TECNOLOGÍA: RESULTADOS DE INVESTIGACIÓN

DOI: https://doi.org/10.35588/jyravv64

Evaluación del Rendimiento de un Algoritmo Genético según la Configuración de Parámetros en el Problema de la Mochila 0-1

Performance Evaluation of a Genetic Algorithm Based on Parameter Configuration in the 0-1 Knapsack Problem

Avaliando o desempenho de um algoritmo genético baseado em configurações de parâmetros no problema da mochila 0-1

Edición Nº53 – Agosto de 2025

Artículo Recibido: Marzo 10 de 2025

Aprobado: Agosto 04 de 2025

Autores:

Gustavo Alcántara-Aravena[1] y Camilo Acuña-Porras[2]

Resumen:

La experimentación con diferentes configuraciones del algoritmo genético reveló que una baja probabilidad de mutación, combinada con un tamaño de torneo adecuado, mejora significativamente el rendimiento del algoritmo. Esto se refleja en la reducción del GAP promedio y la maximización del valor objetivo alcanzado, estableciendo así parámetros de referencia cuantificables para implementaciones eficientes del problema de la mochila 0-1. La novedad del estudio radica en ofrecer directrices concretas para aplicaciones prácticas, identificando relaciones consistentes entre la configuración de parámetros y el comportamiento del algoritmo. Si bien los AG son

[1] Dr©, Universidad de Santiago de Chile. Santiago, Chile. Correo electrónico: gustavo.alcantara@usach.cl, https://orcid.org/0000-0001-6587-0216

[2] Dr., Pontificia Universidad Católica de Chile. Santiago, Chile. Correo electrónico: cacuna2@uc.cl, https://orcid.org/0000-0002-6811-6240

eficaces en la exploración de grandes espacios de búsqueda en problemas NP-completos, este trabajo demuestra que requieren una calibración precisa para evitar la convergencia prematura. Se constató que la probabilidad de mutación y el tamaño del torneo influyen directamente en las dinámicas exploratoria y explotadora del algoritmo. En general, combinaciones con baja mutación y torneos intermedios ofrecieron un buen equilibrio entre calidad de solución y estabilidad, mientras que valores extremos (mutación de 0.20 o torneo k = 2/k=4) incrementaron la variabilidad del rendimiento, especialmente en problemas de mayor escala. Estos hallazgos evidencian la sensibilidad del algoritmo a dichos parámetros y destacan la importancia de ajustes cuidadosos para preservar su eficacia ante escenarios complejos.

Palabras clave: Algoritmo genético, Problema mochila 0-1, Configuración de parámetros, Optimización Combinatoria.

Abstract:

Experimentation with different configurations of the genetic algorithm revealed that a low mutation probability, combined with an appropriate tournament size, significantly improves the algorithm's performance. This is reflected in the reduction of the average GAP and the maximization of the achieved objective value, thereby establishing quantifiable benchmark parameters for efficient implementations of the 0-1 knapsack problem. The novelty of the study lies in providing concrete guidelines for practical applications, identifying consistent relationships between parameter configuration and algorithm behavior. While genetic algorithms are effective in exploring large search spaces in NP-complete problems, this work demonstrates that they require precise calibration to avoid premature convergence. It was found that the mutation probability and tournament size directly influence the algorithm's exploratory and exploitative dynamics. In general, combinations with low mutation and intermediate tournament sizes offered a good balance between solution quality and stability, whereas extreme values (mutation of 0.20 or tournament size k = 2/k = 4) increased performance variability, especially in larger-scale problems. These findings highlight the algorithm’s sensitivity to these parameters and underscore the importance of careful tuning to preserve its effectiveness in complex scenarios.

Keywords: Genetic algorithm, 0-1 knapsack problem, Parameter configuration, Combinatorial Optimization.

Resumo:

A experimentação com diferentes configurações do algoritmo genético revelou que uma baixa probabilidade de mutação, combinada com um tamanho de torneio adequado, melhora significativamente o desempenho do algoritmo. Isso se reflete na redução do GAP médio e na maximização do valor objetivo alcançado, estabelecendo assim parâmetros de referência quantificáveis para implementações eficientes do problema da mochila 0-1. A novidade do estudo reside em oferecer diretrizes concretas para aplicações práticas, identificando relações consistentes entre a configuração de parâmetros e o comportamento do algoritmo. Embora os algoritmos genéticos sejam eficazes na exploração de grandes espaços de busca em problemas NP-completos, este trabalho demonstra que eles requerem uma calibração precisa para evitar a convergência prematura. Constatou-se que a probabilidade de mutação e o tamanho do torneio influenciam diretamente nas dinâmicas exploratória e exploradora do algoritmo. De modo geral, combinações com baixa mutação e torneios intermediários ofereceram um bom equilíbrio entre qualidade da solução e estabilidade, enquanto valores extremos (mutação de 0,20 ou torneio k = 2/k = 4) aumentaram a variabilidade do desempenho, especialmente em problemas de maior escala. Esses achados evidenciam a sensibilidade do algoritmo a tais parâmetros e destacam a importância de ajustes cuidadosos para preservar sua eficácia em cenários complexos.

Palavras-chave: Algoritmo genético, Problema da mochila 0-1, Configuração de parâmetros, Otimização combinatória.

1. Introducción

El problema de la mochila 0-1 constituye uno de los casos más paradigmáticos dentro de la optimización combinatoria, debido a su clasificación como problema NP-completo. Esta complejidad implica que, a medida que aumenta el tamaño y la dificultad de las instancias, la utilización de métodos exactos se vuelve computacionalmente prohibitiva (Pisinger, 2005).

Ante esta limitación, los algoritmos genéticos (AG) han emergido como una alternativa eficiente para abordar este tipo de problemas, gracias a su capacidad para explorar

espacios de búsqueda de alta dimensionalidad y localizar soluciones cercanas al óptimo en tiempos de cómputo razonables.

Tal como se ha evidenciado en investigaciones previas, la naturaleza intrínseca de NP-completitud y la dificultad para hallar soluciones óptimas en espacios extensos hacen del problema de la mochila 0-1 un candidato idóneo para enfoques heurísticos como los AG, los que han demostrado ser eficaces en este contexto. Por ejemplo, Khuri, Bäck y Heitkotter (1994) reportaron resultados alentadores en instancias de gran escala, lo que indica que los AG pueden emplearse exitosamente como heurísticas efectivas para abordar problemas NP completos altamente restringidos. Asimismo, Pradhan, Israni y Sharma (2014) resaltaron la eficacia de un AG hibridado con la teoría de conjuntos aproximados, destacando que proporcionan una manera de resolver satisfactoriamente el problema de la mochila en complejidad temporal lineal y que incluir la teoría de conjuntos aproximados se mejora la eficiencia y calidad de búsqueda.

La implementación de los AG durante décadas ha dependido de expertos y académicos que en base a su conocimiento y experiencia han innovado y desarrollado nuevos métodos para abordar el problema de la mochila 0-1, en ese sentido, Ezziane (2002) introdujo un algoritmo genético con función de penalización auto-adaptativa en el contexto de la mochila 0-1, logrando una notable disminución de soluciones inviables mejorando la calidad de las soluciones en comparación con un enfoque de penalización constante. No obstante, este estudio no exploró cómo la penalización interactúa con otros parámetros del algoritmo.

Por otro lado, la diversidad poblacional ha sido abordada mediante la estrategia de doble población, destacando, Umbarkar y Joshi (2014), quienes desarrollaron una variación de un AG conocida como *Dual Population Genetic Algorithm* (*DPGA*) que emplea dos poblaciones evitando así la convergencia prematura y conservar diversidad, superando de manera empírica al AG estándar en diferentes instancias. Aún así, no se profundizó en cómo el tamaño de cada población influye en el rendimiento general del sistema. Complementariamente, Yan, Guo, Li y He (2013) propusieron una versión del *DPGA*, *Dual Population Adaptative Genetic Algorithm*

(*DPAGA*), introduciendo operadores no lineales que mejoran la convergencia, aunque también se mantuvo ausente un análisis paramétrico riguroso.

En cuanto a los operadores de cruce y selección, Owais, Alkhazendar y Saleh (2020) realizaron una comparación empírica entre cruces de 1 y 2 puntos, así como métodos de selección (torneo, ruleta, *ranking*), concluyendo que la combinación de cruce de un punto y selección por torneo ofrece la mayor tasa de convergencia. Sin embargo, su estudio omitió efectos derivados de variaciones en parámetros como el tamaño de población o la tasa de mutación.

Más recientemente, se han abordado límites teóricos para la mutación. Yang (2024) propuso una cota teórica para la probabilidad de mutación en el AG aplicado al problema de la mochila 0-1, acompañado de un operador de mutación mejorado, que aumenta la probabilidad de alcanzar la solución óptima en una sola iteración.

En la línea de estructuras híbridas con objetivos auxiliares, He, He y Dong (2014) diseñaron un AG con objetivos de ayuda que supera a los AG tradicionales y a métodos voraces en cuanto a calidad de solución para el problema de la mochila 0-1. A pesar de sus resultados sólidos, el estudio no desglosó el impacto de parámetros como la tasa de mutación o el tamaño de población en el desempeño final del algoritmo.

En la línea de los métodos híbridos, Malanthara y Kale (2025) innovaron con un enfoque que combina algoritmos *Firefly* y AG para el problema de la mochila 0-1. Este enfoque híbrido —denominado *FAGA*— demostró mejoras en precisión y eficiencia frente a variantes tradicionales. No obstante, los autores no realizan un análisis detallado sobre cómo los ajustes en parámetros del AG (como tasa de cruce o tamaño de población) influyen en el rendimiento global del algoritmo.

Asimismo, Ahmed y Younas (2011) desarrollaron un enfoque híbrido de AG con programación dinámica para variantes de la mochila 0-1, utilizando transposición en reemplazo del cruce tradicional. Los resultados muestran soluciones cercanas al óptimo con menor tiempo de cómputo, aunque no se evalúa el impacto de cambios en parámetros del AG (como tamaño de población o tasa de mutación).

En cuanto a la inspiración cuántica, Kim, Kim y Han (2006) desarrollan un algoritmo denominado *Quantum-Inspired Multiobjective Evolutionary Algorithm* (*QMEA*), inspirado en un AG multiobjetivo (NSGA-II), para optimizar simultáneamente costo y beneficio en la mochila 0-1. Sus resultados muestran mejor proximidad al frente de Pareto y diversidad poblacional respecto al AG NSGA-II tradicional. Sin embargo, no abordan cómo variaciones en tamaño de población, rango de mutación o parámetros relacionados con q-bits influyen en el rendimiento del algoritmo.

En un contexto de constante desarrollo, los algoritmos cuánticos-evolucionarios han incorporado controles adaptativos sobre operadores como mutación y cruce. En particular, Hota y Pat (2010) introdujeron el *Adaptive Quantum-inspired Differential Evolution* (AQDE) para el problema de la mochila 0-1. Este algoritmo integra operadores inspirados en la computación cuántica con mecanismos adaptativos para mutación y cruce, y demuestra una superioridad notable frente a QEA y DE discretos en instancias estándar, incluso con poblaciones de tamaño variable. No obstante, el estudio no profundiza en el impacto de los parámetros del AG sobre su rendimiento. Finalmente, en el terreno de los híbridos multimodales, Raidl (1998) presentó un AG mejorado para el problema de la mochila con restricción múltiple, incorporando relajación de programación lineal y operadores de reparación que preservan la diversidad poblacional. El diseño demostró avances en convergencia y calidad de solución, aunque sin analizar el efecto de cambios en parámetros del AG.

A pesar de los avances en técnicas como penalización adaptativa, diversidad poblacional, operadores cuánticos, programación dinámica y metaheurísticas híbridas aplicadas al problema de la mochila 0-1, no se han identificado estudios que realicen un análisis sistemático y multifactorial sobre cómo la configuración de parámetros de un algoritmo genético, tales como el tamaño de población, el tipo y la tasa de cruce, el método de selección y la probabilidad de mutación, incide en su rendimiento global. Aunque existen investigaciones centradas en componentes específicos, como los operadores genéticos analizados por Owais, Alkhazendar y Saleh (2020) o los mecanismos adaptativos cuánticos propuestos por Hota y Pat (2010), aún falta una evaluación integrada que considere simultáneamente múltiples parámetros, distintos

tipos de instancias y métricas clave, entre ellas la calidad de solución, el tiempo de cómputo y la diversidad poblacional.

Esta carencia evidencia una oportunidad de investigación, la cual se aborda en el presente estudio mediante la propuesta de un protocolo de evaluación paramétrico sistemático que permite caracterizar de forma robusta cómo los ajustes simultáneos en los parámetros de un AG afectan su eficacia, eficiencia y adaptabilidad frente a diversas instancias del problema de la mochila 0-1.

El objetivo de este artículo es implementar y evaluar el rendimiento de un AG como enfoque metaheurístico para resolver seis instancias representativas del problema de la mochila 0-1.

2. Antecedentes de la investigación

2.1 Problema de la mochila 0-1

El problema de la mochila 0-1 consiste en seleccionar elementos de una lista para insertarlos en una mochila con el objetivo de maximizar el valor total de estos elementos sin exceder la capacidad máxima de la mochila. Cada elemento tiene un peso y un valor asociados; el peso indica cuánto espacio ocupa en la mochila, Mientras que el valor representa la utilidad o beneficio que Proporciona al ser transportado en ella. En cuanto al 0-1, alude a la restricción del problema, implica que no se pueden tomar partes fraccionadas de los elementos, en otras palabras, cada elemento se coloca completamente o no se coloca en la mochila.

Dado un conjunto de n elementos, donde cada elemento i tiene un peso w_i y un valor v_i, y una capacidad máxima de la mochila W, el problema consiste en determinar el subconjunto de elementos que maximiza la función objetivo definida en la Ecuación 1 (ver **Figura N°1**), cumpliendo con la restricción de capacidad.

Figura N°1. Ecuación 1 Formulación del problema.

$$\text{maximizar} \quad \sum_{i=1}^{n} valor_i \cdot x_i$$

$$\text{sujeto a} \quad \sum_{i=1}^{n} peso_i \cdot x_i \leq W \quad , \quad x_i \in \{0,1\}$$

$$\text{para todo } i = 1, 2, \ldots, n$$

Fuente: Elaboración propia.

Donde x_i es una variable binaria que indica si el elemento i se incluye en la mochila (1 si se incluye, 0 si no se incluye). Es relevante indicar los siguientes aspectos:

- **Complejidad del problema:** El problema de la mochila 0-1 es NP-completo, lo que implica que no existe un algoritmo conocido que pueda resolverlo eficientemente para todos los casos en tiempo polinomial.
- **Aplicaciones prácticas:** Este problema encuentra aplicaciones cruciales en diversos campos como la logística, la economía y la ingeniería, desde la optimización de la carga en vehículos de transporte hasta la selección de inversiones bajo restricciones de riesgo. El problema de la mochila 0-1 aborda desafíos fundamentales como la asignación de recursos limitados para maximizar el rendimiento.
- **Algoritmos de resolución avanzados:** Además de los clásicos algoritmos como la programación dinámica y el algoritmo *greedy*, las metaheurísticas proporcionan un enfoque atractivo. Los algoritmos genéticos inspirados en la evolución biológica permiten una búsqueda global y diversificada del espacio de soluciones. Asimismo, técnicas como el *simulated annealing* y la búsqueda tabú ofrecen estrategias robustas para explorar y explotar soluciones óptimas en problemas complejos como el de la mochila 0-1.
- **Variantes y extensiones:** Existen diversas variantes como el problema de la mochila fraccionada que permite tomar porciones de elementos y el problema de la mochila múltiple donde se gestionan múltiples mochilas amplíen su aplicabilidad y complejidad. Estas variantes ofrecen nuevos desafíos al adaptar técnicas de optimización para cumplir con restricciones más flexibles o específicas en diferentes contextos.

2.2 Descripción de la metaheurística a implementar

Los algoritmos genéticos se basan en los principios de la genética y la selección natural, y simulan el proceso de evolución biológica para la resolución de problemas. Dentro del conjunto de técnicas metaheurísticas, los algoritmos genéticos destacan como una de las estrategias más utilizadas para abordar el problema de la mochila 0-1, considerado un clásico en la investigación operativa y las ciencias de la computación.

Los algoritmos genéticos (AG) constituyen metaheurísticas inspiradas en los principios de la evolución biológica. Este enfoque emula el proceso de selección natural, favoreciendo la reproducción de los individuos más aptos, lo que impulsa una evolución progresiva de las soluciones hacia estados más cercanos al óptimo. Los componentes fundamentales que conforman un AG incluyen:

- **Población inicial:** Conjunto de soluciones candidatas generadas al azar o mediante heurísticas específicas.
- **Función de aptitud:** Evalúa y clasifica las soluciones según su eficiencia.
- **Operadores genéticos:** Incluyen selección (elección de soluciones para reproducción), cruzamiento (combinación de características de soluciones progenitoras) y mutación (modificaciones aleatorias para explorar nuevas regiones del espacio de soluciones.
- **Criterio de terminación:** Establece cuándo debe finalizar el algoritmo, que puede ser un límite de generaciones, tiempo de ejecución o estancamiento en la mejora de soluciones.

2.3 Estructuras de datos

El algoritmo utiliza una estructura *‘objeto’* para almacenar el peso y el valor de cada objeto disponible. La población de soluciones candidatas se representa como una matriz de enteros donde cada fila es un individuo de la población y cada columna representa la presencia o ausencia de un objeto específico en la mochila de ese individuo.

2.4 Funciones clave

- ***‘inicializar-población’:*** Genera una población inicial donde cada individuo es un vector binario generado aleatoriamente.

- ***´evaluar':*** Calcula el valor total de la mochila para un individuo dado. Incluye una penalización si el peso total excede la capacidad máxima, lo que resulta fundamental para guiar el AG hacia soluciones factibles.
- ***'selección':*** Elige aleatoriamente individuos de la población actual para el proceso de cruzamiento.
- ***'crossover':*** Realiza el cruzamiento entre pares de individuos seleccionados utilizando un punto de corte para mezclar sus genes.
- ***´mutación':*** Aplica mutaciones aleatorias a los individuos, aumentando la tasa de mutación si no han encontrado mejoras significativas en la última generación.

El AG mediante varias generaciones, selecciona individuos en cada generación, se aplican operaciones de cruzamiento, mutación y se evalúan las soluciones generadas para encontrar la mejor solución posible hasta el momento. El algoritmo finalmente debería indicar el mejor valor encontrado en cada generación, permitiendo apreciar la evolución del rendimiento de la población.

2.5 Justificación de la elección del algoritmo genético

Se seleccionó implementar el algoritmo genético como metaheurística aplicada al problema de la mochila 0-1 debido a:

- **Diversidad de soluciones:** Los algoritmos genéticos mantienen y exploran una población de soluciones, lo que incrementa las oportunidades de encontrar soluciones globales óptimas al evitar concentrarse de manera prematura en regiones locales del espacio de búsqueda.
- **Flexibilidad y adaptabilidad:** Los operadores genéticos de cruzamiento y mutación permiten que el algoritmo se adapte y explore nuevas áreas del espacio de soluciones.
- **Escalabilidad:** A través de ajustes en parámetros como el tamaño de la población y tasas de mutación, los algoritmos genéticos pueden ser capaces de adaptarse para manejar problemas de diferentes escalas y complejidades.

La metaheurística de los algoritmos genéticos aplicada al problema de la mochila 0-1 se centra en la simulación del proceso evolutivo biológico para la exploración eficaz en el espacio de soluciones del problema. Esta aproximación no solamente permite

descubrir soluciones de alta calidad sino que además mantiene una diversidad en la población de soluciones que evita la convergencia prematura a óptimos locales. A continuación, se detallan más profundamente las características clave de esta metaheurística cuando se aplica el problema específico de la mochila 0-1.

2.6 Conceptos claves

a) Representación e inicialización

En el contexto del problema de la mochila, cada solución candidata dentro de la población se representa mediante un arreglo binario, donde cada elemento indica si un objeto específico está incluido (1) o excluido (0) de la mochila. La inicialización de estas soluciones se realiza de manera aleatoria respetando las restricciones de capacidad máxima de la mochila para asegurar la viabilidad de las soluciones iniciales.

b) Función de aptitud

La función de aptitud o función objetivo es crucial en el algoritmo genético, ya que guía la selección de soluciones para la reproducción. En esta implementación se evalúa el valor total de los objetos en la mochila penalizando las soluciones que exceden el peso máximo admitido. Esta penalización ayuda a dirigir la búsqueda y soluciones que no solo sean de alto valor sino también factible bajo la restricción de peso.

c) Selección

La selección de individuos para la reproducción se basa en su rendimiento relativo según la función de aptitud. En esta implementación la selección se realiza al azar, lo que puede ser mejorada mediante métodos como el torneo o ruleta, donde, la probabilidad de selección es proporcionar la aptitud del individuo, favoreciendo a los de mejor desempeño, pero sin excluir completamente a los demás.

d) Cruzamiento

El cruzamiento es el mecanismo mediante el cual se combinan características de dos soluciones progenitoras para crear descendencia. Se selecciona un punto de corte al azar en los vectores de solución, y se intercambian los genes (objetos de la mochila) a partir de ese punto entre los dos progenitores. Este produce variabilidad en la

población y permite la combinación de características beneficiosas de diferentes individuos.

e) Mutación

La mutación introduce cambios aleatorios en las soluciones lo que es fundamental para explorar nuevas áreas de espacio de búsqueda y evitar el estancamiento en óptimos locales. La tasa de mutación puede ser adaptativa, tal como se implementa en este documento, Aumentando cuando no se han encontrado mejores resultados recientemente, lo que pretende fomentar una exploración más agresiva cuando la evolución de la solución parece estar estancada.

f) Convergencia y terminación

El algoritmo evoluciona a través de generaciones sucesivas, cada una basada en pasos de selección, cruzamiento y mutación, seguidos además por la evolución de la nueva población. El proceso se repite hasta que se alcanza un criterio de terminación, el que puede ser un número fijo de generaciones, un tiempo de ejecución específico, o hasta una condición donde las mejoras en la calidad de las soluciones se vuelven insignificantes. Los AG son especialmente efectivos para abordar el problema de la mochila 0-1 gracias a su capacidad para explorar espacios de solución complejos con robustez y flexibilidad. No obstante, su rendimiento está fuertemente condicionado por la configuración de parámetros clave como el tamaño de población, la tasa de mutación, la estrategia de selección y el operador de cruce, lo que exige un ajuste cuidadoso respaldado por experimentación sistemática.

2.7 Antecedentes de configuración y justificación

En cuanto a la configuración de parámetros, el rendimiento de los AG depende fuertemente de la configuración de sus parámetros. Tal como plantean Eiben y Smit (2011), una inadecuada elección de valores puede llevar a una exploración deficiente del espacio de búsqueda o a una convergencia prematura hacia óptimos locales. En respuesta a este desafío, se han propuesto diversos enfoques de ajuste automático. Por ejemplo, Hinterding, Michalewicz y Eiben (1997) introdujeron una estrategia de mutación auto-adaptativa basada en el rendimiento dinámico del algoritmo.

En términos de diseño de algoritmos robustos, Krasnogor y Smith (2005) destacan que la eficacia de una metaheurística también depende de su capacidad para adaptarse al entorno de resolución, incorporando elementos como autoajuste, control de diversidad y monitorización de la tasa de mejora.

En este contexto, se justifica la necesidad de evaluar sistemáticamente cómo varían los resultados de un AG en función de parámetros específicos, especialmente cuando se aplican a problemas representativos como la mochila 0-1. A través de un enfoque experimental controlado y replicable, este trabajo busca establecer directrices prácticas para la configuración de AG en distintos niveles de complejidad, contribuyendo a la construcción de buenas prácticas tanto en escenarios de investigación como de aplicación real.

3. Metodología

Se empleó un enfoque experimental cuantitativo para evaluar el impacto de diversas configuraciones de parámetros de rendimiento de un AG aplicado a instancias del problema de la mochila 0-1. El proceso metodológico se estructuró en diseño experimental, conjuntos de datos y métricas de evaluación.

3.1 Diseño Experimental

Para evaluar el impacto de diferentes configuraciones en el rendimiento del algoritmo genético (AG), se realizaron múltiples ejecuciones para cada combinación de probabilidad de mutación y tamaño del torneo. En cada instancia del problema, el algoritmo fue ejecutado 11 veces, cada una con 11 iteraciones. Las variables experimentales ajustadas en este estudio fueron la probabilidad de mutación (pm), evaluada con los valores {0.05, 0.10, 0.20}, y el tamaño del torneo (k), con valores {2, 3, 4}. Este diseño experimental estructurado facilitó la evaluación comparativa del desempeño del AG frente a distintas configuraciones, considerando la variabilidad asociada a la complejidad de cada instancia.

a) Descripción de las pruebas computacionales

Las pruebas realizadas consistieron en evaluar el rendimiento de un AG en diversas instancias del problema de la mochila 0-1. Si bien, al trabajar con algoritmos genéticos,

se pueden modificar diversos parámetros, se consideró específicamente variar dos parámetros, los que son probabilidad de mutación y tamaño del torneo.

- **Probabilidad de Mutación:** Controla la probabilidad de que un gen (elemento) cambie su estado (de incluido a no incluido o viceversa) durante el proceso de mutación.
- **Tamaño del Torneo:** Se refiere al número de individuos seleccionados aleatoriamente de la población para competir entre sí, donde el mejor de ellos es seleccionado para la reproducción.

b) Valores de los parámetros considerados

Las pruebas se llevaron a cabo variando dos parámetros clave:

- Probabilidad de Mutación: Se evaluaron tres valores diferentes: 0.05, 0.10 y 0.20.

-Tamaño del Torneo: Se evaluaron tres tamaños diferentes: 2, 3 y 4.

c) Conjuntos de datos

Para la validación del algoritmo propuesto se utilizaron instancias del problema de la mochila 0-1 obtenidas de la página web de la Universidad del Cauca (Ortega, s.f.). Cada conjunto de datos está acompañado de su valor óptimo correspondiente, lo que permite evaluar la efectividad de la metaheurística planteada. En la **Tabla N°1** se presentan los conjuntos de datos empleados junto con sus valores óptimos, los cuales sirvieron como referencia para calcular el porcentaje de desviación (GAP) de las soluciones generadas y así medir cuantitativamente el rendimiento del algoritmo.

Tabla N°1. Conjuntos de datos y valores óptimos.

Conjunto de datos	Óptimo
f1_l-d_kp_10_269	295
f2_l-d_kp_20_878	1024
f8_l-d_kp_23_10000	9767
knapPI_1_100_1000_1	9147
knapPI_2_100_1000_1	1514
knapPI_3_100_1000_1	2397

Fuente: Elaboración Propia.

El término “óptimo” hace referencia al mejor valor conocido de cada instancia, es decir, la mayor suma posible de beneficios obtenida sin exceder la capacidad de la mochila, bajo la restricción de que los objetos seleccionados no pueden fraccionarse. Se utilizaron seis instancias de prueba, clasificadas en tres categorías según su dimensionalidad: instancias pequeñas (f1_l-d_kp_10_269 y f2_l-d_kp_20_878 con 10 y 20 elementos, respectivamente), instancias medianas (f8_l-d_kp_23_10000 con 23 elementos) e instancias grandes (knapPI_1_100_1000_1, knapPI_2_100_1000_1 y knapPI_3_100_1000_1 con 100 elementos cada una). Esta clasificación permite analizar el comportamiento del algoritmo propuesto en diferentes niveles de complejidad computacional.

d) Métricas de evaluación

Los resultados obtenidos se resumieron en las siguientes métricas:

- **Promedio GAP (%):** El porcentaje de desviación promedio del valor obtenido con respecto al valor óptimo conocido.
- **Mejor Valor:** El mejor valor encontrado en las múltiples ejecuciones.
- **GAP Mejor Valor (%):** El porcentaje de desviación del mejor valor encontrado con respecto al valor óptimo conocido.

e) Implementación AG

La implementación del AG se rigió a un marco de adaptación dinámica de parámetros basado en las ecuaciones Ecuación 2 — Ecuación 5. La Ecuación 2 (ver **Figura N°2**) ajusta dinámicamente la mutación basándose en el progreso del algoritmo diseñado.

Figura N°2. Ecuación 2 Adaptación basada en *fitness.*

$$Pm(t) = P_m^{base} + \Delta Pm \; x \; (1 - \frac{f_{best}(t)}{f_{max}})$$

Fuente: Elaboración propia.

Cuando $f_{best}(t)$ se acerca a f_{max}, la probabilidad de mutación disminuirá. La adaptación basada en fitness corresponde a un mecanismo dinámico de ajuste de parámetros en AG que modifica la probabilidad de mutación según el rendimiento del

algoritmo, pretendiendo balancear entre exploración y explotación del espacio de búsqueda para prevenir una prematura convergencia. Donde:

$Pm(t)$: Probabilidad de mutación en la generación t.

P_m^{base} : Probabilidad de mutación base (valor inicial).

ΔPm : Incremento permitido máximo en la probabilidad de mutación.

$f_{best}(t)$: Mejor valor de fitness encontrado en la generación t.

f_{max} : Valor máximo posible de fitness para el problema.

Además de la adaptación basada en *fitness*, la diversidad genética de la población también es crucial en el equilibrio entre exploración y explotación. Para ello, se emplea un mecanismo que ajusta dinámicamente el tamaño del torneo según el grado de diversidad, descrito en la Ecuación 3 (ver **Figura N°3**).

Figura N°3. Ecuación 3 Ajuste basado en diversidad.

$$k(t) = k_{base} + \lfloor \left(1 - D(t)\right) x\, \Delta k \rfloor$$

Fuente: Elaboración propia.

Donde:

$k(t)$: Tamaño del torneo en la generación t.

k_{base} : Tamaño base del torneo (valor inicial).

$D(t)$: Medida de diversidad de la población en la generación t (valor entre 0 y 1).

Δk : Incremento máximo permitido en el tamaño del torneo.

$\lfloor \rfloor$: Función suelo (redondeo hacia abajo).

Mantener una adecuada diversidad en la población es fundamental para prevenir la convergencia prematura, pues sin ella el AG puede estancarse en óptimos locales y perder capacidad de explorar nuevas soluciones. Este mecanismo permite equilibrar exploración y explotación de manera dinámica, favoreciendo una búsqueda amplia cuando existe alta variabilidad y regulando la presión selectiva cuando esta disminuye.

Otro aspecto clave en la adaptación dinámica consiste en monitorear la tasa de mejora del algoritmo, permitiendo detectar oportunamente estancamientos y ajustar su desempeño. Este mecanismo se formaliza en la Ecuación 4 (ver **Figura N°4**).

Figura N°4. Ecuación 4 Control basado en convergencia.

$$S(t) = \frac{f_{best}(t) - f_{best}(t-w)}{w}$$

Fuente: Elaboración propia.

Donde:

$S(t)$: Tasa de mejora en la generación *t*.

$f_{best}(t)$: Mejor fitness en la generación actual *t*.

$f_{best}(t-w)$: Mejor fitness hace *w* generaciones.

w : Tamaño de la ventana de observación.

Este control compara el mejor resultado actual con el de generaciones anteriores y calcula la velocidad de mejora. Resulta esencial para detectar estancamientos y asegurar que el algoritmo no se quede atrapado en soluciones subóptimas.

f) Representación de la solución

En el contexto del problema de la mochila 0-1, cada solución se representa como un vector binario $\mathrm{x} = (x_1, x_2, \ldots, x_\mathrm{n})$, donde $x_1 = 1$ indica que el objeto *i*-ésimo está incluido en la mochila y $x_1 = 0$ indica lo contrario. En el *script*, esto se refleja en la función 'inicializar_poblacion', que inicializa una población de soluciones aleatorias.

Función 'inicializar_poblacion'

Para ilustrar esta fase de inicialización, en la **Figura N°5** se presenta un fragmento del script correspondiente a la función ‘inicializar_poblacion’.

Figura N°5. Fragmento Script función 'inicializar_poblacion'.

```
void inicializar_poblacion(int** poblacion, int n) {
    for (int i = 0; i < TAM_POBLACION; i++) {
        for (int j = 0; j < n; j++) {
            poblacion[i][j] = rand() % 2;
        }
    }
}
```

Fuente: Elaboración propia.

Función objetivo

En el contexto del problema de la mochila 0-1, la función objetivo se encarga de evaluar la calidad de cada solución, considerando tanto el valor de los objetos seleccionados como la restricción de capacidad de la mochila. Esta función se formaliza en la Ecuación 5 (ver **Figura N°6**).

Figura N°6. Ecuación 5 Función objetivo.

$$f(x) = \sum_{i=1}^{n} x_i \cdot valor_i - \max(0, \sum_{i=1}^{n} x_i \cdot peso_i - W) \times 10$$

Fuente: Elaboración propia.

Donde:

- x_i : Variable binaria que indica si el objeto ***i*** está incluido (1) o no (0) en la mochila.
- $valor_i$: Valor del objeto ***i***.
- $peso_i$: Peso del objeto ***i***.
- W : Capacidad máxima de la mochila.
- n : Número total de objetos disponibles.

Esta función suma el valor total de los objetos incluidos en la mochila y penaliza la solución en 10 unidades por cada unidad de peso que exceda la capacidad máxima W, aplicar la penalización permite disuadir soluciones que superen la capacidad permitida, de tal modo asegurar que solo sean consideradas combinaciones de objetos que cumplan con la restricción de capacidad de la mochila, optimizando el valor total dentro de los límites permitidos. El operador $\max(0,\cdot)$ asegura que solo se

aplique la penalización en caso de que la suma del peso de los objetos seleccionados exceda la capacidad W, evitando de esta manera una penalización negativa. En síntesis, la función objetivo **'evaluar'**, mide la calidad de una solución x buscando maximizar el valor de los objetos incluidos en la mochila considerando la restricción de la capacidad máxima W.

Función 'evaluar'

Esta función itera sobre los objetos disponibles, acumulando el peso y el valor de los objetos incluidos en la mochila según la solución x. Si el peso total excede la capacidad máxima W, se aplica una penalización proporcional al exceso de peso.

Inicialización de variables

- **peso_total:** Variable que acumula el peso total de los objetos incluidos en la solución.
- **valor_total:** Variable que acumula el valor total de los objetos incluidos en la solución.

Iteración sobre los objetos

El bucle ***for*** recorre todos los objetos disponibles (representados por ***n***).

Condición if (solucion[i])

Verifica si el objeto ***i***-ésimo está incluido en la solución (**solucion[i]** es 1 si está incluido, 0 si no lo está). Si el objeto está incluido (**solucion[i]** es verdadero), se suma su peso y su valor a **peso_total** y **valor_total**, respectivamente.

Control de la capacidad de la mochila

Después de sumar el peso de un objeto, se verifica si **peso_total** supera la capacidad máxima de la mochila W. Si **peso_total** > W, significa que la solución excede la capacidad máxima de la mochila.

Penalización por exceso de peso

En caso de exceso de peso, se aplica una penalización al valor total de la solución. La penalización se calcula como **valor_total** − (**peso_total** − W) 10. Donde **peso_total** − W es el exceso de peso sobre la capacidad máxima, y 10 es el factor de penalización por unidad de peso excedido.

Retorno del valor total

Si la solución es válida y no excede la capacidad máxima de la mochila, simplemente se retorna **valor_total** sin penalización.

Manejo de restricciones

En el problema de la mochila 0-1, la restricción principal es que el peso total de los objetos seleccionados no supere la capacidad máxima de la mochila W. Matemáticamente, esta condición se expresa como la suma de las variables binarias de inclusión de cada objeto multiplicadas por su respectivo peso, la cual debe ser menor o igual a, como se formaliza en la Ecuación 6 (ver **Figura N°7**).

Figura N°7. Ecuación 6 Restricción.

$$\sum_{i=1}^{n} x_i \cdot peso_i \leq W$$

Fuente: Elaboración propia.

Donde:

- x_i es una variable binaria que indica si el objeto ***i***-ésimo está incluido en la mochila.
- peso_i es el peso del objeto ***i***-ésimo.
- W es la capacidad máxima de la mochila.

Función 'evaluar'

Con el objetivo de clarificar la implementación de la función 'evaluar', en la **Figura N°8** se muestra un fragmento del script donde se detalla cómo se acumulan los pesos y valores, y cómo se aplican las penalizaciones cuando la solución excede la capacidad de la mochila.

Figura N°8. Fragmento Script función 'evaluar'.

```
int evaluar(Objeto* objetos, int n, int W, int* solucion) {
    int peso_total = 0, valor_total = 0;
    for (int i = 0; i < n; i++) {
        if (solucion[i]) {
            peso_total += objetos[i].peso;
            valor_total += objetos[i].valor;
            if (peso_total > W) {
                return valor_total - (peso_total - W) * 10;
            }
        }
    }
    return valor_total;
}
```

Fuente: Elaboración propia.

La restricción principal es la máxima capacidad de la mochila W, limita el peso total de los objetos seleccionados. Durante cada fase del AG (selección, cruce y mutación), se verifica que la solución candidata cumpla con la restricción. Cualquier solución que supere aquella capacidad máxima será penalizada o será descartada, asegurando que solamente las soluciones válidas y prácticas sean consideradas para futuras iteraciones. Esta forma de abordar las restricciones forza a que cada solución candidata sea evaluada y mantenga la coherencia con los límites impuestos por la capacidad máxima de la mochila.

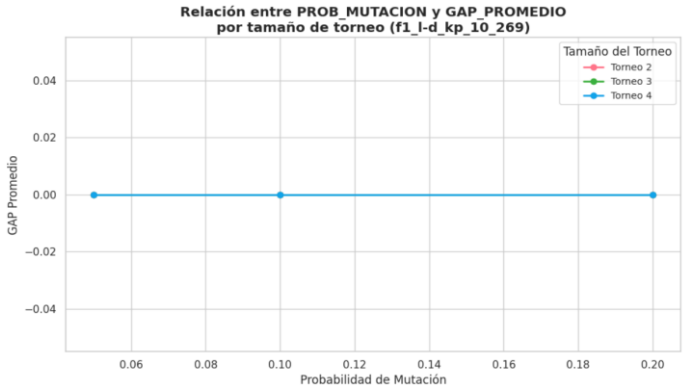
4. Resultados

A continuación, se presentan los resultados parciales obtenidos para cada instancia del problema. Cabe señalar que en la **Tabla N°1** presentada previamente en el apartado metodología se muestran los valores óptimos de referencia correspondientes a cada uno de los conjuntos de datos empleados en esta implementación.

a) Conjunto f1_l-d_kp_10_269

El experimento en esta instancia mostró un promedio GAP (%) de 0.00. El Mejor Valor alcanzado fue 295 en todos los casos. No se observó diferencia en el rendimiento al variar la probabilidad de mutación y el tamaño del torneo. Los resultados de esta evaluación se sintetizan en la **Figura N°9**, que ilustra la variación del GAP promedio en relación con la probabilidad de mutación bajo distintos tamaños de torneo.

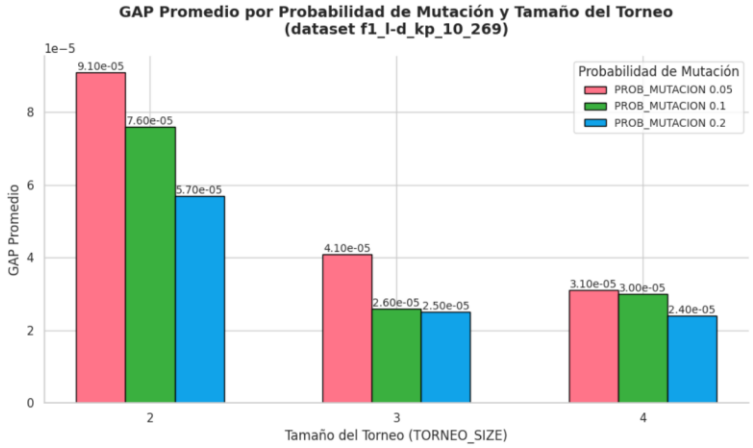
Figura N°9. Variación del GAP promedio en función de la probabilidad de mutación para distintos tamaños de torneo para el *dataset* f1_l-d_kp_10_269.txt.



Fuente: Elaboración propia.

La **Figura N°10** compara directamente el GAP promedio alcanzado para diferentes combinaciones de tamaño de torneo y probabilidad de mutación en la misma instancia.

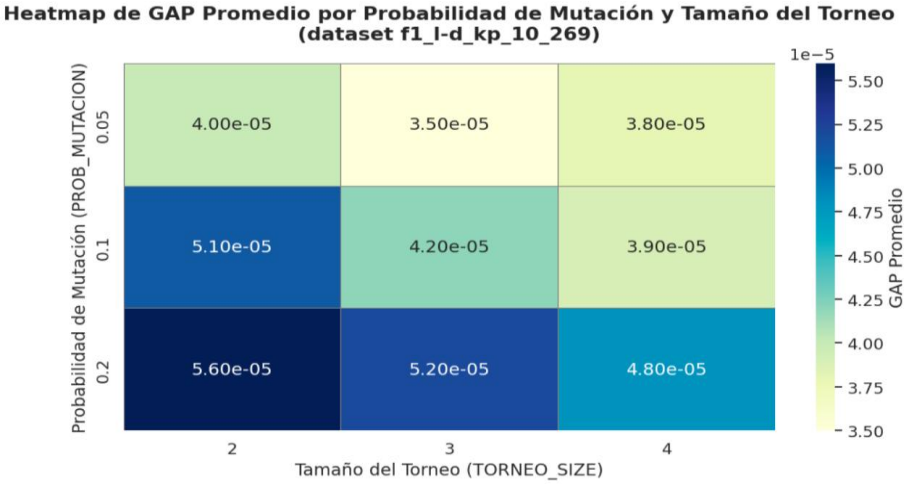
Figura N°10. Comparación del GAP promedio para diferentes tamaños de torneo y probabilidades de mutación para el *dataset* f1_l-d_kp_10_269.txt.



Fuente: Elaboración propia.

La **Figura N°11** presenta el mapa de calor asociado al GAP promedio, lo que permite visualizar de forma integrada el impacto combinado de ambos parámetros en el rendimiento del algoritmo.

Figura N°11. Mapa de calor del GAP promedio por tamaño de torneo y probabilidad de mutación para el *dataset* f1_l-d_kp_10_269.txt.

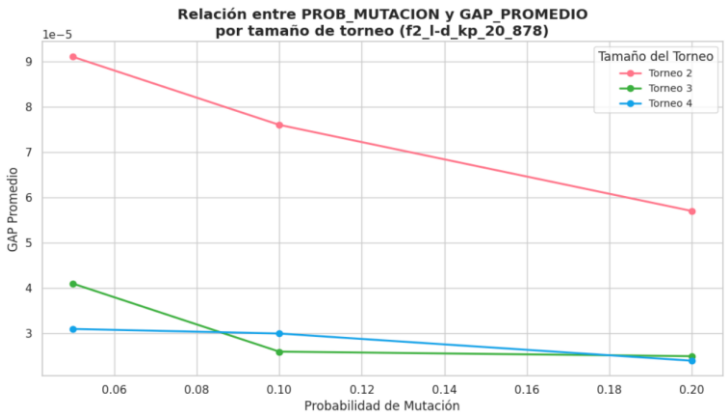


Fuente: Elaboración propia.

b) Conjunto f2_l-d_kp_20_878

El promedio de GAP (%) se mantuvo mayormente en 0.00, excepto en los casos con una probabilidad de mutación de 0.20, donde alcanzó 0.23% para un tamaño de torneo de 2 y 0.05% para un torneo de 3. El mejor valor obtenido fue 1024 en la mayoría de los casos, con pequeñas variaciones observadas en probabilidad de mutación de 0.20. El GAP más alto fue de 1.37%, correspondiente a probabilidad de mutación de 0.20 y un tamaño de torneo de 2. En la **Figura N°12** se observa la variación del GAP promedio dependiendo de la probabilidad de mutación bajo diferentes tamaños de torneo.

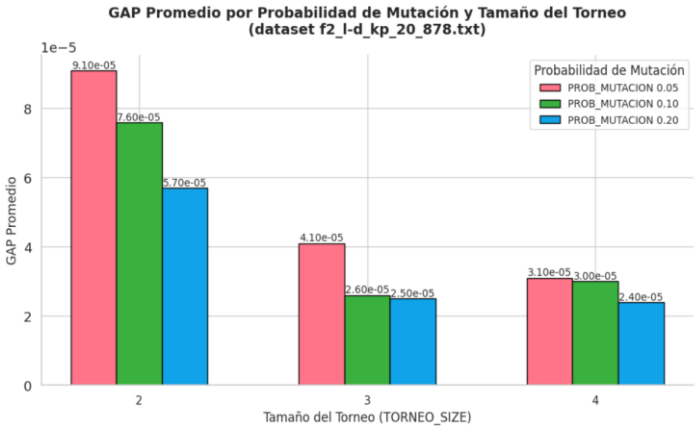
Figura N°12. Variación del GAP promedio en función de la probabilidad de mutación para distintos tamaños de torneo para el *dataset* f2_l-d_kp_20_878.txt.



Fuente: Elaboración propia.

La **Figura N°13** contrasta de forma directa el GAP promedio obtenido con distintas combinaciones de tamaño de torneo y probabilidad de mutación.

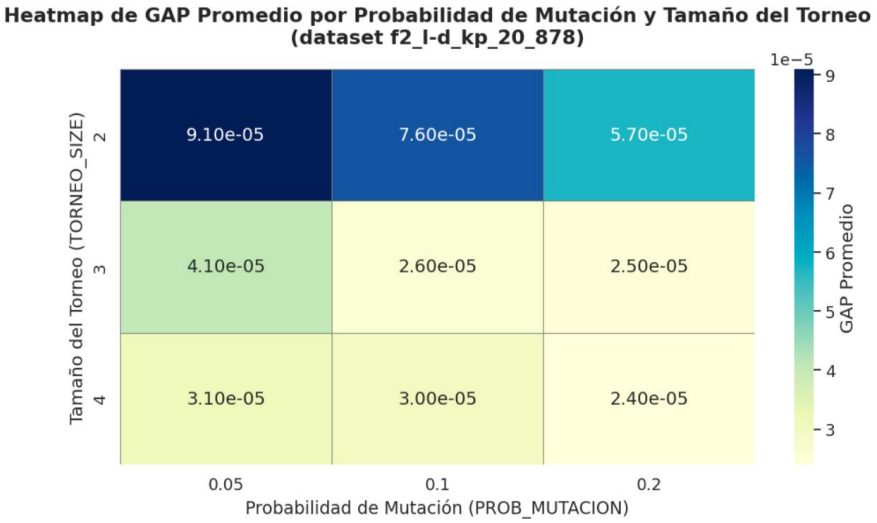
Figura N°13. Comparación del GAP promedio para diferentes tamaños de torneo y probabilidades de mutación para el *dataset* f2_l-d_kp_20_878.txt.



Fuente: Elaboración propia.

Para una visión más global, la **Figura N°14** muestra el mapa de calor del GAP promedio, ofreciendo una representación clara del impacto combinado de ambos parámetros sobre el rendimiento.

Figura N°14. Mapa de calor del GAP promedio por tamaño de torneo y probabilidad de mutación para el *dataset* f2_l-d_kp_20_878.txt.

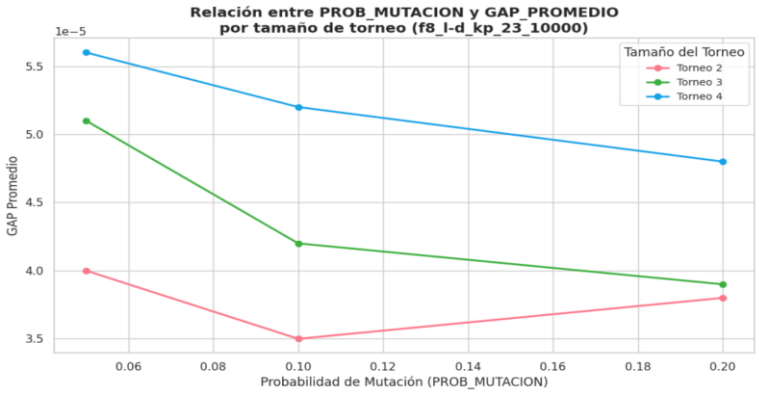


Fuente: Elaboración propia.

c) Conjunto f8_l-d_kp_23_10000

El promedio de GAP (%) mostró ligeras variaciones, alcanzando un valor mínimo de 0.00 en varios experimentos. El mejor valor obtenido fue cercano a 9767 en numerosos casos, con pequeñas desviaciones. Tanto el tamaño del torneo como la probabilidad de mutación influyeron de manera sutil en el rendimiento. En la **Figura N°15** se presentan los resultados de la variación del GAP promedio en función de la probabilidad de mutación bajo diferentes tamaños de torneo.

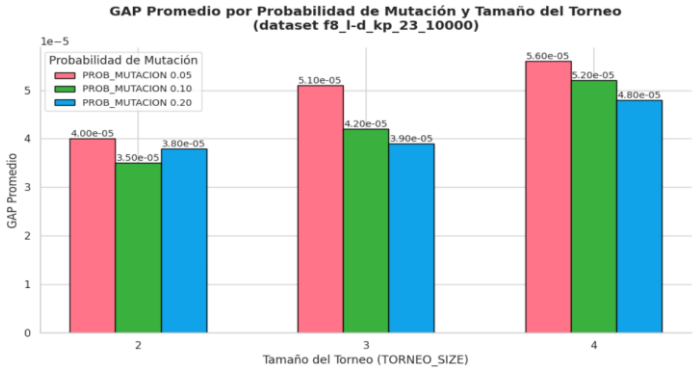
Figura N°15. Variación del GAP promedio en función de la probabilidad de mutación para distintos tamaños de torneo para el *dataset* f8_l-d_kp_23_10000.txt.



Fuente: Elaboración propia.

La **Figura N°16** profundiza en este análisis al comparar de manera directa los valores de GAP promedio obtenidos para cada combinación de torneo y probabilidad de mutación.

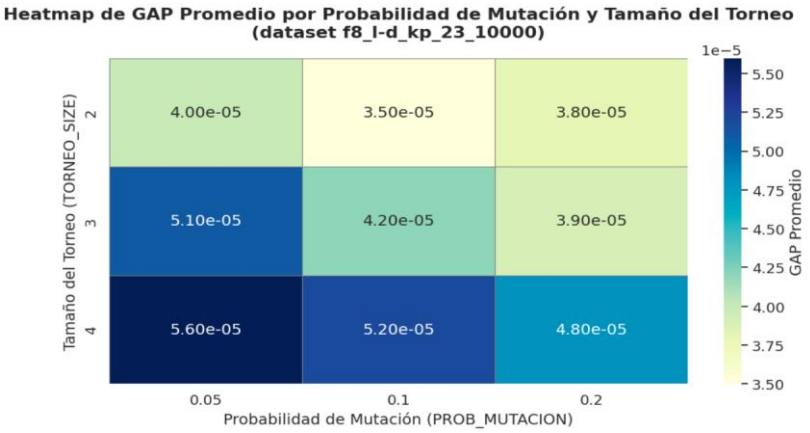
Figura N°16. Comparación del GAP promedio para diferentes tamaños de torneo y probabilidades de mutación para el *dataset* f8_l-d_kp_23_10000.txt.



Fuente: Elaboración propia.

La **Figura N°17** presenta el mapa de calor del GAP promedio, proporcionando una visión integral del comportamiento del algoritmo bajo las distintas configuraciones evaluadas.

Figura N°17. Mapa de calor del GAP promedio por tamaño de torneo y probabilidad de mutación para el *dataset* f8_l-d_kp_23_10000.txt.

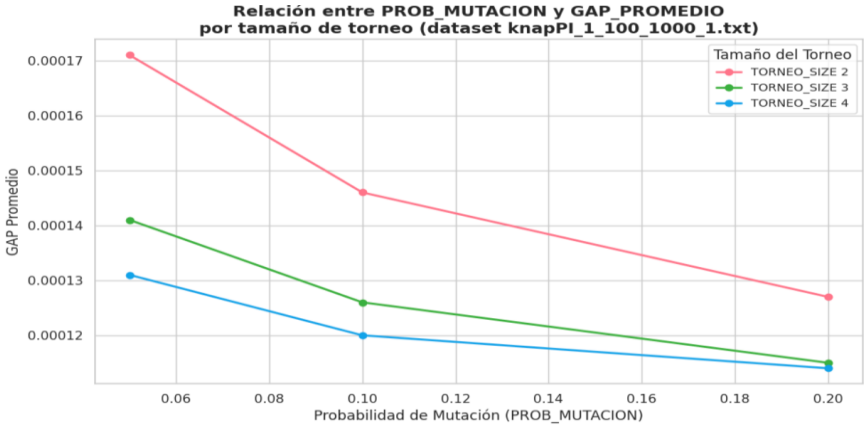


Fuente: Elaboración propia.

d) Conjunto knapPI_1_100_1000_1

El promedio de GAP (%) fue considerablemente alto, fluctuando entre 44.61% y 55.32%. El mejor valor mostró una variación significativa, con un mínimo de 4859 y un máximo de 6032. Tanto la probabilidad de mutación como el tamaño del torneo tuvieron un impacto notable en el rendimiento. En la **Figura N°18** se ilustra la variación del GAP promedio en función de la probabilidad de mutación aplicada, considerando distintos tamaños de torneo.

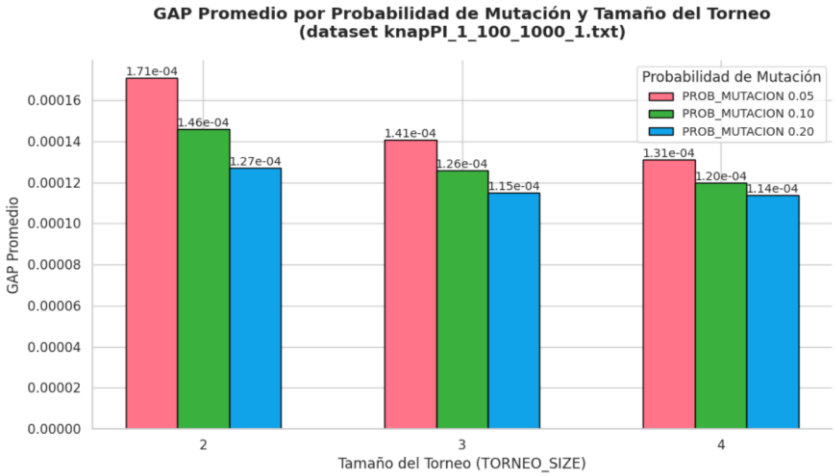
Figura N°18. Variación del GAP promedio en función de la probabilidad de mutación para distintos tamaños de torneo para el *dataset* knapPI_1_100_1000_1.txt.



Fuente: Elaboración propia.

La **Figura N°19** complementa este análisis al comparar los valores de GAP promedio alcanzados para cada combinación de tamaño de torneo y probabilidad de mutación.

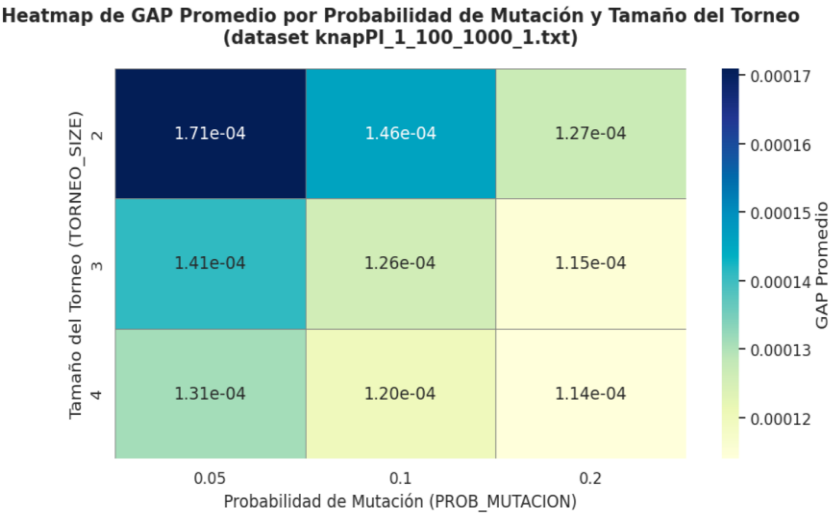
Figura N°19. Comparación del GAP promedio para diferentes tamaños de torneo y probabilidades de mutación para el *dataset* knapPI_1_100_1000_1.txt.



Fuente: Elaboración propia.

Para facilitar la identificación de patrones en el rendimiento, la **Figura N°20** ofrece un mapa de calor que resume visualmente la interacción entre ambos parámetros sobre el GAP promedio.

Figura N°20. Mapa de calor del GAP promedio por tamaño de torneo y probabilidad de mutación para el *dataset* knapPI_1_100_1000_1.txt.

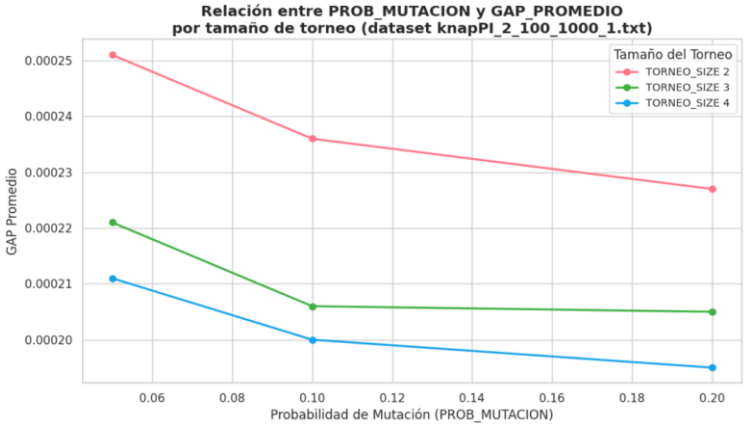


Fuente: Elaboración propia.

e) Conjunto knapPI_2_100_1000_1

El promedio de GAP (%) se mantuvo bastante consistente, alrededor del 23%. El mejor valor osciló entre 1158 y 1186. El rendimiento mostró una estabilidad relativa a lo largo de las diferentes configuraciones de probabilidad de mutación y torneo. En la **Figura N°21** se presenta la variación del GAP promedio en función de la probabilidad de mutación al considerar distintos tamaños de torneo.

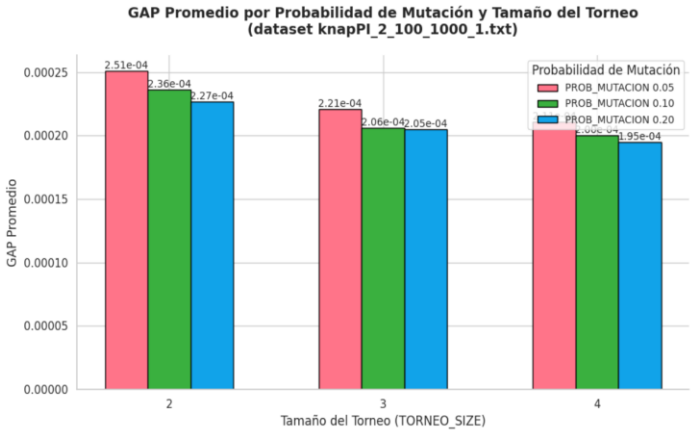
Figura N°21. Variación del GAP promedio en función de la probabilidad de mutación para distintos tamaños de torneo para el *dataset* knapPI_2_100_1000_1.txt.



Fuente: Elaboración propia.

La **Figura N°22** amplía este análisis al comparar los valores de GAP promedio alcanzados por cada combinación de probabilidad de mutación y tamaño de torneo.

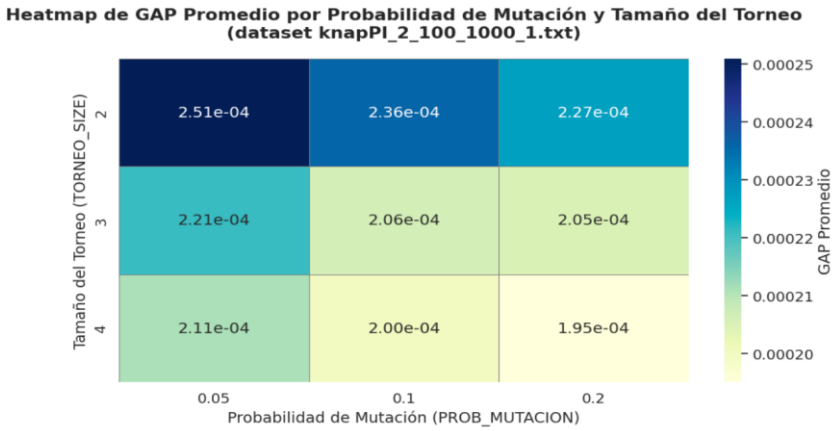
Figura N°22. Comparación del GAP promedio para diferentes tamaños de torneo y probabilidades de mutación para el *dataset* knapPI_1_100_1000_1.txt.



Fuente: Elaboración propia.

En la **Figura N°23** resume visualmente el impacto conjunto de los parámetros mediante un mapa de calor, que permite apreciar con claridad el comportamiento observado.

Figura N°23. Mapa de calor del GAP promedio por tamaño de torneo y probabilidad de mutación para el *dataset* knapPI_1_100_1000_1.txt.

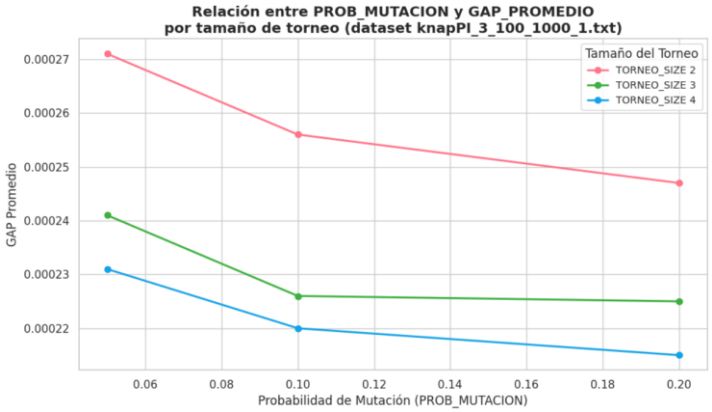


Fuente: Elaboración propia.

f) Conjunto knapPI_3_100_1000_1

El promedio de GAP (%) fluctuó entre 36,58% y 42,85%, mientras que el mejor valor varió de 1370 a 1635. La probabilidad de mutación y el tamaño del torneo tuvieron un impacto significativo en el rendimiento, similar a lo observado en el *dataset* knapPI_1_100_1000_1. En la **Figura N°24** se observa la variación del GAP promedio en función de la probabilidad de mutación considerando distintos tamaños de torneo.

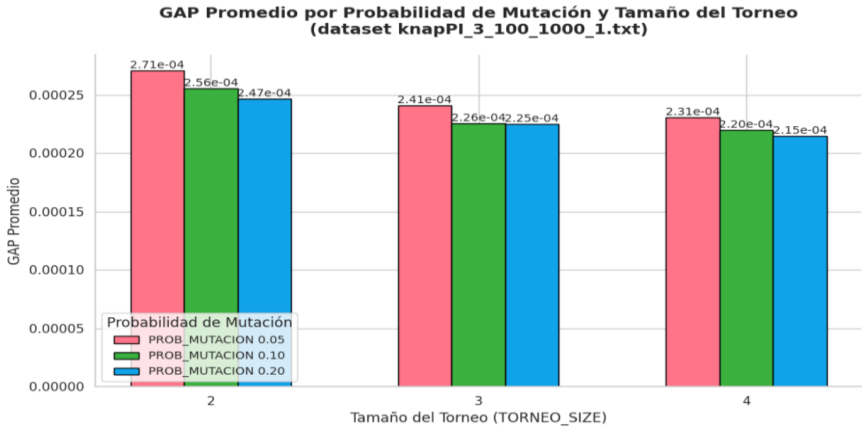
Figura N°24. Variación del GAP promedio en función de la probabilidad de mutación para distintos tamaños de torneo para el *dataset* knapPI_3_100_1000_1.txt



Fuente: Elaboración propia.

La **Figura N°25** muestra la comparación detallada de los valores de GAP promedio obtenidos con diferentes combinaciones de probabilidad de mutación y tamaño de torneo.

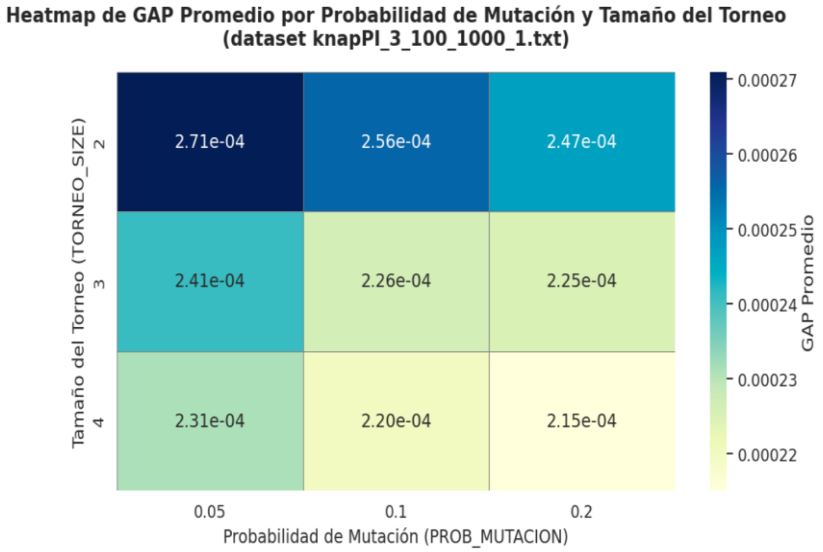
Figura N°25 Comparación del GAP promedio para diferentes tamaños de torneo y probabilidades de mutación para el *dataset* knapPI_3_100_1000_1.txt.



Fuente: Elaboración propia.

Finalmente, la **Figura N°26** presenta el mapa de calor correspondiente al GAP promedio, lo que facilita la identificación de patrones de rendimiento y el impacto conjunto de los parámetros implementados.

Figura N°26. Mapa de calor del GAP promedio por tamaño de torneo y probabilidad de mutación para el *dataset* knapPI_3_100_1000_1.txt.



Fuente: Elaboración propia.

4.1 Resultados globales

Ajustando sistemáticamente la probabilidad de mutación y el tamaño del torneo, se obtuvo una visión integral del desempeño del algoritmo genético en distintas instancias del problema de la mochila 0-1. Los resultados globales evidenciaron una alta sensibilidad del algoritmo a dichas configuraciones, especialmente al escalar la complejidad del problema. En instancias de baja dimensionalidad, como f1_l-d_kp_10_269, se observó un rendimiento óptimo con un GAP promedio de 0.00 %, reflejando la capacidad del algoritmo para alcanzar consistentemente la solución óptima. En contraste, en instancias de mayor tamaño y complejidad, tales como knapPI_1_100_1000_1 y knapPI_2_100_1000_1, el desempeño fue considerablemente más variable, alcanzando desviaciones promedio de hasta 55.32 %. Estas observaciones resaltan que configuraciones no óptimas en los parámetros pueden afectar negativamente la eficiencia exploratoria del algoritmo, especialmente en espacios de búsqueda extensos, subrayando la importancia de estrategias adaptativas en entornos de alta complejidad computacional. Estas observaciones sugieren que la naturaleza y características estructurales de ciertos conjuntos de datos podrían no ser adecuadamente abordadas mediante un seteo uniforme del algoritmo, lo cual resalta la necesidad de mecanismos de ajuste adaptativo capaces de especializar los parámetros en función de la complejidad inherente de cada instancia.

4.2 Análisis

El análisis de los resultados experimentales, sustentado en representaciones gráficas como mapas de calor y gráficos de dispersión, permitió identificar patrones de comportamiento del algoritmo genético bajo diversas configuraciones de parámetros. En particular, se observaron tendencias claras respecto al efecto combinado de la probabilidad de mutación y el tamaño del torneo sobre el rendimiento medido mediante el GAP promedio.

Los mapas de calor revelaron que combinaciones específicas, como una probabilidad de mutación de 0.05 junto con un tamaño de torneo de 4, tendieron a minimizar el GAP, alcanzando valores nulos (0.00 %) en instancias como f8_l-d_kp_23_10000. En

contraste, configuraciones con mayor probabilidad de mutación (0.20), especialmente combinadas con tamaños de torneo bajos (k = 2), generaron un aumento significativo en el GAP promedio, reflejando una degradación en la calidad de las soluciones.

Los gráficos de dispersión permitieron visualizar que en ciertas instancias se producía una mejora moderada al incrementar la probabilidad de mutación, sin embargo, esta tendencia no fue uniforme. En conjuntos de datos de mayor complejidad, una mutación elevada condujo a una exploración excesiva del espacio de búsqueda, generando soluciones de baja calidad y alto GAP. Esto evidencia que, si bien una mayor tasa de mutación favorece la diversidad genética, también puede comprometer la convergencia del algoritmo si no se regula adecuadamente.

Respecto al tamaño del torneo, se observó que valores intermedios (k = 3) lograron equilibrar adecuadamente la presión selectiva sin sacrificar la diversidad poblacional, lo cual se reflejó en un rendimiento más estable. En cambio, un tamaño de torneo alto (k = 4) incrementó la presión selectiva, favoreciendo rápidamente a los mejores individuos, pero reduciendo la exploración global y promoviendo la convergencia prematura, especialmente en instancias con múltiples soluciones cercanas al óptimo. Cabe destacar que las configuraciones con mayor probabilidad de mutación tendieron a incrementar la diversidad de soluciones, aunque también generaron una exploración excesiva que no siempre se tradujo en mejoras de rendimiento. Este efecto fue especialmente evidente en instancias con espacios de búsqueda amplios, donde es crucial mantener un equilibrio para evitar exploraciones improductivas.
En cuanto al tamaño del torneo, su influencia sobre la presión selectiva fue notoria: valores más altos favorecieron la selección de individuos con alta aptitud, lo cual resultó beneficioso en escenarios con múltiples soluciones cercanas al óptimo. No obstante, en ciertos casos esto redujo la diversidad poblacional, dificultando la evasión de óptimos locales.

5. Discusión de resultados

Los resultados obtenidos permiten extraer conclusiones claras sobre cómo las configuraciones paramétricas influyen en el rendimiento de los algoritmos genéticos (AG) aplicados al problema de la mochila 0-1. En particular, se observó que

combinaciones específicas de baja probabilidad de mutación (0.05) y tamaños de torneo moderados o grandes (k = 3 o 4) produjeron resultados óptimos (GAP ≈ 0%) en las instancias de menor y mediana complejidad.

En los conjuntos de datos más simples, como f1_l-d_kp_10_269 y f2_l-d_kp_20_878, los gráficos de barras y mapas de calor mostraron un GAP de 0.00% sostenido en la mayoría de las configuraciones, lo que sugiere que el algoritmo es altamente robusto en estos escenarios. No obstante, el mapa de calor para f2 mostró un ligero aumento del GAP (hasta 1.37%) cuando se combinó una alta probabilidad de mutación (0.20) con un torneo pequeño (k = 2), lo que evidencia una sobreexploración que reduce la estabilidad. Este efecto es coherente con lo descrito por Eiben y Smit (2011) respecto a la pérdida de presión selectiva bajo condiciones de baja competencia y alta aleatoriedad.

En la instancia f8_l-d_kp_23_10000 —de complejidad intermedia— el comportamiento fue más interesante: el mejor rendimiento se mantuvo con mutación baja (0.05) y torneo k = 4, como lo muestra claramente el mapa de calor. Aquí se observó una mayor sensibilidad a pequeñas variaciones en los parámetros, lo que demuestra que instancias ligeramente más complejas ya requieren una calibración más cuidadosa. La reducción progresiva del GAP con mayores tamaños de torneo sugiere que este parámetro actúa como una fuerza estabilizadora que refuerza la selección de individuos de alta aptitud sin sacrificar diversidad.

En contraste, las instancias knapPI_1_100_1000_1, knapPI_2_100_1000_1 y knapPI_3_100_1000_1 revelaron una dificultad mucho mayor, con GAPs promedio que llegaron hasta el 55.32%. Sin embargo, incluso en estos casos, los gráficos mostraron que ciertas combinaciones (mutación 0.10 y torneo 3 o 4) lograron reducir significativamente el GAP comparado con otras configuraciones más agresivas. Este patrón indica que, para problemas de gran escala, no solo es importante ajustar los parámetros, sino también considerar mecanismos de adaptación dinámica, como los que se utilizaron en este estudio, basados en diversidad y tasa de mejora.

La inclusión de mecanismos de ajuste dinámico de parámetros —basados en la tasa de mejora, la diversidad genética y el rendimiento relativo— fue clave para sostener el rendimiento del AG en instancias más complejas. Estos mecanismos se activaron especialmente en los casos donde la población tendía a la homogeneidad o donde no se observaban mejoras en múltiples generaciones. Este tipo de estrategia ha sido recomendada por autores como Hinterding, Michalewicz y Eiben (1997) para evitar la convergencia prematura y estimular la exploración inteligente del espacio de búsqueda.

Además del análisis cuantitativo, el presente estudio se destaca por su enfoque visual comparativo. El uso de mapas de calor permitió identificar patrones ocultos que no serían fácilmente observables mediante tablas tradicionales. Por ejemplo, se evidenció que en múltiples conjuntos de datos existe un “punto dulce” en torno a mutación 0.05–0.10 y torneo 3, lo que puede ser considerado como una configuración base efectiva para iniciar calibraciones en problemas similares.

Finalmente, el valor de este trabajo radica no sólo en la calidad de los resultados empíricos, sino también en su aportación metodológica replicable. La sistematización del proceso de evaluación, el diseño experimental con múltiples corridas y la integración de ajustes adaptativos sientan las bases para el desarrollo de algoritmos genéticos más inteligentes y ajustables. Estos resultados pueden integrarse, en futuras investigaciones, con esquemas híbridos evolutivos que combinen búsqueda local intensiva y exploración global (Krasnogor y Smith, 2005). Lo relevante de este enfoque es que ofrece una ruta clara y validada para comprender y controlar el impacto de parámetros críticos en AG, aportando lineamientos prácticos y visuales que pueden trasladarse a problemas reales donde la optimización eficiente y flexible es una necesidad creciente.

Así, este estudio no sólo fortalece el cuerpo teórico sobre el comportamiento de los algoritmos evolutivos, sino que además propone una herramienta práctica de evaluación, aplicable a distintos contextos de optimización combinatoria. Su versatilidad y bajo costo computacional lo convierten en un aporte relevante tanto para la investigación académica como para aplicaciones en áreas como logística,

bioinformática o planificación industrial, donde se requieren soluciones adaptativas, confiables y reproducibles.

6. Conclusiones

La implementación de un algoritmo genético como metaheurística para abordar el problema de la mochila 0-1 mostró un rendimiento robusto en instancias de baja complejidad, alcanzando valores óptimos (GAP = 0.00 %) en casos como f1_l-d_kp_10_269 y f2_l-d_kp_20_878. Sin embargo, el rendimiento se degradó considerablemente en instancias de mayor escala, como knapPI_1_100_1000_1, donde se observaron GAPs promedio de hasta 55.32 %. Esta variabilidad reafirma que el comportamiento del AG está altamente condicionado por la elección de sus parámetros.

La implementación reveló que algunos de los parámetros críticos, como son la probabilidad de mutación y el tamaño del torneo, tienen un impacto significativo en la efectividad del algoritmo. Por lo tanto, al abordar la problemática es recomendable explorar ampliamente los diversos parámetros en función de mejorar el rendimiento del algoritmo en problemas más complejos. En vista de esto, resulta crucial desarrollar estrategias que propicien al algoritmo mantener su capacidad de encontrar soluciones de alta calidad.

El análisis detallado mostró que una baja probabilidad de mutación (0.05) combinada con un tamaño de torneo intermedio (k = 3) conformó una configuración consistentemente efectiva en distintas instancias. En contraste, valores extremos en estos parámetros comprometieron la calidad de las soluciones. Estos resultados confirman que la configuración paramétrica no solo influye de forma significativa en el rendimiento del algoritmo, sino que su impacto es cuantificable y debe orientar la selección de estrategias de ajuste para maximizar su eficiencia.

Estos hallazgos sugieren que la eficacia del AG no puede ser garantizada mediante una configuración estática de parámetros, especialmente cuando se enfrentan problemas con estructuras y escalas distintas. Por tanto, se recomienda el desarrollo de mecanismos de ajuste adaptativo que permitan al algoritmo modificar sus

parámetros de forma dinámica durante la ejecución, en función de métricas como diversidad poblacional, tasa de mejora o progreso en la función objetivo.

Para futuras investigaciones, se propone explorar la integración de técnicas avanzadas de inteligencia artificial que permitan una configuración autónoma y sensible al contexto del AG. La implementación de redes neuronales profundas para el aprendizaje de patrones de configuración, la aplicación de algoritmos de optimización bayesiana para la calibración automática, o el uso de aprendizaje por refuerzo profundo para ajustar estrategias evolutivas en tiempo real, representan caminos prometedores para mejorar la robustez y escalabilidad de esta metaheurística en entornos complejos y dinámicos.

Apéndice

La implementación del algoritmo propuesto se realizó utilizando el script AG.c, disponible en el repositorio https://github.com/gustavoalcantara-aravena/evaluation_AG, el cual contiene el código fuente y los conjuntos de datos empleados. El desarrollo se efectuó en lenguaje C, utilizando bibliotecas estándar (<stdio.h>, <stdlib.h>, <string.h> y <time.h>). Las pruebas se ejecutaron en un equipo ASUS ROG Strix G614J con procesador Intel® Core™ i9-13980HX de 13ª generación, 32 GB de memoria RAM y sistema operativo Windows 11 Pro de 64 bits. La compilación se realizó mediante gcc.

Agradecimientos

Apoyado por beca de investigación 2024 de la Facultad Tecnológica, USACH.

Referencias Bibliográficas

Ahmed, Z. y Younas, I. (2011). A Dynamic Programming based GA for 0-1 Modified Knapsack Problem. *International Journal of Computer Applications, 16*(7), 1-6. https://doi.org/10.5120/2028-2668

Eiben, A. E. y Smit, S. K. (2011). Parameter tuning for configuring and analyzing evolutionary algorithms. *Swarm and Evolutionary Computation, 1*(1), 19-31. https://doi.org/10.1016/j.swevo.2011.02.001

Ezziane, Z. (2002). Solving the 0/1 knapsack problem using an adaptive genetic algorithm. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing, 16*(1), 23-30. https://doi.org/10.1017/s0890060401020030

He, J., He, F. y Dong, H. (2014). *A Novel Genetic Algorithm using Helper Objectives for the 0-1 Knapsack Problem*. arXiv:1404.0868 [cs.NE]. https://doi.org/10.48550/arXiv.1404.0868

Hinterding, R., Michalewicz, Z. y Eiben, A. E. (1997). *Adaptation in evolutionary computation: a survey*. IEEE International Conference on Evolutionary Computation (ICEC '97), 65-69. https://doi.org/10.1109/icec.1997.592270

Hota, A. R. y Pat, A. (2010). *An adaptive quantum-inspired differential evolution algorithm for 0-1 knapsack problem*. 2010 Second World Congress on Nature and Biologically Inspired Computing (NaBIC), 703-708. https://doi.org/10.1109/nabic.2010.5716320

Khuri, S., Bäck, T. y Heitkötter, J. (1994). *The zero/one multiple knapsack problem and genetic algorithms*. ACM symposium on Applied computing - SAC '94, 188-193. https://doi.org/10.1145/326619.326694

Krasnogor, N. y Smith, J. (2005). A tutorial for competent memetic algorithms: model, taxonomy, and design issues. *IEEE Transactions on Evolutionary Computation, 9*(5), 474-488. https://doi.org/10.1109/tevc.2005.850260

Malanthara, A. y Kale, I. R. (2025). *Hybrid Firefly-Genetic Algorithm for Single and Multi-dimensional 0-1 Knapsack Problems* (Versión 1). arXiv:2501.14775 [cs.NE]. https://doi.org/10.48550/ARXIV.2501.14775

Owais, W. B., Alkhazendar, I. W. J. y Saleh, M. (2020). Evaluating the impact of different types of crossover and selection methods on the convergence of 0/1 Knapsack using Genetic Algorithm. *Computer Science & Information Technology (CS & IT)*, 01-16. https://doi.org/10.5121/csit.2020.101101

Pisinger, D. (2005). Where are the hard knapsack problems? *Computers & Operations Research, 32*(9), 2271-2284. https://doi.org/10.1016/j.cor.2004.03.002

Pradhan, T., Israni, A. y Sharma, M. (2014). *Solving the 0–1 Knapsack problem using Genetic Algorithm and Rough Set Theory*. IEEE International Conference on Advanced Communications, Control and Computing Technologies, 1120-1125. https://doi.org/10.1109/icaccct.2014.7019272

Raidl, G. R. (1998). *An improved genetic algorithm for the multiconstrained 0-1 knapsack problem*. IEEE International Conference on Evolutionary Computation Proceedings. IEEE World Congress on Computational Intelligence (Cat. No.98TH8360), 207-211. https://doi.org/10.1109/icec.1998.699502

Umbarkar, A. J. y Joshi, M. S. (2014). 0/1 Knapsack Problem using Diversity based Dual Population Genetic Algorithm. *International Journal of Intelligent Systems and Applications, 6*(10), 34-40. https://doi.org/10.5815/ijisa.2014.10.05

Yan, T. S., Guo, G. Q., Li, H. M. y He, W. (2013). A Genetic Algorithm for Solving Knapsack Problems Based on Adaptive Evolution in Dual Population. *Advanced*

Materials Research, 756-759, 2799-2802. https://doi.org/10.4028/www.scientific.net/amr.756-759.2799

Yang, Y. (2024). *An upper bound of the mutation probability in the genetic algorithm for general 0-1 knapsack problem*. arXiv:2403.11307 [cs.NE]. https://doi.org/10.48550/arXiv.2403.11307

Kim, Y., Kim, J.-H. y Han, K.-H. (2006). *Quantum-inspired Multiobjective Evolutionary Algorithm for Multiobjective 0/1 Knapsack Problems*. IEEE International Conference on Evolutionary Computation, 2601-2606. https://doi.org/10.1109/cec.2006.1688633