Informe Final: Comparación de Métodos Bioinspirados para el Problema de la Mochila

Informe Comparativo: Métodos Bioinspirados para el Problema de la Mochila

En este informe se presenta la comparación de dos algoritmos bioinspirados aplicados a la resolución del problema de la mochila (Knapsack problem): Enfriamiento Simulado (Simulated Annealing, SA) y Optimización por Colonia de Hormigas (Ant Colony Optimization, ACO). Se parte de una instancia con límite de capacidad de 20 kg (que vienen dado al archivo Mochila_capacidad_maxima_20kg) y un conjunto de ítems con diferentes pesos, valores y cantidades máximas disponibles. El objetivo es maximizar el valor total transportado sin exceder la capacidad que ofrece la mochila (20kg).

3.1 Esquema de representación y estrategia de generación de soluciones

Tanto para Simulated Annealing (SA) como para Ant Colony Optimization (ACO), se utiliza una representación basada en listas de cantidades de objetos. Cada posición en la lista representa la cantidad de objetos de su correspondiente tipo, el cual está incluido en la mochila. Por ejemplo, la solución [1, 3, 0, 2] indica que se incluyen 1 objeto del tipo 1, 3 del tipo 2, ninguno del tipo 3 y 2 del tipo 4.

Todas las soluciones tanto en SA como en ACO, se codifican como un vector de enteros de longitud *N*, donde *N* es el número de tipos de ítems disponibles. Donde cada posición *i* indica la cantidad de unidades del ítem *i* que se incluyen en la mochila.

Las funciones para evaluar una solución se basan en los siguientes cálculos:

$$Peso\ total = \sum_{i=0}^{n-1} cantidad[i] \cdot peso[i]$$

$$Valor\ total = \sum_{i=0}^{n-1} cantidad[i] \cdot valor[i]$$

Solo se consideran soluciones válidas (factibles), es decir, aquellas cuyo peso total no excede la capacidad máxima (20 kg). Estas soluciones válidas son las que participan en los procesos de actualización (ACO) o aceptación (SA).

Representación de Ant Colony Optimization (ACO)

En la implementación de ACO, cada solución es una lista de longitud *N* (número de objetos), donde cada posición indica cuántas unidades del objeto correspondiente (como ya se explicó antes) han sido seleccionadas. La construcción de soluciones se realiza de forma **incremental y probabilística**, guiada por dos factores clave:

- **Feromonas**: representan la experiencia acumulada de las hormigas previas. Se inicializan aleatoriamente en un rango estrecho (0.8, 1.2) esto con el fin evitar sesgos iniciales ya que nos dimos cuenta que cuando se inicializan las feromonas todas en 1 el algoritmo empieza a sufrir sesgos y deja de explorar nuevas variantes, es por eso que se optó por este modelo, después de la inicialización de estas se actualizan luego de cada generación en función de la calidad de las soluciones encontradas.

$$\tau_{ij} = random.uniform(0.8, 1.2)$$

- **Heurística**: basada en la razón valor/peso de cada ítem, define una medida estática de atractivo.
- **Construcción de solución**: cada hormiga recorre las posiciones de la mochila en un orden aleatorio, en cada paso se selecciona un item i, que tiene una probabilidad dada por:

$$p_{ij} \propto \tau_{ij}^{\alpha} \cdot \eta_{ij}^{\beta}$$

  Donde η_{ij}^{β} es la heurística del item, α y β son hiperparametros que controlan la influencia de las feromonas y la heurística, respectivamente.

- **Actualización de feromonas:** Al terminar la generación, cada hormiga refuerza el τ_{ij} de la ruta que le tocó con un refuerzo proporcional a la calidad de su solución, es por esto, que se agrego ese hiperparametro de Q. No es indispensable en el algoritmo, pero si le aporta una mayor calidad, esta es la razón por la cual se colocó.
- **Evaporación:** por último, se evaporan las feromonas de modo:

$$\tau_{ij} = (1 - \rho)\tau_{ij}$$

La probabilidad de seleccionar un ítem en cada paso se calcula mediante una combinación ponderada de ambos factores, usando los parámetros **α** (influencia de la feromona) y **β** (influencia heurística) los cuales son pasados como inicialización del algoritmo. Cada hormiga construye su solución eligiendo ítems uno por uno hasta alcanzar el límite de capacidad (20kg en nuestro caso). Esta estrategia mejora significativamente respecto a versiones anteriores, en las cuales las soluciones eran generadas asignando cantidades aleatorias a cada ítem sin un control efectivo del peso total ni una guía clara. La versión actual logra una mejor ocupación de la mochila y mayor calidad global en las soluciones generadas.

Representación de Simulated Annealing (SA)

En la implementación de Enfriamiento Simulado, también se emplea una lista como representación de soluciones. La generación inicial se basa en un enfoque heurístico que ordena los ítems por su razón valor/peso (con una ligera aleatorización con el fin de explorar un poco más el espacio), seleccionando la mayor cantidad posible de cada uno hasta llenar la mochila. Este método permite partir de una solución factible y competitiva desde el inicio. La solución inicial se construye con un heurístico valor/peso: ordenamos los ítems de mayor a menor eficiencia (valor/peso) y vamos agregando la máxima cantidad posible de cada uno hasta agotar capacidad.

Las soluciones vecinas se generan aplicando dos tipos de modificaciones:

- **Ajuste de unidades**: se incrementan o decrementan pequeñas cantidades (entre 1 y 3 unidades) en uno o más ítems aleatoriamente.
- **Intercambio entre ítems**: se transfiere una unidad de un ítem a otro, manteniendo el tamaño del conjunto.

Criterio de aceptación:

- Si el vecino tiene valor mayor o igual al valor actual, se acepta directamente.
- Si no es así, se deberá aceptar con una probabilidad basada en la temperatura:

  $$e^{-\varepsilon/T}$$

  Donde ε equivale a la perdida del valor, es decir:

  $$\varepsilon = new\_val - old\_val$$

  Y donde **T** equivale a la temperatura actual.

Enfriamiento:

Partimos de una temperatura inicial, la cual viene dada desde los hiperparametros, y se va reduciendo de manera gradual variando por **α** (Alpha) hasta que lleguemos a un umbral que lo llamamos *stoppingT* o hasta que se cumplan la cantidad máxima de iteraciones.

En las primeras versiones, la generación de vecinos estaba limitada a la modificación de un único ítem, lo que restringía la exploración del espacio de búsqueda y conducía a convergencia prematura. Para resolver este problema, se introdujo algo de ruido múltiples y se refinaron los mecanismos de enfriamiento, ajustando tanto la tasa **α** (Alpha) como la temperatura mínima de parada. Estas mejoras permitieron una exploración más diversa y una convergencia más estable hacia soluciones de mayor calidad.

Evaluación y validez

En ambos algoritmos, cada solución es evaluada mediante la suma total del valor y del peso, verificando que el total no exceda la capacidad de la mochila. Solo las soluciones válidas (factibles) son consideradas para actualización de feromonas (ACO) o aceptación (SA), lo que garantiza el cumplimiento de las restricciones del problema.

Hiperparámetros y Justificación de las Configuraciones Utilizadas

Para evaluar el comportamiento de los algoritmos **Simulated Annealing (SA)** y **Ant Colony Optimization (ACO)** para resolver el problema de la mochila, se diseñaron cuatro configuraciones distintas por cada uno de ellos: **original**, **exploratoria**, **competitiva** y **débil**. Cada configuración responde a un enfoque estratégico con dos objetivos principales:

- Analizar desde la robustez de cada algoritmo.
- Hallar los límites de desempeño de cada modelo.

Simulated Annealing (SA)

Las configuraciones de SA se definieron a partir de tres parámetros clave:

- **T**: Temperatura inicial del sistema. Valores altos favorecen la exploración al inicio del proceso.
- **α (alpha)**: Tasa de enfriamiento. Controla qué tan rápido disminuye la temperatura, afectando la capacidad de escapar de óptimos locales.
- **stopping_T**: Temperatura mínima para detener el proceso.
- **stopping_iter**: Iteraciones máximas permitidas. (adicional)

Cada configuración tiene el siguiente propósito:

1. **Original**:
    - **T = 15000**: Temperatura inicial alta, diseñada con el fin de permitir una exploración amplia en etapas tempranas, aceptando soluciones subóptimas que puedan llevar a regiones de alto valor.
    - **alpha = 0.996**: Un coeficiente de enfriamiento muy **lento**, que permite muchas iteraciones de búsqueda antes de estancarse.
    - **stopping_T = 1e-12**: Un umbral muy bajo que asegura que la temperatura disminuya hasta valores insignificantes, permitiendo observar el efecto del enfriamiento prolongado.

- **stopping_iter = 1000**: Límite alto para iteraciones, ideal para observar el comportamiento completo del algoritmo sin restricciones prematuras

Diseñada como línea base, esta configuración busca un equilibrio entre exploración y explotación. La alta temperatura inicial combinada con un enfriamiento muy suave permite que el sistema explore ampliamente antes de estabilizarse.

2. **Exploratoria**:

- **T = 12000**: Una temperatura inicial alta, aunque más baja que la original, permite buena **variabilidad inicial sin descontrol**.
- **alpha = 0.985**: Enfriamiento **más rápido**, lo que reduce la persistencia en estados subóptimos y **fomenta más movimiento** entre soluciones diversas.
- **stopping_T = 1e-7**: Umbral de temperatura menos estricto, deteniendo el proceso más pronto y permitiendo evaluar exploraciones sin convergencia extrema.
- **stopping_iter = 1000**: Se mantiene el número de iteraciones para garantizar que se explore de manera razonable del espacio de búsqueda.

Se favorece una búsqueda más diversa a través de un enfriamiento más rápido (α más bajo) y una temperatura inicial ligeramente menor. El objetivo es provocar más variabilidad en las soluciones, aún a costa de precisión.

3. **Competitiva**:

- **T = 13000**: Temperatura media-alta, suficiente para arrancar con diversidad sin desviarse tanto.
- **alpha = 0.991**: Enfriamiento moderadamente lento, diseñado para alcanzar convergencia precisa en menos tiempo.
- **stopping_T = 1e-11**: Umbral estricto que permite exprimir las fases finales del algoritmo buscando **soluciones muy refinadas**.
- **stopping_iter = 700**: Se reduce el número de iteraciones intencionalmente para **evaluar eficiencia** sin perder calidad de solución.

Esta configuración está diseñada para ser **eficiente y precisa**, ideal para comparación con escenarios reales. También optimiza el tiempo de búsqueda con una temperatura también un poco alta y un enfriamiento un poco lento, con el fin de hallar la convergencia. Es ideal para encontrar buenas soluciones en menos tiempo y simula un comportamiento "eficiente y preciso".

4. **Débil**:

   - **T = 8000**: Temperatura inicial moderada, restringida para no aceptar muchos movimientos al principio.
   - **alpha = 0.899**: Enfriamiento extremadamente rápido, lo que hace que el algoritmo pierda capacidad de exploración rápidamente.
   - **stopping_T = 1e-4**: Umbral alto de parada, detiene el algoritmo cuando aún podría seguir mejorando.
   - **stopping_iter = 300**: Muy pocas iteraciones, pensada para simular una ejecución limitada y poco efectiva.

   Configurada para obtener soluciones de menor calidad, esta variante incluye una temperatura más baja y un enfriamiento agresivo. Es útil como control negativo, ya que demuestra qué sucede cuando se limita intencionalmente la capacidad de búsqueda.

Ant Colony Optimization (ACO)

Para ACO se consideraron los siguientes hiperparámetros:

- **ant_count**: Número de hormigas en cada generación.
- **generations**: Número total de iteraciones del algoritmo.
- **α (alpha)**: Influencia relativa de las feromonas.
- **β (beta)**: Influencia del valor heurístico (valor/peso).
- **ρ (rho)**: Tasa de evaporación de feromonas.
- **Q**: Intensidad con la que se refuerzan los caminos más exitosos. (adicional)

Cada configuración tiene la siguiente intención:

1. **Original**:
   - **ant_count = 10**: Número estándar de hormigas, balanceando carga computacional y diversidad.
   - **generations = 1000**: Altísimo número de generaciones, ideal para evaluar el comportamiento a largo plazo.
   - **alpha = 1.0**: Peso equilibrado entre feromonas y heurística.
   - **beta = 2.0**: Alta influencia del heurístico (valor/peso), típico en problemas de mochila.
   - **rho = 0.1**: Evaporación moderada, permite preservar caminos buenos mientras elimina ruido.
   - **Q = 100**: Refuerzo estándar.

   Utiliza una configuración estándar, balanceando adecuadamente entre la información heurística y la acumulación de feromonas. Nuevamente, esta configuración sirve como punto de referencia general para comparar otras variantes.

2. **Exploratoria**:
    - **ant_count = 20**: Aumento del número de hormigas para ampliar la exploración del espacio de soluciones.
    - **generations = 1000**: Se mantiene el número alto de generaciones para evaluar mejoras a largo plazo.
    - **alpha = 1.1**: Ligero aumento en la influencia de feromonas, con el fin de mantener flexibilidad.
    - **beta = 1.0**: Reducción de la influencia heurística para dar más libertad a las decisiones basadas en experiencia colectiva.
    - **rho = 0.3**: Evaporación alta, fuerza a la colonia a explorar nuevas rutas constantemente.
    - **Q = 120**: Refuerzo más agresivo para caminos exitosos.

    Aumenta tanto el número de hormigas como la tasa de evaporación para forzar la diversificación de rutas. Disminuye la influencia del heurístico, incentivando caminos menos tradicionales y favoreciendo la exploración.

3. **Competitiva**:
    - **ant_count = 12**: Cantidad moderada de hormigas para mantener eficiencia sin perder diversidad.
    - **generations = 700**: Reducción de generaciones para buscar resultados en menor tiempo.
    - **alpha = 1.3**: Mayor peso de las feromonas, favoreciendo las rutas con historial positivo.
    - **beta = 1.5**: Peso moderado del heurístico, se mantiene la evaluación del valor/peso sin dominar.
    - **rho = 0.25**: Alta evaporación, incentiva aprendizaje rápido y actualización constante.
    - **Q = 80**: Refuerzo contenido, para evitar sobreespecialización prematura.

    Esta configuración prioriza resultados de calidad en tiempo eficiente. Busca una rápida convergencia a soluciones óptimas mediante una mayor influencia de las feromonas y una evaporación moderadamente alta. Es adecuada para escenarios donde se requiere rendimiento eficiente con pocas generaciones.

4. **Débil**:
    - **ant_count = 8**: Menor número de hormigas, se reduce la diversidad.
    - **generations = 300**: Pocas generaciones, el algoritmo tiene poco tiempo para aprender.
    - **alpha = 0.7**: Baja influencia de feromonas, lo cual reduce el aprendizaje colectivo.
    - **beta = 0.8**: Heurística poco valorada, mayor aleatoriedad.
    - **rho = 0.4**: Alta evaporación, elimina rápidamente el aprendizaje acumulado.
    - **Q = 50**: Refuerzo bajo, los caminos exitosos no se destacan con claridad.

Al reducir tanto la influencia de feromonas como del heurístico y aumentar la evaporación, esta configuración promueve un comportamiento casi aleatorio. Se emplea como escenario de referencia con bajo rendimiento intencional.

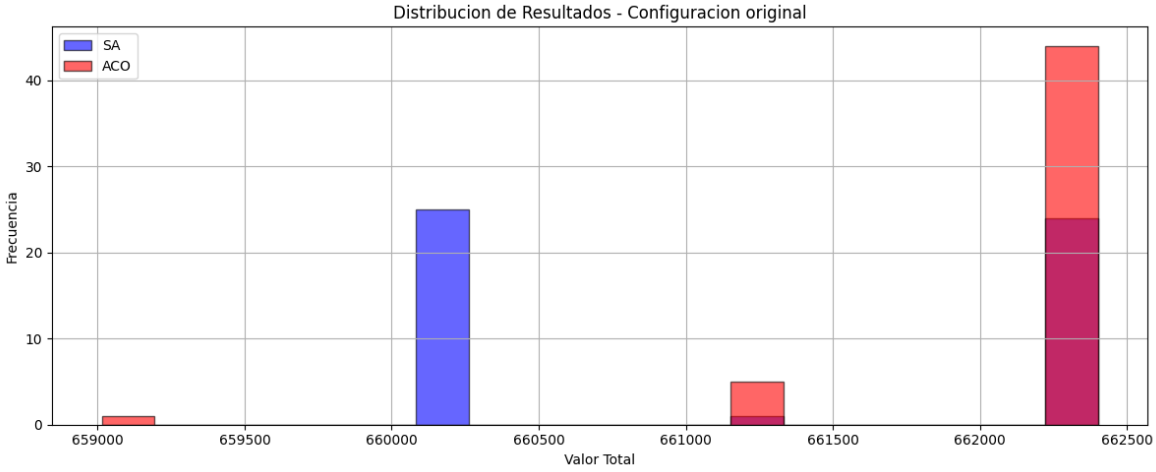
NOTA:

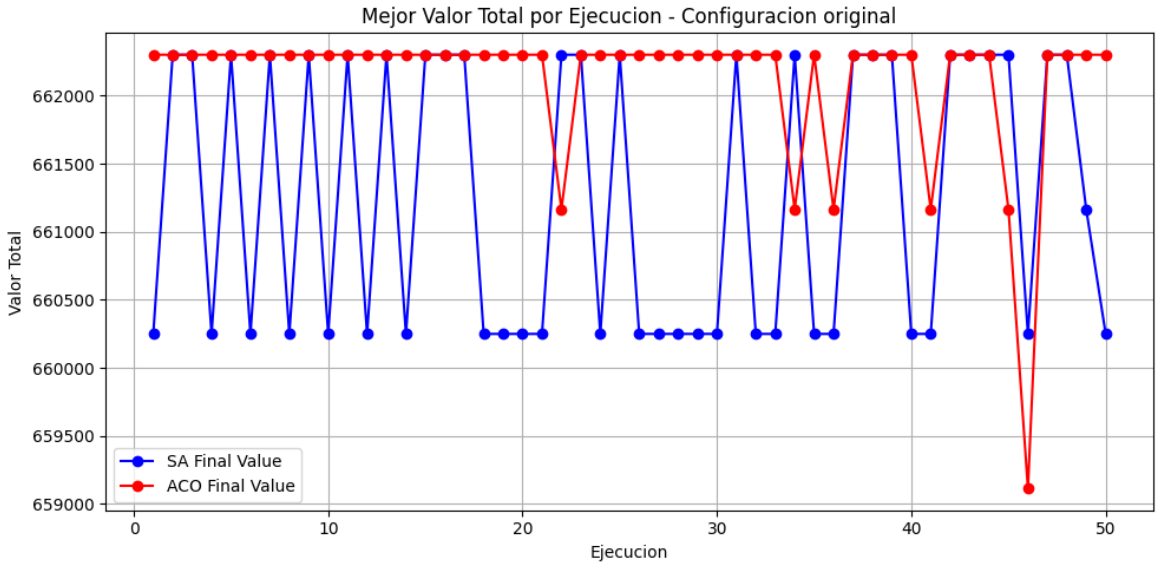
Ahora después de haber explicado cada una de las configuraciones se harán pruebas con 50 iteraciones sobre cada una de estas con el fin de mostrar sencillamente su funcionamiento y por qué se consideran útiles en el trabajo. Esto no hace parte de ningún ítem requerido en el trabajo, sencillamente se quiere demostrar como fue el funcionamiento de cada una de las configuraciones. Como no hace parte de ningún ítem de los requerimientos lo puede omitir y pasar directamente a la comparativa entre dos Configuraciones que si hace parte del ítem 3.2.

Resultados por configuración

Configuración Original

Metrica	SA	ACO
Min Valor	660250.0	659114.0
Max Valor	662301.0	662301.0
Media Valor	661252.78	662123.66
Varianza Valor	1030345.2516000001	300739.50440000003
Tiempo Medio (s)	0.010969605445861817	1.8272574996948243
Varianza Tiempo	1.362902423829837	0.00568458810852726
Iter. Convergencia Media	336.76	358.22
Iter. Convergencia Varianza	139320.38239999997	52615.97160000001



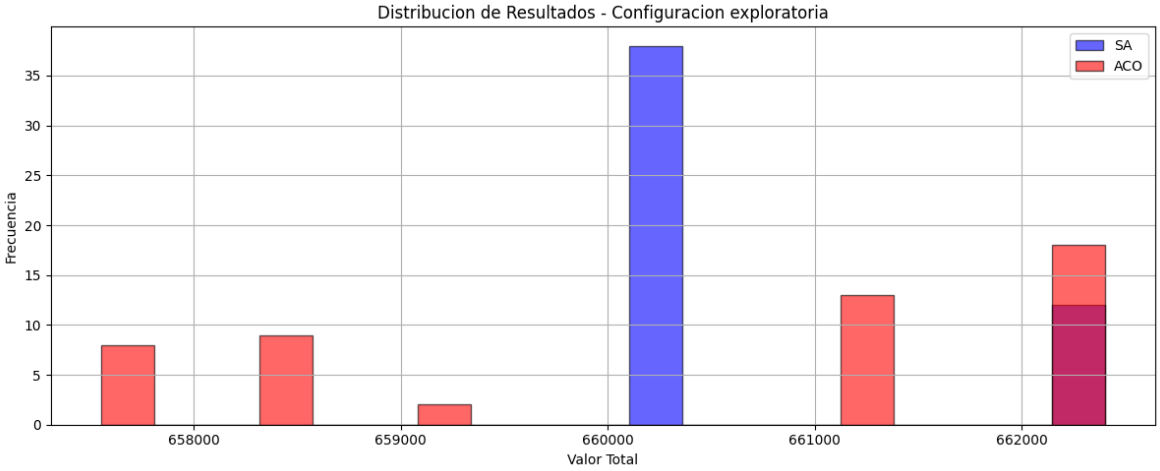


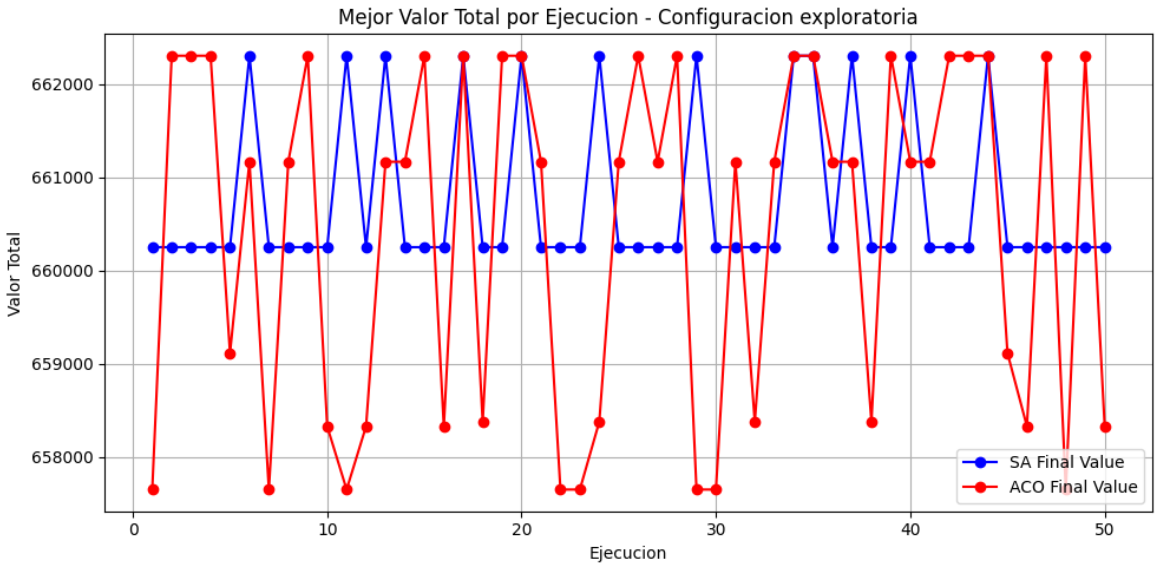
Análisis:

ACO mostró un desempeño más consistente y preciso, logrando un valor medio superior al de SA y con una varianza significativamente menor. SA también fue competitivo, pero su rendimiento fluctuó más entre ejecuciones. En cuanto a eficiencia, SA fue mucho más rápido, aunque con más riesgo de inestabilidad.

Configuración Exploratoria

Metrica	SA	ACO
Min Valor	660250.0	657653.0
Max Valor	662301.0	662301.0
Media Valor	660742.24	660423.7
Varianza Valor	767284.0223999998	3481000.45
Tiempo Medio (s)	0.015782785415649415	5.277522811889648
Varianza Tiempo	9.764149738202831e-06	0.5552764628937097
Iter. Convergencia Media	102.86	430.06
Iter. Convergencia Varianza	50641.7204	81159.01640000001



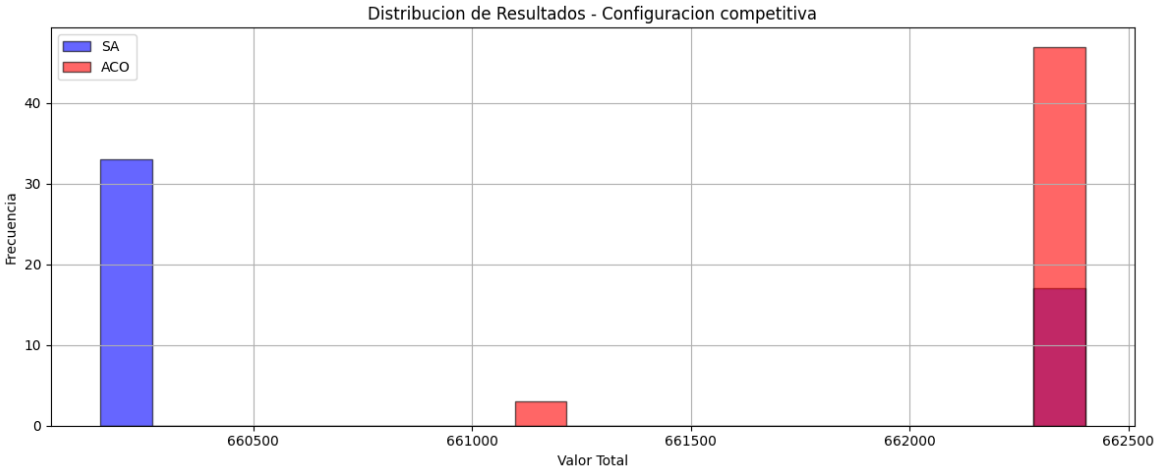


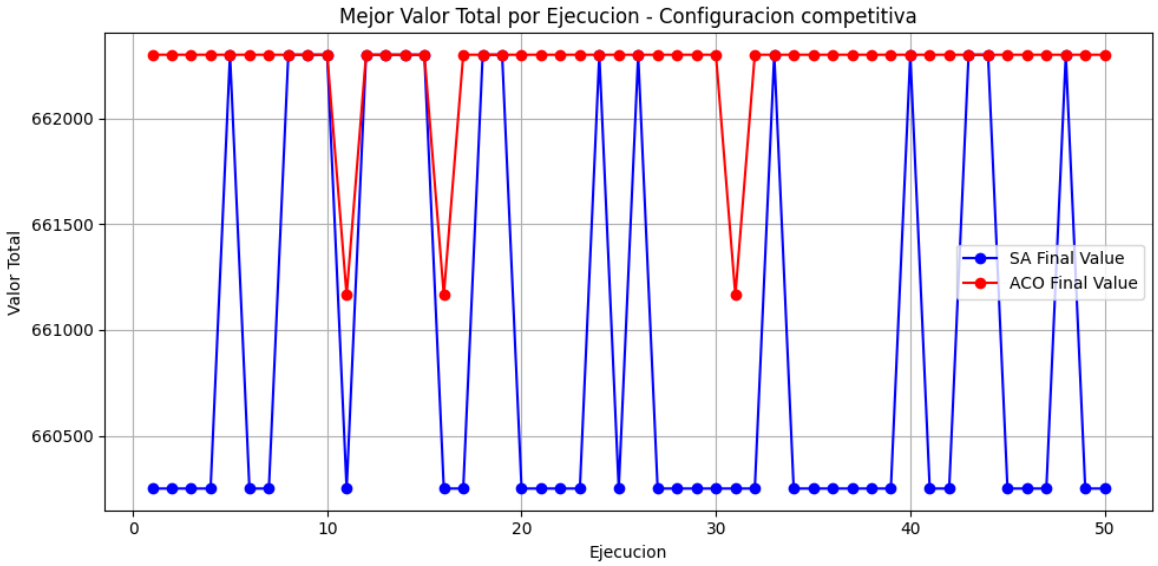
Análisis:

En esta configuración sorprendentemente SA superó a ACO en rendimiento medio y estabilidad. ACO tuvo una alta varianza y un valor mínimo mucho más bajo, lo que indica que su capacidad exploratoria produjo muchas soluciones subóptimas. Esta configuración confirmó el rol de SA como buen explorador con resultados consistentes

Configuración Competitiva

Metrica	SA	ACO
Min Valor	660250.0	661165.0
Max Valor	662301.0	662301.0
Media Valor	660947.34	662232.84
Varianza Valor	943961.2644	72783.9744
Tiempo Medio (s)	0.007911658287048 34	1.6156236600875855
Varianza Tiempo	3.4398983245864655	0.018296597683666 48
Iter. Convergencia Media	133.6	286.64
Iter. Convergencia Varianza	50210.11999999999	29061.790399999994



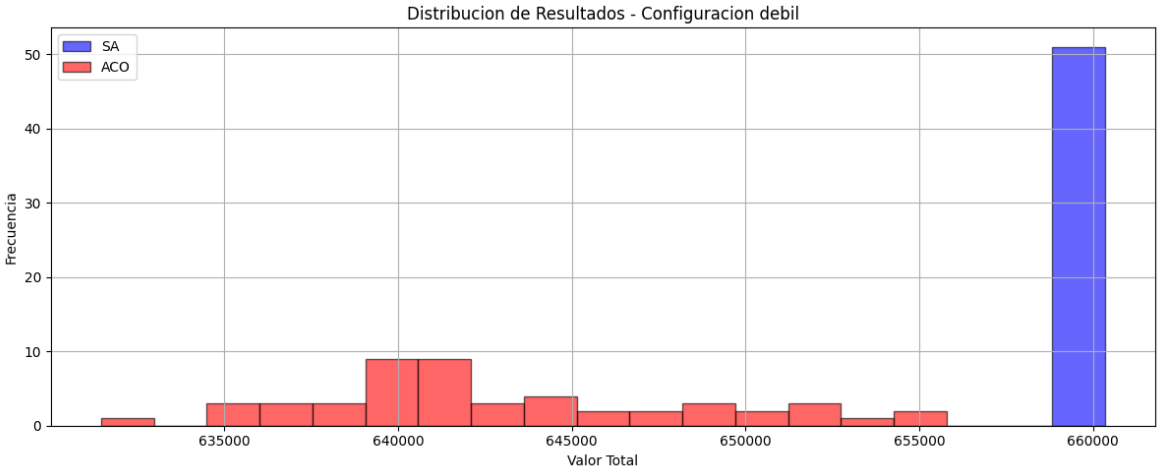


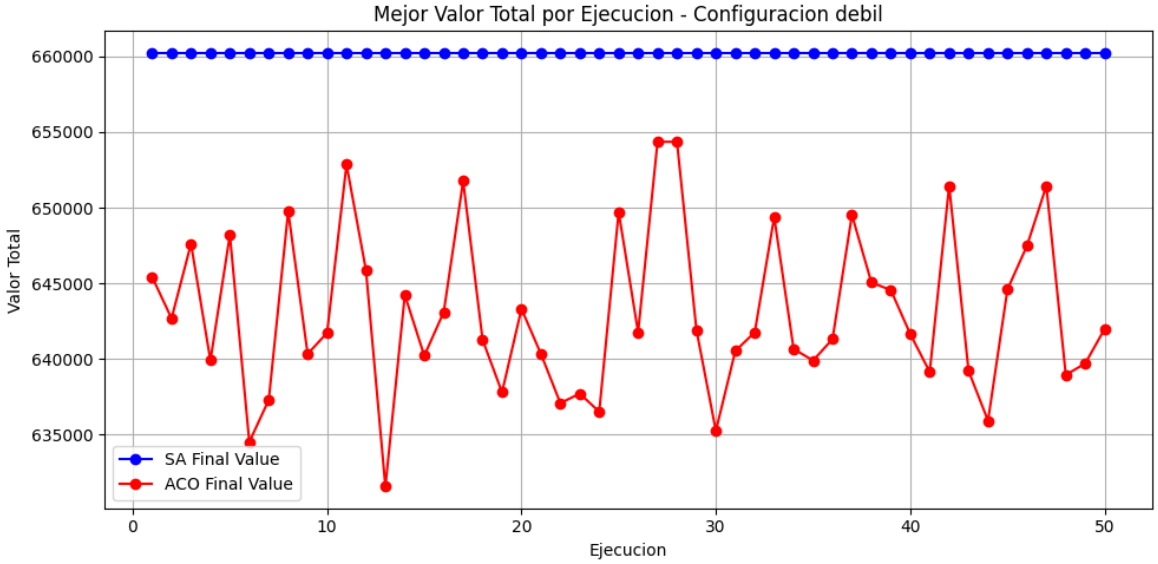
Análisis:

ACO logró en su mayoría de veces soluciones óptimas, pero en algunas ejecuciones logro valores bajo, lo cual puede demostrar que el algoritmo si está encontrando soluciones, pero pierde algo de exploración. SA, por el contrario, mostró buen rendimiento, aunque no perfecto. Nuevamente SA fue más rápido, consiguiendo picos de exploración fáciles de ver en las gráficas, lo que indica que el algoritmo está explorando bastante, mientras que ACO fue altamente preciso pero determinista.

Configuración Débil

Metrica	SA	ACO
Min Valor	660250.0	631548.0
Max Valor	660250.0	654359.0
Media Valor	660250.0	643044.04
Varianza Valor	0.0	28911050.198400002
Tiempo Medio (s)	0.001902294158935547	0.3995834255218506
Varianza Tiempo	4.219646143610589	0.0007992360180759535
Iter. Convergencia Media	1.0	163.16
Iter. Convergencia Varianza	0.0	5685.774399999999





Análisis:

SA mostró una solución fija en todas las ejecuciones, lo que sugiere que no exploró en absoluto bajo esta configuración. ACO, en cambio, produjo resultados altamente variables, con valores tanto bajos como competitivos. Esto muestra que incluso en condiciones pobres, ACO mantiene algo de capacidad exploratoria, mientras que SA queda completamente estancado.

Comparativas entre Dos Configuraciones

Como ya se explicaron todas las configuraciones que se usaron en el proyecto, pasaremos a responder lo que pide el ítem 3.2 "Por medio de gráficas y tablas resuma los resultados obtenidos. Por cada método debe mostrar los resultados obtenidos con mínimo dos configuraciones diferentes de parámetros, incluyendo una configuración con la mejor combinación de parámetros encontrado."

Se hará la comparativa entre la métrica **Original** y **Competitiva**, a continuación, se muestran los resultados obtenidos:

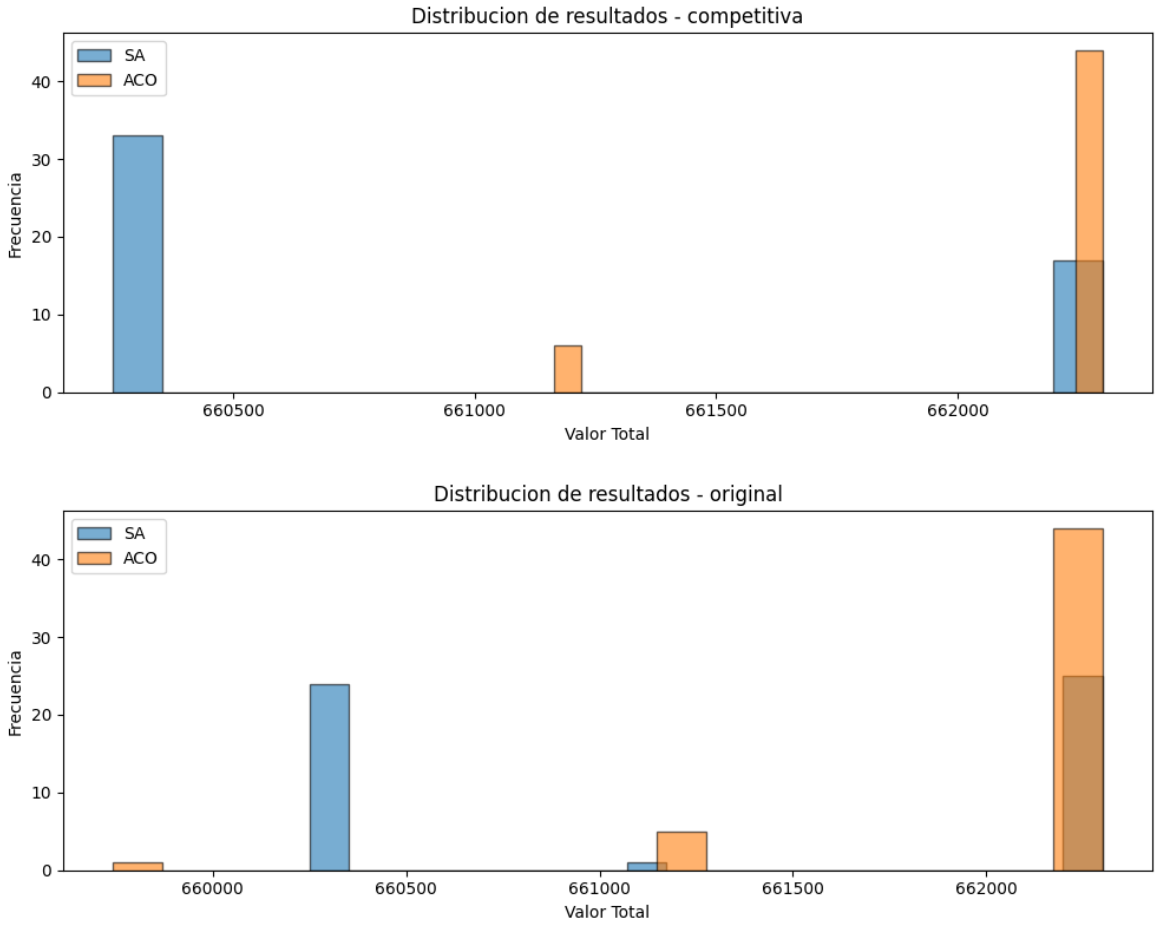
Configuración Original:

Configuración	Original	Competitiva
sa_min_	660250.0	660250.0
sa_max	662301.0	662301.0
sa_mean	661293.8	660947.34
sa_var	1030526.56	943961.2644
sa_time	0.0105s	0.0086s

Configuración Competitiva:

Configuración	Original	Competitiva
aco_min	659741.0	661165.0
aco_max	662301.0	662301.0
aco_mean	662136.2	662164.68
aco_var	232962.56	136276.3776
aco_time	1.7382s	1.7411s

Gráficos de Distribución

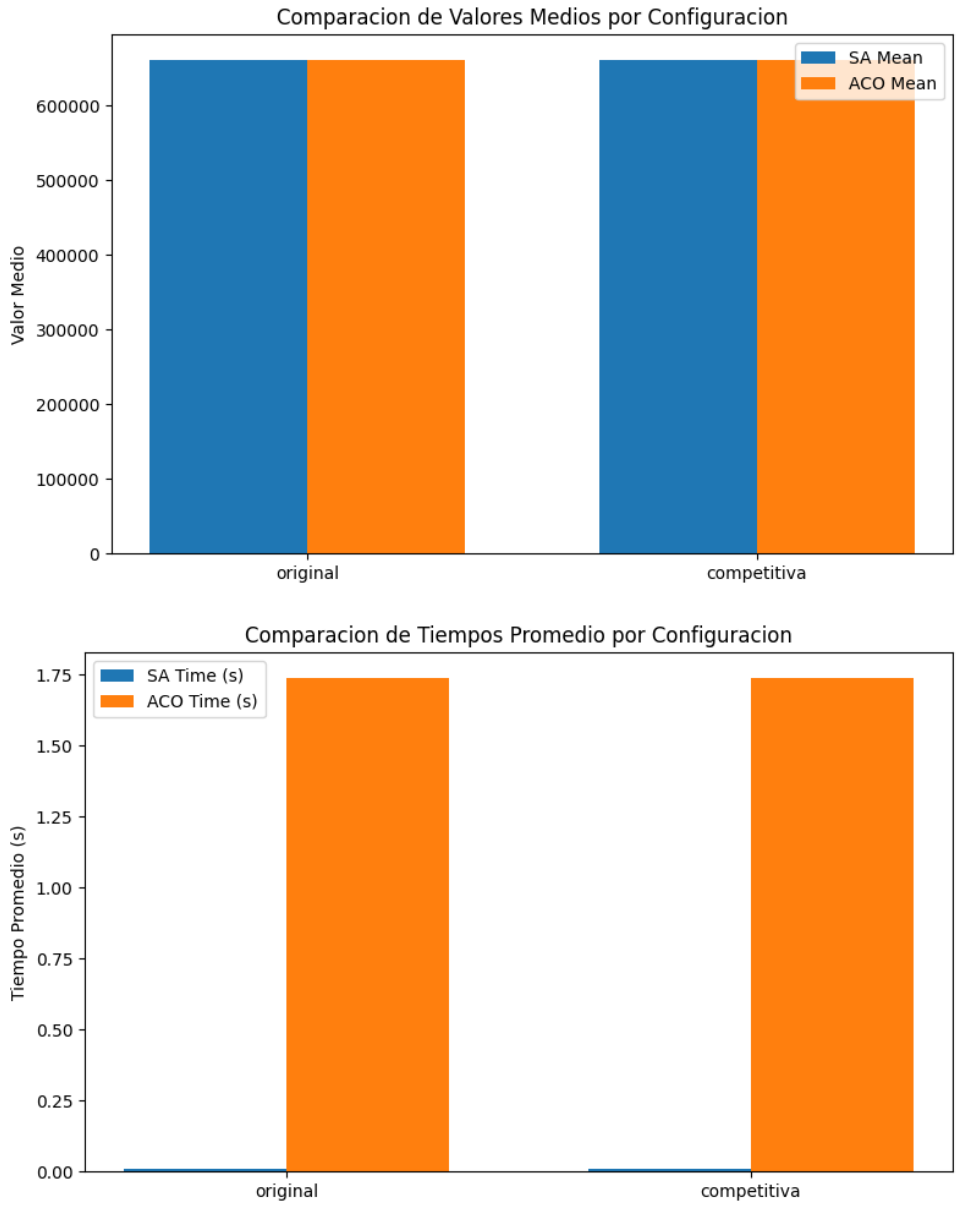


Este gráfico muestra la distribución de los valores obtenidos por cada algoritmo bajo dos configuraciones distintas (Original y Competitiva).

- En **(SA)**, ambas configuraciones están centradas en torno al valor máximo, pero la configuración *competitiva* muestra una distribución ligeramente más concentrada, lo cual se alinea con su menor varianza.
- En **ACO**, la distribución correspondiente a la configuración *competitiva* también es más estrecha y desplazada hacia valores más altos, lo que refleja su mayor consistencia y mejor valor promedio.

Este gráfico permite visualizar la dispersión y estabilidad del algoritmo, complementando los resultados numéricos de varianza y promedio.

Histograma de Valores y Tiempo Promedio por Configuración



Este gráfico compara simultáneamente el valor promedio obtenido y el tiempo promedio de ejecución de cada configuración, entre rendimiento y eficiencia temporal.

- En **SA**, se observa una clara ventaja en tiempo (por debajo de 0.011s) en ambas configuraciones, con una ligera pérdida de calidad en la configuración competitiva.
- En **ACO**, el valor promedio fue mayor en la configuración competitiva, aunque los tiempos de ejecución se mantuvieron prácticamente iguales (~1.74s).

El gráfico evidencia que SA es más rápido, mientras que ACO obtiene valores promedio superiores a costa de mayor tiempo de cómputo.

La Mejor Configuración

Simulated Annealing (SA):

- Original obtuvo una media de 661 293.8 (varianza 1 030 526) en 50 ejecuciones, frente a 660 947.3 (varianza 943 961) de la configuración competitiva.
- Aunque la versión competitiva es ligeramente más rápida (0.0086 s vs 0.0105 s) y tiene algo menos de varianza, priorizamos la calidad de la solución (media de valor). Por ello, la configuración original se considera óptima para SA.

Ant Colony Optimization (ACO)

- La configuración competitiva alcanzó la mayor media de 662 164.7 (varianza 136 276) frente a 662 136.2 (varianza 232 962) de la original.
- Además, mantiene tiempos prácticamente idénticos (1.741 s vs 1.738 s) pero con mayor consistencia y valor promedio. En ACO, la configuración competitiva es, por tanto, la óptima.

Resumiendo, SA se adapta mejor a la configuración **original** y ACO tiene mejor desempeño con la configuración **competitiva**. Es por esto que estas son las combinaciones de hiperparámetros que maximizan el valor promedio y equilibran estabilidad y eficiencia.

Test con estos hiperparametros

SA – Original

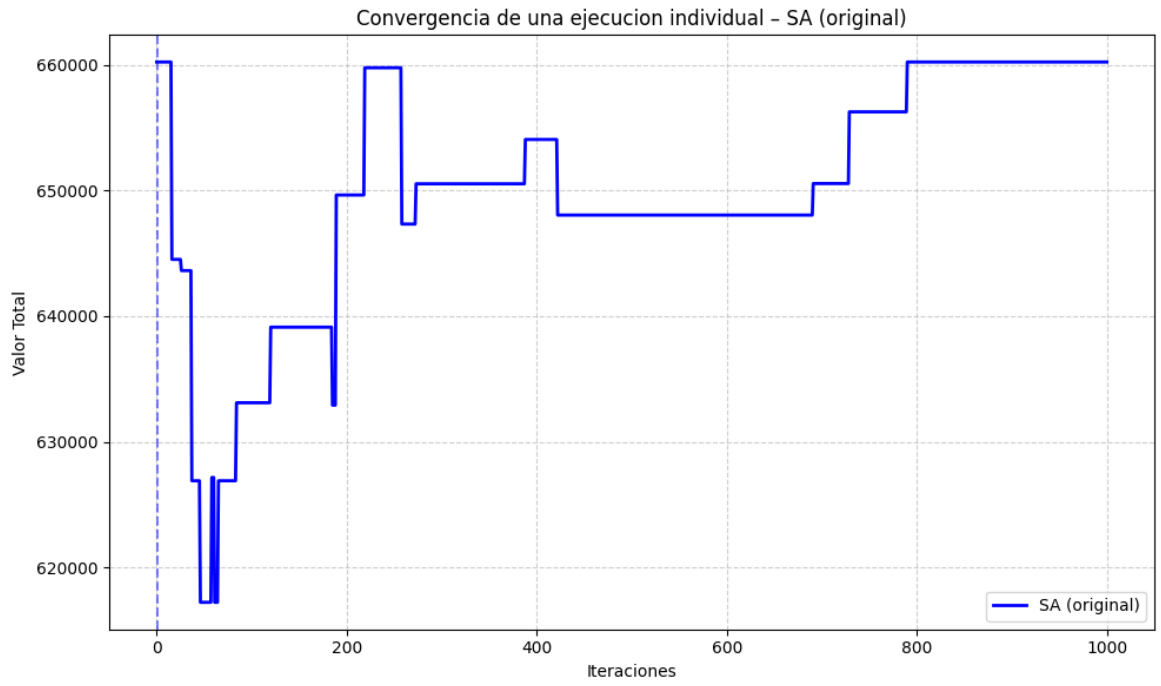
Analicemos los resultados obtenidos

Item	Cantidad	Peso	Valor
2	2	1.034	40 954
3	7	1.457	38 964
4	5	1.0	47 814
9	1	1.007	25 050
13	1	1.566	41 474
Total	16	19.84	660 250

Necesito las siguientes iteraciones y tiempo

Iteraciones	1000
Convergencia	1
Tiempo (s)	0.011

Visualicemos la convergencia del método



El algoritmo logró generar una solución, alcanzando un valor total de **660,250** con un peso total de **19.84 kg**, justo por debajo del límite de capacidad. La mejor solución se obtuvo en solo 1 iteración**,** lo que indica que el diseño de soluciones es bueno. El tiempo de ejecución fue muy bajo (0.011 segundos), Como apreciamos en la gráfica, hizo una exploración, pero encontrando pocas veces mejores soluciones a la solución inicial.

ACO – Competitiva

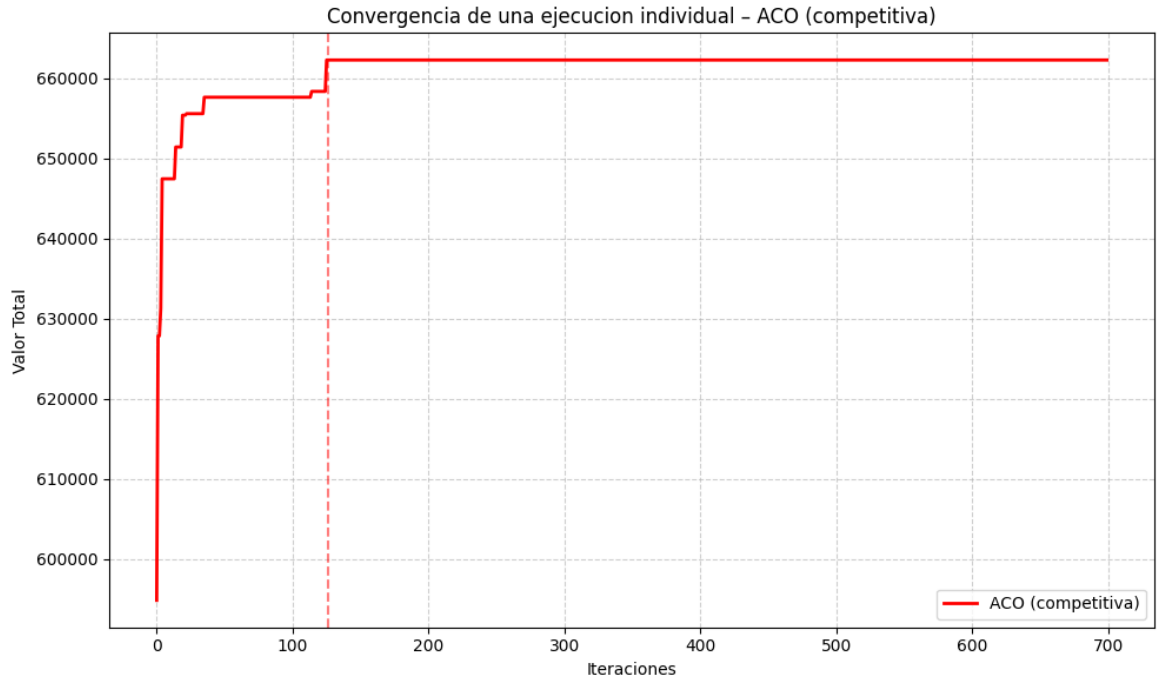
Analicemos los resultados obtenidos

Item	Cantidad	Peso (kg)	Valor
2	2	1.034	40 954
3	6	1.457	38 964
4	5	1.0	47 814
11	1	1.176	31 072
12	1	1.391	34 993
13	1	1.566	41 474
Total	16	19.94	662 301

Necesito las siguientes iteraciones y tiempo

Iteraciones	700
Convergencia	126
Tiempo (s)	1.496

Visualicemos la convergencia del método



Este algoritmo alcanzó una solución de alto valor (**662,301**) con un peso total de **19.94 kg**, utilizando eficientemente la capacidad de la mochila. La convergencia se logró en **126 iteraciones**, y el tiempo de ejecución fue de **1.496 segundos**. Estos resultados reflejan una construcción efectiva de soluciones guiada por feromonas e información heurística, mostrando un buen desempeño, como se aprecia en la gráfica, hace una exploración muy buena hasta llegar a la convergencia.

Observaciones sobre el dataset utilizado

El dataset no es complejo y es por eso que se llegan a soluciones tan rápidamente y cada uno de los algoritmos llega a encontrar una solución optima, acá hay algunas de las razones por las que el dataset no representa complejidad**:**

1. **Espacio de búsqueda limitado:** La cantidad máxima por objeto es baja, por lo que el número de combinaciones posibles no es excesivamente alto. Esta es una razón por las cuales los algoritmos llegan a las mismas soluciones, sin necesidad de muchas iteraciones.
2. **Relación valor/peso clara:** Muchos objetos tienen eficiencia obvia, lo que facilita que heurísticas simples como valor/peso guíen hacia buenas soluciones.
3. **Pesos homogéneos:** No existen objetos que sobresalgan por un peso exageradamente alto o bajo, lo cual hace más predecible la selección.

Esta estructura del dataset permite que los modelos converjan rápidamente a soluciones óptimas. Es ideal para pruebas, validación de algoritmos y comparaciones de rendimiento como supongo que es el objetivo de este trabajo, por eso hay que decir que este dataset **NO** representa un reto extremo en optimización combinatoria para ninguno de los dos algoritmos.

Análisis de Convergencia y Desempeño de los Métodos

Con el objetivo de evaluar de manera más precisa el comportamiento de los métodos implementados, se realizó el análisis estadístico requerido en el ítem 3.3. Cada método fue ejecutado 50 veces de forma independiente, lo que permite observar no solo su capacidad de convergencia, sino también la estabilidad y robustez de los resultados obtenidos.

Durante estas ejecuciones, se recopilaron datos clave como:

- Solución promedio encontrada.
- Número promedio de iteraciones necesarias para converger.
- Tiempo promedio de ejecución.
- Varianza de cada uno de estos indicadores.

Para el análisis de convergencia se seleccionó la **Configuración Competitiva** para ambos métodos, dado que mostraron la mejor combinación entre estabilidad, calidad de solución y eficiencia. A continuación, se presentarán los resultados de cada uno de estos algoritmos con dicha configuración.

Configuración Competitiva

Metrica	SA	ACO
Min Valor	660 250.0	661 165.0
Max Valor	662 301.0	662 301.0
Media Valor	660 947.34	662 232.84
Varianza Valor	943 961.2644	727 83.9744
Tiempo Medio (s)	0.00791s	1.61562s
Varianza Tiempo	3.439898	0.018296
Iter. Convergencia Media	133.6	286.64
Iter. Convergencia Varianza	50210.1199	29061.79039

A continuación, se muestran los resultados de la mejor Solución encontrada por el algoritmo:

SA:

Item	Cantidad	Peso	Valor
2	2	1.03	40954
3	6	1.46	38964
4	5	1.00	47814
11	1	1.18	31072
12	1	1.39	34993
13	1	1.57	41474
Total	16	19.94	662301
Iter.Conv.	108		
Tiempo(s)	0.007		

ACO:

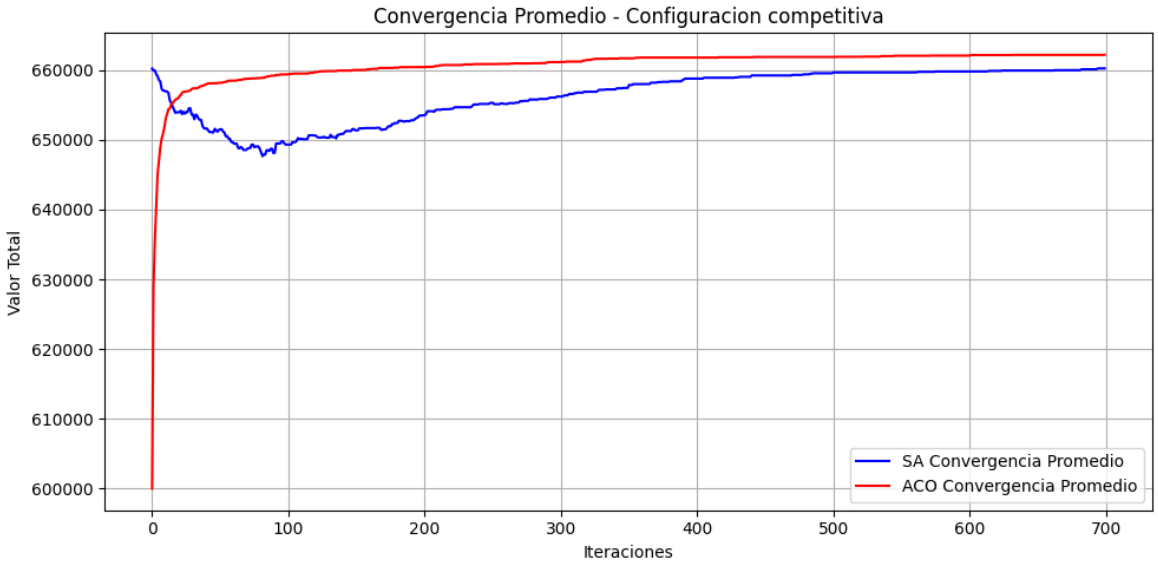
Item	Cantidad	Peso	Valor
2	2	1.03	40954
3	6	1.46	38964
4	5	1.00	47814
11	1	1.18	31072
12	1	1.39	34993
13	1	1.57	41474
Total	16	19.94	662301
Iter.Conv.	167		
Tiempo(s)	1.551		

Ambos algoritmos encontraron la misma solución en términos de los elementos seleccionados, su cantidad, peso y valor totales:

- **Cantidad total de ítems:** 16
- **Peso total:** 19.94
- **Valor total:** 662,301

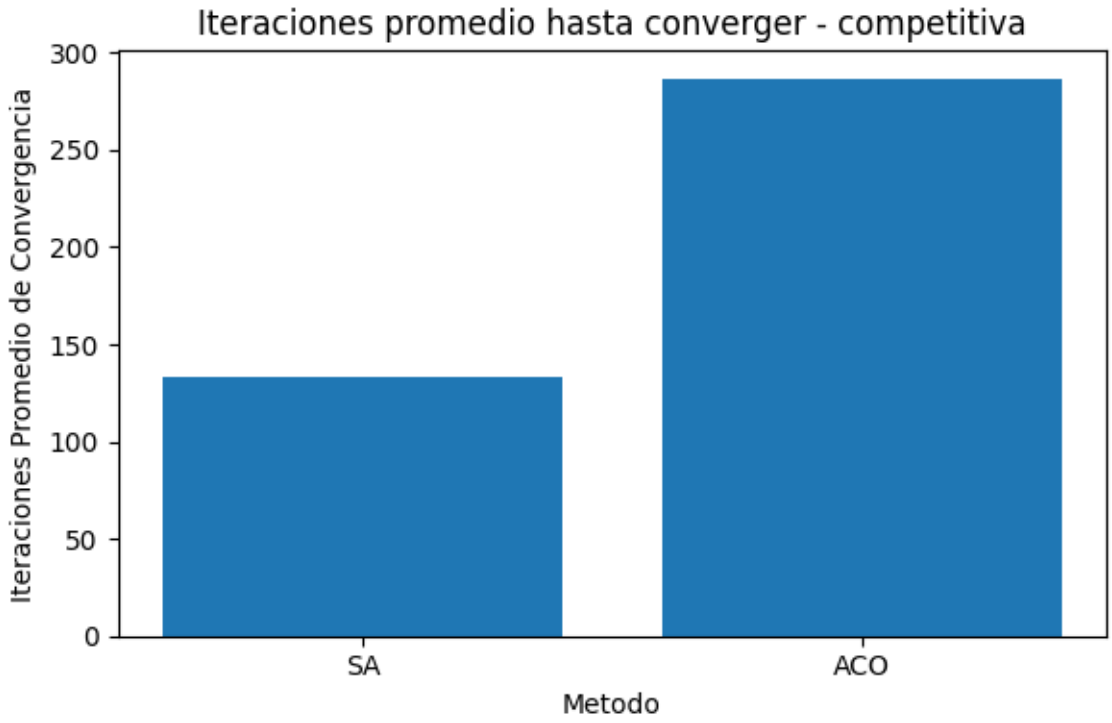
Esto indica que tanto SA como ACO fueron capaces de alcanzar una solución óptima o muy cercana al óptimo para el problema propuesto.

Grafica de convergencia



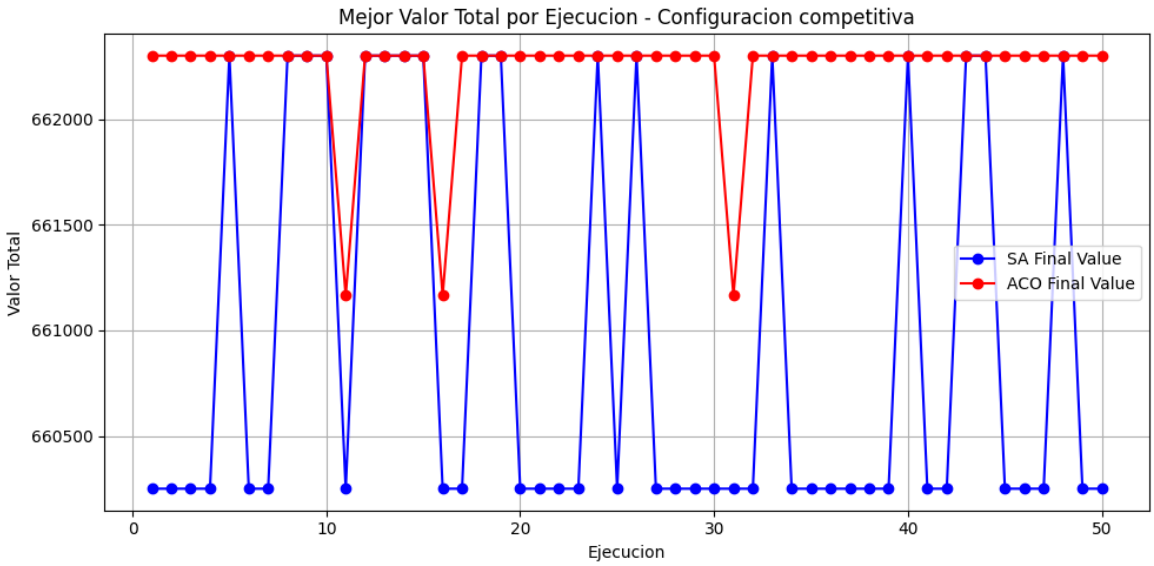
Recordemos que en la configuración competitiva cada uno de los algoritmos hace **700** iteraciones internas para hallar su resultado, esta grafica muestra la convergencia promedio de las **50** iteraciones que se hicieron sobre cada uno de estos algoritmos.

Iteraciones Promedio hasta Converger



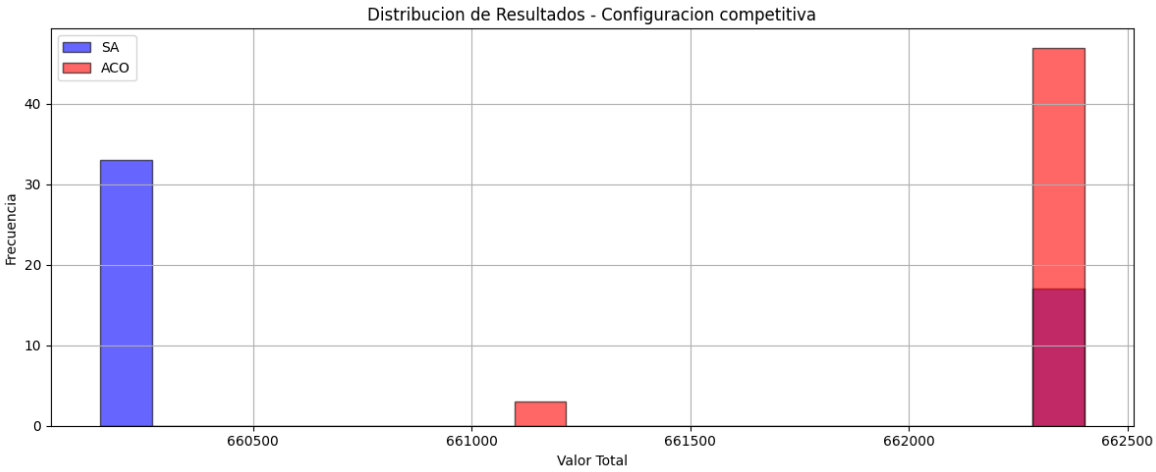
En esta grafica se muestran las iteraciones que tardan en promedio cada uno de los algoritmos hasta llegar a la convergencia.

Mejores Valores en las 50 Iteraciones



Podemos apreciar que a pesar de que se vean picos bruscos en SA, los rangos de valores en los que estos se encuentran no soy muy grandes ≈ 1500, esto nos demuestra que el algoritmo siempre está en constante búsqueda de los resultados y muy cerca de estos valores deseados. Con respecto ACO, por lo general se encuentra en la solución optima del problema a excepción de en 3 iteraciones, donde se encuentra con pequeños picos (no tan bruscos como los de SA), demostrando robustes y poca variabilidad.

Distribución de los Resultados



Respecto a la distribución, podemos apreciar que los valores totales de ACO tienen su mayor frecuencia en los resultados óptimos alcanzando por lo menos más de 40 veces el mejor resultado, en cambio SA la mayoría de sus valores se encuentran muy por debajo de ese optimo, poco menos de 20 veces llego a conseguir el optimo esperado, demostrando que muy posiblemente en la mayoría de las iteraciones se haya quedado atrapado en locales.

Conclusión sobre este apartado

Como se evidencia en los resultados, ambos métodos logran encontrar la mejor solución posible en múltiples ejecuciones. No obstante, **Ant Colony Optimization** destaca por su mayor consistencia (menor varianza en la solución), aunque a costa de un mayor tiempo y número de iteraciones promedio.

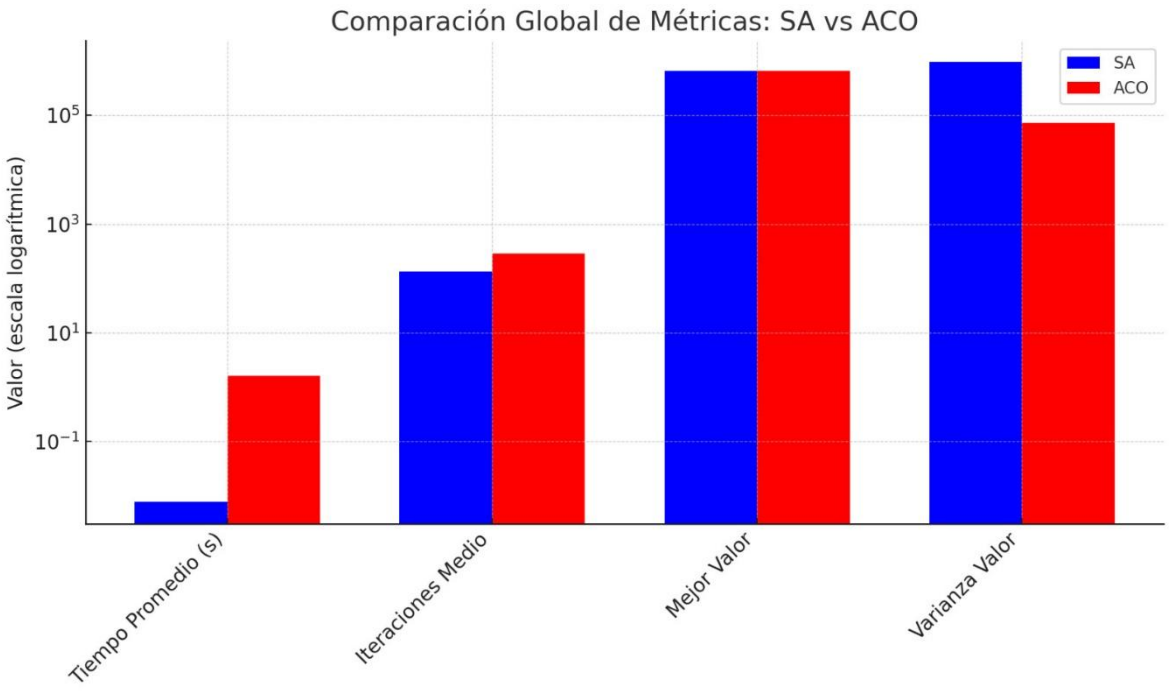
Por su parte, **Simulated Annealing** se caracteriza por ser más rápido y converger con menos iteraciones, aunque con una mayor dispersión en los resultados.

Gracias a esto permite nos concluir que **ACO** es más robusto, mientras que **SA** es más eficiente computacionalmente, siendo la elección del algoritmo dependiente de las restricciones del problema (tiempo vs. estabilidad). Como en el aparatado el trabajo se especifica cual es la métrica principal. Se dejará a la elección del lector, dados los argumentos elegir la mejor opción, dependiendo de los debidos argumentos presentados con anterioridad.

Comparación General entre Métodos

A continuación, se presenta una comparación directa entre los dos métodos evaluados: **Simulated Annealing (SA)** y **Ant Colony Optimization (ACO)**, con base en los resultados obtenidos con la mejor configuración de parámetros (Configuración Competitiva).

Métrica	Simulated Annealing (SA)	Ant Colony Optimization (ACO)
Mejor Solución Encontrada	662301	662301
Tiempo Promedio de Ejecución	0.0079 s	1.6156 s
Iteraciones Promedio	133.6	286.64
Varianza de la Solución	943961.26	72783.97
Varianza del Tiempo	3.44	0.018
Varianza de Iteraciones	50210.12	29061.79



La comparación entre **SA** y **ACO** revela que ambos métodos tienen la capacidad de alcanzar soluciones óptimas, logrando en múltiples ejecuciones el valor máximo del problema (662301). No obstante, lo hacen con enfoques y desempeños diferenciados, lo cual permite mostrar claramente sus ventajas y limitaciones en función del criterio de evaluación.

Desde una perspectiva de **eficiencia computacional**, SA se posiciona como el método más adecuado: presenta un tiempo promedio de ejecución extremadamente bajo (0.0079 s), y requiere menos iteraciones en promedio para alcanzar la convergencia. Esta combinación entre velocidad y eficiencia lo hace altamente recomendable en momentos donde se necesite una solución con mayor velocidad y menor tiempo de espera, a costa de perder algo de precisión.

Por otro lado, ACO demuestra una mayor robustez y estabilidad en sus resultados. Aunque su tiempo de ejecución es significativamente mayor (alrededor de 1.6 s en promedio), su baja varianza en la solución encontrada (72783.97 frente a 943961.26 de SA) indica que es menos sensible al azar y más consistente a lo largo de múltiples ejecuciones. Esto lo convierte en una opción preferente en escenarios donde la fiabilidad de los resultados es crítica, incluso si ello implica un mayor costo computacional.

Adicionalmente, la varianza en **el número de iteraciones** también refleja esta diferencia: ACO muestra una menor dispersión en sus diseños sobre los espacios de búsqueda, lo que sugiere un comportamiento más controlado y menos propenso al error. SA, en contraste, aunque rápido, puede presentar errores mucho mas marcados en sus caminos de exploración.

En conclusión:

- **Simulated Annealing** es ideal cuando se prioriza la velocidad de ejecución y el uso eficiente de recursos, siendo capaz de obtener soluciones de alta calidad en muy poco tiempo, a costa de cierta variabilidad.
- **Ant Colony Optimization** es más adecuado cuando se **busca estabilidad y menor sensibilidad a las condiciones iniciales,** garantizando resultados consistentes aunque con mayor demanda de tiempo y recursos computacionales.

La elección del método dependerá, por tanto, del contexto del problema y las prioridades del sistema en el que se implemente. Para este trabajo, cualquiera de las dos es una opción válida, ya que como se explico en un apartado, el dataset no es muy complejo, tiene pocos datos y los valores de los ítems están dentro de los rangos adecuados para satisfacer la capacidad máxima permitida, es por ende, que la elección de cualquiera de los dos métodos es completamente valida y funcional.