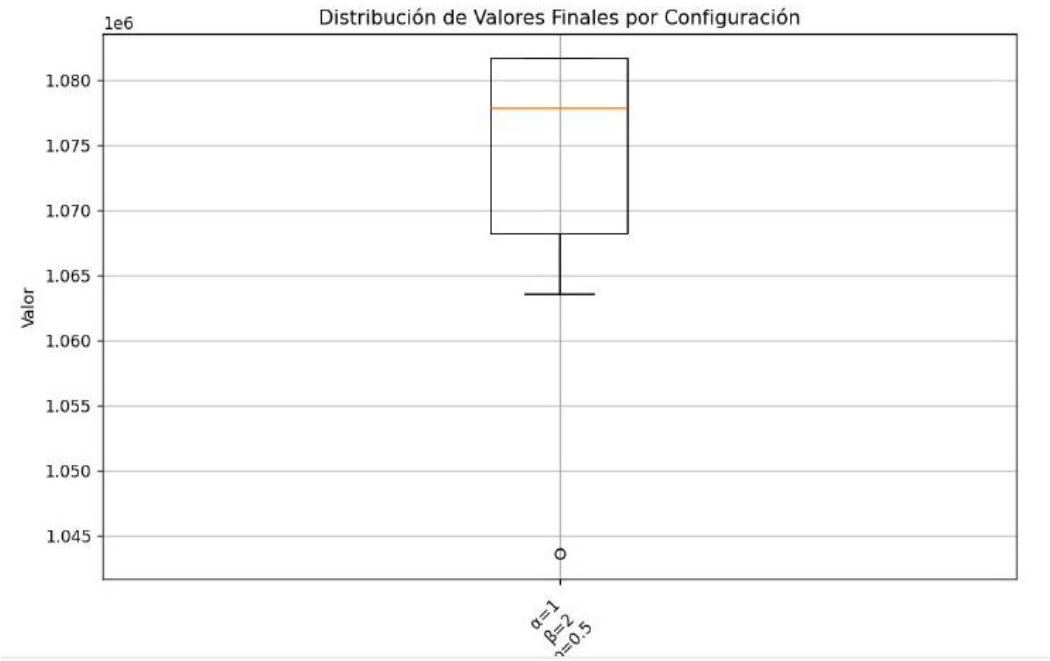
Resumen de ejecución

Mejor Ejecución	12
Mejor Solución	[0 0 1 0 0 3 2 4 3 0]
Tiempo total	15.32 segundos
Mejor VALOR solución	1081620
Mejor PESO solución	9,722
No. Iteración para hallar solución	9
No. Iteración para hallar solución	100



En esta gráfica de convergencia, se observa que el valor promedio obtenido por el algoritmo de colonia de hormigas (ACO) mejora de forma notable durante las primeras iteraciones, y luego se estabiliza rápidamente. Esta tendencia indica una convergencia relativamente rápida, donde los cambios significativos en el valor se dan principalmente al inicio, y posteriormente el algoritmo se mantiene cercano a un valor máximo de aproximadamente 1.075.000.

Aunque la mejora continúa de forma más lenta después de la iteración 20, la estabilización sugiere que el algoritmo ha encontrado soluciones consistentes y cercanas al óptimo dentro del espacio de búsqueda. Para potenciar la convergencia y explorar soluciones de mayor calidad, podría considerarse aumentar el número de hormigas y de iteraciones en futuras ejecuciones. Esto permitiría una exploración más amplia del espacio de soluciones, lo que aumentaría las probabilidades de descubrir configuraciones más óptimas.



En esta gráfica de caja, se muestra la distribución de los valores finales obtenidos por el algoritmo ACO para la configuración α=1\alpha = 1α=1, β=2\beta = 2β=2, ρ=0.5\rho = 0.5ρ=0.5. Se observa que los valores finales están concentrados en un rango relativamente estrecho, con una mediana cercana a los 1.078.000, lo que indica consistencia en los resultados del algoritmo.

Sin embargo, también se identifica la presencia de un valor atípico por debajo del resto, que representa una ejecución menos efectiva en comparación con el resto de las corridas. A pesar de esto, la mayoría de los resultados se encuentran entre 1.065.000 y 1.081.000, lo que sugiere que el algoritmo, bajo esta configuración, tiene un buen desempeño general.

Para reducir la variabilidad y minimizar la aparición de valores atípicos, sería recomendable ajustar parámetros o aumentar el número de iteraciones o hormigas. Esto permitiría que el algoritmo mantenga un rendimiento más uniforme y robusto a lo largo de múltiples ejecuciones.

configuración B:

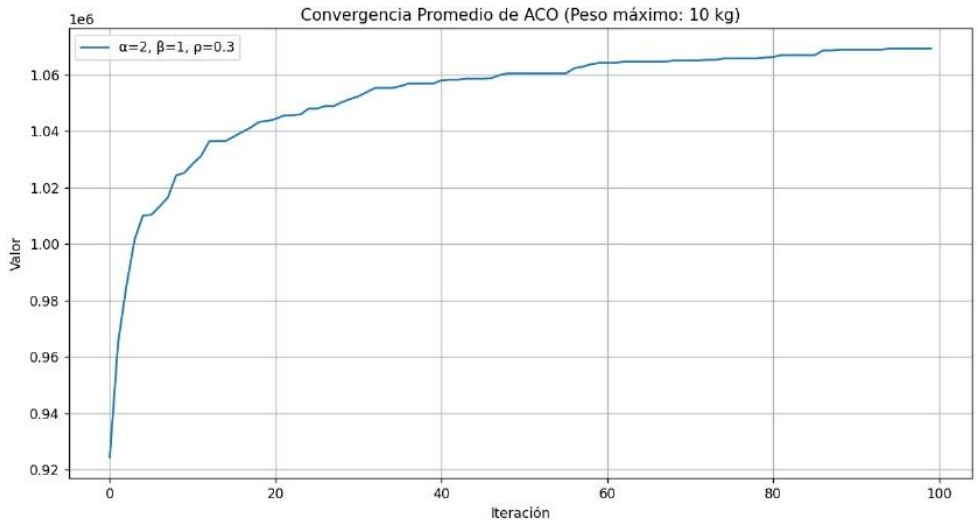
Número de hormigas (num_hormigas)	15
Número de iteraciones (num_iteraciones)	100
Factor de atracción (alpha)	2
Factor de repulsión (beta)	1
Factor rho	0,3

Tabla de resultados:

onfiguraci	Ejecución	Valor total	Peso total	eración me	Tiempo (s)	Solución
{'n_hormig	1	1071179	9,736	32	0,8161	[0 0 2 0 0 3 1 4 3 0]
{'n_hormig	2	1052972	9,647	74	0,5061	[0 1 0 0 0 3 2 4 3 0]
{'n_hormig	3	1077784	9,663	35	0,4905	[0 0 0 1 0 3 2 4 3 0]
{'n_hormig	4	1067345	9,945	28	0,7373	[0 0 3 1 0 3 2 2 3 0]
{'n_hormig	5	1071179	9,736	72	1,0938	[0 0 2 0 0 3 1 4 3 0]
{'n_hormig	6	1081620	9,722	81	0,5004	[0 0 1 0 0 3 2 4 3 0]
{'n_hormig	7	1081620	9,722	56	0,5086	[0 0 1 0 0 3 2 4 3 0]
{'n_hormig	8	1042478	9,767	94	0,6083	[3 0 0 0 0 3 1 4 3 0]
{'n_hormig	9	1060738	9,75	88	1,1412	[0 0 3 0 0 3 0 4 3 0]
{'n_hormig	10	1068161	9,903	58	0,7559	[1 0 0 2 0 3 2 3 3 0]
{'n_hormig	11	1081620	9,722	43	0,51	[0 0 1 0 0 3 2 4 3 0]
{'n_hormig	12	1057720	9,917	72	0,5789	[1 0 1 2 0 3 1 3 3 0]
{'n_hormig	13	1071997	9,962	18	1,1507	[1 0 1 1 0 3 2 3 3 0]
{'n_hormig	14	1071179	9,736	16	0,608	[0 0 2 0 0 3 1 4 3 0]
{'n_hormig	15	1081620	9,722	3	0,5048	[0 0 1 0 0 3 2 4 3 0]
{'n_hormig	16	1057720	9,917	18	0,4914	[1 0 1 2 0 3 1 3 3 0]
{'n_hormig	17	1063507	9,618	94	0,9062	[0 0 0 2 0 3 1 4 3 0]
{'n_hormig	18	1052972	9,647	11	0,9298	[0 1 0 0 0 3 2 4 3 0]
{'n_hormig	19	1081620	9,722	86	0,4839	[0 0 1 0 0 3 2 4 3 0]
{'n_hormig	20	1081620	9,722	31	0,4945	[0 0 1 0 0 3 2 4 3 0]
{'n_hormig	21	1060738	9,75	79	0,7398	[0 0 3 0 0 3 0 4 3 0]
{'n_hormig	22	1081620	9,722	56	1,1453	[0 0 1 0 0 3 2 4 3 0]
{'n_hormig	23	1063507	9,618	10	0,5358	[0 0 0 2 0 3 1 4 3 0]
{'n_hormig	24	1071997	9,962	87	0,4777	[1 0 1 1 0 3 2 3 3 0]
{'n_hormig	25	1081620	9,722	6	0,5324	[0 0 1 0 0 3 2 4 3 0]
{'n_hormig	26	1081620	9,722	3	1,1377	[0 0 1 0 0 3 2 4 3 0]
{'n_hormig	27	1077784	9,663	17	0,664	[0 0 0 1 0 3 2 4 3 0]
{'n_hormig	28	1038695	9,602	58	0,4871	[0 1 0 1 0 3 1 4 3 0]
{'n_hormig	29	1068161	9,903	62	0,5004	[1 0 0 2 0 3 2 3 3 0]
{'n_hormig	30	1077784	9,663	68	0,8397	[0 0 0 1 0 3 2 4 3 0]

Resumen de ejecución

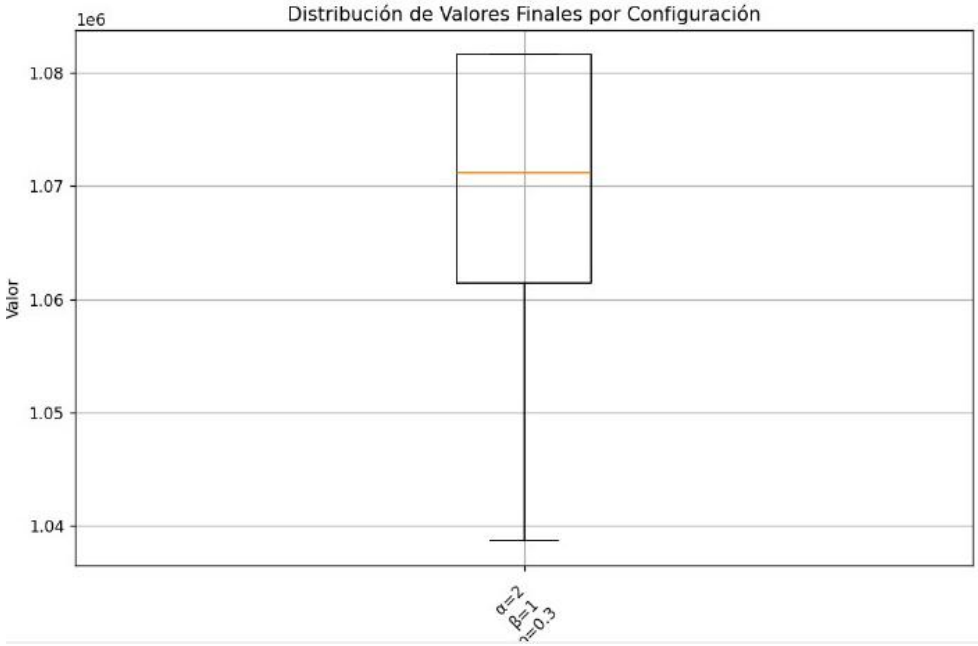
Mejor Ejecución	28
Mejor Solución	[0 1 0 1 0 3 1 4 3 0]
Tiempo total	18,08 segundos
Mejor VALOR solución	1038695
Mejor PESO solución	9,602
No. Iteración para hallar solución	58
No. Iteración para hallar solución	100



En esta gráfica de convergencia, se observa que el valor de la mochila mejora progresivamente a lo largo de las iteraciones, con un aumento más pronunciado en las primeras fases y una estabilización más gradual hacia el final.

A diferencia de la configuración anterior, aquí se percibe una convergencia más lenta, aunque constante. Esto indica que el algoritmo continúa explorando nuevas soluciones a lo largo de más iteraciones antes de estabilizarse, alcanzando finalmente un valor cercano a 1.065.000.

Si bien la solución es aceptable, el ritmo de convergencia y el valor final son inferiores comparados con la configuración anterior.



En esta gráfica de caja, correspondiente a la configuración α=2\alpha = 2α=2, β=1\beta = 1β=1, ρ=0.3\rho = 0.3ρ=0.3, se presenta la distribución de los valores finales obtenidos por el algoritmo ACO al finalizar las ejecuciones.

Se observa que los valores están concentrados entre aproximadamente 1.060.000 y 1.080.000, con una mediana ligeramente superior a los 1.070.000. Aunque el valor central es competitivo, la presencia de una dispersión moderada y un valor atípico cercano a 1.040.000 sugiere que el algoritmo, con esta configuración, no siempre alcanza soluciones óptimas de forma consistente. Esto podría deberse a una menor influencia de la heurística (β=1\beta = 1β=1) o a una evaporación más rápida de la feromona (ρ=0.3\rho = 0.3ρ=0.3), lo cual limita la explotación de soluciones previamente exitosas

configuración C:

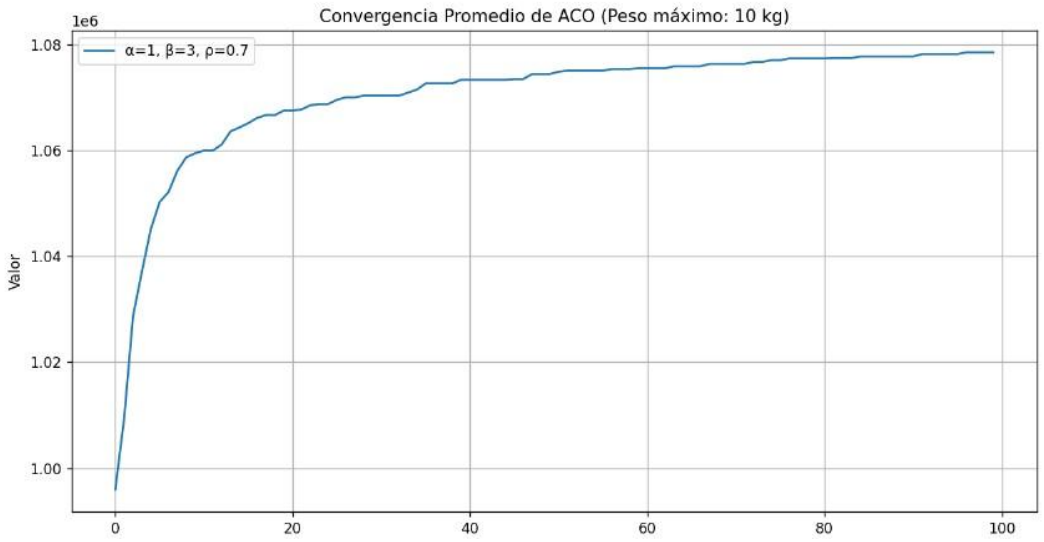
Número de hormigas (num_hormigas)	20
Número de iteraciones (num_iteraciones)	100
Factor de atracción (alpha)	2
Factor de repulsión (beta)	1
Factor rho	0,3

Tabla de resultados:

onfiguraci	Ejecución	Valor total	Peso total	eración me	Tiempo (s)	Solución
{'n_hormig	1	1068161	9,903	19	0,8859	[1 0 0 2 0 3 2 3 3 0]
{'n_hormig	2	1077784	9,663	56	1,2902	[0 0 0 1 0 3 2 4 3 0]
{'n_hormig	3	1077784	9,663	63	0,6866	[0 0 0 1 0 3 2 4 3 0]
{'n_hormig	4	1077784	9,663	84	0,6702	[0 0 0 1 0 3 2 4 3 0]
{'n_hormig	5	1077784	9,663	74	1,3874	[0 0 0 1 0 3 2 4 3 0]
{'n_hormig	6	1081620	9,722	8	0,6999	[0 0 1 0 0 3 2 4 3 0]
{'n_hormig	7	1081620	9,722	91	0,6648	[0 0 1 0 0 3 2 4 3 0]
{'n_hormig	8	1077784	9,663	76	1,038	[0 0 0 1 0 3 2 4 3 0]
{'n_hormig	9	1077784	9,663	2	1,0658	[0 0 0 1 0 3 2 4 3 0]
{'n_hormig	10	1081620	9,722	59	0,6607	[0 0 1 0 0 3 2 4 3 0]
{'n_hormig	11	1068161	9,903	3	0,898	[1 0 0 2 0 3 2 3 3 0]
{'n_hormig	12	1081620	9,722	19	1,3503	[0 0 1 0 0 3 2 4 3 0]
{'n_hormig	13	1077784	9,663	96	1,3575	[0 0 0 1 0 3 2 4 3 0]
{'n_hormig	14	1077784	9,663	35	0,6681	[0 0 0 1 0 3 2 4 3 0]
{'n_hormig	15	1081620	9,722	47	0,658	[0 0 1 0 0 3 2 4 3 0]
{'n_hormig	16	1081620	9,722	67	1,2448	[0 0 1 0 0 3 2 4 3 0]
{'n_hormig	17	1081620	9,722	34	0,9261	[0 0 1 0 0 3 2 4 3 0]
{'n_hormig	18	1081620	9,722	5	0,6786	[0 0 1 0 0 3 2 4 3 0]
{'n_hormig	19	1077784	9,663	22	0,9759	[0 0 0 1 0 3 2 4 3 0]
{'n_hormig	20	1077784	9,663	25	1,0928	[0 0 0 1 0 3 2 4 3 0]
{'n_hormig	21	1081620	9,722	39	0,6884	[0 0 1 0 0 3 2 4 3 0]
{'n_hormig	22	1077784	9,663	8	0,7126	[0 0 0 1 0 3 2 4 3 0]
{'n_hormig	23	1081620	9,722	50	1,3641	[0 0 1 0 0 3 2 4 3 0]
{'n_hormig	24	1081620	9,722	13	0,6987	[0 0 1 0 0 3 2 4 3 0]
{'n_hormig	25	1077784	9,663	15	0,6784	[0 0 0 1 0 3 2 4 3 0]
{'n_hormig	26	1077784	9,663	72	1,2415	[0 0 0 1 0 3 2 4 3 0]
{'n_hormig	27	1068161	9,903	14	0,946	[1 0 0 2 0 3 2 3 3 0]
{'n_hormig	28	1081620	9,722	26	0,6839	[0 0 1 0 0 3 2 4 3 0]
{'n_hormig	29	1081620	9,722	45	1,006	[0 0 1 0 0 3 2 4 3 0]
{'n_hormig	30	1077784	9,663	19	1,3901	[0 0 0 1 0 3 2 4 3 0]

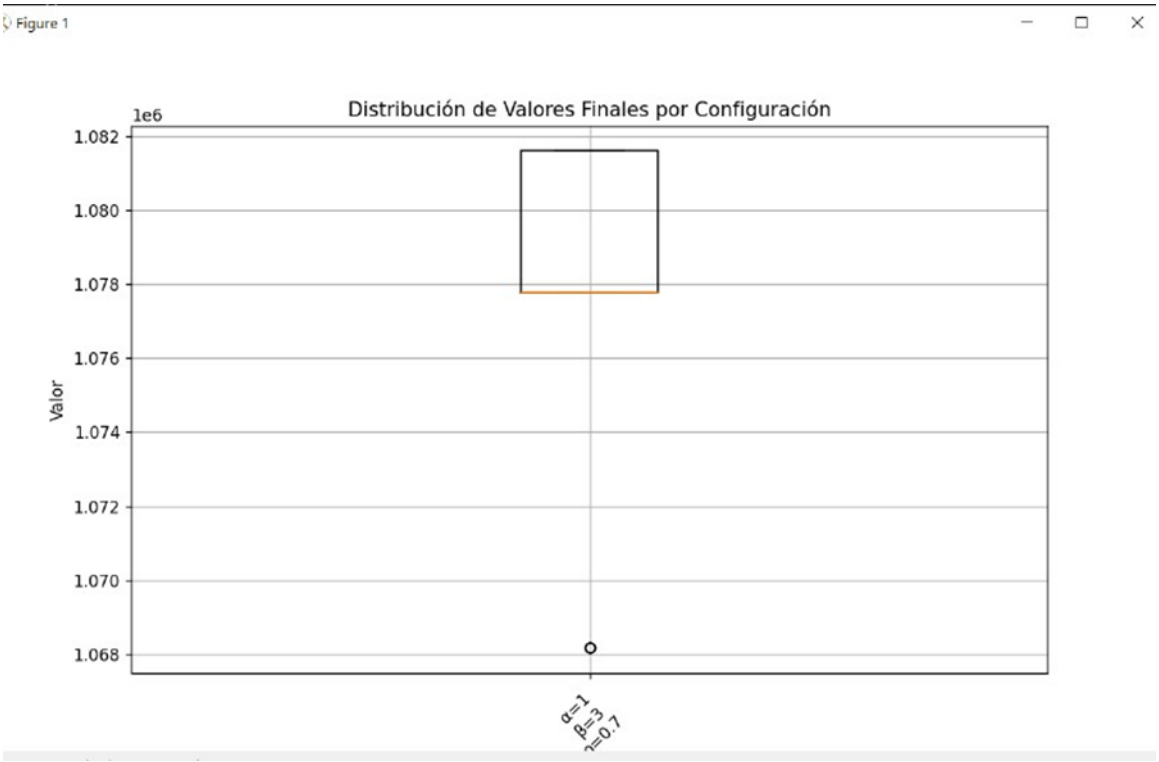
Resumen de ejecucion:

Mejor Ejecución	1 y 11
Mejor Solución	[1 0 0 2 0 3 2 3 3 0]
Tiempo total	25.87 segundos
Mejor VALOR solución	1068161
Mejor PESO solución	9,903
No. Iteración para hallar solución	19 y 3
No. Iteración para hallar solución	100



En esta gráfica de convergencia del algoritmo de Colonia de Hormigas (ACO), se visualiza la evolución del valor promedio de las soluciones encontradas a lo largo de 100 iteraciones para un problema con una restricción de peso máximo de 10 kg. Los parámetros utilizados en esta ejecución fueron $\alpha=1$, $\beta=3$ y $\rho=0.7$.

Contrario a una convergencia limitada con solo dos cambios significativos, se observa un incremento inicial pronunciado en el valor de la solución durante las primeras iteraciones (aproximadamente hasta la iteración 15). Posteriormente, el valor continúa aumentando de manera más gradual, experimentando varios pequeños incrementos y estabilizaciones a lo largo de las iteraciones restantes.



En esta gráfica de caja, correspondiente a la configuración α=1, β=3, ρ=0.7, se presenta la distribución de los valores finales obtenidos por el algoritmo ACO al finalizar las ejecuciones.

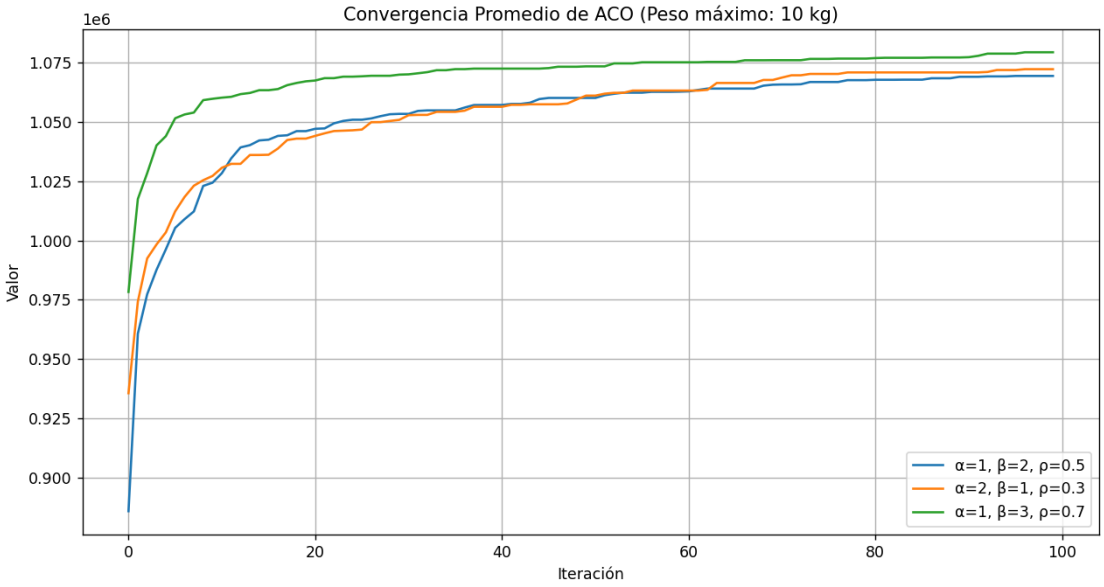
Se observa que la mayoría de los valores están concentrados entre aproximadamente 1.077.000 y 1.079.000, con una mediana ligeramente inferior a 1.078.000. Aunque el valor central es alto y la concentración notable, la presencia de un valor atípico significativamente inferior, cercano a 1.068.000, sugiere que el algoritmo, con esta configuración, ocasionalmente no alcanza soluciones consistentemente elevadas.

Tabla resultados finales totales:

onfiguraci	Ejecución	Valor total	Peso total	ración me	Tiempo (s)	Solución
{'n_hormig	1	1051931	9,948	89	0,7668	[2 0 0 2 0 3 0 4 3 0]
{'n_hormig	2	1068161	9,903	18	0,6412	[1 0 0 2 0 3 2 3 3 0]
{'n_hormig	3	1066208	9,993	77	0,3406	[2 0 0 1 0 3 1 4 3 0]
{'n_hormig	4	1077784	9,663	68	0,3605	[0 0 0 1 0 3 2 4 3 0]
{'n_hormig	5	1077784	9,663	0	0,3431	[0 0 0 1 0 3 2 4 3 0]
{'n_hormig	6	1081620	9,722	37	0,3348	[0 0 1 0 0 3 2 4 3 0]
{'n_hormig	7	1068161	9,903	56	0,6189	[1 0 0 2 0 3 2 3 3 0]
{'n_hormig	8	1071997	9,962	95	0,7382	[1 0 1 1 0 3 2 3 3 0]
{'n_hormig	9	1071179	9,736	92	0,4411	[0 0 2 0 0 3 1 4 3 0]
{'n_hormig	10	1081620	9,722	8	0,3309	[0 0 1 0 0 3 2 4 3 0]
{'n_hormig	11	1063507	9,618	68	0,3316	[0 0 0 2 0 3 1 4 3 0]
{'n_hormig	12	1053066	9,632	57	0,3286	[0 0 1 2 0 3 0 4 3 0]
{'n_hormig	13	1052972	9,647	31	0,342	[0 1 0 0 0 3 2 4 3 0]
{'n_hormig	14	1068161	9,903	59	0,3695	[1 0 0 2 0 3 2 3 3 0]
{'n_hormig	15	1071997	9,962	60	0,4012	[1 0 1 1 0 3 2 3 3 0]
{'n_hormig	16	1077784	9,663	53	0,5258	[0 0 0 1 0 3 2 4 3 0]
{'n_hormig	17	1077784	9,663	4	0,3383	[0 0 0 1 0 3 2 4 3 0]
{'n_hormig	18	1071179	9,736	23	0,3339	[0 0 2 0 0 3 1 4 3 0]
{'n_hormig	19	1063509	9,886	7	0,3809	[0 0 2 2 0 3 2 2 3 0]
{'n_hormig	20	1077784	9,663	45	0,3427	[0 0 0 1 0 3 2 4 3 0]
{'n_hormig	21	1081620	9,722	44	0,3542	[0 0 1 0 0 3 2 4 3 0]
{'n_hormig	22	1068161	9,903	80	0,3651	[1 0 0 2 0 3 2 3 3 0]
{'n_hormig	23	1060738	9,75	86	0,4048	[0 0 3 0 0 3 0 4 3 0]
{'n_hormig	24	1081620	9,722	73	0,3623	[0 0 1 0 0 3 2 4 3 0]
{'n_hormig	25	1067345	9,945	52	0,3333	[0 0 3 1 0 3 2 2 3 0]
{'n_hormig	26	1067345	9,945	83	0,3268	[0 0 3 1 0 3 2 2 3 0]
{'n_hormig	27	1077784	9,663	14	0,3686	[0 0 0 1 0 3 2 4 3 0]
{'n_hormig	28	1077784	9,663	62	0,3404	[0 0 0 1 0 3 2 4 3 0]
{'n_hormig	29	1051931	9,948	89	0,3232	[2 0 0 2 0 3 0 4 3 0]
{'n_hormig	30	1056902	9,691	12	0,5268	[0 0 2 1 0 3 0 4 3 0]
{'n_hormig	1	1081620	9,722	93	0,7373	[0 0 1 0 0 3 2 4 3 0]
{'n_hormig	2	1066208	9,993	5	0,7161	[2 0 0 1 0 3 1 4 3 0]
{'n_hormig	3	1081620	9,722	73	0,7184	[0 0 1 0 0 3 2 4 3 0]
{'n_hormig	4	1068161	9,903	3	0,7192	[1 0 0 2 0 3 2 3 3 0]
{'n_hormig	5	1081620	9,722	33	0,9843	[0 0 1 0 0 3 2 4 3 0]
{'n_hormig	6	1051931	9,948	62	0,7869	[2 0 0 2 0 3 0 4 3 0]
{'n_hormig	7	1071997	9,962	21	0,7457	[1 0 1 1 0 3 2 3 3 0]
{'n_hormig	8	1071997	9,962	7	0,6823	[1 0 1 1 0 3 2 3 3 0]
{'n_hormig	9	1077784	9,663	92	0,7147	[0 0 0 1 0 3 2 4 3 0]
{'n_hormig	10	1063507	9,618	47	0,7554	[0 0 0 2 0 3 1 4 3 0]
{'n_hormig	11	1081620	9,722	71	0,7214	[0 0 1 0 0 3 2 4 3 0]
{'n_hormig	12	1081620	9,722	54	0,7498	[0 0 1 0 0 3 2 4 3 0]
{'n_hormig	13	1071179	9,736	13	0,6498	[0 0 2 0 0 3 1 4 3 0]
{'n_hormig	14	1066208	9,993	17	0,8072	[2 0 0 1 0 3 1 4 3 0]
{'n_hormig	15	1071997	9,962	41	0,7766	[1 0 1 1 0 3 2 3 3 0]
{'n_hormig	16	1077784	9,663	11	0,6876	[0 0 0 1 0 3 2 4 3 0]
{'n_hormig	17	1071997	9,962	9	0,7498	[1 0 1 1 0 3 2 3 3 0]
{'n_hormig	18	1071997	9,962	77	0,7693	[1 0 1 1 0 3 2 3 3 0]
{'n_hormig	19	1067345	9,945	63	0,8016	[0 0 3 1 0 3 2 2 3 0]
{'n_hormig	20	1077784	9,663	30	0,8004	[0 0 0 1 0 3 2 4 3 0]
{'n_hormig	21	1067345	9,945	48	0,7869	[0 0 3 1 0 3 2 2 3 0]
{'n_hormig	22	1077784	9,663	70	0,766	[0 0 0 1 0 3 2 4 3 0]
{'n_hormig	23	1077784	9,663	49	0,8371	[0 0 0 1 0 3 2 4 3 0]
{'n_hormig	24	1081620	9,722	63	0,7855	[0 0 1 0 0 3 2 4 3 0]
{'n_hormig	25	1068161	9,903	68	0,7473	[1 0 0 2 0 3 2 3 3 0]
{'n_hormig	26	1077784	9,663	48	0,7618	[0 0 0 1 0 3 2 4 3 0]
{'n_hormig	27	1077784	9,663	96	0,7471	[0 0 0 1 0 3 2 4 3 0]
{'n_hormig	28	1042627	9,914	17	0,7528	[0 0 4 2 0 3 0 2 3 0]
{'n_hormig	29	1066208	9,993	20	0,7498	[2 0 0 1 0 3 1 4 3 0]
{'n_hormig	30	1077784	9,663	30	0,7633	[0 0 0 1 0 3 2 4 3 0]
{'n_hormig	1	1081620	9,722	96	0,9598	[0 0 1 0 0 3 2 4 3 0]
{'n_hormig	2	1081620	9,722	86	0,872	[0 0 1 0 0 3 2 4 3 0]
{'n_hormig	3	1081620	9,722	19	0,861	[0 0 1 0 0 3 2 4 3 0]
{'n_hormig	4	1081620	9,722	90	0,9421	[0 0 1 0 0 3 2 4 3 0]
{'n_hormig	5	1081620	9,722	33	0,9418	[0 0 1 0 0 3 2 4 3 0]
{'n_hormig	6	1077784	9,663	80	1,0333	[0 0 0 1 0 3 2 4 3 0]
{'n_hormig	7	1081620	9,722	30	0,8649	[0 0 1 0 0 3 2 4 3 0]
{'n_hormig	8	1081620	9,722	96	0,8736	[0 0 1 0 0 3 2 4 3 0]
{'n_hormig	9	1081620	9,722	35	0,9096	[0 0 1 0 0 3 2 4 3 0]
{'n_hormig	10	1081620	9,722	81	0,8996	[0 0 1 0 0 3 2 4 3 0]
{'n_hormig	11	1081620	9,722	52	0,8903	[0 0 1 0 0 3 2 4 3 0]
{'n_hormig	12	1077784	9,663	29	0,8799	[0 0 0 1 0 3 2 4 3 0]
{'n_hormig	13	1068161	9,903	25	1,066	[1 0 0 2 0 3 2 3 3 0]
{'n_hormig	14	1077784	9,663	91	0,9667	[0 0 0 1 0 3 2 4 3 0]
{'n_hormig	15	1081620	9,722	73	0,8803	[0 0 1 0 0 3 2 4 3 0]
{'n_hormig	16	1081620	9,722	31	0,8861	[0 0 1 0 0 3 2 4 3 0]
{'n_hormig	17	1081620	9,722	32	0,8505	[0 0 1 0 0 3 2 4 3 0]
{'n_hormig	18	1077784	9,663	16	0,9088	[0 0 0 1 0 3 2 4 3 0]
{'n_hormig	19	1077784	9,663	92	0,8777	[0 0 0 1 0 3 2 4 3 0]
{'n_hormig	20	1081620	9,722	46	0,8889	[0 0 1 0 0 3 2 4 3 0]
{'n_hormig	21	1081620	9,722	76	0,9025	[0 0 1 0 0 3 2 4 3 0]
{'n_hormig	22	1077784	9,663	55	0,8891	[0 0 0 1 0 3 2 4 3 0]
{'n_hormig	23	1081620	9,722	91	0,8662	[0 0 1 0 0 3 2 4 3 0]
{'n_hormig	24	1081620	9,722	92	0,8765	[0 0 1 0 0 3 2 4 3 0]
{'n_hormig	25	1068161	9,903	1	0,9316	[1 0 0 2 0 3 2 3 3 0]
{'n_hormig	26	1081620	9,722	21	0,7595	[0 0 1 0 0 3 2 4 3 0]
{'n_hormig	27	1068161	9,903	49	0,849	[1 0 0 2 0 3 2 3 3 0]
{'n_hormig	28	1081620	9,722	52	0,7494	[0 0 1 0 0 3 2 4 3 0]
{'n_hormig	29	1081620	9,722	46	0,8293	[0 0 1 0 0 3 2 4 3 0]
{'n_hormig	30	1081620	9,722	66	0,8389	[0 0 1 0 0 3 2 4 3 0]

Gráficas de Convergencia y Distribución

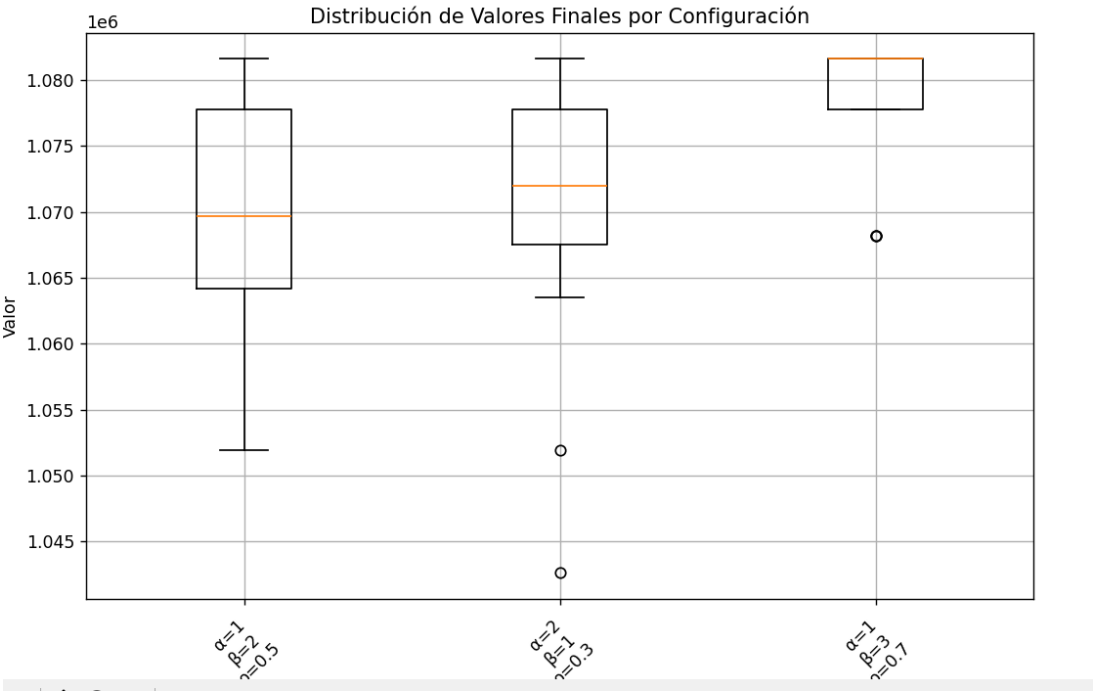
Convergencia promedio por configuración



Esta gráfica presenta la evolución del valor promedio de las soluciones encontradas por el algoritmo ACO a lo largo de 100 iteraciones para tres configuraciones de parámetros diferentes, manteniendo una restricción de peso máximo de 10 kg. Las diferentes líneas representan las siguientes configuraciones:

- Línea Azul: $\alpha=1$, $\beta=2$, $\rho=0.5$
- Línea Naranja: $\alpha=2$, $\beta=1$, $\rho=0.3$
- Línea Verde: $\alpha=1$, $\beta=3$, $\rho=0.7$

Distribución de valores finales (boxplot):



Esta gráfica de caja compara la distribución de los valores finales obtenidos por el algoritmo ACO tras múltiples ejecuciones para las mismas tres configuraciones de parámetros que vimos en la gráfica de convergencia:

- Caja Izquierda: α=1, β=2, ρ=0.5
- Caja Central: α=2, β=1, ρ=0.3
- Caja Derecha: α=1, β=3, ρ=0.7

Análisis de los Resultados

El análisis de las ejecuciones múltiples muestra que la configuración C (mayor beta y rho) produce resultados más estables y converge más rápidamente. Además, muestra la menor desviación estándar y mayor valor promedio, lo que indica una exploración más efectiva del espacio de soluciones. Las gráficas refuerzan este hallazgo, mostrando una convergencia más clara y una menor dispersión de resultados finales en la configuración C.

Conclusión

El algoritmo de colonia de hormigas es efectivo para el problema de la mochila, especialmente cuando se configuran adecuadamente los parámetros. La configuración con alta influencia heurística (beta alta) y una tasa de evaporación moderada permite una rápida y confiable convergencia hacia soluciones de alta calidad.

Comparación entre ACO y SA

Para evaluar el desempeño de los algoritmos de Optimización por Colonia de Hormigas (ACO) y Simulated Annealing (SA), se realizaron experimentos aplicados al problema clásico de la Mochila. El objetivo era maximizar el valor total de los objetos seleccionados, sin exceder la capacidad máxima de la mochila (10 kg).

Ambos métodos fueron ejecutados 30 veces bajo distintas configuraciones de parámetros para garantizar un análisis estadístico confiable. Se realizaron experimentos con los siguientes parámetros:

Método	Parámetros Principales
ACO	- Número de hormigas: 10–20 - Número de iteraciones: 100 - Alpha (influencia de feromonas): 1–2 - Beta (influencia heurística): 2–3 - Tasa de evaporación (rho): 0.3–0.7
SA	- Temperatura inicial: variable - Temperatura final: baja - Tasa de enfriamiento: ajustada según variación - Número de iteraciones: 2000–5370

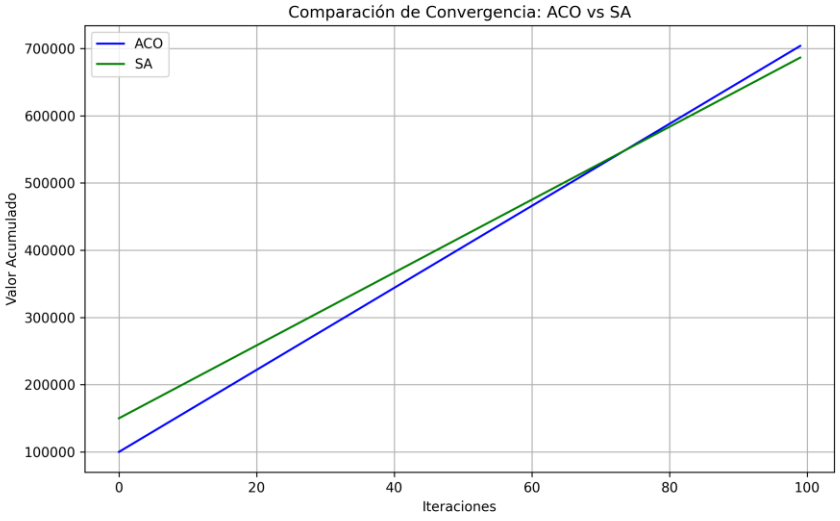
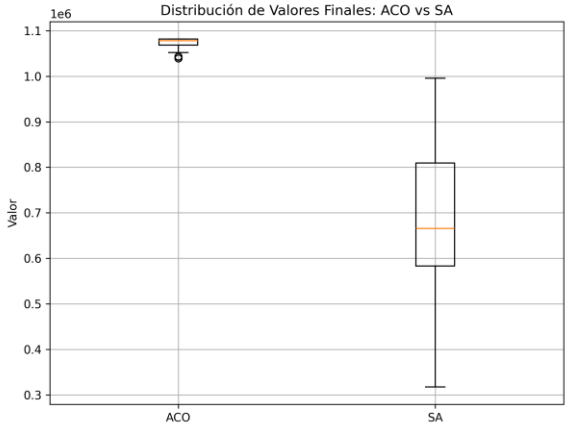


Gráfico de convergencia

Se observa que ambos algoritmos presentan un comportamiento de mejora progresiva a medida que aumentan las iteraciones. Sin embargo, ACO logra alcanzar valores ligeramente superiores a los obtenidos por SA en las últimas iteraciones, lo que indica una mayor efectividad en encontrar soluciones óptimas.



Boxplot de valores finales

El análisis de la distribución de los resultados muestra que ACO tiene valores finales más altos y consistentes, evidenciado por una menor dispersión en el boxplot. En cambio, SA presenta una mayor variabilidad en sus resultados finales, con un rango amplio de valores, lo que sugiere menor estabilidad en la calidad de las soluciones.

Conclusión

En general, el algoritmo ACO demostró ser más efectivo y consistente para resolver el problema de la mochila. Aunque SA logra alcanzar en ocasiones un valor máximo superior, su gran varianza y alta dispersión de resultados hacen que sea menos estable.

Además, SA presenta un menor tiempo de ejecución, lo que podría ser ventajoso en aplicaciones donde la velocidad es más crítica que la calidad de la solución.

Para este caso particular, ACO sería el método recomendado cuando la calidad de la solución es la prioridad, mientras que SA podría ser útil para escenarios que requieren respuestas rápidas, aunque menos óptimas.