Recibido: 18-ene-2026 Aceptado: 30-mar-2026 e-ISSN 2954-4890

Sección: *¿Cómo funciona?*

Algoritmos genéticos: de la teoría evolutiva a la eficiencia industrial

Genetic algorithms: from evolutionary theory to industrial efficiency

Byron Cristian Guzmán-Marín[1]
Ailén Dumont-Viollaz[2,3,4*]

[1]*Facultad de Ingeniería y Ciencias, Univ. Adolfo Ibáñez, Av. Diagonal Las Torres 2700, 7910000 Santiago, Chile.*
[2]*Programa de Doctorado en Medicina de la Conservación, Universidad Andrés Bello, República 440, Santiago, Chile.*
[3]*One Health Institute, Faculty of Life Sciences, Universidad Andrés Bello, Santiago, 8370251, Chile.*
[4]*Escuela de Medicina Veterinaria, Facultad de Ciencias de la Vida, Universidad Andrés Bello, Santiago 8370134, Chile.*

**Autor para la correspondencia: a.dumontviollaz@uandresbello.edu*

RESUMEN

La optimización de procesos ha buscado inspiración en la naturaleza para superar las limitaciones de los métodos matemáticos tradicionales. Este enfoque bio-inspirado ha sido fundamental para abordar problemas donde el cálculo convencional resulta insuficiente. Los Algoritmos Genéticos (AG) permiten explorar vastos espacios de búsqueda de manera eficiente, fortaleciendo la toma de decisiones y optimizando recursos sin la dependencia exclusiva de la experimentación física. El presente artículo explora de dónde provienen teóricamente los AG y demuestra su potencial a través de dos casos de estudio. Se evidencia cómo estas herramientas bio-inspiradas se aplican con éxito para resolver problemas industriales complejos.

Palabras clave: Bioprocesos, optimización, simulación computacional.

SUMMARY

Process optimization has looked to nature for inspiration to overcome the limitations of traditional mathematical methods. This bio-inspired approach has been fundamental in addressing problems where conventional computation proves insufficient. Genetic algorithms (GAs) allow for the efficient exploration of vast search spaces, strengthening decision-making and optimizing resources without relying solely on physical experimentation. This article explores the theoretical origins of GAs and demonstrates their potential through two case studies. It shows how these bio-inspired tools are successfully applied to solve complex problems in industry.

Keywords: Bioprocesses, optimization, computational simulation.

El arte de simular lo imposible

Desde el plegamiento de proteínas hasta el destino celular, la naturaleza resuelve problemas complejos cada segundo. A este proceso de hallar la mejor solución entre infinitas opciones, lo llamamos optimización. Ante un número astronómico de combinaciones, el cálculo exhaustivo es computacionalmente inmanejable. Para superar esta limitación, recurrimos a la simulación como un atajo inteligente. Hoy podemos crear "mundos virtuales" y usar estrategias inspiradas en la biología y la física para resolver problemas de alta complejidad que antes se consideraban inabarcables.

Desglosando el concepto: de algoritmos a metaheurísticas

Un algoritmo es una secuencia finita y bien definida de instrucciones para resolver un problema. Pensemos en una receta de cocina, una serie de pasos para lograr un plato. Son procedimientos deterministas.

La heurística, del griego "heuriskein" ("hallar"), es una técnica para encontrar una solución "buena" de manera eficiente, especialmente cuando los métodos exactos son demasiado lentos [1]. Esta se adapta y se diseña para resolver un problema específico [2]. Pensemos en un conductor en una ciudad congestionada, no calcula matemáticamente cada ruta, sino que usa su experiencia para encontrar un camino rápido para ese trayecto en particular. Son atajos inteligentes.

Mientras que la heurística es específica, una metaheurística es un algoritmo de propósito general que puede ser aplicado para resolver casi cualquier problema de optimización [3]. No son algoritmos que resuelven un problema específico directamente, sino "estrategias maestras" que guían a otros algoritmos para explorar el espacio de soluciones. No garantizan la solución perfecta, pero encuentran una suficientemente buena en un tiempo razonable, crucial cuando el espacio de búsqueda es tan vasto que resulta computacionalmente intratable. Las metaheurísticas no se limitan exclusivamente a estas magnitudes extremas, son herramientas fundamentales para resolver cualquier problema donde los métodos exactos consuman demasiado tiempo o recursos para ser viables en la práctica. Entre las técnicas más importantes y reconocidas en este campo se encuentran los Algoritmos Genéticos (AG).

¿Qué es un AG?

Un AG es una técnica de búsqueda y optimización heurística inspirada en los principios de la evolución natural y la genética de poblaciones descritos por Darwin [4]. A diferencia de los métodos matemáticos tradicionales que siguen un camino lineal fijo, los AG trabajan con una "población" de soluciones candidatas que evolucionan simultáneamente a lo largo del tiempo [5]. En general, el flujo de trabajo o pseudocódigo consiste en los siguientes pasos [6] (Figura 1A):

1. **Inicialización (i)**: se genera una población inicial aleatoria de cromosomas que representan las posibles soluciones. Es fundamental asegurar diversidad estructural para explorar ampliamente el espacio de búsqueda y evitar una convergencia prematura. El "genotipo" de estos cromosomas puede codificarse, por ejemplo, mediante codificación binaria (ceros y unos para decisiones de inclusión/exclusión) o codificación de permutación (secuencias numéricas para problemas de ordenamiento o asignación).

2. **Evaluación (f(X))**: se aplica una función de aptitud (fitness) a cada cromosoma para cuantificar qué tan buena es la solución codificada. En problemas con restricciones estrictas (como límites de peso

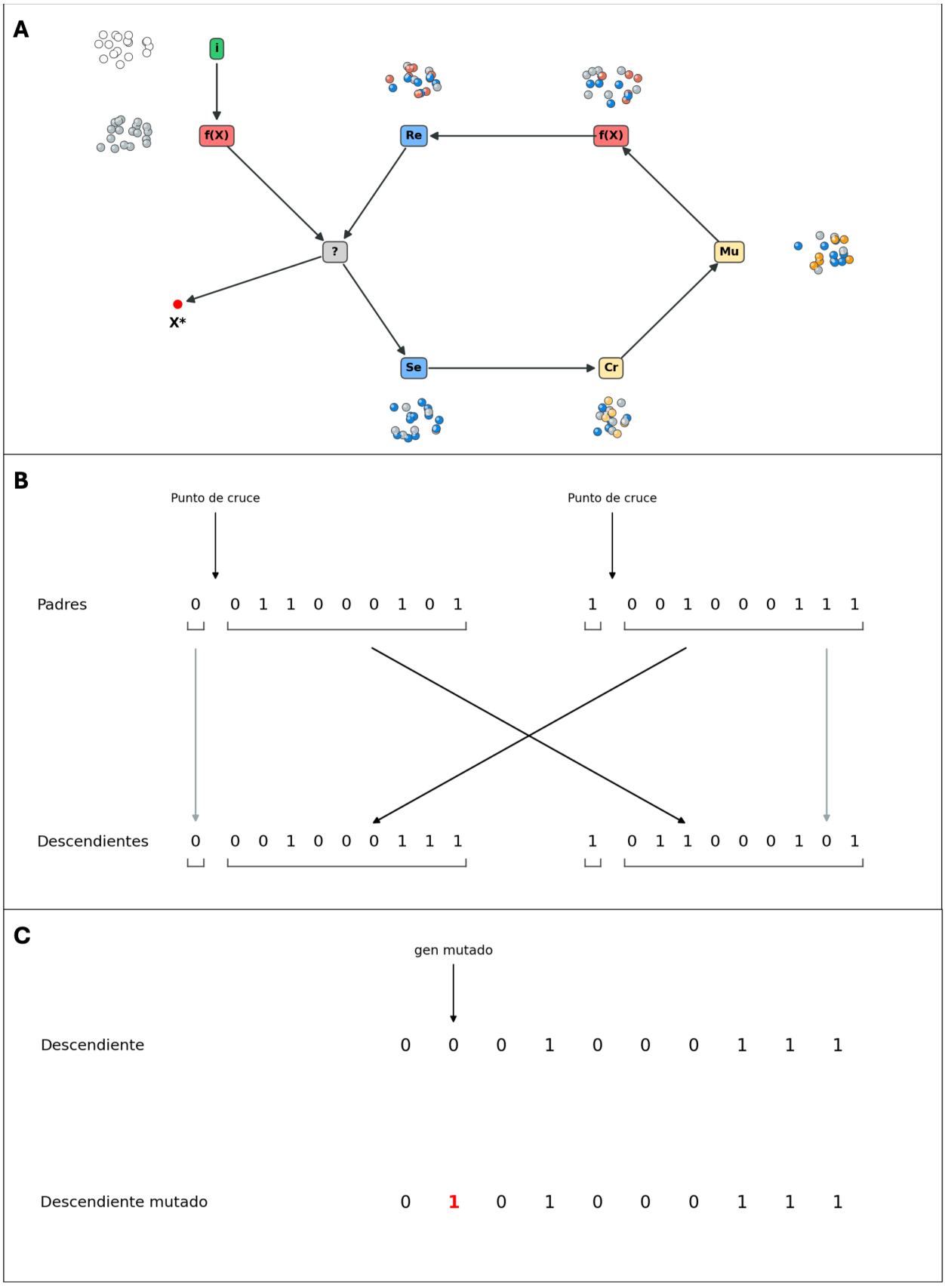

Figura 1. *(A) Ciclo de un Algoritmo Genético. i: inicialización, f(X): evaluación, ?: condición de término, Se: selección, Cr: cruzamiento, Mu: mutación, Re: reemplazo, X*: mejor solución encontrada. Adaptado de Ticona Melo [6]. (B) Operador de cruce (ejemplo ilustrativo empleando codificación binaria). (C) Operador de mutación (ejemplo ilustrativo empleando codificación binaria).*

o de capacidad), suele incorporar mecanismos de penalización, que reducen drásticamente la aptitud de aquellos individuos que proponen soluciones físicamente inviables, forzando a la población a evolucionar dentro de los límites permitidos.

3. **Ciclo evolutivo (Iteración)**: a través de operadores genéticos, la población se transforma generación tras generación explorando nuevas y mejores soluciones. Este ciclo iterativo consta de:

    - **Selección (Se)**: se eligen los cromosomas que serán padres en la siguiente generación. Individuos con mejor aptitud tienen mayor probabilidad de ser seleccionados para transmitir sus genes. Esta elección puede darse por "selección por ruleta" (la probabilidad es directamente proporcional a la aptitud del individuo) o "selección por torneo" (se enfrentan subgrupos de individuos y sobrevive el mejor), entre otros.

    - **Cruzamiento (Cr)**: es el principal operador genético y representa la reproducción sexual. Opera sobre dos cromosomas a la vez para generar descendientes que combinan las características de ambos padres (Figura 1B).

    - **Mutación (Mu)**: introduce cambios aleatorios en los genes, alcanzando zonas del espacio de búsqueda no cubiertas por la población actual, manteniendo la diversidad genética (Figura 1C).

    - **Reemplazo (Re)**: una vez aplicados los operadores, se seleccionan los mejores individuos (padres e hijos) para conformar la población de la generación siguiente.

4. **Condición de término (?)**: se define el criterio para finalizar la búsqueda iterativa. Aunque el objetivo es detenerse al encontrar la solución óptima, al ser comúnmente desconocida, se aplican otros límites prácticos como un número máximo de generaciones o la ausencia de cambios significativos en la población.

Tabla 1. *Rendimiento operativo: Biorreactor vs. Proteína.*

	Proteína 1	Proteína 2	Proteína 3	Proteína 4
Biorreactor A	9	12	8	10
Biorreactor B	10	9	11	13
Biorreactor C	14	10	12	9
Biorreactor D	11	13	10	8

De la fábrica a la célula: aplicaciones transversales

La versatilidad de los AG les permite resolver problemas críticos en múltiples sectores [4]. En la industria, se usan para optimizar la programación de la producción y minimizar costos [7]; en ciencias de los materiales, aceleran el descubrimiento de estructuras; en medicina asisten en el diagnóstico clínico [8]. En el campo de la bioingeniería, permiten el ensamblado preciso de secuencias de ADN y la detección bioinformática de inhibidores virales, abriendo paso al diseño de enzimas mediante la integración con modelos de lenguaje [9].

Estudio de caso 1: planificación de producción en biorreactores

En una planta de bioprocesos, la eficiencia lo es todo [10, 11]. Imaginemos una planta que debe producir 4 proteínas recombinantes diferentes utilizando 4 biorreactores disponibles. La Tabla 1 de tiempos de proceso estimados, representa cuántas horas tarda un biorreactor en alcanzar la densidad celular objetivo para una proteína. Se debe

encontrar la combinación de asignaciones que minimice el tiempo total de operación de la planta, respetando que cada equipo procese una única proteína.

Para aplicar el AG, convertimos este rompecabezas en un problema evolutivo. Aquí, cada posible plan de producción se trata como un individuo con su propio código genético:

- **Genotipo (el plan de producción)**: es el "ADN" de nuestra solución, es decir, una secuencia numérica única que dicta qué proteína se cultiva en qué tanque. Un genotipo 3, 1, 4, 2 significa: "El biorreactor A produce la proteína 3, el B la proteína 1, el C la proteína 4 y el D la proteína 2".
- **Función de aptitud (el cronómetro)**: el algoritmo suma las horas que cada biorreactor tardaría según el genotipo propuesto. Aquí, los planes de producción con la suma de tiempo más baja son los más "aptos".
- **Restricción (la capacidad física)**: para que la solución sea válida en la realidad, impusimos una regla física inquebrantable: Un Biorreactor = Una Proteína. El algoritmo no puede proponer que el Biorreactor A produzca dos proteínas simultáneamente ni dejar una proteína sin asignar, descartando cualquier mutación que viole esta regla.
- **Dinámica evolutiva (el proceso en acción)**: imaginemos una generación intermedia dentro de una población constante de 8 individuos, donde se evalúan pares para elegir al más apto. Un ganador con genotipo 1, 4, 2, 3 se traduce en una asignación física (fenotipo) donde los biorreactores A, B, C y D producen las proteínas 1, 4, 2 y 3 respectivamente, con un tiempo total de 42 horas. A través del operador de cruzamiento con otro individuo, se genera descendencia asegurando no duplicar proteínas. Posteriormente, el operador de mutación introduce variabilidad intercambiando aleatoriamente la posición de dos proteínas. Si el nuevo genotipo resulta en 3, 1, 2, 4, la nueva asignación física reduce el tiempo a 36 horas. Este ciclo se repite iterativamente, descartando las combinaciones ineficientes.

Tras ejecutar la simulación mediante un código desarrollado por los autores en Python (utilizando librerías estándar para la generación de secuencias aleatorias), el algoritmo convergió exitosamente identificando la configuración óptima con un tiempo total de 36 horas, y genotipo 3, 1, 2, 4. El criterio de paro establecido para finalizar la búsqueda iterativa fue completar un límite máximo de 10 generaciones evolutivas, lo cual demostró ser computacionalmente suficiente para garantizar la convergencia dada la magnitud del espacio de búsqueda en este caso de estudio.

Si el código de esta simulación se ejecuta múltiples veces, el camino evolutivo hacia la solución variará, dada la naturaleza estocástica (probabilística) inherente a los AG. El algoritmo rara vez repetirá la misma secuencia de pasos intermedios. No obstante, la robustez de esta técnica bio-inspirada radica precisamente en que, independientemente de la ruta exploratoria aleatoria que tome, el sistema converge consistentemente hacia la solución óptima global al contar con los parámetros de aptitud correctos.

Estudio de caso 2: el problema de la mochila (asignación de recursos)

Se tiene una mochila con una capacidad máxima de peso (C) y una serie de n ítems disponibles. Cada ítem posee un peso específico (p_i) y aporta un beneficio determinado (b_i). El objetivo es seleccionar qué ítems introducir en la mochila para maximizar el beneficio total, asegurando que la suma de sus pesos no supere la capacidad física de la mochila. A modo de ejemplo consideraremos C=120 y n=

10, donde:

Tabla 2. *Parámetros de los ítems disponibles.*

Ítem	1	2	3	4	5	6	7	8	9	10
Peso	20	25	20	20	35	40	35	30	35	25
Beneficio	50	35	45	20	35	40	50	50	45	40

Este problema requiere un enfoque distinto:

- **Genotipo (Codificación binaria)**: el "ADN" de cada solución se representa mediante una cadena de ceros (el ítem es descartado) y unos (el ítem es incluido). Por ejemplo, el genotipo 1 1 0 1 0 1 1 1 0 0 propone llevar los ítems 1, 2, 4, 6, 7 y 8, dejando fuera al resto.
- **Función de aptitud con penalización**: se introduce un mecanismo de penalización. Si la suma de los pesos es menor o igual a 120, la aptitud (*F*) equivale al total del beneficio obtenido. Si la mochila se "rompe" (el peso supera la capacidad de 120), el individuo es penalizado y su aptitud se calcula restando la suma total de los pesos al beneficio total (*F* = Beneficios - Pesos). Evaluando el genotipo anterior (1 1 0 1 0 1 1 1 0 0), la suma de beneficios es 245 y la de pesos es 170. Al exceder el límite, la penalización resulta en *F* = 245 - 170 = 75.
- **Dinámica evolutiva (Operadores aplicados)**: se utiliza un método probabilístico proporcional. Los individuos con mayor aptitud obtienen intervalos de probabilidad más amplios (por ejemplo, el individuo con *F*=160 abarca el intervalo de selección [0,58-0,89]), aumentando sus posibilidades de ser seleccionados como progenitores.
- **Cruce**: una vez seleccionados dos padres, se define un punto de corte en sus cromosomas y se intercambian sus segmentos para generar descendientes que combinan los objetos de ambas mochilas parentales.
- **Mutación**: con una baja probabilidad, se altera aleatoriamente un bit del cromosoma (un ítem que estaba dentro se saca, o viceversa), inyectando nueva diversidad para evitar que el algoritmo se estanque.
- **Resultado final**: el algoritmo converge en la configuración óptima representada por el genotipo 1 1 1 1 0 0 0 1 0 0 con un beneficio máximo de *F* = 200.

Conclusiones

Los AG aplican la eficiencia biológica a la ingeniería. Este estudio lo demuestra mediante dos enfoques: la optimización de tiempos en bioprocesos usando permutación, y la resolución del problema de la mochila con codificación binaria y penalizaciones. Ambos casos prueban que los modelos evolutivos hallan configuraciones óptimas sin recurrir a cálculos exhaustivos. La optimización bio-inspirada trasforma la ciencia en una realidad operativa más precisa y económica.

Disponibilidad de datos y código

El código fuente en Python desarrollado para la simulación de los dos casos de estudio (Planificación de Biorreactores y Problema de la Mochila) se encuentra disponible de forma abierta para su consulta y replicación en el siguiente repositorio de Google Colab: https://colab.research.google.com/drive/1lqGJ9AV-t92kV8dssmGYAxbJxCzGJzrq5

Agradecimientos

Los autores agradecen a la Agencia Nacional de

Investigación y Desarrollo (ANID)/ Programa de Becas/Doctorado Becas Chile/2024-21241524 y 2026-21262510

Gemini se utilizó únicamente para la edición del lenguaje y la mejora de la claridad durante la preparación de este manuscrito sin comprometer su originalidad.

Referencias

[1] Martí, R. (2003). Procedimientos metaheurısticos en optimización combinatoria. *Matemátiques*, Universidad de Valencia, 1(1), 3-62.

[2] Talbi, E. (2009). *Metaheuristics. From design to implementation.* John Wiley & Sons Inc.

[3] Glover, F. (1986). Future paths for integer programming and links to artificial intelligence. *Computers and Operation Research* 13(5),533-549. https://doi.org/10.1016/0305-0548(86)90048-1

[4] McCall, J. (2005). Genetic algorithms for modelling and optimisation. *Journal of Computational and Applied Mathematics* 184(1), 205-222. https://doi.org/10.1016/j.cam.2004.07.034

[5] Katoch, S., Chauhan, S. S., Kumar, V. (2021). A review on genetic algorithm: past, present, and future. *Multimedia tools and applications* 80(5), 8091-8126. https://doi.org/10.1007/s11042-020-10139-6

[6] Ticona Melo, L. R., Oliveira, R., Echegaray, R., Bittencourt, T. N. (2010). *Optimización de estructuras metálicas para puentes mediante algoritmos genéticos con el programa PUENFLEX Ver. 2.0. Mecánica Computacional* 29, 9327-9344.

[7] Angulo Guerrero, R. J., Garcia Camacho, D. J. (2023). Optimización de procesos de producción mediante el uso de algoritmos genéticos. *Ingeniería y sus Alcances, Revista de Investigación* 7(18), 316-324. https://doi.org/10.33996/revistaingenieria.v7i18.109

[8] Ghaheri, A., Shoar, S., Naderan, M., Hoseini, S. S. (2015). The applications of genetic algorithms in medicine. *Oman medical journal* 30(6), 406. https://doi.org/10.5001/omj.2015.82

[9] Nana Teukam, Y. G., Zipoli, F., Laino, T., Criscuolo, E., Grisoni, F., Manica, M. (2025). Integrating genetic algorithms and language models for enhanced enzyme design. *Briefings in bioinformatics* 26(1), bbae675. https://doi.org/10.1093/bib/bbae675

[10] Sarkar, D., Modak, J. M. (2004). Genetic algorithms with filters for optimal control problems in fed-batch bioreactors. *Bioprocess and Biosystems Engineering* 26(5), 295-306. https://doi.org/10.1007/s00449-004-0366-0

[11] Roubos, J. A., Van Straten, G., Van Boxtel, A. J. B. (1999). An evolutionary strategy for fed-batch bioreactor optimization; concepts and performance. *Journal of biotechnology* 67(2-3), 173-187. https://doi.org/10.1016/S0168-1656(98)00174-6