2022

Informe de
ASESORAMIENTO
y TRANSFERENCIA

047-22

Optimización metaheurística con algoritmos genéticos

Hernández Rodríguez, Matías Ezequiel

Resumen

Este informe trata sobre los algoritmos genéticos, un tipo de inteligencia artificial inspirada en la evolución biológica. En el mismo, se explican detalladamente su funcionamiento, sus principales componentes y operadores, lo cual, junto a su correspondiente pseudocódigo, proporciona al lector el conocimiento necesario para programar un algoritmo genético genérico en lenguajes de alto nivel como Fortran, Python, R o MatLab.
La eficacia y versatilidad de esos algoritmos queda ilustrada mediante los resultados obtenidos al aplicarlos en situaciones reales que involucran diversos problemas de optimización, tales como, entre otros, la estimación de los parámetros de la ecuación logística de Verhulst y de un modelo de competencia entre especies.

Ministerio de Agricultura, Ganadería y Pesca
Argentina

INIDEP
INSTITUTO NACIONAL DE INVESTIGACIÓN
Y DESARROLLO PESQUERO

Optimización metaheurística con algoritmos genéticos

Hernández Rodríguez, Matías Ezequiel

Instituto Nacional de Investigación y Desarrollo Pesquero - INIDEP

Resumen

Este informe trata sobre los algoritmos genéticos, un tipo de inteligencia artificial inspirada en la evolución biológica. En el mismo, se explican detalladamente su funcionamiento, sus principales componentes y operadores, lo cual, junto a su correspondiente pseudocódigo, proporciona al lector el conocimiento necesario para programar un algoritmo genético genérico en lenguajes de alto nivel como Fortran, Python, R o MatLab.

La eficacia y versatilidad de esos algoritmos queda ilustrada mediante los resultados obtenidos al aplicarlos en situaciones reales que involucran diversos problemas de optimización, tales como, entre otros, la estimación de los parámetros de la ecuación logística de Verhulst y de un modelo de competencia entre especies.

Palabras Clave

Optimización, Algoritmos genéticos, Metaheurística, Sistemas dinámicos.

Introducción

El presente informe trata sobre *algoritmos genéticos*, un tipo de *inteligencia artificial* que emula la evolución biológica. La exposición de la teoría en conexión con aplicaciones y problemas prácticos concretos tiene como finalidad proporcionar al lector las herramientas necesarias para implementar su propio algoritmo adaptado a las exigencias y necesidades de los problemas que forman parte de su labor como investigador. La preparación matemática usual de los cursos de cálculo, probabilidad y álgebra es más que suficiente para leerlo.

Para situar este informe en el contexto adecuado, justificando no sólo a quién va dirigido, sino también su necesidad, es preciso comenzar considerando un concepto clave en el desarrollo científico-tecnológico actual: la optimización.

La optimización está presente en todos los aspectos de la existencia, y se manifiesta a través de multitud de problemas que emergen en dominios tan diversos como la robótica, la medicina, la economía, la ingeniería y la salud pública. Incluso, aparece en los pormenores de la vida cotidiana, como cuando, por ejemplo, las personas planifican sus vacaciones al menor costo posible, o cuando organizan la rutina semanal procurando maximizar las horas de sueño sin llegar tarde al trabajo (Grass, et al. 2008; Lenhart y Workman 2007; Yang 2008).

En el ámbito netamente científico los problemas de optimización aparecen de manera natural ante la necesidad de controlar o modificar el comportamiento de un sistema dinámico para lograr objetivos deseados, los cuales típicamente consisten en maximizar un rendimiento o minimizar un costo. Por ejemplo, cada una de las siguientes cuestiones se pueden plantear con un problema de optimización: ¿cuánta biomasa de una especie animal se debe capturar para minimizar el riesgo de que la biomasa de esa especie caiga por debajo de un determinado umbral?, ¿cómo controlar las poblaciones de especies invasoras para que su impacto en el medio ambiente sea mínimo?, ¿cómo aplicar un fungicida durante un período de tiempo prestablecido si se pretende minimizar la cantidad de moho que afecta una especie, la cantidad de fungicida utilizada y costo del proceso?, ¿cuál es la superficie mínima con la que se puede construir un objeto geométrico que tenga un volumen predeterminado?

Los problemas de optimización también aparecen cuando se estiman los parámetros que forman parte de modelos matemáticos que describen procesos de la naturaleza (desde la dinámica entre diferentes especies de peces hasta el cálculo de la trayectoria de una misión espacial), cuando se minimizan (o maximizan) simultáneamente objetivos que están en conflicto (como, por ejemplo, el diseño de un tratamiento de quimioterapia que busque minimizar simultáneamente la cantidad de drogas utilizadas

y el número de células cancerígenas) y en el diseño de políticas de desarrollo sustentable (Barrea y Hernández 2016; Barrea, Hernández y Spies 2017; Barrea y Hernández 2012; De Lara y Doyen 2008; Hernández 2014; Van Regenmortel 2019; Vogel 2002).

Por ser la optimización tan importante en los procesos de tomas de decisiones y en el desarrollo de la ciencia actual, la cual está caracterizada por la interdisciplinariedad, resulta de capital importancia que el investigador científico-tecnológico, sea este del área que sea, esté familiarizado no sólo con los problemas de optimización, sino también con sus métodos de resolución. En ese sentido, la matemática proporciona una gran variedad de métodos deterministas, tales como, por nombrar algunos, el método Simplex, el método del gradiente, el método de multiplicadores de Lagrange y los métodos que se desprenden de las condiciones de Kuhn-Tucker, sustentados, todos ellos, por una teoría sólida; sin embargo, la teoría es una cosa y la práctica otra. En Muchas ocasiones, los algoritmos anteriores suelen presentar las siguientes dificultades fácticas: (i) son métodos de búsqueda local; (ii) requieren que las funciones a optimizar sean derivables, pero en muchas ocasiones no lo son o, en caso de serlo, el cálculo de sus derivadas es extremadamente dificultoso; (iii) los problemas reales suelen ser lo suficientemente complejos como para no poder demostrarse las condiciones necesarias y/o suficientes para la convergencia de los susodichos algoritmos

Para hacer frente a dificultades como las antes señaladas se desarrollan los denominados *métodos metaheurísticos*[1] de optimización, los cuales, a diferencia de los métodos matemáticos (algoritmos y métodos iterativos), se caracterizan por ser métodos de búsqueda global y estar signados por el azar. En muchas ocasiones, los métodos metaheurísticos son la única herramienta disponible para aproximar la solución de un problema de optimización, y en muchas otras son utilizados para encontrar un buen punto de partida para los métodos deterministas. En el campo de la metaheurística de la optimización, algunos de los métodos más notables son los algoritmos genéticos, la optimización por enjambre de partículas, la optimización por colonias de hormigas, el enfriamiento simulado, la búsqueda tabú y los algoritmos meméticos.

Este informe, como se indicó al comienzo, trata sobre los algoritmos genéticos. De aquí en adelante, está organizado de la siguiente manera: después de formalizar el concepto de problema de optimización, se explica detalladamente cada una de las componentes y operadores de un algoritmo genético estándar. Luego, en un movimiento que va de lo general a lo particular, se abordan diferentes aplicaciones prácticas provenientes de situaciones reales, lo cual pone de manifiesto que un algoritmo genético es lo suficientemente versátil como para adaptarse a diferentes tipos de problemas. Esas aplicaciones son: *función de Rosenbrock*, *ecuación logística de Verhulst*, *ecuaciones de Lotka-Volterra* y la *optimización multiobjetivo*.

Finalmente, el informe culmina con una breve conclusión.

Formulación matemática del problema de optimización

Existen diferentes maneras de formalizar la idea de optimización, lo cual da lugar a diversos subcampos de estudio, tales como, entre otros, la programación lineal, la programación no lineal, la programación dinámica, la teoría de control óptimo y el cálculo de variaciones. Este informe está enfocado en el problema de minimizar o maximizar un funcional sujeto a restricciones, ya que, como veremos en las aplicaciones prácticas, muchas de las ideas que de ahí surgen son fértiles a la hora de abordar otros problemas de optimización más complejos. En términos matemáticos, y del modo más genérico posible, el problema de optimizar un funcional se formula así:

$$\min_{x \in \mathbb{R}^n} f(x), \qquad (1)$$

[1] ***NOTA sobre la palabra methaheurística***
La palabra *metaheurística* combina el prefijo griego *meta*, que significa más allá, y *heurística*, que significa *encontrar*.

sujeto a $h_j(x) = 0\ (j = 1, \dots, M)$ y $g_k(x) < 0\ (k = 1, \dots, N)$, donde $f, h_j\ y\ g_k$ son funciones escalares[2]. Las componentes, $x_i(i = 1, \dots, n)$, de x se denominan *variables de decisión*, y pueden ser números enteros, reales o mixtos. La función f se conoce como *función objetivo* o *función de costo*, mientras que las funciones h_j y g_k se denominan *restricciones de igualdad* y *desigualdad*, respectivamente. Además, el subconjunto, **Ω**, de $\mathbb{R}^n$ determinado por las restricciones se denomina *espacio de búsqueda*; el conjunto $f(\Omega)$, *espacio de soluciones*; el elemento $x^* \in \Omega$ que minimiza (maximiza) a f, *mínimo* (*máximo*) u *óptimo*.

Un ejemplo simple, aunque muy concreto, de problema de optimización es el de minimizar la función $f(x, y) = x^2 + y^2$, sujeta a que $(x, y) \in [-10{,}10] \times [-10{,}10]$, es decir, encontrar el mínimo de f sobre un rectángulo. Utilizando las herramientas del cálculo elemental, es fácil probar que f tiene un mínimo global en (0,0).

Algoritmos genéticos

El dominio de la optimización se extiende más allá de la esfera netamente humana, ya que los seres vivientes parecen evolucionar en el tiempo siguiendo estrategias que maximicen su capacidad de supervivencia. Pensar en el proceso de evolución como el gran algoritmo de la vida que selecciona, luego de muchas iteraciones, qué características producen una especie de organismos aptos para la supervivencia, inspiró a John Henrry Holland, en la década del 70, a desarrollar una metaheurística para la optimización conocida como algoritmos genéticos (Haupt y Haupt 2004).

Los algoritmos genéticos presentan variaciones dependiendo de su naturaleza[3] y del problema para el cual han sido diseñados; sin embargo, sus componentes y operadores son esencialmente los mismos –aunque, claro está, existen diferentes criterios para diseñarlos–. A continuación, se discuten los principales elementos de un algoritmo genético; a saber, la *inicialización*, la *decodificación*, la *evaluación*, la *selección*, el *entrecruzamiento cromosómico* y la *mutación* (Goldberg y Holland 1989; Haupt y Haupt 2004; Michalewicz 2000).

Inicialización

Una vez que se han definido la función de costo, los parámetros y las variables del algoritmo, se genera una población inicial constituida por individuos denominados *cromosomas*, los cuales son puntos elegidos normalmente al azar en el espacio de búsqueda. La población inicial se representa mediante una matriz que tendrá tantas filas como cromosomas; pero el número de columnas dependerá del número, N_{var}, de variables de la función de costo y de la naturaleza del algoritmo.

[2] ***NOTA sobre el problema de maximizar un funcional***
Es evidente que el problema de maximizar un funcional, f, es equivalente al de minimizar su negativo, es decir, $-f$; por lo tanto, todos los métodos que se desarrollen para resolver el problema (1) servirán también para maximizar un funcional sujeto a restricciones.

[3] ***NOTA sobre la clasificación de los algoritmos genéticos***
Según cómo se representen sus cromosomas, los algoritmos genéticos se clasifican en continuos o binarios

Un algoritmo es *binario* cuando los cromosomas son representados por ceros y unos. Como cada cromosoma es un punto del espacio de búsqueda, el cual es un subconjunto de $\mathbb{R}^{var}$, los ceros y unos que lo representan poseen toda la información necesaria para decodificar, utilizando el sistema de representación binario y una transformación adecuada, el valor de las variables del cromosoma. Para realizar la decodificación se reserva un conjunto, sea N_{bits} su cardinalidad, de ceros y unos para representar cada variable. A su vez, de esos ceros y unos, una proporción está destinada a la representación de la parte decimal y la otra a la parte entera. El conjunto de ceros y unos utilizados para representar cada variable se conoce como *gen*, cada cromosoma contiene N_{var} genes.

En términos de lo que acabamos de explicar, la población inicial de un algoritmo binario es una matriz
de tamaño $N_{pop}N_{bits}N_{var}$, cuyos elementos son ceros y unos elegidos al azar. Un ejemplo de población inicial de un algoritmo genético binario es el siguiente:

$$P = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 \end{pmatrix} \quad (2)$$

En este caso $N_{pop} = 4$ porque P tiene 4 filas. Si fuese $N_{bits} = 9$; entonces, como el número de columnas es 18, $N_{var} = 2$. Además, si de los 9 bits utilizados para representar cada variable, 6 son utilizados para la parte entera y los 3 restantes para la decimal; entonces, haciendo uso del sistema binario se puede calcular los elementos de la población en notación decimal, los cuales del primero al último son: (53,6250, 12,1250), (19,1250, 58,8750), (42,5000, 33,3750) y (39,3750, 7,2500). Luego, dependiendo del espacio de búsqueda, a esos puntos se les aplica una transformación para que estén dentro del susodicho espacio (ese aspecto quedará más claro cuando se explique el proceso de decodificación).

En los algoritmos continuos los cromosomas se representan utilizando el sistema decimal; por lo tanto, la población inicial será una matriz de números reales de tamaño $N_{pop}N_{var}$. Por ejemplo, de la siguiente matriz:

$$\begin{pmatrix} 1{,}2340 & -5{,}1265 & 9{,}4294 & 0{,}2934 \\ 28{,}1297 & 0{,}0000 & -11{,}6584 & 1{,}2253 \\ 3{,}2718 & 31{,}6541 & 1{,}4198 & 66{,}0714 \end{pmatrix}$$

se deduce que $N_{var} = 4$ y $N_{pop} = 3$, es decir, que la población tiene 3 cromosomas, los cuales son los siguientes puntos de $\mathbb{R}^4$: (1,2340, -5,1265, 9,4294, 0,2934), (28,1297, 0,0000, -11,6584, 1,2253) y (3,2718, 31,6541, 1,4198, 66,0714).

Una vez definida la población inicial comienza a ejecutarse la parte del algoritmo que simula el proceso de evolución propiamente dicho. En cada iteración se seleccionan los cromosomas más aptos, es decir, aquellos que están más cerca de optimizar la función de costo, se los cruza, se reemplazan los cromosomas menos dotados (por la descendencia de los cromosomas seleccionados) y se producen las mutaciones. El algoritmo se ejecutará a lo sumo un número máximo de iteraciones, N_{it}.

Decodificación

La *decodificación* sólo tiene lugar en los algoritmos binarios, pues para poder evaluar la función de costo en cada cromosoma es necesario expresarlos en el sistema decimal y luego transformarlos en puntos del espacio de búsqueda. Por ejemplo, si se desea minimizar la función $f(x,y) = x^2 + y^2$, sujeta a que $(x,y) \in [-10,10] \times [-10,10]$, utilizando un algoritmo genético binario con los parámetros $N_{pop} = 4$, $N_{bits} = 9$ y $N_{var} = 2$, y se dispone de la población dada por la matriz (2), ya como población inicial, ya como población de alguna instancia del algoritmo, el proceso de decodificación comienza expresando los cromosomas en sistema decimal. Como se ya se señaló, los cromosomas en el sistema decimal son (53,6250, 12,1250), (19,1250, 58,8750), (42,5000, 33,3750) y (39,3750, 7,2500), los cuales, evidentemente no están en el espacio de búsqueda $[-10,10] \times [-10,10]$. Entonces, debe aplicarse una trasformación, $T(x)$, que lleve esos cromosomas al espacio de búsqueda. Una forma de hacer es con la transformación lineal[4] $T(x) = -10 + \frac{20}{63.7500}x$, mediante la cual los cromosomas de la población quedan decodificados, respectivamente, en los puntos (6,8235, -6,1961), (-4,000, 8,4706), (3,3333, 0,4706) y (2,3529, -7,7255).

Evaluación

El proceso de evaluación consiste en evaluar la función de costo en cada cromosoma. Si el algoritmo es continuo, simplemente se calcula el valor de la función de costo en cada uno de los cromosomas, es decir, en cada uno de los puntos elegidos al azar en el espacio de búsqueda; si el algoritmo es binario, se evalúa la función de costo en los puntos del espacio de búsqueda decodificados a partir de los cromosomas representados por ceros y unos. Continuando con el ejemplo anterior, la evaluación de $f(x,y) = x^2 + y^2$ en los puntos (6,8235, -6,1961), (-4,000, 8,4706), (3,3333, 0,4706) y (2,3529, -7,7255) es, respectivamente, 84,9543, 87,7511, 11,3323, y 65,2190, lo cual significa que la función de costo toma su menor valor en el tercer cromosoma (el más apto) de la matriz (2), su segundo menor valor en el cuarto cromosoma (segundo mejor apto), su tercer menor valor en el segundo cromosoma (el tercero más apto) y cuarto menor valor en el primer cromosoma (el menos apto).

Una vez que se ha realizado la decodificación (en el caso de los algoritmos binarios) y la evaluación del costo, se reordenan las filas de la matriz que contiene los cromosomas de menor a mayor según el costo de cada cromosoma. Por ejemplo, la matriz (2) quedaría como:

$$P = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \quad (3)$$

Selección

La selección consiste en descartar los cromosomas menos aptos seleccionando una fracción x_{rate} de los N_{pop} cromosomas de la población. Determinar qué proporción de cromosomas seleccionar es una cuestión bastante arbitraria. Si se selecciona una pequeña cantidad, es decir, si x_{rate} es muy pequeño; entonces, se limita la existencia de buenos genes en la próxima generación. Por otro lado, si x_{rate} es

[4] ***NOTA sobre la transformación T(x)***

La transformación que lleva los cromosomas expresados en sistema decimal hacia el espacio de búsqueda depende de la geometría de este. En el caso de que cada componente, x_i, de un cromosoma deba satisfacer $x_i \in [a_i, b_i]$, suele utilizarse la transformación lineal $T(z) = a_i + \frac{(b_i - a_i)}{M}z$, donde M es el máximo valor que se puede representar con N_{bits}.

muy grande, suele obtenerse malos resultados. Lo usual es considerar $x_{rate} = 0.5$. Sea cual sea el valor que se le asigne a x_{rate}, la cantidad de individuos que se van a seleccionar puede calcularse así: $N_{keep} = ceil(x_{rate})N_{pop}$, donde $ceil(x) = min\{k \in \mathbb{Z} : x \leq k\}$. Si en el ejemplo que se viene considerando, $x_{rate} = 0{,}5$; entonces, $N_{keep} = ceil(0{,}5) \times 4 = 2$, lo cual significa que se seleccionan la primera y segunda fila de la matriz (3), mientras que los cromosomas de la tercera y cuarta fila son eliminados. Entonces, la población quedaría reducida a:

$$P = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 \end{pmatrix} \quad (4)$$

Entrecruzamiento y reemplazo

Después de la selección, viene el proceso de entrecruzamiento y reemplazo. Los N_{keep} cromosomas seleccionados deben ser apareados para generar una descendencia que complete los $N_{pop} - N_{keep}$ cromosomas que han sido descartados. La manera en que se aparean dos cromosomas depende de la naturaleza del algoritmo, es decir, de si es binario o continuo; pero en ambos casos de dos cromosomas resultan dos descendencias. Si $N_{pop} - N_{keep}$ es un número par; entonces, por cada apareamiento podríamos considerar los dos descendientes e incorporarlos en la población de cromosomas hasta que esta llegue a poseer nuevamente N_{pop} cromosomas. Sin embargo, no siempre $N_{pop} - N_{keep}$ es un número par, con lo cual una opción es considerar una de las dos descendencias al azar e incorporarla en la población de cromosomas hasta que esta llegue a los N_{pop} individuos.

En el contexto de los algoritmos binarios el entrecruzamiento cromosómico de dos cromosomas, C_1 y C_2, consiste en elegir una posición, s, al azar (algunos autores consideran siempre la misma posición) entre la primera y la última de los ceros y unos de los cromosomas, y luego se construyen las dos descendencias, C_3 y C_4, de la siguiente manera: la primera parte de C_3 se forma con los s primeros elementos de C_1 y el resto, es decir desde la posición $s + 1$ en adelante, con los elementos de C_2 que se encuentran desde la posición $s + 1$ en adelante. De modo análogo se construye C_4. Por ejemplo, si $s = 14$, el entrecruzamiento de los cromosomas de la matriz (4) se realiza de la siguiente manera: $C_1 = [\underbrace{10101010010000}_{Parte\ destina\ a\ C_3}\ \underbrace{1011}_{Parte\ destinada\ a\ C_4}]$, $C_2 = [\underbrace{10011101100011}_{Parte\ destina\ a\ C_4}\ \underbrace{1010}_{Parte\ destinada\ a\ 3}]$, con lo cual $C_3 = [10101010010000 1010]$ y $C_4 = [10011101100011 1011]$. Ahora, el reemplazo podría consistir en incorporar directamente C_3 y C_4 en (4) para obtener la siguiente nueva población:

$$P = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 \end{pmatrix} \quad (5)$$

El entrecruzamiento de dos cromosomas, C_1 y C_2, de un algoritmo continuo puede realizarse de la siguiente manera: C_3=$C_1 + \beta(C_2 - C_1)$ y C_4=$C_2 + \beta(C_1 - C_2)$, donde β es un número aleatorio uniformemente distribuido en $[0,1]$ (Haupt y Haupt 2004).

Pero resulta que, en analogía con lo que ocurre en la naturaleza, de los N_{keep} cromosomas seleccionados no todos necesariamente se van a aparear, el azar siempre juega su parte en estos algoritmos. Existen diferentes maneras de seleccionar qué cromosomas se aparean, la mayoría de las cuales le dan más probabilidad de apareamiento a los cromosomas más aptos, incluso, podría ocurrir que un cromosoma se aparee más de una vez. A continuación, se explica algunos de los métodos más utilizados para seleccionar los cromosomas que van a aparearse.

- **Apareamiento de arriba hacia abajo.** Consiste en aparear el primer cromosoma con el segundo, y así sucesivamente. Evidentemente esta metodología, la más simple de todas, funciona si N_{keep} es un número par, de lo contrario debe añadirse o quitarse un cromosoma de los seleccionados.
- **Apareamiento aleatorio.** Cada vez que se realiza un apareamiento, se eligen dos cromosomas al azar de los N_{keep} que sobrevivieron. Este método, como los restantes, da la posibilidad de que un cromosoma pueda aparearse más de una vez, incluso, de que pueda aparearse consigo mismo.
- **Apareamiento ponderado por filas.** Al cromosoma de la fila n dentro del grupo de los N_{keep} seleccionados, se le asigna la probabilidad:

$$P_n = \frac{N_{keep} - n + 1}{\sum_{j=1}^{N_{keep}} j} \quad (6)$$

  El numerador de (6) nos da la posición (distancia) del cromosoma n con respecto al cromosoma más apto. La ventaja de este método es que las probabilidades se calculan una sola vez, antes de que comiencen a ejecutarse las iteraciones del algoritmo genético, mientras que su principal desventaja es que la probabilidad de elegir siempre el mismo cromosoma para aparearse es alta en poblaciones pequeñas.
- **Apareamiento ponderado por costos.** Esta metodología consiste en calcular las probabilidades de apareamiento de los cromosomas ponderando la función de costo. A cada uno de los cromosomas seleccionados se le asigna una probabilidad normalizada de apareamiento, la cual se calcula dividiendo la diferencia entre el costo del cromosoma y el costo del más apto de los cromosomas descartados por la suma de esas diferencias calculadas en todos los cromosomas seleccionados, es decir:

$$P_n = \left| \frac{C_n}{\sum_{j=1}^{N_{keep}} C_j} \right|,$$

para $n = 1, \ldots, N_{keep}$, donde C_n es la diferencia entre el costo del cromosoma n, c_n, y el costo del mejor de los peores cromosomas, $c_{N_{keep}}$, es decir, $C_n = c_n - c_{N_{keep}}$.

La principal ventaja de este método es que tiende a ponderar el cromosoma más apto cuando existe una gran dispersión entre los costos, y tiende a ponderar los cromosomas uniformemente cuando estos tienen costos similares. La desventaja de este método es que en cada iteración deben calcularse las probabilidades P_n.

- **Selección por torneo.** Este enfoque consiste en elegir al azar un pequeño grupo (dos o tres) de cromosomas de la población seleccionada, y de ese pequeño grupo se elige el individuo más apto para aparearse. El torneo se repite hasta alcanzar una cantidad preestablecida de individuos para aparearse. Lo interesante de este método, que suele funcionar mejor en poblaciones grandes, es que no requiere que se ordenen los cromosomas.

Mutación

El proceso de mutación, el más simple de programar, es de capital importancia, ya que sin mutaciones el algoritmo podría converger rápidamente sin haber explorado todo el espacio de búsqueda. Cada uno de los elementos de la matriz de cromosomas tiene la misma probabilidad de mutar, y viene determinada por una tasa de mutación, μ, que suele definirse entre 0.01 (muta el %1 de los elementos de la matriz de cromosomas) y 0.2 (muta el %20 de los elementos de la matriz de cromosomas).

Ventajas y desventajas de los algoritmos genéticos

Los algoritmos genéticos tienen sus ventajas y desventajas. Algunos de los principales puntos a favor son siguientes:

- Se aplican tanto a problemas con variables discretas como a problemas con variables continuas.
- Se aplican con éxitos a problemas de optimización con funcionales muy complejos.
- Son efectivos en problemas con un elevado número de variables.
- Proveen de un conjunto de posibles soluciones.
- Buscan soluciones globales.

Entre las principales contras podemos citar:

- Pueden demandar mucho tiempo de cómputo.
- Pueden tardar mucho en converger, hacerlo de manera prematura o directamente no hacerlo.

Antes de escribir un programa, como puede ser el caso un algoritmo genético, en el lenguaje de programación que ha sido elegido, es útil escribir primero su pseudocódigo, ya que este proporciona una mirada integral del algoritmo.

Usualmente los pseudocódigos omiten todos los detalles que no son esenciales para la comprensión humana del algoritmo; describen informalmente todos los pasos a seguir por el programa de un modo sencillo, general y lo más cercana posible al lenguaje que se utilizará.

A continuación, se presenta el pseudocódigo de un algoritmo genético genérico.

Pseudocódigo de un algoritmo genético

Entrada: La función de costo, f, los parámetros y las variables del algoritmo.

$t \leftarrow 0$.
Inicialización de la población $P(t)$.
Mientras (no se verifique alguna de las condiciones de corte) **hacer**
Decodificación de $P(t)$ (si el algoritmo es binario).
Evaluación de $P(t)$.
Selección de N_{keep} cromosomas de $P(t)$.
Cruce de los individuos seleccionados.
Reemplazo de los individuos menos aptos por la descendencia.
Mutación.
$t \leftarrow t + 1$
Chequeo de la condición de corte.
Fin
Salida $P(t)$.

Materiales y métodos

Para discutir la eficacia de los algoritmos genéticos en la resolución de problemas de optimización se han programado cuatro algoritmos genéticos: (i) un algoritmo genético binario que pondera por filas, AGBPF; (ii) un algoritmo genético binario que pondera por costos, AGBPC; (iii) un algoritmo genético continuo que pondera por filas, AGCPF; y un algoritmo genético continuo que pondera por costos, AGCPC.
Los algoritmos han sido programados en Fortran siguiendo las instrucciones explicadas más arriba; pero pueden programarse en cualquier otro lenguaje de alto nivel, tales como R, Octave o Python, por mencionar algunos.
Luego, se abordaron aplicaciones prácticas extraídas de la literatura científica, tal como se especifica señalando las correspondientes citas bibliográficas. La eficacia del o los algoritmos utilizados se cuantifica con el error relativo y se ilustra mediante las gráficas pertinentes.

Resultados

A continuación, se presentan los resultados obtenidos después de abordar aplicaciones reales.
La función de Rosenbrock
La *función de Rosenbrock*, definida como $f(x,y) = (1-x)^2 + 100(y-x^2)^2$, tiene un mínimo global en el $(x^*,y^*) = (1,1)$, algo muy fácil de probar mediante las herramientas del cálculo elemental.
Sin embargo, como se puede ver en la Figura 1 (a), el mínimo se encuentra sobre una región plana en forma de parábola, lo cual dificulta la convergencia de los métodos determinista.
Se consideró el problema de optimizar la función de Rosenbrock sobre el dominio $[-10,10] \times [-10,10]$ utilizando el algoritmo AGBPF con los parámetros que se muestran en la

Tabla 1, y se obtuvo la aproximación $(1{,}0079, 1{,}0158)$, con un error relativo igual a 0,012491.

Tabla 1. Parámetros del algoritmo

N_{it}	N_{pop}	N_{bits}	s	μ	x_{rate}
20	180	16	8	0.02	0.5

Comparar el rendimiento de los diferentes tipos de algoritmos genéticos está más allá del objetivo de este informe; sin embargo, dado que la función Rosenbrock es una función test que se utiliza para probar el rendimiento de los algoritmos de optimización, ha resultado interesante observar las

aproximaciones obtenidas con los demás algoritmos. Usando los mismos parámetros (para los algoritmos continuos, evidentemente, N_{bits} y s no son necesarios), los algoritmos AGBPC, AGCPF y

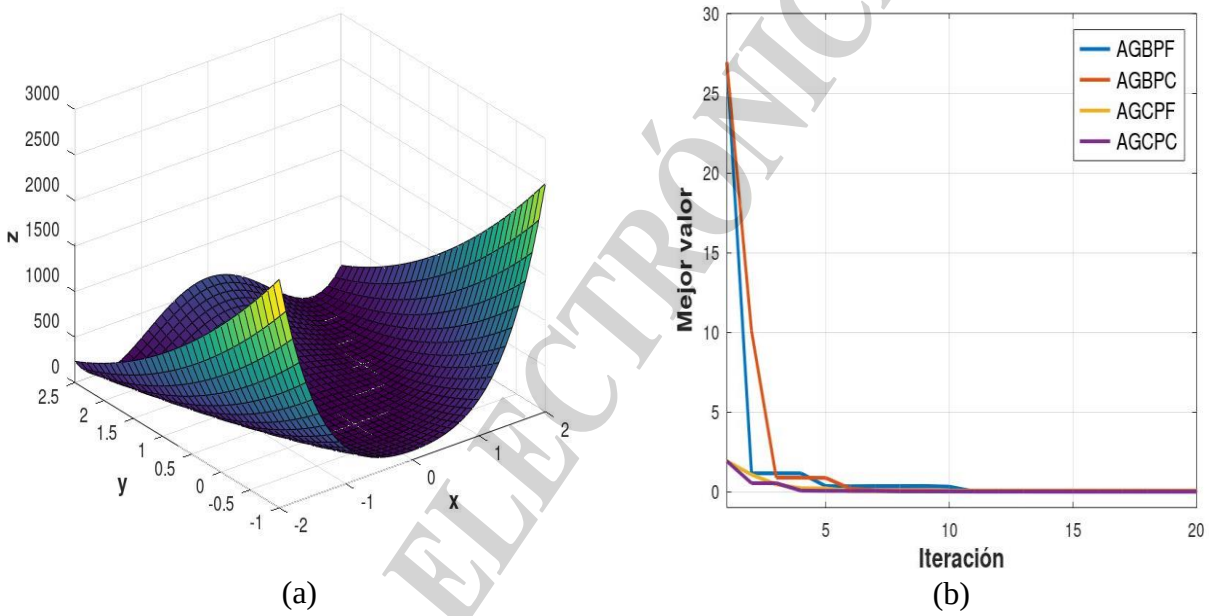
AGCPC proporcionaron, las siguientes aproximaciones (1,2502, 1,5630), (1,1594, 1,3437) y (0,9999, 0,9999), con errores relativos iguales a 0,4356, 0,2679 y 0,0001, respectivamente. La Figura 1 (b) muestra la evolución de los cuatro algoritmos.

Los algoritmos genéticos también se utilizan en el ámbito de los problemas inversos, es decir, en la estimación de los parámetros que forman parte de un modelo a partir de los datos observados en la realidad. Se presentan a continuación algunas aplicaciones en ese ámbito.

Ecuación logística de Verhulst

En (Spier, et al. 2009) se investigó el crecimiento de hongos durante la fermentación de pulpa cítrica para la producción de fitasa. La biomasa se estimó mediante la extracción de ergosterol basada en el método de Seitz con algunas modificaciones, y se mide en g por g de sustrato seco [g/gds]. Los

datos experimentales para el crecimiento de la biomasa se muestran en la Tabla 2.



(a) (b)

Figura 1. (a) Función de Rosenbrock. (b) Evolución de los algoritmos.

Tabla 2. Biomasa en función del tiempo

Tiempo	Biomasa $[\times 10^{-2} g/gds]$
0	10,2700
24	24,9500
72	58,9300
96	81,5200
120	92,1600

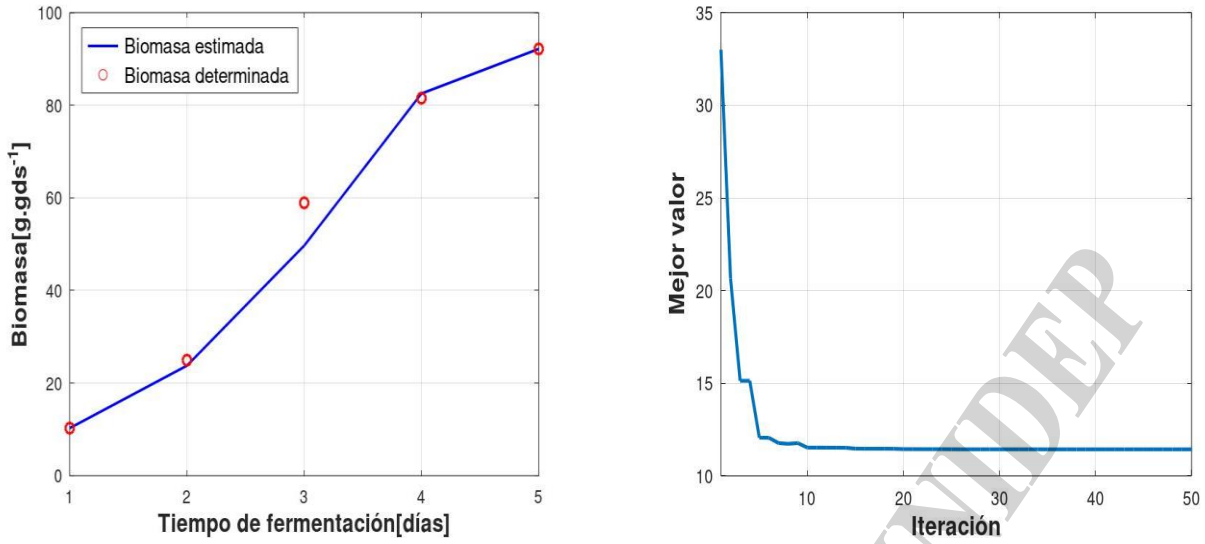
Los datos parecen indicar que la biomasa sigue un crecimiento logístico discreto, el cual en términos matemáticos viene dado por la ecuación:

$$B(t+1) = B(t) + rB(t)\left(1 - \frac{B(t)}{K}\right),$$

donde $B(t)$ es la biomasa del hongo en el tiempo t (con una condición inicial conocida $B(0) = B_0$); $r > 0$, la tasa de crecimiento per cápita; K, la capacidad de carga del hábitat. La ecuación anterior se conoce como *ecuación logística de Verhulst*, o simplemente ecuación logística.
Bajo la hipótesis de que el hongo sigue un crecimiento logístico se utilizó el algoritmo AGBPF para estimar los parámetros r y K. Para medir cuán bien se ajusta los datos la dinámica determinada por un par de parámetros (r, K), se eligió la función de costo $f(r,K) = \sum_{i \in A} |B(i) - N(i)|$, donde $N(i)$ es la biomasa del hongo en el tiempo i; $B(i)$, el valor de la biomasa en el tiempo i según la ecuación logística con los parámetros r y K, y la condición inicial $B(0) = 10.2700$; $A = \{0, 24, 72, 96, 120\}$.

Aquí se pudo observar una de las ventajas de los algoritmos genéticos: pueden buscar mínimos globales de funciones que, como $f(r,K)$, no son diferenciales.

Una vez definida la función de costo, se definió el espacio de búsqueda. En ese hubiese sido de capital importancia la opinión de un especialista que pueda indicar en qué rango debe buscarse el valor (r^*, K^*) que minimiza la función de costo. No obstante, a partir de los datos, graficados mediante círculos rojos en la Figura 2 (a), se asumió que $K \in [0{,}150]$. Respecto a r se consideró que se encuentra en [0,1] (Spier, y otros 2009). El algoritmo genético binario determinado por los parámetros de la Tabla 1 arrojó la aproximación $r = 0{,}5667$ y $K = 96{,}0952$, lo cual arroja un error relativo igual a 0,0299.



(a) (b)

Figura 2. (a) Datos vs modelo. (b) Evolución del algoritmo AGBPF

Ecuaciones de Lotka-Volterra

Las *ecuaciones de Lotka-Volterra*, introducidas de forma independiente por Alfred J. Lotka en 1925 y Vito Volterra en 1926, describen la dinámica de dos poblaciones que interactúan mediante la relación cazador-presa. Un entendimiento profundo de las mismas se puede encontrar en la bibliografía proporcionada al final de la obra (Britton 2003, Hadeler, Mackey y Stevens 2017). Para nuestro propósito basta con recordar que si representamos por $x(t)$ e $y(t)$ a las poblaciones de presas y cazadores respectivamente; entonces, el modelo de Lotka-Volterra establece que:

$$\begin{cases} \dot{x}(t) = a_1 x(t) - a_2 x(t) y(t) \\ \dot{y}(t) = -b_1 y(t) + b_2 x(t) y(t) \\ x(0) = x_0, y(0) = y_0 \end{cases} \qquad (7)$$

En la ecuación (7), a_1, a_2, b_1 y b_2 son los parámetros del modelo.

La compañía canadiense *Hudson Bay* registró las capturas de conejos y linces en el período 1900 - 1921 (Mahay 2018). Se supone que esas capturas fueron una muestra representativa del tamaño real de las poblaciones de esas especies, las cuales parecen seguir la dinámica del modelo (7). Bajo ese supuesto, aplicó el algoritmo AGBPF para aproximar los parámetros del modelo y se obtuvo el siguiente resultado: $a_1 = 0{,}3984$, $a_2 = 0{,}0210$, $b_1 = 0{,}9999$ y $b_2 = 0{,}0031$, con un error relativo igual a 0,1450.

Optimización multiobjetivo

Muchos problemas prácticos requieren minimizar (o maximizar) simultáneamente varias funciones, denominadas *objetivos*, que se encuentran en conflicto, lo cual significa que la reducción en el valor de un objetivo implica el aumento de otros. Un ejemplo es el problema de encontrar un tratamiento de quimioterapia buscando al mismo tiempo minimizar el número de células cancerígenas y la cantidad de drogas utilizadas (Hernández 2014).

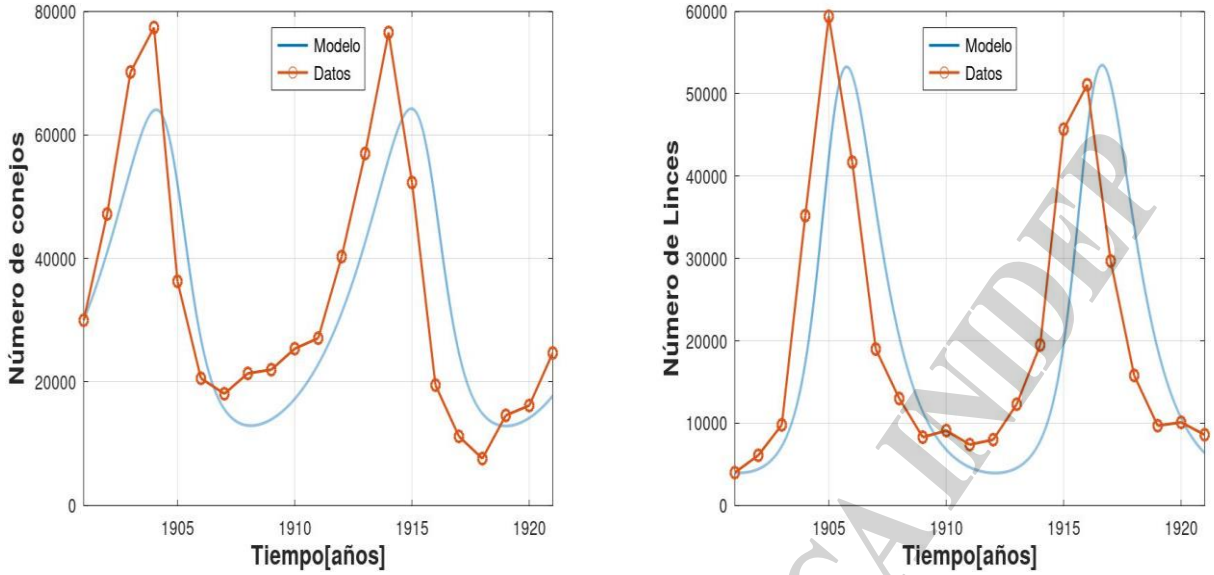


Figura 3. Datos vs modelo

En términos matemáticos, el problema de optimización multiobjetivo se puede expresar así:

$$\min_{x \in U \subset \mathbb{R}^m} (f_1(x), \ldots, f_n(x))$$

En general el problema anterior no tiene una única solución que minimice a todos los objetivos a la vez; pero existe un conjunto de puntos igualmente eficientes conocido como *frente de Pareto*, P. Un elemento, x^*, de P se caracteriza por el hecho de no poder reducir ninguno de los objetivos sin aumentar al menos otro. En términos matemáticos, $x^* \in P$ si y sólo si $\nexists x \in U$ tal que $f_i(x) \leq f_i(x^*), \forall\, i = 1,\ldots,n$; y $f_j(x) < f_j(x^*)$ en al menos una j.

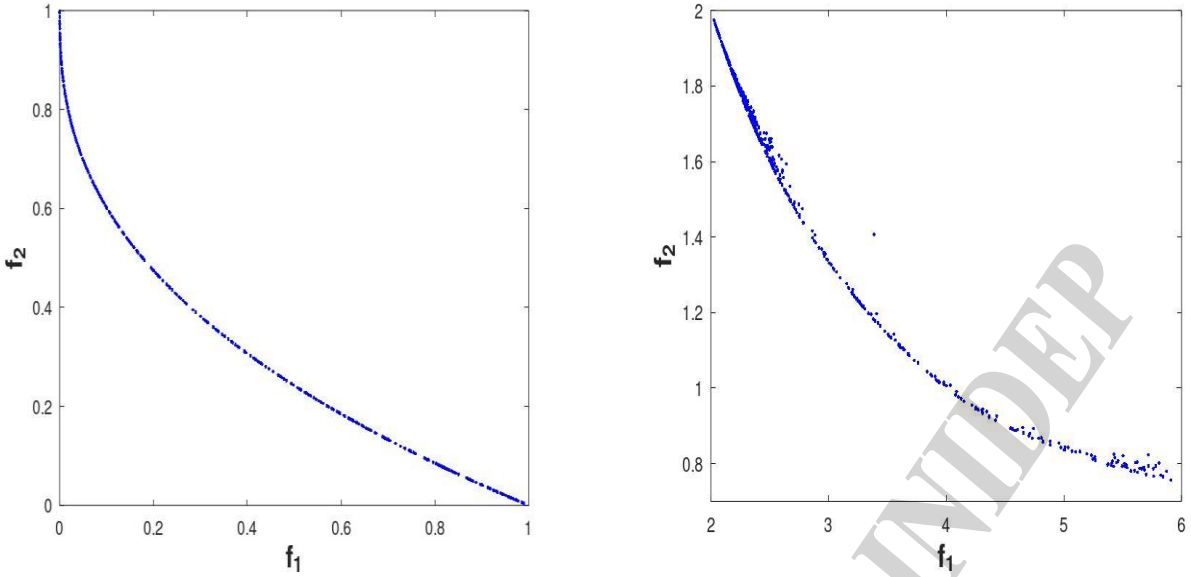
Existen diferentes métodos metaheurísticos diseñados especialmente para aproximar el frente de Pareto de un problema multiobjetivo; pero cualquiera de los algoritmos que hemos comentados se pueden adaptar para hacer esa tarea a partir del método determinista conocido como *método de sumas ponderadas* (Liu, Yang y Whidborne 2007).
Las Figura 4 (a) y (b) muestran, respectivamente, los frentes de Pareto aproximados por el algoritmo AGBPF de los problemas de los siguientes problemas de optimización multiobjetivo:

$$\min_{x \in [0,1]} (x^5, (1-x)^2)$$

y

$$\min_{(x,y) \in [1,2] \times [1,2]} \left(x + y^2, \frac{1}{x} + \frac{1}{y^2}\right)$$

respectivamente.



(a) (b)

Figura 4. Frentes de Pareto aproximados

Conclusiones

En este informe de asesoramiento se proporcionó la información necesaria para que el lector del mismo comprenda el funcionamiento de un algoritmo genético y esté en condiciones de programarlo en el lenguaje que utilice habitualmente en el desempeño de su profesión.

Los resultados obtenidos ponen de relieve la eficiencia de los algoritmos genéticos en la resolución de problemas de optimización muy variados, los cuales van desde minimizar un funcional con restricciones hasta la optimización multiobjetivo, pasando por la estimación de parámetros. Esa versatilidad justifica la necesidad de que el científico que participa en la investigación interdisciplinaria vinculada a las tomas de decisiones y el desarrollo de políticas sustentables tenga conocimiento sobre esa metaheurística.

Para terminar, es importante aclarar que este informe lejos está de agotar el tema que aborda, y se recomienda al lector interesado en diseñar y aplicar algoritmos genéticos hechos a la medida de los problemas que trata en sus quehaceres científicos, profundizar en el tema.

Bibliografía

Barrea A, Hernández M and Spies R. 2017. Optimal chemotherapy schedules from tumor entropy. Computational and Applied Mathematics (Springer).

Barrea A and Hernández M. 2016. Optimal control of a delayed breast cancer stem cells nonlinear model. Optimal Control Applications and Methods.

Barrea A and Hernández M. 2012. Pareto front for chemotherapy schedules. Applied Mathematical Sciences.

Britton NF. 2003. Essential mathematical biology. London: Springer.

De Lara M and Doyen L. 2008. Sustainable management of natural resources. Springer Science & Business Media.

Eichfelder G. 2009. An adaptive scalarization method in multiobjective optimization. SIAM Journal on Optimization.

Goldberg DE and Holland JH. 1989. Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley Publishing Company.

Grass D,Caulkins JP, Feichtinger G, Tragler G and Behrens DA. 2008. Optimal control of nonlinear processes. Springer.

Hadeler KP Mackey MC and Stevens A. 2017. Topics in mathematical biology. Springer.

Haupt RL and Haupt SE. 2004. Practical genetic algorithms. John Wiley & Sons.

Hernández ME. 2014. Optimización y sustentabilidad de protocolos de quimioterapia. Córdoba: Universidad Nacional de Córdoba.

Lenhart S and Workman JT. 2007. Optimal control applied to biological models. Chapman and Hall/CRC.

Liu GP, Yang JB and Whidborne JF. 2007. Multiobjective Optimisation and Control. Research Studies Press Ltd.

Mahay JM. 2018. Continuous Models Lotka-Volterra. Department of Mathematics and Statistics Dynamical Systems Group Computational Sciences Research.

Michalewicz Z. 2000. Genetic Algorithms+ Data Structures= Evolution programs. Springer.

Spier MR, Letti LAJ, Woiciechowski AL and Soccol CR. 2009. A simplified model for A. niger FS3 growth during phytase formation in solid state fermentation. Brazilian Archives of Biology and Technology.

Van Regenmortel MH. 2019. Development of a preventive HIV vaccine requires solving inverse problems which is unattainable by rational vaccine design. In HIV/AIDS: Immunochemistry, Reductionism and Vaccine Design. Springer.

Vogel CR. 2002. Computational methods for inverse problems. Society for Industrial and Applied Mathematics.

Yang X. 2008. Introduction to mathematical optimization. From linear programming to metaheuristics. Ca,bridge International Science Publishing.