“OPTIMIZACIÓN DE LA ASIGNACIÓN DE ÍTEMS A BINS UTILIZANDO ALGORITMOS GENÉTICOS”

Autor/Autores: Guerrero Jorge, Mena Mario, Salazar Wilian, Trujillo Diego

**Resumen-** El problema de asignación de objetos a contenedores es un problema clásico de optimización combinatoria. Este se puede describir de la siguiente manera, dada una cantidad limitada de contenedores con capacidad constante, y un conjunto de objetos de diversos tamaños se debe asignar los objetos a los contenedores de tal forma que se minimice la suma total del tamaño de los objetos no asignados y el área no utilizada en cada contenedor. En este trabajo se propone un algoritmo genético mono-objetivo para abordar este problema. La codificación de la solución (cromosoma) consiste en una representación basada en el número de objetos, la cual permite utilizar los operadores clásicos de selección, cruzamiento y mutación. Asimismo, se utiliza una función objetivo (fitness) que sintetiza los requerimientos del problema, transformándolo a un problema de maximización. La selección de los operadores genéticos se basó en la experimentación de diversas configuraciones de estos, eligiendo la que obtuvo mejor fitness promedio. El algoritmo se evaluó en varios problemas para los cuales se contaba con la solución exacta. En esta evaluación el algoritmo obtuvo la solución exacta para el 37% de problemas y en los demás se observó que el área no utilizada de los contenedores era nula para los primeros y crecía a medida que se acerca al último. Con lo cual este algoritmo muestra que cumple con la regla de optimización requerida en el problema.

——————————◆——————————

1. **Introducción**

La tarea de asignación de objetos a contenedores es un problema complejo que es considerado NP-Completo (NP y NP-Duro). Este que ha sido estudiado por muchos años debido a que está relacionado a muchas aplicaciones en la industria tales como almacenamiento de inventario en bodegas, corte de materiales, carga de contenedores en barcos, asignación de capital, planificación del estacionamiento de autos, asignación de recursos computacionales en la nube, entre otros [1]. La obtención de una solución cercana a la óptima para este problema es crucial para minimizar costos y maximizar el espacio utilizado, ello ayudará a lograr beneficios económicos y crear un impacto positivo en el medio ambiente [2]. Existen varias versiones de este problema [3], por las dimensiones de los objetos se pueden clasificar en los de una, dos o tres dimensiones. Por las características de los contenedores se pueden clasificar en contenedores del mismo tamaño o de tamaños diferentes. Asimismo, existe una versión del problema donde hay una cantidad limitada de contenedores y otra versión donde pueden agregarse los contenedores que se necesiten. Adicionalmente, existe una versión offline donde se tiene acceso a todos los objetos a la vez y una versión online en la que solo se tiene acceso a un objeto a la vez. La versión que se abordará en este trabajo es la que tiene objetos de una dimensión y se accede a ellos de forma offline. Asimismo, los contenedores son de tamaño constante y cantidad predefinida.

Varios enfoques se han propuesto para abordar este problema, incluyendo algoritmos heurísticos y metaheurísticos. Entre las heurísticas más conocidas se tiene la de primer encaje (FF por sus siglas en inglés) y la de mejor encaje [2]. Estas heurísticas son simples de implementar, pero por lo general no llegan a conseguir una solución óptima. Asimismo, en [4] se abordó el problema bajo un enfoque de programación entera mixta (MIP por sus siglas en inglés) y se obtiene un algoritmo de ramificación y poda exacto.Sin embargo, este algoritmo es bastante lento comparado con otros métodos. Entre los trabajos que han utilizado algoritmos genéticos para abordar este problema se tienen [4] y [5], en los cuales la representación de los cromosomas está basada en grupos. Esta representación, en comparación a otras, permite dar un mayor impulso a la transmisión de los mejores genes. Sin embargo, genera que los cromosomas no sean de tamaño fijo, con lo cual el cruzamiento de padres se realiza con metodologías más complejas que las clásicas.

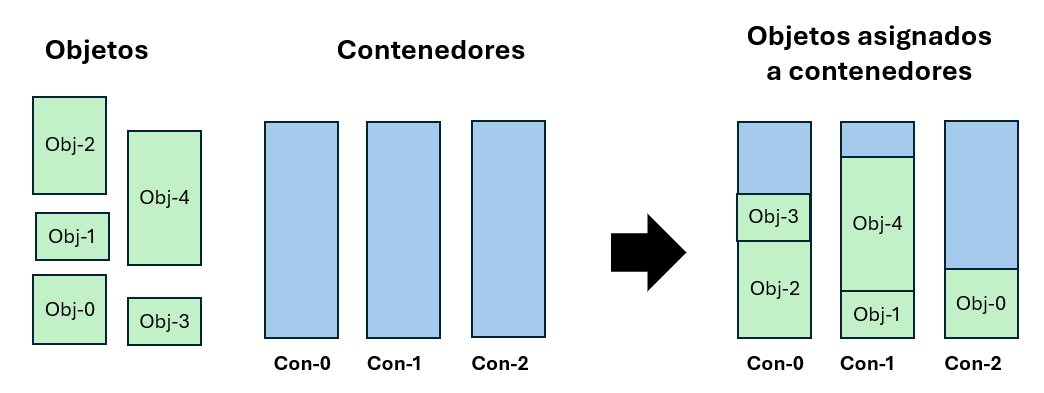
El presente trabajo presenta un algoritmo genético (AG) que permita encontrar soluciones eficientes para la asignación de objetos a contenedores, minimizando la suma total del tamaño de los objetos no asignados y el área no utilizada en cada contenedor. Este algoritmo propone la utilización de una representación basada en el número de objetos para codificar los cromosomas con el fin de que las operaciones selección, cruzamiento y mutación sean simples y rápidas. Además, se diseña una función de Fitness que integra los dos objetivos del problema para obtener una única solución y no una lista de soluciones como en el caso de algoritmos multi-objetivos. La solución entregada por el algoritmo genético es procesada para darle un formato simple que pueda ser interpretado fácilmente por el usuario. Con este algoritmo se propone encontrar soluciones cercanas a las óptimas en términos de eficiencia y utilización de espacio. Asimismo, que este pueda adaptarse a diferentes configuraciones de contenedores y objetos, proporcionando soluciones robustas y versátiles para este problema.

1. **Metodología**

El diseño del algoritmo propuesto se dividió en cuatro partes. En la primera se propone una representación basada en el número de objetos para cada individuo. Seguidamente, se diseña una función de Fitness para integrar los objetivos del problema. A continuación, se seleccionan los operadores genéticos en base a experimentación. Finalmente, se desarrolla un post-procesamiento de la solución encontrada por el algoritmo genético para que sea interpretada por el usuario.

**2.1 Representación del individuo**

La codificación de los individuos se basó en el número de objetos que desean asignarse. En esta codificación cada individuo de la población se representa como una lista de enteros. Los índices de la lista hacen referencia a la identidad de los objetos y los elementos de la lista identifican el contenedor al que el objeto es asignado. Por lo tanto, el tamaño de cada individuo es igual al número de objetos. Por ejemplo, si tenemos 5 objetos y 3 contenedores, un individuo podría representarse como [2, 1, 0, 0, 1], el cual indica que el primer objeto está en el contendor 2, el segundo en el contenedor 1, el tercero en el contenidor 0, el cuarto contenedor 0 y el quinto en el contenedor 1 (ver Figura 1). Por otra parte, para almacenar los tamaños de los objetos se construye otra lista, donde cada elemento de la lista es el tamaño del objeto y los índices de la lista hacen referencia a la identidad de los objetos. Puesto que los índices de ambas listas hacen referencia a lo mismo, entonces se mantiene la coherencia durante las operaciones evolutivas.



**Figura 1. Ejemplo de asignación de objetos a contenedores**

Utilizando esta codificación se asume que el número de contenedores es igual al número de objetos, sin embargo el proceso evolutivo del algoritmo intenta reducir al mínimo el número de contenedores. Asimismo, luego del post-procesamiento del mejor individuo encontrado, se entrega al usuario una solución en la que solo toma en cuenta los contenedores reales que existen.

Se eligió este tipo de representación de individuos porque permite adaptar de forma directa los operadores genéticos clásicos y obtener soluciones factibles de forma rápida.

**2.2 Función Fitness**

El objetivo del problema es asignar los objetos en los contenedores de tal manera que se minimice la suma total del tamaño de los objetos no asignados y el área no utilizada en cada contenedor. La función de fitness propuesta transforma este problema de minimización multi-objetivo en un problema de maximización mono-objetivo. En este enfoque, lo que se intenta maximizar es la capacidad ocupada de cada contenedor. Con lo cual, se minimiza la cantidad de espacio no ocupado en los contenedores y a la vez se incentiva a que los objetos que queden fuera sean los de menor tamaño. Para este fin se adaptó la función propuesta en (2) obtenido lo siguiente:

**, … (1)**

**,**

**P = ( ,**

donde Tj es el tamaño del objeto j, Mi es la cantidad de objetos que están asignados al bin i. C es la capacidad de los contenedores, N es el número de contenedores ocupados. Por lo tanto, al optimizar esta función se está maximizando la suma total del cuadrado del ratio de capacidad ocupada por contenedor , y al dividir esta sumatoria entre el número de contenedores ocupados se trata de minimizar esta cantidad. Asimismo, la penalidad P desincentiva que las soluciones tengan contenedores donde la suma del tamaño de objetos asignados exceda el tamaño del contenedor.

**2.3 Operadores genéticos**

La selección de los operadores genéticos para este algoritmo se basó en la experimentación. Con el fin de tener resultados estadísticamente representativos se iteró diez veces cada configuración de operadores propuesta y se eligió la que tuvo el mejor fitness promedio. En la elección de los operadores de selección de padres y cruzamiento se deshabilitó el operador de mutación para que no influenciara en la evaluación. Una vez seleccionados estos dos operadores, se procedió a seleccionar el operador de mutación y su respectiva tasa. A continuación, se detallan los operadores seleccionados.

1. **Selección de padres:** Se implementaron dos métodos de selección de padres: ruleta y torneo. En la selección por ruleta, los individuos tienen una probabilidad de ser seleccionados proporcional a su valor de fitness, siendo esta seleccionada para la solución.
2. **Cruzamiento:** Se implementaron dos operadores de cruzamiento: cruzamiento de un punto y cruzamiento uniforme. En el cruzamiento uniforme, cada gen de los hijos se hereda de uno de los padres de forma aleatoria, siendo este cruzamiento seleccionado para la solución.
3. **Mutación:** Se implementaron dos operadores de mutación: mutación de bit-flip y mutación multi-bit flip. En la mutación de bit-flip, se selecciona un gen aleatorio y se cambia su valor, esta mutación fue seleccionada para la solución óptima. Asimismo, la tasa de mutación seleccionada fue de 0.9.
4. **Selección de sobrevivientes:** Se implementó un método de selección de sobrevivientes basado en ranking. En este método, se combinan la población actual y la población de descendientes, se ordenan los individuos por valor de fitness y se seleccionan los mejores individuos para la siguiente generación.

**2.4 Post-procesamiento de mejor individuo**

El algoritmo genético entrega una solución codificada en la cual debe ser decodificada para obtener la solución requerida por el problema. Puesto que este algoritmo asume que la cantidad de objetos y contenedores es igual, entonces se empieza identificando todos los contenedores que tienen por lo menos un objeto el resto se descarta. Luego, se calcula la suma de los tamaños de objetos por contenedor y se ordena estos de mayor a menor. Se le asigna una nueva etiqueta a cada contenedor que es igual a su índice de ordenamiento (empieza en cero). Se considera como solución a todos los contenedores cuya etiqueta es menor al número de contenedores reales.

1. **Experimentación y Resultados**

**3.1 Enfoque multi-objetivo**

Primero, se construye una matriz con dimensiones "número de contenedores" por "número de artículos". Luego, la selección de artículos se realiza de manera binaria, donde cada individuo o cromosoma es representado por una matriz de dimensiones "número de contenedores" por "número de artículos", compuesta por valores de unos y ceros.

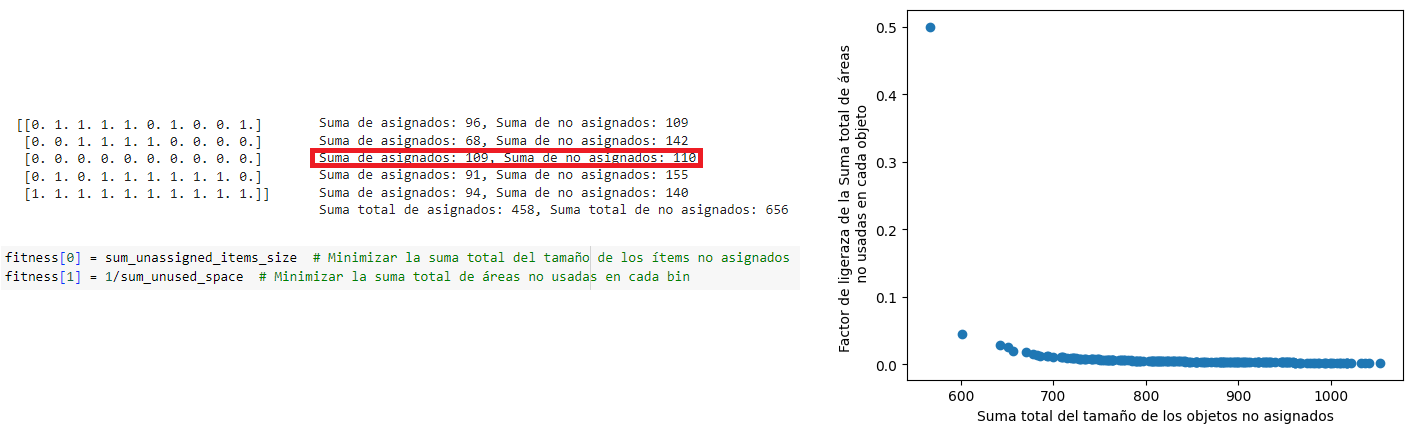
Hiperparámetos utilizados:

* Número de objetos = 54
* Máximo tamaño posible de cada objeto = 50
* Número de contenedores = 5
* Capacidad de contenedor = 100

Parámetros de la población utilizados en cada solución:

* Tamaño de la población = 200
* Tamaño mínimo de la población = 200
* Tamaño máximo de la población = 200

Para la evaluación se utiliza la suma total del tamaño de los objetos no asignados y el factor de ligereza en cada objeto que es calculado como la división del primer indicador con la unidad.



**Figura 2: Gráfico de pareto para las soluciones encontradas.**

Como resultado se obtuvo una gráfico de pareto con la finalidad de encontrar las mejores soluciones. Del gráfico, tenemos que existe una solución que se ajusta a la solución que necesitamos, pero no cumplirá con los objetivos

**3.2 Enfoque mono-objetivo**

Primero, se construye un vector fila llamado "ítems" donde cada posición representa el tamaño de cada "ítem" (objeto). Luego, se crea un vector fila para cada individuo, donde cada posición del vector representa el bin donde se asignará un "ítem".

Entonces, el problema consiste en asignar un conjunto de "ítems", cada uno con un tamaño específico, a un conjunto de bins de manera que se maximice el uso del espacio disponible en los bins sin exceder su capacidad máxima.

Hiperparámetos utilizados:

* Capacidad de contenedor = 100
* Número de objetos = 40
* Máximo número posible de objetos = 50
* Tamaño de la población = 50
* Número de generaciones = 300

Para la evaluación se utiliza la función fitness anteriormente definida en la sección de metodología.

Describir los experimentos hechos (qué componentes/ parámetros se probaron, qué valores, qué estrategia de validación).

Los parámetros antes descritos corresponden a las dos soluciones planteadas pero de ellas escogimos quedarnos con la segunda porque podemos llegar a mejores soluciones. Se realizó la optimización de parámetros y se probó con una mutación de 0, al final se realiza una comparación de los parámetros utilizados y en cuáles de ellos se obtuvo el mejor resultado.

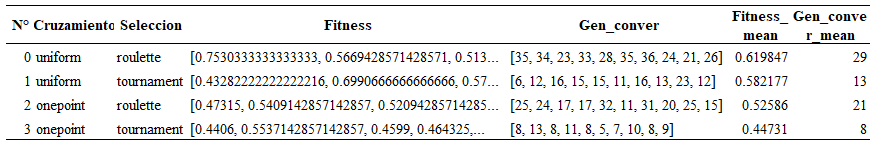
El segundo enfoque desarrollado sí resuelve el problema con algunas excepcionalidades que en la parte de resultados se explicara, pero las mismas se identifican y pueden ser detectadas a tiempo.

**3.3 Resultados y discusión**

**Algoritmo con mutación cero**

En la parte metodología se desarrolló dos enfoques, pero en esta parte de resultados solo describiremos los encontrados en la solución por contenedores. Una de las primeras acciones desarrolladas fue generar los resultados con una mutación “0” que se presentan en la tabla 1, el mejor fitness logrado es de 0.75, con una media de 0.61 porque se realizaron 10 interacciones. Los dos mejores resultados se logran con cruzamiento “uniforme” y en el caso del de selección es por “roulette” que se logra el mejor fitness seguido por “tournament”. En el caso del mejor gen encontrado, no se repiten ninguno de los elementos.

**Tabla 1: Resultados obtenidos con mutación “0”**

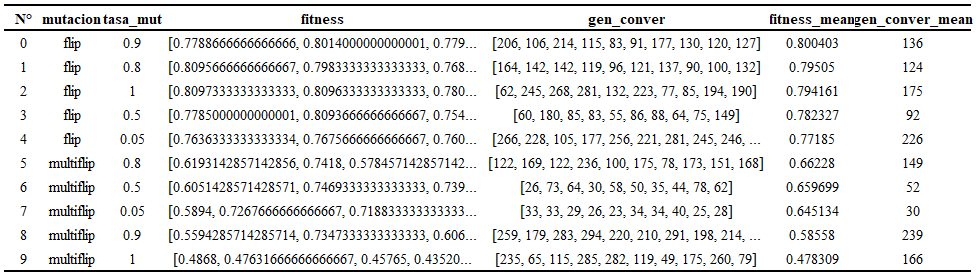


gen\_cover (orden de los genes gemerados), fitnesss\_mean (media de los fitness hallados) y gen\_cover\_mean (medias de los genes encontrados)

**Algoritmo con tasa de mutación diferente a cero**

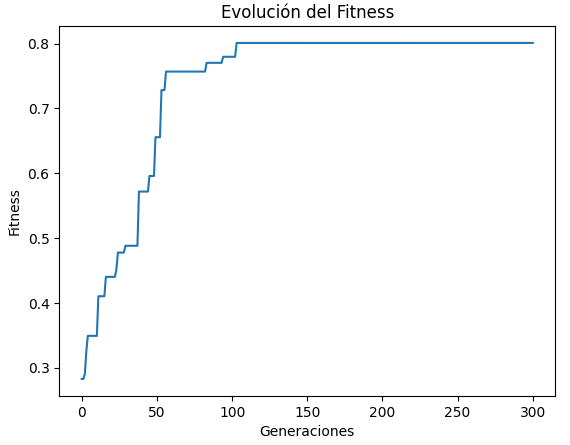
Después de realizadas las primeras interacciones se decidió probar con los siguientes valores de mutación: 0.05, 0.5, 0.8, 0.9 y 1, los resultados obtenidos se pueden observar en la tabla 2. En el caso de cruzamiento se utilizó “uniform” y el método de selección fue “roulette”, que fue fueron los parámetros que se obtuvo el mejor resultado con mutación “0”. En el caso de la mutación se probó con “flip” y “multiflip”, obteniéndose los mejores resultados con “flip”. La mejor combinación encontrada es con una tasa de mutación de 1 y con el método de mutación “flip”, el fitness promedio llega hasta un valor de 0.94, que quiere decir que falta 6% para que todos los contenedores (bins) estén ocupando toda su capacidad.

**Tabla 2: Resultados obtenidos con diferentes valores de mutación**



Al igual que con una tasa de mutación de uno, también con 0.8, 0.5 y 0.9 se logran valores promedios de fitness mayores a 0.9, siempre utilizando “flip”. El mejor resultado encontrado con “multiflip” es de un fitness promedio de 0.70, que es menor al último fitness logrado con “flip” que es de 0.86. Los resultados mostrados nos indican que es mejor utilizar tasas de mutación a que ésta sea cero, esto utilizando como comparación el fitness, que más adelante veremos que no es la mejor solución.

Después de los resultados obtenidos anteriormente se generó iteraciones con la selección de la mejor configuración encontrada, los parámetros son: tasa de mutación uno, mutación “flip”, seleccionador de tipo “roulette” y con cruzamiento “uniform”. La evolución del fitness de la configuración se puede observar en la figura 3. llega a valores de 0.8 después de 500 generaciones.

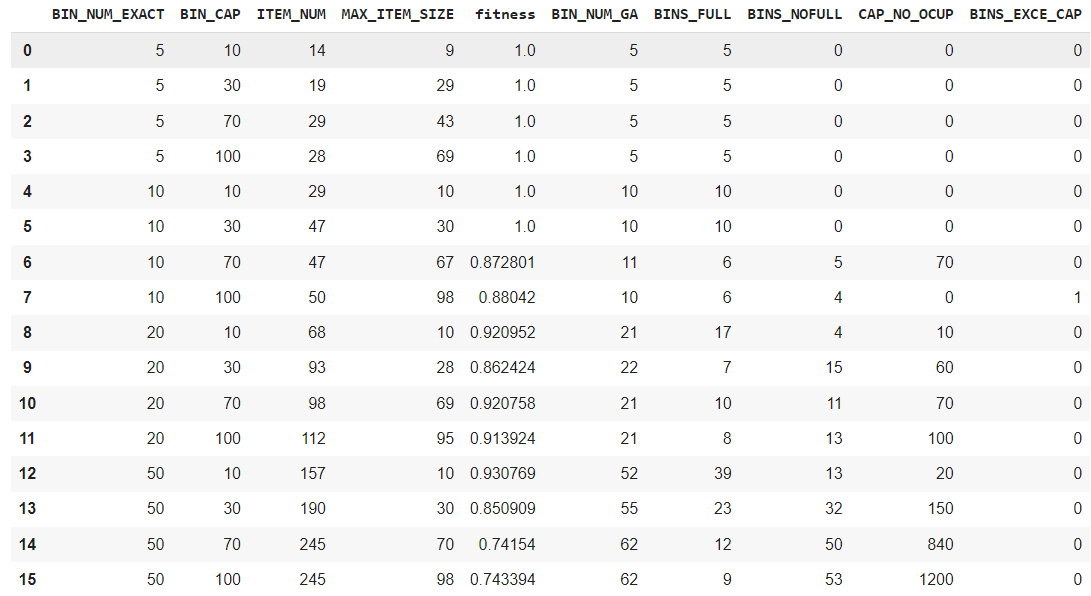


**Figura 3: Evolución del fitness para la mejor configuración**

**Prueba con solución exacta**

En las anteriores soluciones se obtienen valores altos del fitness, pero no nos garantizan que la solución sea exacta, puede ser que algunos estén semillenos o alguno de ellos pueden estar excedidos. Para ello se generan medidores que nos digan el número de contenedores llenos, la capacidad no ocupada de cada uno de ellos y el número de contenedores excedidos de la capacidad. Los resultados se pueden observar en la tabla 3, en las primeras seis iteraciones en las soluciones encontradas podemos observar que los contenedores están llenos totalmente, pero en la solución N° 7, podemos encontrar que hay diez contenedores, pero la solución encuentra 11, porque hay un contenedor que excede su capacidad y lo cuenta como uno más, generando el contenedor excedido de la solución. También se puede observar valores de fitness por encima de 0.9, que tienen capacidad no utilizada eso ocurre en los N° 10, 11 y 12.

**Tabla 3: Comparación de los resultados calculados a partir de las diferentes configuraciones e iteraciones.**



BIN\_NUM\_EXACT: Número exacto de (bins) requeridas para la solución óptima generada.; BIN\_CAP: Capacidad máxima de cada (bin) en la instancia generada; ITEM\_NUM: Número total de elementos a empacar; MAX\_ITEM\_SIZE: Tamaño máximo de los elementos en la instancia generada; BIN\_NUM\_GA: Número de (bins) utilizadas por la solución encontrada por el algoritmo genético; BINS\_FULL: Número de bins que se llenaron completamente en la solución del algoritmo genético; BINS\_NOFULL: Número de bins que no se llenaron completamente en la solución del algoritmo genético; CAP\_NO\_OCUP: Capacidad total no ocupada en las bins (suma de espacios vacíos) en la solución del algoritmo genético; BINS\_EXCE\_CAP: Número de bins que exceden la capacidad máxima en la solución del algoritmo genético.

El problema de la asignación de objetos en contenedores es posible solucionar con la aplicación de métodos heurísticos, los mejores resultados encontrados son con tasas de mutación y las soluciones encontradas pueden dar soluciones generales y no solo quedarse en óptimos locales.

La presentación de los resultados debe facilitar el entendimiento. En general se deben usar figuras y/o tablas. Recuerde, todos los resultados deben interpretarse. Esforzarse para explicar el formato de las curvas presentadas, dar detalles del tiempo de simulación.

1. **Conclusión**
2. El algoritmo genético encuentra soluciones cercanas o iguales a la solución óptima (fitness = 1.0) para instancias pequeñas y medianas del problema (hasta aproximadamente 50 elementos).
3. En la prueba con soluciones exactas, se logró encontrar esta solución en el 37.5% (6/16) de los casos.
4. Para instancias más grandes (más de 100 elementos), el algoritmo comienza a tener dificultades para encontrar la solución óptima, y el valor de fitness disminuye.
5. En las instancias donde el algoritmo genético no encuentra la solución óptima, se puede observar que:

* El número de bins (BIN\_NUM\_GA) utilizado es mayor que el número mínimo requerido (BIN\_NUM\_EXACT).
* Hay bins que no se llenan completamente (BINS\_NOFULL > 0 y CAP\_NO\_OCUP > 0).
* En algunos casos, se utilizan bins que exceden la capacidad máxima (BINS\_EXCE\_CAP > 0).

1. A medida que aumenta el número de elementos (ITEM\_NUM) y el tamaño máximo de los elementos (MAX\_ITEM\_SIZE), el algoritmo genético tiene más dificultades para encontrar soluciones óptimas o cercanas a la óptima.

Estos resultados son consistentes con el hecho de que el problema asignación de ítems a bins es un problema NP-difícil, y los algoritmos genéticos, siendo métodos aproximados, pueden tener dificultades para encontrar soluciones óptimas en problemas de gran escala.

1. **Sugerencias de trabajos futuros**

Queda pendiente comparar este algoritmo con heurísticas simples como primer encaje para ver cual es la diferencia de tiempo en entregar una solución y que tan diferentes son las soluciones.

En futuros trabajos se piensa comparar este algoritmo con otros algoritmos genéticos como el que usa la representación por agrupación.

En lugar de una función de fitness único, se podría considerar una función de aptitud multicapa, donde se optimicen diferentes objetivos en distintas capas. Por ejemplo, una capa podría maximizar la suma de tamaños, otra capa podría minimizar el número de mochilas, y otra capa podría buscar una distribución equilibrada de elementos.

Se podría evaluar una variante del problema donde se tienen múltiples conjuntos de bins con diferentes capacidades, y se deben asignar los elementos a las bins correspondientes. Esto requeriría ajustes en la representación de los cromosomas y en la función de fitness.

1. **Link del repositorio del trabajo**

<https://github.com/DiegoTrux/Comp_Evolutiva>

1. **Declaración de contribución de cada integrante**

Generación de algoritmos, JG, MM; redacción de introducción, DT, WS, JG; redacción de metodología, CIA, redacción de resultados WS, JG, DT, MM, revisión de informe y código, CIA.

1. **Referencias**
2. Guo, B.; Zhang, Y.; Hu, J.; Li, J.; Wu, F.; Peng, Q.; Zhang, Q. Two-Dimensional Irregular Packing Problems: A Review. *Front. Mech. Eng.* **2022**, *8*, 966691, doi:10.3389/fmech.2022.966691.
3. Tsao, Y.-C.; Tai, J.-Y.; Vu, T.-L.; Chen, T.-H. Multiple Bin-Size Bin Packing Problem Considering Incompatible Product Categories. *Expert Syst. Appl.* **2024**, *247*, 123340, doi:10.1016/j.eswa.2024.123340.
4. Wu, W.; Fan, C.; Huang, J.; Liu, Z.; Yan, J. Machine Learning for the Multi-Dimensional Bin Packing Problem: Literature Review and Empirical Evaluation 2023.
5. Falkenauer, E.; Delchambre, A. A Genetic Algorithm for Bin Packing and Line Balancing. In Proceedings of the Proceedings 1992 IEEE International Conference on Robotics and Automation; May 1992; pp. 1186–1192 vol.2.
6. Quiroz-Castellanos, M.; Cruz-Reyes, L.; Torres-Jimenez, J.; Gómez S., C.; Huacuja, H.J.F.; Alvim, A.C.F. A Grouping Genetic Algorithm with Controlled Gene Transmission for the Bin Packing Problem. *Comput. Oper. Res.* **2015**, *55*, 52–64, doi:10.1016/j.cor.2014.10.010.
7. Kaaouache, M.A.; Bouamama, S. Solving Bin Packing Problem with a Hybrid Genetic Algorithm for VM Placement in Cloud. *Procedia Comput. Sci.* **2015**, *60*, 1061–1069, doi:10.1016/j.procs.2015.08.151.