Practica 01

Problemas de optimización y Heurísticas

Alberto Valentín Velásquez Santos[†], **Rodolfo Morocho Caballero**[†], **Max Houston Ramirez Martel**[†] **and Harold Mondragon Tavara**[†]

[†]*Estos autores contribuyeron igualmente a este trabajo.*

Este archivo fue compilado en 26 de Octubre, 2024

Abstract

Este artículo explora técnicas avanzadas de optimización y heurísticas aplicadas a problemas NP-hard, abordando el Max-Cut, el problema del vendedor viajero y la gestión de portafolios financieros. Se examinan algoritmos metaheurísticos como los algoritmos genéticos, el recocido simulado y las colonias de hormigas, y se introduce el método Johnson y enfoques greedy para resolver problemas de gran escala. En aplicaciones prácticas, se destaca el uso de algoritmos heurísticos en la detección de fraudes en plataformas de streaming y en la predicción de archivos en servicios de almacenamiento. Los casos de estudio demuestran la efectividad de estas técnicas para encontrar soluciones eficientes y escalables, resaltando su valor en la adaptación a entornos cambiantes y la optimización de procesos complejos.

Keywords: *optimización, heurísticas, greedy, knapsack, Machine Learning, reconocimiento heurístico, np-hard*

Received: October 26, 2024 **Revised:** October 26, 2024 **Accepted:** October 26, 2024 **Published:** October 26, 2024

Rho LaTeX Class © This document is licensed under Creative Commons CC BY 4.0.

1. Problemas de optimización

Muchos problemas en campos cuantitativos como las finanzas y la ingeniería son problemas de optimización. Los problemas de optimización se encuentran en el centro de la compleja toma de desiciones y la definición de estrategias.

1.1. Max-Cut y Problema del Vendedor Viajero

La computación cuántica aborda por Qiskit [10], problemas de optimización, específicamente los problemas de Max-Cut y el Problema del Vendedor Viajero (TSP). Ambos son problemas NP-completos, lo que significa que son difíciles de resolver clásicamente para un conjunto N grande de nodos. El problema Max-Cut busca dividir los nodos de un grafo $G = (V, E)$ en dos subconjuntos S y $\bar{S}$, maximizando la función objetivo:

$$\text{Max-Cut}(S) = \sum_{(i,j)\in E} w_{ij} x_i (1 - x_j) \tag{1}$$

donde:

- w_{ij} representa el peso de la arista entre los nodos i y j
- $x_i \in \{0, 1\}$ indica la asignación del nodo i al subconjunto

El TSP se puede fórmular matemáticamente como:

$$\min \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij} \tag{2}$$

sujeto a:

$$\sum_{j=1}^{n} x_{ij} = 1, \quad \forall i \quad \sum_{i=1}^{n} x_{ij} = 1, \quad \forall j \quad x_{ij} \in 0, 1, \quad \forall i, j \tag{3}$$

Para un grafo con $n = 4$ nodos, el espacio de soluciones es:

$$\text{Total de combinaciones} = 2^n = 2^4 = 16 \tag{4}$$

La complejidad computaciónal crece exponencialmente con el número de nodos:

$$O(2^n) \tag{5}$$

Para cada combinación $c \in \{0, 1\}^n$, se evalúa la función de costo:

$$f(c) = \sum_{(i,j)\in E} w_{ij} c_i (1 - c_j) \tag{6}$$

La solución óptima c^* es aquella que maximiza esta función:

$$c^* = \max_{c\in\{0,1\}^n} f(c) \tag{7}$$

En este ejemplo particular, la solución óptima alcanza un costo de 4.0 unidades.

Figure 1 Grafo con 4 vertices.

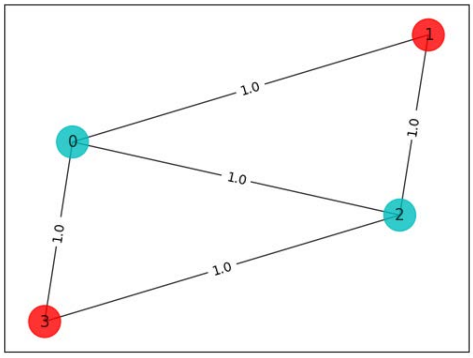

Figure 1. Grafo con 4 vertices

Si bien este enfoque funciona para grafos pequeños, su escalabilidad es muy limitada. El número de combinaciones posibles crece exponencialmente con el número de nodos, siguiendo la función:

$$f(n) = 2^n \tag{8}$$

Para ilustrar esta explosión combinatoria, consideremos un grafo con $n = 1000$ nodos:

$$f(1000) = 2^{1000} \approx 1.071 \times 10^{301} \tag{9}$$

Este número es astronómicamente grande, haciendo que el enfoque de fuerza bruta sea computaciónalmente inviable. Para contextualizar:

- El número de átomos en el universo observable es aproximadamente 10^{80}
- El número de operaciones por segundo de la supercomputadora más rápida es del orden de 10^{18}

Por lo tanto, incluso si pudiéramos evaluar un millón de combinaciones por segundo:

$$\text{Tiempo de cómputo} \approx \frac{2^{1000}}{10^6} \text{ segundos} \tag{10}$$

Este tiempo excede por mucho la edad actual del universo, demostrando la inviabilidad práctica del enfoque de fuerza bruta para problemas de gran escala.

1.2. Gestión de asignación de potencia utilizando el algoritmo del problema de la mochila(Knapsack)

Este método explora un sistema de gestión de asignación de energía que utiliza el problema de la mochila para optimizar el consumo energético en hogares. Morimoto et al. [5] proponen una solución que consiste en enchufes inteligentes y un controlador centralizado que se comunican a través de Wi-Fi. El controlador utiliza un algoritmo de programación dinámica para resolver el problema de la mochila, decidiendo qué electrodomésticos encender o apagar para maximizar la calidad de vida del usuario sin superar un límite de consumo total. Se presentan resultados experimentales que demuestran que el sistema puede funcionar de manera eficiente y práctica estas desiciones se hacen siguiendo las siguientes fórmulas.

$$\max \sum_{i=1}^{n} p_i x_i \tag{11}$$

$$\text{s.t.} \sum_{i=1}^{n} w_i x_i \leq c, x_i \in \{0,1\} \tag{12}$$

Entonces para poder llevarlo a un lenguaje computaciónal tendríamos el siguiente pseudocódigo.

Algorithm 1 Algoritmo de Programación Dinámica para Knapsack

Require: n items con pesos w_i y valores p_i, capacidad c
Ensure: Tabla P con valores óptimos

```
for j = 0 to c do
    P(0, j) ← 0
end for
for i = 1 to n do
    for j = 0 to c do
        if j ≥ w_i then
            P(i, j) ← max{P(i − 1, j − w_i) + p_i, P(i − 1, j)}
        else
            P(i, j) ← P(i − 1, j)
        end if
    end for
end for
```

1.3. Optimización de Portafolio Financiero

La optimización de portafolios financieros es fundamental en la gestión de inversiones, y su objetivo es lograr un equilibrio adecuado entre el rendimiento y el riesgo en la selección de activos. La Teoría Moderna de Portafolio (MPT), propuesta por Markowitz en 1952, establece un marco para maximizar el rendimiento esperado y minimizar el riesgo a través de una diversificación eficiente [2]. El principal desafío radica en considerar la incertidumbre inherente de los mercados financieros y los riesgos asociados a cada inversión.
La MPT se basa en dos principios fundamentales: maximizar el rendimiento esperado y minimizar el riesgo, generalmente representado por la varianza del rendimiento. Para formalizar estos conceptos, sea un portafolio compuesto por n activos, donde el peso de cada activo i en el portafolio está dado por w_i. El rendimiento esperado $E(R_P)$ del portafolio es:

$$E(R_P) = \sum_{i=1}^{n} w_i E(R_i) \tag{13}$$

donde $E(R_i)$ es el rendimiento esperado del activo i

El riesgo asociado, representado por la varianza del portafolio σ_p^2, está dado por:

$$\sigma_p^2 = \sum_{i=1}^{n} \sum_{j=1}^{n} w_i w_j Cov(R_i, R_j) \tag{14}$$

donde $Cov(R_i, R_j)$ es la covarianza entre los rendimientos de los activos i, j

El objetivo de la optimización del portafolio es resolver el siguiente problema de programación cuadrática:

$$\min_w \tfrac{1}{2} w^t \Sigma w \text{ sujeto a } w^t 1 = 1, w_i \geq 0$$

donde Σ es la matriz de covarianzas y w representa el vector de pesos.

La optimización de portafolios según Markowitz se enfrenta a varios desafíos prácticos, entre ellos:

- **Dimensionalidad alta:** A medida que crece el número de activos en el portafolio, la complejidad computaciónal de la optimización crece exponencialmente debido a la necesidad de calcular las covarianzas entre los activos.
- **Inestabilidad de los parámetros:** Las estimaciones de rendimientos y covarianzas son frecuentemente volátiles, afectando la robustez de las soluciones.
- **Costos de transacción y restricciones prácticas:** Los modelos clásicos no consideran costos de transacción ni restricciones de mercado, lo que limita su aplicabilidad [2].

Para abordar estas limitaciones, se han propuesto enfoques avanzados que incluyen algoritmos heurísticos y metaheurísticos, así como modelos robustos que tienen en cuenta la incertidumbre de los parámetros.

Los algoritmos metaheurísticos, como los **algoritmos genéticos (GA)**, el **recocido simulado (simulated annealing)** y el **algoritmo de colonia de hormigas**, han demostrado ser eficaces para abordar la optimización de portafolios en escenarios complejos y no lineales. Estos enfoques no garantizan una solución óptima global, pero tienen la ventaja de explorar de manera eficiente grandes espacios de búsqueda [1], [3].

- **Algoritmos Genéticos**: Utilizan principios de evolución biológica y se destacan en problemas con restricciones no lineales [2].
- **Recocido Simulado**: Inspirado en el enfriamiento de metales, permite escapar de óptimos locales y realizar una exploración más profunda del espacio de soluciones [1].
- **Algoritmo de Colonia de Hormigas**: Emula el comportamiento colectivo de las hormigas y ha mostrado buenos resultados en problemas con restricciones de cardinalidad y costos de transacción [3].

Estas técnicas permiten abordar de manera eficiente problemas que involucran múltiples restricciones y objetivos, proporcionando una mejora en la robustez de las soluciones en comparación con los enfoques tradicionales.

Además de los enfoques metaheurísticos, los modelos robustos han ganado popularidad en los últimos años. Estos modelos abordan la incertidumbre en las estimaciones de los parámetros, utilizando técnicas como el enfoque de conjuntos robustos para considerar una gama de escenarios adversos en la optimización del portafolio . El enfoque robusto busca soluciones que sean menos sensibles a las variaciones de los datos de entrada, lo que es especialmente útil en mercados financieros volátiles [4].

2. Heurística

2.1. Método Johnson

El método de Johnson se presenta como una heurística desarrollada para resolver problemas de secuenciación de tareas y planificación, especialmente aquellos donde se busca optimizar el tiempo total de procesamiento en un sistema. En este contexto, el término *heurística* implica que el método de Johnson no necesariamente garantiza la solución óptima en todos los escenarios, sino que ofrece una aproximación que logra un equilibrio entre la calidad de la solución y el costo computaciónal. Esto es particularmente útil cuando el problema es de gran escala, lo que hace inviable analizar todas las soluciones posibles.

Aplicado a la planificación de redes eléctricas, el método de Johnson permite la asignación eficiente de puntos de demanda a subestaciones mediante una secuencia que minimiza el tiempo y los costos operacionales. En el análisis de Breda y Mestria [9], el método Johnson se emplea específicamente para estructurar la red eléctrica de forma óptima. La heurística de Johnson asigna cada punto de demanda a una subestación cercana de acuerdo con una secuencia de desiciones que minimiza el costo incremental. Esto, a su vez, logra una asignación balanceada que mantiene una operación estable y confiable en la red. Según el estudio, este método heurístico ofrece una solución eficiente y de alta calidad, lo que lo convierte en una herramienta poderosa para abordar problemas de gran escala en el ámbito de la optimización de redes.

2.2. Algoritmos Genéticos

Los Algoritmos Genéticos **(Genetic Algorithms, GA) Michalewicz1999**, **Back1997**, **Diaz1996** son una familia de modelos computaciónales inspirados en los mecanismos de herencia y evolución natural.

Mediante una imitación del mecanismo genético de los organismos biológicos, los algoritmos genéticos exploran el espacio de soluciones asociado a un determinado problema. Se establece una codificación apropiada de las soluciones del espacio de búsqueda y una forma de evaluar la función objetivo para cada una de estas codificaciones. Las soluciones se identifican con individuos que pueden formar parte de la población de búsqueda. La codificación de una solución se interpreta como el cromosoma del individuo compuesto de un cierto número de genes a los que les corresponden ciertos alelos. El gen es la característica del problema; alelo, el valor de la característica; genotipo, la estructura y fenotipo, la estructura sometida al problema. Cada cromosoma es una estructura de datos que representa una de las posibles soluciones del espacio de búsqueda del problema. Los cromosomas son sometidos a un proceso de evolución que envuelve evaluación, selección, recombinación sexual o cruce y mutación. Después de varios ciclos de evolución la población deberá contener individuos más aptos. Se consideran dos operaciones básicas: ***la mutación*** y ***el cruce***. La mutación de un individuo consiste en modificar uno o varios genes cambiando al azar el alelo correspondiente. El cruce de dos individuos, llamados padres, produce un individuo hijo tomando un numero k (elegido al azar) de genes de uno de los padres y los $t-k$ del otro. La población evoluciona de acuerdo a las estrategias de selección de individuos, tanto para las operaciones como para la supervivencia. La selección se puede hacer simulando una lucha entre los individuos de la población con un procedimiento que, dados dos individuos, selecciona uno de ellos teniendo en cuenta su valoración (la función objetivo) y la adaptación al ambiente y a la población (criterios de diversidad, representatividad).

Mediante una imitación de este mecanismo, los algoritmos genéticos exploran el espacio de soluciones asociado a un determinado problema. **Algorithm** 2 presenta el esquema básico del algoritmo genético. En cada iteración t, se mantiene una población de individuos $P(t) = \{x_1^t, \cdots, x_n^t\}$ donde cada individuo representa una potencial solución del problema a resolver. Cada solución x_1^t se evalúa para obtener cierta medida de su calidad o fitness. Luego, se genera una nueva población $P(t+1)$ tomando como base a los mejores individuos; algunos de los cuales sufren transformaciones a partir de la aplicación de operadores "genéticos".

Algorithm 2 Esquema básico de un Algoritmo Genético

```
1:  procedure AG
2:      t ← 0
3:      inicializar(P(t))
4:      evaluar(P(t))
5:      while no-finalizacion do
6:          t ← t + 1
7:          Seleccionar P(t) desde P(t − 1)
8:          Hacer-Cruzamientos(P(t))
9:          Aplicar-Mutaciones(P(t))
10:     end while
11:     return mejor solución
12: end procedure
```

Típicamente, estos operadores son: cruzamiento o crossover (nuevos individuos son generados a partir de la combinación de dos o más individuos), y mutación (nuevos individuos son generados a partir de pequeños cambios en un individuo). La idea intuitiva del operador de cruce es permitir el intercambio de información entre individuos de la población. La mutación, en cambio, permite incorporar nueva información que no se puede generar a partir de la clausura transitiva del cruce sobre la población. Se puede entender el mecanismo de selección si se plantea cada iteración del Algoritmo Genético como un proceso en dos etapas. En la primera, se aplica algún mecanismo de selección sobre la "población actual" para crear una "población intermedia"; y en la segunda etapa, a partir de esta población, se aplica cruce y mutación para generar una "nueva población". El resultado del proceso de selección es un conjunto de entrecruzamiento formado por los mejores individuos de la población. En base a ellos, la etapa de cruce elegirá a los padres y generará a los hijos; sobre la salida de esta etapa, se ejecutará la etapa de mutación. La nueva población puede estar formada solamente por los hijos, o por padres e hijos en alguna proporción adecuada.

2.3. Algoritmos Greedy

El método *greedy* es una heurística que se destaca en problemas de optimización combinatoria debido a su enfoque basado en desiciones inmediatas y locales. La esencia de este método radica en seleccionar, en cada etapa del proceso, la opción que parece más ventajosa en ese momento, es decir, la que maximiza el beneficio o minimiza el costo de forma inmediata, sin analizar el impacto que dicha elección puede tener en etapas futuras. Este enfoque no asegura que se alcance siempre la solución óptima global; sin embargo, es sumamente útil para problemas en los que el análisis de todas las combinaciones posibles sería impráctico o consumiría demasiados recursos.

Al operar bajo esta lógica, el método *greedy* se adapta bien a situaciones donde el tiempo y los recursos de cálculo son limitados, ofreciendo soluciones rápidas y lo suficientemente buenas en la mayoría de los casos. Mari [7] lo aplica en problemas NP-hard, como el de cobertura máxima y el conjunto independiente, donde el número de posibles soluciones es exponencial. En cada uno de estos problemas, la estrategia *greedy* selecciona iterativamente la opción con el mayor beneficio local inmediato, logrando así una solución aceptable sin necesidad de explorar todas las alternativas. A través de esta aproximación, Mari demuestra que el método *greedy* es una herramienta versátil y eficaz en problemas de alta complejidad, ya que permite obtener resultados satisfactorios de manera eficiente y con menor costo computaciónal que los métodos exactos. En resumen, el método *greedy* se consolida como una heurística valiosa cuando el objetivo es

encontrar soluciones de calidad en menor tiempo, sin exigir precisión absoluta.

3. Casos reales

3.1. Detección de abuso y fraude en servicios de streaming mediante Machine Learning con reconocimiento heurístico

La Plataforma de streaming Netflix desarrollo un framework para la detección de fraudes en su aplicación empleando algoritmos heurísticos desarrollados en la compañía y Machine Learning (ML), usando diferentes fuentes de datos como la trazabilidad de los usuarios, conexiones, contenido visualizado, ubicaciones, entre otros. Los autores Esmaeilzadeh et al. (2022) [8], expresan en su artículo que desarrollar un modelo ML para la detección de anomalías en el contexto de su aplicación implica muchos retos, entre ellos están, el análisis en tiempo real que puede llegar a ser costoso en términos de infraestructura y poca escalable, la definición de anomalía depende del contexto de negocio y aplicación y la cantidad de datos para alimentar al modelo, ya que al tratarse de casos no recurrentes se presenta una disparidad en la cantidad de datos. Debido a estas razones los autores plantean las siguientes soluciones. Primero, una solución basada en reglas que permita identificar irregularidades tomando como base el conocimiento y experiencia de los expertos de negocio, brindando características esenciales y contexto sobre los incidentes que permitan elaborar algoritmos heurísticos. Segundo, aplicar otra solución basada en modelo que permita identificar los casos anormales en la plataforma de forma automática. Para ejecutar este marco de trabajo primero se separaron en tres tipos de fraude, de cuentas, servicios y contenido, entonces la primera solución planteada complementa a la segunda, ya que ayuda a etiquetar y limpiar los datos que ingesta a la segunda solución, con la finalidad de identificar cual es el fraude que más se comete en la aplicación y poder identificar la cuenta asociada al cliente que infringen las políticas de la aplicación. Por último, con algoritmos heurísticos desarrollados con base a la experiencia y conocimiento de los expertos, complementa un modelo ML incrementando su precisión hasta en un 86% en el análisis de fraude en tiempo real.

3.2. Uso de Machine Learning para predecir el próximo archivo

Un segundo caso de éxito lo aplica la empresa de servicio de almacenamiento en la nube Dropbox, en su artículo de ingeniería el autor Kumar (2019) [6] presenta como una funcionalidad de predicción de archivos que parece simple para el usuario tiene una gran complejidad por detrás, esto se debe a que para poder desarrollarlo realizaron algunos algoritmos heurísticos tomando como punto de inicio variables como la frecuencia con la que se accede a un archivo y el acceso reciente, sobre esto implementaron los algoritmos para obtener la probabilidad de cuál sería el próximo archivo que el usuario necesitaría, sin embargo, algunos obstáculos en ciertos escenario y el incremento en la complejidad computaciónal es que optaron por construir un primer modelo de Machine Learning que pueda analizar el comportamiento del usuario sobre los archivos que crea y modifica dentro de la aplicación.

4. Conclusiones

- Las soluciones clásicas son óptimas siempre y cuando no se tengan que analizar grandes cantidades de datos de ser el caso pasamos a un tiempo computaciónal fuera de las posibilidades y entendimiento humano por lo que se tiene que explorar otras soluciones que estén referenciadas en tiempo humano.
- Los algoritmos heurísticos se han mostrado efectivos para resolver problemas complejos en diversos contextos, ofreciendo soluciones escalables y eficientes. Al aplicar estas técnicas, es posible superar limitaciones de costo computaciónal y disponibilidad de datos, creando modelos que se adaptan bien a condiciones cambiantes y que permiten optimizar tanto el rendimiento como la capacidad de respuesta ante desafíos específicos. Estos casos de éxito subrayan la importancia de la heurística como herramienta fundamental en la búsqueda de soluciones prácticas y de alta eficiencia.
- De los casos de éxito se llega a la conclusión que se puede aplicar heurística en diferente entornos y problemas, mediante el desarrollo de algoritmos heurísticos se puede desarrollar soluciones escalables y eficientes, a modo de ejemplo el caso de Netflix que mediante estos algoritmos pudo entrenar un modelo de ML mucho más grande y solucionar el problema de fraude en su plataforma, también el caso de Dropbox que pudo desarrollar una funcionalidad para el usuario final brindando facilidad en la navegación entre los archivos de su plataforma.

Tabla de Contenidos

References

[1] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, “Optimization by simulated annealing”, *Science*, vol. 220, no. 4598, pp. 671–680, 1983.

[2] K. Deb, *Multi-Objective Optimization Using Evolutionary Algorithms*. Chichester: Wiley, 2001.

[3] M. Dorigo and T. Stützle, *Ant Colony Optimization*. Cambridge, MA: MIT Press, 2004.

[4] A. Ben-Tal, L. El Ghaoui, and A. Nemirovski, *Robust Optimization*. Princeton, NJ: Princeton University Press, 2009.

[5] N. Morimoto, Y. Fujita, M. Yoshida, *et al.*, “A power allocation management system using an algorithm for the knapsack problem”, in *2014 IEEE 38th Annual International Computers, Software and Applications Conference Workshops*, IEEE, 2014, pp. 590–595, ISBN: 978-1-4799-3578-9. DOI: 10.1109/COMPSACW.2014.99.

[6] N. Kumar, *Using machine learning to predict what file you need next*, May 2019. [Online]. Available: https://dropbox.tech/machine-learning/content-suggestions-machine-learning.

[7] A. Mari, “Aplicación de métodos greedy en problemas np-hard”, *Journal of Optimization*, 2020.

[8] S. Esmaeilzadeh, N. Salajegheh, A. Ziai, and J. Boote, *Abuse and fraud detection in streaming services using heuristic-aware machine learning*, Mar. 2022. [Online]. Available: https://arxiv.org/abs/2203.02124.

[9] A. Breda and M. Mestria, “Optimización de redes eléctricas usando el método johnson”, *Nombre de la Revista*, 2023.

[10] Qiskit Community. “Maximum cut and traveling salesman problems using qiskit”. (2024), [Online]. Available: https://qiskit-community.github.io/qiskit-optimization/locale/es_UN/tutorials/06_examples_max_cut_and_tsp.html (visited on 05/25/2024).