CIMAT

MTY Modelando el siglo XXI

Centro de Investigación en Matemáticas (CIMAT).
Unidad Monterrey

Maestría en Cómputo Estadístico

Optimización

Proyecto: Resolviendo un problema de empaquetamiento en contenedores a través de un algoritmo genético

Autores
Pedro Omar Gonzalez Rodriguez
Armando Salinas Lorenzana

Supervisores
Dra. Martha Selene Casas Ramírez
Dr. Alejandro Rosales Pérez.

Monterrey, Nuevo León A. 27 de mayo de 2024

1. Introducción.

El *Problema de empaquetamiento en contenedores* (Bin Packing Problem) se puede formular de la siguiente manera: dados n objetos con pesos $w_1, \ldots, w_n$ y considerando un conjunto de contenedores con una capacidad C. El objetivo es encontrar el menor número de contenedores en donde se coloquen todos los objetos sin violar la capacidad de los contenedores.

El problema de Bin Packing es un problema NP-duro, por lo que un método exacto para resolver dicho problema requiere un gran número de variables y demasiado tiempo de ejecución. Es por ello que ha sido estudiado y abordado desde hace décadas en diversas disciplinas, como la informática, la logística, la manufactura y la investigación de operaciones [1, 2]. Algunos trabajos relacionados al problema de Bin Packing son los siguientes:
El algoritmo MTP (Multiple Tuned Packing) fue introducido por Martello en 1990 [3], se centró en encontrar una solución al problema del Bin Packing que minimizara el número de contenedores utilizados. Utilizó una combinación de estrategias de empaquetamiento inicial, como First-Fit Decreasing (FFD) o Best-Fit Decreasing (BFD).
En el estudio realizado por Falkenauer en 1996 [4], se llevó a cabo una comparación entre el método MTP y un enfoque híbrido denominado HGGA (Hybrid Genetic Grouping Algorithm). El enfoque híbrido HGGA integra dos componentes principales: un algoritmo genético de agrupamiento (GGA) adaptado específicamente para abordar problemas de agrupamiento, y una técnica de optimización local basada en criterios de dominación.
En 2004 Kasap aplicó una Red Neuronal Aumentada (AugNN) para resolver BPP [5], combinando reglas de prioridad y aprendizaje iterativo para manejar problemas difíciles y dividirlos en subproblemas, usando un enfoque que se beneficia tanto de la estructura del problema como de los límites superior e inferior establecidos.

Por nuestra parte, el objetivo de este trabajo fue implementar una estrategia metaheurística (algoritmo genético) al problema del empaquetamiento de contenedores con el fin de realizar un análisis exploratorio en su espacio de soluciones y encontrar una buena solución óptima.
Para justificar el uso de un algoritmo genético (AG) en el problema del *Bin Packing Problem* (BPP), las razones son las siguientes:

- Flexibilidad en la modelación: Los AG no requieren que el problema sea lineal ni que cumpla con otras restricciones estrictas que sí pueden ser necesarias en métodos de optimización más tradicionales. Esto los hace particularmente adaptativos para diferentes variantes del BPP, incluyendo restricciones adicionales como diferentes tamaños de bins o propiedades especiales de los objetos.
- Diversidad de soluciones: Los algoritmos genéticos exploran el espacio de soluciones de manera estocástica, lo que les permite generar una amplia variedad de soluciones candidatas. Esto es particularmente útil en el BPP, donde puede haber muchas configuraciones posibles de objetos en los bins.
- Capacidad de escape de óptimos locales: A diferencia de los métodos de búsqueda tradicionales que pueden quedar atrapados en óptimos locales, los algoritmos genéticos tienen una mayor capacidad para escapar de estos óptimos gracias a operadores como la mutación, que introduce variabilidad en las soluciones.

Por los argumentos dados, se consideró usar un algoritmo genético en el BPP ya que resultan ser una buena opción para tratar problemas de optimización combinatoria. Sin embargo, es importante tener en cuenta que la eficacia de estos algoritmos dependen significativamente de cómo se diseñe la tasa de mutación, el método de cruzamiento, la selección y reemplazo de individuos en la población.

2. Descripción del problema

El modelo de BPP tiene una cantidad diversa de modificaciones, sin embargo para este proyecto usaremos la presentada por *Martello y Toth (1990)* [1], este modelo es de programación binaria mixta, antes de presentar el modelo, primero haremos una descripción de sus variables y parámetros.

2.1. Parámetros:

Los parámetros a considerar para el problema BPP son los siguientes:

- Número de objetos, es decir, la cardinalidad del conjunto de objetos ($|I| = n$) que queremos ingresar en los contenedores.
- Pesos de los objetos (w_i).
- Número de contenedores disponibles en los que se ingresarán los objetos ($|y| = n$),
- Capacidad de los contenedores (C).

2.2. Variables de decisión.

Nuestra variable estará expresada por una matriz binaria, la cual solo contendrá valores binarios, donde cada 1 indica que el j-ésimo elemento será ingresado al i-ésimo contenedor y 0 en caso contrario.

2.3. Función objetivo y restricciones

La función objetivo para el BPP es la cantidad de contenedores que se usan, es decir, $z = \sum_{i=1}^{n} y_i$, donde y es un vector binario que indica los contenedores que han sido usados, es decir, $y_i = 0$ en caso de no usarse ó 1 en caso de usarse. Mientras que el problema de optimización es minimizar la función objetivo. Una vez que hemos definido todos sus parámetros y variables involucradas, procedemos a presentar el modelo el cual es definido de

la siguiente manera[1, 6]:

$$
\begin{aligned}
& min\ z = min \sum_{i=1}^{n} y_i, \\
& S.a: \\
& \sum_{j=1}^{n} w_j x_{ij} - C y_i \leq 0, \text{ para } i, j = 1, ..., n. \\
& \sum_{j=1}^{n} x_{i,j} = 1, \forall i \in i = 1, ..., n
\end{aligned}
\tag{2.1}
$$

El conjunto de ecuaciones que mostramos en la ecuación (2.1) es una generalización del BPP. A manera de ejemplo, presentamos la Figura (2.1), donde se tienen 7 objetos que son introducidos en 4 contenedores con una capacidad máxima de 8, vemos que la solución de esta imagen es factible porque los contenedores no sobrepasan su capacidad máxima. Por otra parte, su función de costos tiene un valor de 3, debido a que solo uso 3 contenedores.

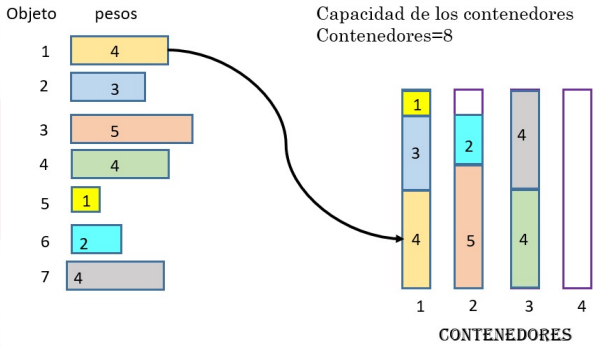

Figura 2.1: Ejemplo del BPP

La diferencia entre este trabajo y el artículo que cita al modelo matemático es que en este trabajo se implementó una metaheurística de "algoritmo genético" para buscar soluciones al problema, mientras que en el trabajo citado se implementaron heurísticas como "First-fit" y "Best-fit Decreasing" para encontrar las soluciones al problema.

3. Metodología

La estructura de un algoritmo genético está formado por los siguientes elementos: Una población de individuos (soluciones) y un conjunto de operadores (cruza y muta) que actúan sobre la población para ir obteniendo nuevas generaciones de individuos [6, 7]. La población se conforma por un conjunto de individuos y cada individuo se representa por un vector binario y recibe el nombre de chromosoma [6, 8]. A continuación presentamos la serie de pasos que sigue la versión simple o canónica de un AG y damos una descripción a detalle [6, 7, 8, 9, 10]:

1. Crear una población inicial de soluciones. La **Población** es una colección (de tamaño N) de individuos que se genera a partir de un proceso aleatorio, y cada individuo tiene información que se encuentra codificada en su cromosoma, esta información contiene soluciones del problema a optimizar.
2. Seleccionar las soluciones más adecuadas de la población actual. El operador de **selección** es muy importante debido a que es el encargado en transmitir el código genético de los individuos más aptos para las futuras generaciones. Hay distintas formas de realizar la selección, por torneo, elitista, por ruleta, etc.
3. Cruzar las soluciones seleccionadas para obtener su descendencia (a los cuales también se les conoce como hijos). El operador de **cruce** consiste en el intercambio genético de cromosomas de los padres dando como resultado los hijos. Hay distintas formas de realizar la cruza, cruce en un solo punto, de dos puntos, de multiples puntos, uniforme, etc.

4. Introducir alguna mutación en los hijos para obtener variantes mutadas. El operador de **mutación** esencialmente modifica al azar algunos de los genes del cromosoma del individuo a mutar. Además, a este proceso se la da una probabilidad de ocurrencia por lo que la mutación puede ocurrir o no.
5. Seleccionar las soluciones que permanecerán en la siguiente generación. Este paso consiste en agregar a los hijos en la población, es decir, los hijos reemplazarán a elementos de la población (preferentemente los menos aptos); conformando la siguiente generación del AG.
6. Repetir de manera iterativa a partir del paso 2, hasta que se cumpla un criterio de parada.

Una vez que se terminan todos los pasos, la mejor solución de la población actual es la que se propone como solución del problema [7].

3.1. Algoritmo genético para el problema del empaquetamiento de contenedores

En base a la descripción de AG, se describe de manera detallada la implementación del algoritmo genético a nuestro proyecto, resaltando las modificaciones y detalles que se contemplaron para poder adaptar esta metodología a nuestra problema del BPP.

1. **Población inicial:** Se crea una población inicial de forma aleatoria, para cada individuo de la población, cada elemento es asignado a un contenedor de manera aleatoria. Este proceso genera una población con soluciones que podrían no ser factibles.
2. **Factibilidad de la población inicial:** Dado que algunas soluciones de la población inicial pueden no ser factibles, la población se somete a un método de factibilidad que consiste en identificar primeramente aquellas soluciones en la que sus contenedores hayan sido sobrepasados (soluciones no factibles), se identifican los índices de las soluciones no factibles. Posteriormente para cada solución no factible se identifican los contenedores que sobrepasaron su límite de capacidad, posteriormente se identifican los elementos contenidos en los contenedores que sobrepasaron su capacidad, se selecciona un elemento al azar y se coloca en otro contenedor donde pueda ingresar sin violar la capacidad del contenedor, posteriormente se verifica si el contenedor ya no viola su capacidad, de ser así se continua con el siguiente contenedor "lleno", de lo contrario se vuelve a intercambiar otro elemento al azar hasta ya no revazar su capacidad, este proceso continua hasta que todos los contenedores respeten su capacidad y de igual forma con las demás soluciones no factibles.
3. **Selección de los padres por torneo:** Se elige un subconjunto de manera aleatoria del tamaño de un 30 % de la población total, se verifica su evaluación en la función objetivo y se eligen a los dos mejores individuos con mejor evaluación en la función objetivo (menor número de contenedores ocupados). Estos individuos serán los padres que darán lugar a las soluciones hijos.
4. **Cruza de los padres:** Una vez identificados los padres para la cruza, se escoge un número entero al azar en el intervalo $[int(n/3), n - int(n/3)]$ para poder asegurar que en la cruza se intercambia suficiente información. En base a este número se realiza un corte vertical, de ambas matrices padres. Y se realiza el intercambio de información de acuerdo a la opción 1 o la opción 2 (ver 3.1), cada una con una probabilidad del 50 % de ocurrir.

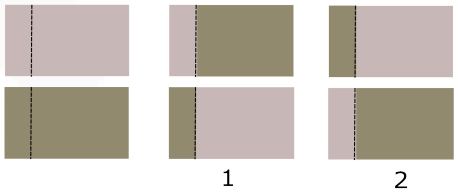

Figura 3.1: Opciones de cruza.

Una vez que se tienen las dos nuevas soluciones, es probable que se obtengan soluciones no factibles, por lo que las dos soluciones se someten a una función de factibilidad, explicado en la parte dos (factibilidad de la población). De esta forma tenemos las dos soluciones factibles y se evalúan en la función objetivo para elegir al mejor candidato de los dos.

5. **Mutación:** Una vez que se obtienen la solución a intercambiar, pasa por un proceso de mutación, el cual se realiza de forma aleatoria, siendo un 50 % de probabilidad que ocurra o no la mutación, de esta forma, si la mutación ocurre, se selecciona de manera aleatoria un contenedor y a su vez un elemento al azar y se coloca de manera aleatoria en otro contenedor cuidando la capacidad del contenedor, en caso de no acceder a un contenedor se escoge otro de forma aleatoria y se pruba si puede ingresar, este proceso ocurre hasta que el elemento es asignado a otro contenedor, de esta forma respetamos la factibilidad de la solución mutada.
6. **Remplazamiento:** Una vez que se tiene la solución a remplazar, se prosigue a verificar si la solución ya se encuentra en la población, de ser así es rechazada la solución y se vuelve a obtener otra, en caso contrario se prosigue a evaluar las soluciones de la población en la función objetivo y se obtiene la peor solución, y está es la que se remplaza por la nueva solución obtenida.
7. **Condición de parada:** Se verifica si se cumple la condición de parada, en este caso se determina por la cantidad de generaciones indicadas, donde una generación equivale a realizar en la población el número de reemplazamientos igual al número de población. Si se cumple la condición se termina el calculo de las generaciones, de lo contrario se vuelve al paso 3 (selección de los padres por torneo).

4. Experimentación computacional

El ambiente computacional donde se realizó la programación y ejecución del algoritmo se muestra a continuación.

Hardware	
Procesador	Intel(R) Core(TM) i5-1035G1 CPU @ 1.00GHz 1.19 GHz
RAM instalada	12.0 GB (11.7 GB usable)
Tipo de sistema	Sistema operativo de 64 bits, procesador basado en x64
Sistema operativo	
Edición	Windows 11 Pro
Versión	24H2
Experiencia	Windows Feature Experience Pack 1000.26100.1.0
Sofware	**Versión**
Conda	23.7.4
Jupyter Notebook	6.5.4
Python	3.11.5

Cuadro 4.1: Software y hardware del sistema de computo implementado.

Se relizaron pruebas del algoritmo con cuatros distintas instancias que se presentan a continuación.

	Instancia 1	**Instancia 2**	**Instancia 3**	**Instancia 4**
Capacidad	10	10	120	120
Pesos	[1,10]	[1,10]	[20,100]	[20,100]
Num. Elementos	50	100	50	100
Población	50	50	50	50
Generaciones	3000	3000	3000	3000

Cuadro 4.2: Instancias. Las primeras dos instancias son iguales a excepción del número de elementos, de igual forma para las instancias 3 y 4.

Todas las instancias fueron ejecutadas 5 veces, por lo que se registraron las evaluaciones en la función objetivo y el tiempo de cómputo. Los resultado que se obtuvieron se resumen en la siguiente tabla

	Instancia 1	Instancia 2	Instancia 3	Instancia 4
Mejor valor	26	61	33	54
Valor Promedio	27	62	34	55
Peor valor	27	62	34	56
Tiempo promedio (s)	77.3652	130.0338	71.0461	128.1832
Valor BFD	25	59	33	50

Cuadro 4.3: Resultados. Se reporta el mejor valor, el peor, el valor promedio, el tiempo promedio y el valor BFD, éste último fue el valor óptimo encontrado con la heurística "Best-Fit Decreasing", el cual sirve de referencia para comparar con el mejor valor obtenido con el algoritmo genético aplicado.

Podemos destacar que los peores valores de la función objetivo no sobrepasan a más de 6 contenedores con respecto a los valores obtenidos por la heurística, y los valores en promedio están entre este rango, entre el óptimo encontrado con la heurística y seis contenedores más, el mejor valor no sobrepasa 4 cuatro contenedores, incluso en la instancia 3 se obtuvo el mismo valor que el de la heurística, aunque con pruebas más prolongas (mayor número de generaciones), los mejores valores pueden llegar a disminuir uno o dos en la evaluación de la función objetivo. También se destaca que entre mayor número de elementos se tengan, el tiempo de ejecución del algoritmo incrementa. En general podemos observar buenos resultados en comparación con la heurística, en tiempo razonables.

5. Conclusiones

El BPP es un problema de optimización combinatoria, ante esta situación, recurrimos a AG (algoritmos genéticos) debido a que sus características podrían adaptarse a las condiciones del BPP, la más importante es su capacidad para escapar de óptimos locales; es decir, asegura acercamiento óptimo global.
De modo que la estructura del AG en este trabajo es: Crea una población inicial de soluciones (de manera aleatoria), la población es enviada a un método de factibilidad, cuando la población es factible se hace una selección (por torneo) de los padres, se hace una cruza de los padres para obtener los hijos, se decide si los hijos mutan y finalmente los hijos terminan siendo agregados a la población, este proceso se repite hasta que se cumple la condición de parada. Si hablamos del método de factibilidad, identifica las soluciones no factibles, se corrigen al identificar los contenedores que sobrepasan su capacidad y los objetos que están en dichos contenedores, se elige uno de estos objetos, se reempaqueta de manera aleatoria en contenedores con espacio disponible. Esto se hace manera iterativa hasta que el contenedor ya no sobrepase su capacidad y se procede a aplicar el mismo proceso para los demás contenedores que sobrepasen su capacidad. En cuanto a la cruza, se corta (con un punto de corte aleatorio) de forma vertical las matrices que tiene la información genética de los padres y se mezcla la información de ambas matrices. En cuanto a la mutación, si ocurre; selecciona de manera aleatoria un contenedor y un elemento contenido en el, se selecciona un contenedor de manera aleatoria, en caso de que el contenedor este lleno; se escoge otro de forma aleatoria, este proceso se repite hasta que el elemento es asignado a otro contenedor. En cuanto al reemplazamiento, los hijos reemplazan a los peores elementos de la población. Por ultimo, la condición de parada es dada por un numero finito de generaciones.
En base a las adaptaciones que se hicieron para aplicar un AG, podemos concluir que los resultados obtenidos logran estar muy cerca de la solución óptima, incluso el peor valor que se encontró en cada instancia se queda muy cerca del valor óptimo. Por ultimo, el aumento en el número de objetos a empaquetar requiere un aumento considerable en los recursos computacionales y eso se ve reflejado en el tiempo de ejecución, esto puede deberse a que el número de restricciones se incrementa.
Las áreas de oportunidad de este trabajo son diversas y aplicables en múltiples contextos. Por ejemplo, puede ser aplicable a empresas en cuestiones de logística y distribución, donde podemos optimizar el espacio dentro de contenedores, camiones, o almacenes. La finalidad es maximizar la cantidad de mercancía que pueda ser enviada o almacenada; dejando el mínimo espacio vacio. Si hablamos de la industria de manufactura, podemos optimizar el corte de materiales tales como el metal, tela o papel, para reducir desperdicios. Además, en la gestión de proyectos y recursos humanos, facilita la programación eficiente de tareas para maximizar la productividad en horarios laborales, asignando tareas de manera que se "empaquen" de la forma más eficiente posible.

Referencias

[1] V. A. D. M.-S. A. Z.-D. J. C. M.-R. A. E.-E. H. Perez Ortega Joaquin, Castillo Zacatelco Hilda, "*Una nueva estrategia heurística para el problema de Bin Packing*," vol. XVII, pp. 155–168, abril-junio 2016.

[2] I. D. M. N. YÁÑEZ, *HIBRIDACIÓN DE ALGORITMOS METAHEURÍSTICOS PARA PROBLEMAS DE BIN PACKING*. PhD thesis, INSTITUTO TECNOLÓGICO DE CD. MADERO, Ubicación de la Universidad, Diciembre 2007.

[3] S. Martello and P. Toth, *Knapsack problems: algorithms and computer implementations*. John Wiley & Sons, Inc., 1990.

[4] E. Falkenauer, "A hybrid grouping genetic algorithm for bin packing," *Journal of heuristics*, vol. 2, pp. 5–30, 1996.

[5] N. Kasap and A. Agarwal, "Augmented-neural networks approach for the bin packing problem," in *Proceedings of the 4th International Symposium on Intelligent Manufacturing Systems*, pp. 348–358, 2004.

[6] T. M. J. C. S. Hernández Romero Norberto, Medina Marín Joselito, *Introducción a Matlab para Resolver Problemas de Ingeniería aplicando Algoritmos Genéticos*. ICBI, 2012.

[7] M. V. J. M. Belén Melián Batista Belén, Moreno Pérez José A., "Algoritmos genéticos. una visión práctica," *Números Revista de Didáctica de las matemáticas*, vol. 71, pp. 29–47, agosto 2009.

[8] Z. Michaelewicz, *Genetic Algorithms+ Data Structures*. Springer, third ed., 1995.

[9] B. G. W. Javier, *Aplicación de algoritmos genéticos en sistemas de control*. PhD thesis, Universidad de los Andes, Bogota D.C., 2005.

[10] N. L. Laura, "La gestión de clientes en el comercio electrónico. aplicación de algoritmos genéticos (ag) al crm.," *Economia industrial*, vol. IV, no. 340, pp. 83–92, 2001.