ISSN 1870-4069

Comparativa de operadores de reemplazo en algoritmos genéticos aplicado a problemas de optimización

Carlos Alberto Rolon-Gonzalez[1], Yenny Villuendas-Rey[2]

[1] Instituto Politécnico Nacional,
Centro en Investigación en Computación,
México

[2] Instituto Politécnico Nacional,
Centro de Innovación y Desarrollo Tecnológico en Cómputo,
México

carlosagrolon@gmail.com, yvilluendasr@ipn.mx

Resumen. Los algoritmos genéticos han demostrado ser metodologías viables para resolver problemas de optimización, búsqueda y aprendizaje en las máquinas. El algoritmo genético clásico da lugar a una gran variedad de modelos, tomando en consideración los operadores genéticos que se implementen. Uno de los operadores genéticos con mayor impacto entre iteraciones es el operador de reemplazo. En este trabajo se presenta una comparativa y análisis de los resultados obtenidos por distintos operadores de reemplazo del estado del arte aplicado a funciones de minimización.

Palabras clave: Algoritmos genéticos, operadores de reemplazo, algoritmos evolutivos, optimización de funciones.

Comparison of Replacement Operators in Genetic Algorithms Applied to Optimization Problems

Abstract. Genetic algorithms have proven to be viable methodologies for solving optimization, search, and machine learning problems. The classical genetic algorithm gives rise to a wide variety of models, taking into account the genetic operators implemented. One of the genetic operators with the greatest impact between iterations is the replacement operator. This paper presents a comparison and analysis of the results obtained by different state-of-the-art replacement operators applied to minimization functions.

Keywords: Genetic algorithms, replacement operators, evolutionary algorithms, function optimization.

1. Introducción

En los últimos años existe un creciente interés por el análisis de experimentos en el ámbito de los algoritmos evolutivos y las metaheurísticas [1]. Entre las técnicas evolutivas destacan los algoritmos genéticos (AG) por su adaptabilidad a los problemas y robustes.

Los AG son una técnica de búsqueda que se basa en la teoría de la evolución y selección de sistemas biológicos de Darwin [2]. Son métodos adaptativos y generalmente son utilizados en problemas de búsqueda y optimización de parámetros.

El desarrollo de los algoritmos genéticos se debe en gran parte a los trabajos de John Holland y sus colaboradores. En la década de 1960, Holland establece las teorías que imitaban el funcionamiento de selección natural y su adaptación en sistemas robustos [3]. Con base en las estrategias del cómputo evolutivo de Rechenberg [4, 5], Holland propone en [6] una heurística basada en los principios genéticos y evolución biológica como metodología para resolver problemas llamada algoritmo genético.

El algoritmo genético clásico toma una representación de un problema (codificación) y genera un grupo de posibles soluciones (llamada población). La idea de AG es que durante la ejecución del algoritmo la población evolucione, para que esto ocurra son aplicados los operadores genéticos: selección, cruce, mutación y reemplazo.

En el proceso de selección se eligen aquellas soluciones que presenten los mejores valores en su función de ajuste o *fitness*. En el proceso de cruce los individuos seleccionados producen soluciones descendientes de manera que contengan información parcial de cada uno de los progenitores. La mutación es un mecanismo que modifica de forma aleatoria algunos valores de los descendientes, de esta forma se desplazan valores numéricos hacia zonas del espacio de búsqueda que no pueden ser alcanzadas por los otros operadores genéticos. Por último, la técnica de reemplazo indica que miembros formarán parte de la siguiente población.

La capacidad de optimización y adaptabilidad de los AG permite que los investigadores puedan aplicarlos en áreas como robótica [7], economía [8], energía [9], medicina [10], geografía [11], entre otros. Dentro de los problemas de optimización encontramos las funciones de minimización. El problema de minimizar funciones [12] consiste en obtener el mínimo valor de una función real seleccionando sistemáticamente los valores de entrada, sujeto a un conjunto permitido, y computando el valor de la función. El desafío de estos problemas es obtener la mejor solución con la metodología más eficiente posible, es decir, aquella que utilice menos recursos y pueda generar un mayor rendimiento.

El algoritmo genético clásico da lugar a una gran variedad de modelos, tomando en consideración los operadores genéticos que se implementen. El operador de reemplazo juega un papel importante en el desarrollo de los AG; es el procedimiento encargado de decidir qué miembros de la población se preservarán entre generaciones. Bajo el teorema no-free-lunch [13] se puede aseverar que no existe el algoritmo ideal para todas las funciones posibles, por lo cual se analiza en este documento el impacto que tienen los operadores de reemplazo más utilizados en el estado del arte aplicados al problema de minimización de funciones.

2. Trabajos relacionados

2.1. Cómputo evolutivo

La computación evolutiva (CE) [14] es la rama de la computación que engloba a las técnicas inspiradas en la evolución biológica para la resolución de problemas, generalmente métodos de búsqueda estocásticos basados en ideas evolutivas de la selección natural y la genética. Los algoritmos de CE destacan por su fuerte adaptabilidad y autoorganización.

Aunque existe una variedad de modelos computacionales evolutivos, estos comparten una base conceptual común: la simulación de la evolución de estructuras individuales a través de procesos biológicos (llamados operadores genéticos). Estos procesos dependen del desempeño percibido de las estructuras individuales definidas por un entorno.

Los algoritmos evolutivos mantienen una población de estructuras que evolucionan de acuerdo con reglas de selección y otros operadores genéticos, como el cruce y la mutación. Cada individuo de la población es medido cuantitativamente por su aptitud al medio ambiente. La selección centra la atención en las miembros con mejor adaptación al ambiente. El cruce y la mutación perturban a esos individuos, proporcionando heurísticas generales para la exploración.

Entre los algoritmos más conocidos de la computación evolutiva están la programación evolutiva [15], estrategias evolutivas [16], programación genética [17] y los algoritmos genéticos [2]. Es en estos últimos se centra este artículo.

2.2. Algoritmos genéticos

Los algoritmos genéticos son técnicas que simulan la selección natural y el proceso evolutivo de los organismos vivos para resolver problemas de optimización, búsqueda y aprendizaje en las máquinas.

Esta estrategia pertenece a las técnicas basadas en poblaciones. Dado un problema particular a resolver, la entrada de AG es un conjunto de posibles soluciones al problema (llamado conjunto de individuos) codificadas de alguna manera y que, asociada una función llamada *fitness*, evalúa cuantitativamente cada solución candidata.

En cada iteración del algoritmo (llamado generación) se le aplican a la población los cuatro operadores de los AG: cruce, selección, mutación y reemplazo, en donde las soluciones candidatas son sometidas a acciones aleatorias semejantes a la evolución biológica.

Durante la ejecución del algoritmo el tamaño de la población se mantiene constante. Después de aplicar un número determinado de generaciones, la salida del AG es el mejor miembro de la población, es decir, la solución con el mejor valor *fitness*. El proceso de los AG se observa en la Fig. 1 y se explica a continuación:

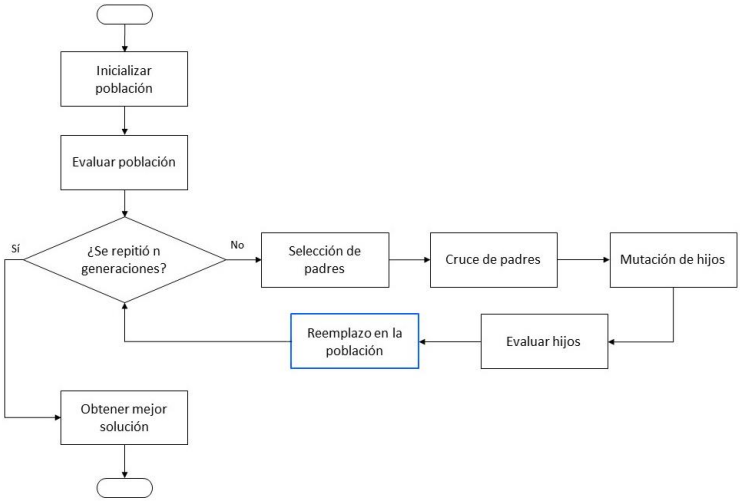

Fig. 1. Proceso de un algoritmo genético. La etapa de reemplazo en la población (remarcada en color azul) es el proceso de estudio de este trabajo.

- Inicializar población: Se genera una población inicial de soluciones candidatas, usualmente de forma uniformemente aleatoria
- Evaluación. Paso donde se calcula el valor *fitness* de cada solución candidata. Este proceso se realiza después de obtener la población inicial o cuando se obtiene una nueva solución descendiente.
- Selección: Se escogen qué individuos van a disponer de oportunidades de reproducirse. Las soluciones elegidas son llamadas *padres*.
- Cruce: Se genera soluciones descendientes (llamados *hijos*) a partir de las soluciones *padres*. La descendencia resulta de la recombinación de los *padres*.
- Mutación: Produce variaciones de modo aleatorio en cada una de las soluciones *hijo*, usualmente mediante un cambio aleatorio en la vecindad de la solución. La mutación es aplicada (con una tasa muy baja) a cada gen de la solución.
- Reemplazo: La población descendiente que se crea mediante la selección, cruce y mutación compite para reemplazar a algún miembro de la población inicial siguiendo algún criterio determinado. Estos son encargados de determinar qué soluciones continuarán en la siguiente generación.

En la siguiente sección se describirán algunos de los métodos de reemplazo más utilizados en el estado del arte.

2.3. Teorema de no free-lunch

Al estudiar los algoritmos genéticos se puede observar la gran gama de métodos que existen en cada operador de los que lo componen. Este trabajo analiza al operador de

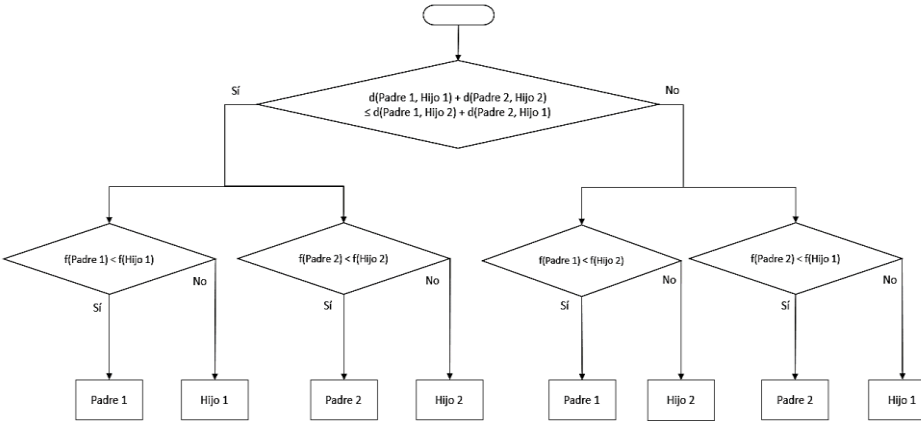

Fig. 2. Proceso del reemplazo crowding. En primera instancia, se emparejan los padres con los hijos de acuerdo con una métrica de similitud (función *d*). En una segunda fase, toma la decisión sobre cuál de ellos sobrevive en la población considerando el fitness (función *f*).

reemplazo; estos algoritmos son encargados de elegir, mediante un criterio, los individuos de la población que se preservarán entre generaciones. La importancia de los algoritmos de reemplazo es que estos orientan al espacio de búsqueda preservando do soluciones con mayor adaptación observada hasta el momento, y manteniendo la diversidad genética.

En este contexto, cada investigador de cómputo evolutivo diseña y crea algoritmos genéticos con funciones de reemplazo esperando que la salida sea el mínimo global del problema, sin embargo, la demostración del teorema de no-free-lunch [13] excluye la existencia de un algoritmo o modelo ideal. El teorema de no-free-lunch indica que no existe un algoritmo que sea universalmente mejor que los demás, siempre existirán escenarios con una distribución de probabilidad en la que falle. Este teorema es muy importante y gobierna la efectividad de los algoritmos de optimización [18].

Ahora, teniendo en cuenta lo demostrado en el teorema de no-free-lunch, se presenta esta investigación donde se implementan una serie de algoritmos de reemplazo más empleados en el estado del arte con el objetivo de analizar su comportamiento en funciones de minimización.

3. Materiales y métodos

3.1. Operadores de reemplazo

Los operadores de reemplazo utilizados en este trabajo son de los principales en el estado del arte [19], y se encuentran descritos a continuación.

- Reemplazo elitista: En este método consideran las soluciones *hijos* y toda la población para la competencia, la nueva población es compuesta por las mejores soluciones.

- Reemplazo de padres: Se obtiene espacio para la nueva descendencia (*hijos*) liberando el espacio ocupado por las soluciones *padres*.

- Reemplazo de padres por competencia: Los *hijos* compiten con los *padres* por ocupar su espacio en la población, entre este grupo son elegidas las soluciones con mejor *fitness*.

- Reemplazo proporcional. A cada miembro de la población e *hijos* se le asigna una probabilidad de ser elegido teniendo en cuenta su valor *fitness*. Los mejores individuos recibirán una mayor probabilidad de ser elegidos para la nueva población, pero no la certeza.

- Reemplazo crowding: Introducido por De Jong [20], es una técnica utilizada en algoritmos genéticos para preservar la diversidad en la población y evitar la convergencia prematura a óptimos locales. Consiste en emparejar cada descendencia con un individuo similar en la población actual y decidir cuál permanecerá en la población. En la Fig. 2 se puede observar este proceso.

3.2. Funciones de prueba

Las funciones utilizadas para probar los algoritmos genéticos fueron obtenidas de [21, 22, 23] y se muestran en la Tabla 1, estas son utilizadas como función *fitness* de cada algoritmo genético. Todas las funciones de prueba son problemas de minimización definidos como:

$$\text{Min } f(\mathbf{x}), \mathbf{x} = [x_1, x_2, \dots, x_D],$$

donde D es la dimensión de la función. Las funciones empleadas en este experimento presentan su valor de mínimo global cuando todas las componentes del vector solución son iguales a cero.

3.3. Parámetros del experimento

Los parámetros considerados para la ejecución de los algoritmos genéticos se muestran en la Tabla 2. Para cada función, se realizaron 10 ejecuciones independientes. La dimensión *D* se definió como *D=15* y los valores de cada dimensión se definieron como $x_i \in$ [-100, 100].

La metodología del operador de selección proporcional está definida en [24, 25], el operador de cruce Blx-α en [26] y el operador de mutación Gaussina en [27]. Los valores de probabilidad de cada operador se definieron considerando [28]. Los algoritmos genéticos fueron desarrollados en el lenguaje de programación Python (versión 3.6) e implementados en Google Colaboratory [29].

Tabla 1. Funciones utilizadas.

Nombre	Función	
Sphere	$f(\mathbf{x}) = \sum_{i=1}^{D} x_i^2,$	(1)
Rastrigin	$f(\mathbf{x}) = \sum_{i=1}^{D} (x_i^2 - 10\cos(2\pi x_i) + 10),$	(2)
Schwefel	$f(\mathbf{x}) = \sum_{i=1}^{D} \left(\sum_{j=1}^{i} x_j^2 \right)^2,$	(3)
Rosenbrock	$f(\mathbf{x}) = \sum_{i=1}^{D-1} (100(x_i^2 - x_{i+1})^2 + (x_i - 1)^2),$	(4)
Zakharov	$f(\mathbf{x}) = \sum_{i=1}^{D} x_i^2 + \left(\sum_{i=1}^{D} 0.5x_i \right)^2 + \left(\sum_{i=1}^{D} 0.5x_i \right)^4,$	(5)
Discus	$f(\mathbf{x}) = 10^6 x_1^2 + \sum_{i=2}^{D} x_i^2,$	(6)
HGBat	$f(\mathbf{x}) = \left\| \left(\sum_{i=1}^{D} x_i^2 \right)^2 - \left(\sum_{i=1}^{D} x_i \right)^2 \right\|^{1/2} + \left(.5 \sum_{i=1}^{D} x_i^2 + \sum_{i=1}^{D} x_i \right)/D + .5,$	(7)
Different Powers	$f(\mathbf{x}) = \sqrt{\sum_{i=1}^{D} \lvert x_i \rvert^{2+4\frac{i-1}{D-1}}},$	(8)
Happy Cat	$f(\mathbf{x}) = \left\| \sum_{i=1}^{D} x_i^2 - D \right\|^{1/4} + \left(.5 \sum_{i=1}^{D} x_i^2 + \sum_{i=1}^{D} x_i \right)/D + .5,$	(9)
Griewank	$f(\mathbf{x}) = \sum_{i=1}^{D} \frac{x_i^2}{4000} - \prod_{i=1}^{D} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1.$	(10)

3.4. Métricas de desempeño

Existen diversas métricas para medir el desempeño de algoritmos, la más utilizadas para los problemas de minimización son las medidas de tendencia central [30]: mínimo, máximo, promedio y desviación estándar. A continuación, se muestra cómo se calculan estas métricas:

$$Min(Y) = \min\{y_1, y_2, \dots, y_n\}, \tag{11}$$

$$Max(Y) = \max\{y_1, y_2, \dots, y_n\}, \tag{12}$$

$$Promedio(Y) = \bar{Y} = \frac{1}{n} \sum_{i=i}^{n} y_{i,} \tag{13}$$

$$Desviación\ Estandar(Y) = \sqrt{\frac{\sum_{i=i}^{n} (y_i - \bar{Y})^2}{n}}, \tag{14}$$

Tabla 2. Parámetros de los algoritmos genéticos.

Parámetro	Valor
Tamaño de población	10
Iteraciones	250
Máxima cantidad evaluaciones	500
Operador de selección	Proporcional
Operador de cruce	Blx-α
Probabilidad de cruce	0.9
Operador de mutación	Gaussiana
Probabilidad de mutación (por gen)	0.08
Dimensión de la función	15
Valor posible en cada dimensión	$x_i \in$ [-100, 100]
Ejecuciones independientes	10

Tabla 3. Mejores valores de los algoritmos genéticos con distintos métodos de reemplazo en funciones de minimización.

Función	Reemplazo elitista	Reemplazo proporcional	Reemplazo de padres	Reemplazo de padres por competencia	Reemplazo Crowding
Sphere	0.49	14.06	5.94E+04	**0.03**	1.26
Rastrigin	**67.67**	161.23	6.11E+04	97.12	122.29
Schwefel	938.43	1.16E+03	2.48E+05	836.42	**679.59**
Rosenbrock	36.39	103.03	7.80E+09	**5.85**	129.15
Zakharov	1.80E+03	3.41E+03	8.51E+04	**180.99**	626.70
Discus	25.23	31.62	1.24E+06	**0.19**	0.52
HGBat	0.46	34.46	7.28E+04	**0.41**	1.23
Different Powers	**1.07**	69.98	2.18E+05	7.58	1.80
Happy Cat	**0.44**	2.59	2.24E+03	0.69	0.65
Griewank	**0.54**	1.00	1.00	0.82	0.79
Total Ganados	4	0	0	**5**	1

donde $n = 10$ y $Y = [y_1, y_2, \ldots, y_n]$ son los valores obtenidos de las 10 ejecuciones independientes del algoritmo genético.

4. Resultados y discusión

En este trabajo se evalúan algoritmos genéticos con distintos operadores de reemplazo con el objetivo de optimizar funciones de minimización. Las métricas

Tabla 4. Peores valores de los algoritmos genéticos con distintos métodos de reemplazo en funciones de minimización.

Función	Reemplazo elitista	Reemplazo proporcional	Reemplazo de padres	Reemplazo de padres por competencia	Reemplazo Crowding
Sphere	16.23	1.47E+03	3.49E+05	**10.89**	215.84
Rastrigin	**131.83**	2.84E+03	1.83E+05	223.27	309.65
Schwefel	**9.14E+03**	1.26E+04	3.43E+06	1.24E+04	9.45E+03
Rosenbrock	7.79E+05	2.59E+06	5.88E+11	**2.19E+05**	1.33E+07
Zakharov	1.18E+04	2.50E+04	4.02E+09	**1.53E+04**	4.07E+04
Discus	5.43E+05	3.79E+07	7.23E+09	1.29E+05	**1.98E+03**
HGBat	**4.26**	880.17	4.32E+05	103.40	315.17
Different Powers	**131.91**	2.29E+03	1.86E+06	307.39	606.71
Happy Cat	**1.66**	73.54	1.25E+04	2.29	6.52
Griewank	**0.96**	1.28	3.16	0.98	0.97
Total Ganados	**6**	0	0	3	1

utilizadas para evaluar y comparar el desempeño de cada algoritmo fueron las medidas de tendencia central.

Los resultados del experimento se pueden observar en las Tablas 3-6, cada tabla corresponde a una métrica; en la Tabla 3 se observa el mejor valor *fitness* (el más cercano al mínimo global) dentro de las ejecuciones de cada algoritmo genético, la Tabla 4 muestra el peor valor *fitness* (el más lejano al mínimo global), en la Tabla 5 se presentan un promedio de valores *fitness* obtenidos por cada algoritmo y en la Tabla 6 la desviación estándar de estos valores. Cada columna representa el operador de reemplazo probado, mientras que cada fila corresponde a la función de minimización empleada. Los mejores resultados para cada función de minimización en particular se muestran en negrita.

Los resultados muestran que el operador de reemplazo de padres por competencia logró el mejor valores *fitness* (Tabla 3) en cinco de las diez funciones de minimización, seguido del reemplazo elitista logrando el mejor valor en cuatro funciones y el método Crowding en una función. Los algoritmos de reemplazo proporcional y reemplazo de padres no logran mejores resultados en ninguna de las funciones comparadas.

En los resultados de los peores valores *fitness* (Tabla 4) destaca el operador de reemplazo elitista al obtener el menor valor en seis funciones. El algoritmo de reemplazo de padres con competencia logra el menor valor entre los peores valores en tres funciones y, como en el caso anterior, el método Crowding en una función. Se identifica que los peores valores *fitness* para todas las funciones son obtenidos por el reemplazo proporcional.

El promedio de los valores obtenido en las ejecuciones, mostrado en la Tabla 5, muestran como mínimo valor de promedio el reemplazo elitista para cinco funciones,

Tabla 5. Promedio de los resultados de algoritmos genéticos con distintos métodos de reemplazo en funciones de minimización.

Función	Reemplazo elitista	Reemplazo proporcional	Reemplazo de padres	Reemplazo de padres por competencia	Reemplazo Crowding
Sphere	4.49	232.49	1.66E+05	**2.83**	58.63
Rastrigin	**93.01**	726.77	1.09E+05	141.21	188.14
Schwefel	4.35E+03	6.08E+03	9.27E+05	5.07E+03	**3.85E+03**
Rosenbrock	7.99E+04	3.06E+05	1.81E+11	**2.65E+04**	1.34E+06
Zakharov	1.16E+04	1.54E+04	1.01E+09	**7.01E+03**	1.06E+04
Discus	6.95E+04	4.57E+06	2.43E+09	1.50E+04	**8.03E+03**
HGBat	**1.58**	261.77	1.52E+05	12.37	55.20
Different Powers	**50.44**	725.93	6.55E+05	120.70	105.73
Happy Cat	**0.98**	20.79	5.34E+03	1.44	2.68
Griewank	**0.88**	1.03	1.54	0.95	0.92
Total Ganados	**5**	0	0	3	2

Tabla 6. Desviación estándar de los resultados de algoritmos genéticos con distintos métodos de reemplazo en funciones de minimización.

Función	Reemplazo elitista	Reemplazo proporcional	Reemplazo de padres	Reemplazo de padres por competencia	Reemplazo Crowding
Sphere	4.95	430.20	8.95E+04	**3.13**	65.46
Rastrigin	**20.26**	778.10	3.75E+04	37.53	65.01
Schwefel	2.45E+03	3.33E+03	8.71E+05	3.77E+03	**2.44E+03**
Rosenbrock	2.33E+05	7.66E+05	1.83E+11	**6.50E+04**	3.97E+06
Zakharov	**5.93E+03**	6.49E+03	1.36E+09	6.21E+03	1..07E+04
Discus	1.62E+05	1.13E+07	2.10E+09	3.81E+04	**714.77**
HGBat	**1.08**	236.79	9.96E+04	30.44	92.35
Different Powers	**38.62**	575.71	5.20E+05	99.75	186.62
Happy Cat	**0.40**	20.49	2.58E+03	0.53	1.83
Griewank	0.12	0.08	0.68	**0.04**	0.05
Total Ganados	**5**	0	0	3	2

seguido del reemplazo de padres por competencia, en tres funciones, y el reemplazo Crodwing (en dos funciones).

n los valores de la desviación estándar de los valores *fitness* (Tabla 6) presentan menor dispersión en los resultados obtenidos por el operador de reemplazo elitista en seis funciones. Escenario similar expuesto anteriormente, también logra destacar por su

menor valor de desviación estándar el reemplazo de padres por competencia (en dos funciones) y el reemplazo Crodwing (en una función).

Bajo el esquema del teorema de no-free-lunch, de que no existe el algoritmo genético ideal, no se presenta el escenario donde un operador de reemplazo muestre los mejores resultados para todas las funciones de minimización empleadas. Al analizar los resultados en conjunto, los tres operadores de reemplazo que destacan, independiente de la métrica, son el reemplazo de padres con competencia, el reemplazo elitista y el reemplazo Crodwing. Se reafirma el operador de remplazo de padres por competencia como una mejora al método de padres, esto al mostrar mejores resultados.

5. Conclusiones

En esta investigación se presenta un informe detallado sobre los experimentos realizados en algoritmos genéticos con distintos operadores de reemplazo conocidos en el estado del arte.

Entre los diversos operadores de reemplazo revisados y probados destacan el reemplazo elitista, el reemplazo de padres por competencia y el reemplazo Crodwing al generar soluciones con los mejores valores mínimos. Los algoritmos genéticos desarrollados en este trabajo representan una solución viable para optimizar funciones de búsqueda de soluciones.

Para futuras investigaciones, sugerimos modificar los parámetros de los algoritmos genéticos y la dimensión y dominio en las funciones de minimización. En este sentido creemos que el uso de algoritmos genéticos puede ser de gran beneficio en la búsqueda de resultados en problemas de optimización.

Referencias

1. Wang, Z., Qin, C., Wan, B.: A Comparative Study of Common Nature-Inspired Algorithms for Continuous Function Optimization. Entropy, 23(7), 874 (2021) doi: 10.3390/ e23070874.
2. Buontempo, F.: Genetic Algorithms and Machine Learning for Programmers: Create AI Models and Evolve Solutions. Pragmatic Bookshelf (2019)
3. Holland, J.H.: Outline for a Logical Theory of Adaptive Systems. Journal of the ACM, 9(3), pp. 297–314 (1962) doi: 10.1145/321127.321128.
4. Rechenberg, I.: Cybernetic Solution Path of an Experimental Problem. Royal Aircraft Establishment Library Translation, 1122 (1965)
5. Rechenberg, I.: Evolutionsstrategien. Simulationsmethoden in der Medizin und Biologie, pp. 83–114 (1973)
6. Holland, J. H.: Adaptation in Natural and Artificial Systems. University of Michigan Press (1975)
7. López-González, A., Campaña, J.M., Martínez, E.H.: Multi Robot Distance Based Formation Using Parallel Genetic Algorithm. Applied Soft Computing, 86, 105929 (2020) doi: 10.1016/j.asoc.2019.105929.
8. NoParast, M., Hematian, M., Ashrafian, A.: Development of a Non-Dominated Sorting Genetic Algorithm for Implementing Circular Economy Strategies in the Concrete Industry. Sustainable Production and Consumption, 27, pp. 933–946 (2021) doi: 10.1016/j.spc.2021.02.009.

9. Elsoragaby, S., Yahya, A., Mahadi, M.R.: Applying Multi-Objective Genetic Algorithm (MOGA) to Optimize the Energy inputs and Greenhouse Gas Emissions (GHG) in Wetland Rice Production. Energy Reports, 6, pp. 2988–2998 (2020) doi: 10.1016/j.egyr. 2020.10.010.
10. Książek, W., Gandor, M. Pławiak, P.: Comparison of Various Approaches to Combine Logistic Regression with Genetic Algorithms in Survival Prediction of Hepatocellular Carcinoma. Computers in Biology and Medicine, 134, 104431 (2021) doi: 10.1016/j.compbiomed.2021.104431.
11. Parimi, P., Rout, R.R.: Genetic Algorithm Based Rumor Mitigation in Online Social Networks Through Counter-Rumors: A Multi-Objective Optimization. Information Processing & Management, 58(5), 102669 (2021) doi: 10.1016/j.ipm.2021.102669.
12. Hussain, K., Salleh, M.N.M., Cheng, S.: Common Benchmark Functions for Metaheuristic Evaluation: A Review. JOIV: International Journal on Informatics Visualization, 1(4-2), pp. 218–223 (2017) doi: 10.30630/joiv.1.4-2.65.
13. Wolpert, D.H., Macready, W.G.: No Free Lunch Theorems for Optimization. In: IEEE Transactions on Evolutionary Computation, 1(1), pp. 67–82 (1997) doi: 10.1109/ 4235.585893.
14. Fogel, D.B.: Evolutionary Computation: Toward a New Philosophy of Machine Intelligence, 1, John Wiley & Sons (2006)
15. Yao X.: Evolutionary Computation. Evolutionary. Optimization. International Series in Operations Research & Management Science, 48. Springer, Boston, MA (2003) doi: 10.1007/0-306-48041-7_2.
16. Goldberg, D.E.: Genetic Algorithms in Search. Optimization and Machine Learning, Addison Wesley (1989)
17. Koza, J.R.: Genetic Programming. MIT Press (1992)
18. Joyce, T., Herrmann, J.M.: A Review of no Free Lunch Theorems, and their Implications for Metaheuristic Optimization. Nature-Inspired Algorithms and Applied Optimization, pp. 27–51 (2018) doi: 10.1007/978-3-319-67669-2_2.
19. Lozano, M., Herrera, F., Cano, J.R.: Replacement Strategies to Preserve Useful Diversity in Steady-State Genetic Algorithms. Information Sciences, 178(23), pp. 4421–4433 (2008) doi: 10.1016/j.ins.2008.07.031.
20. De Jong, K.A.: Analysis of the Behavior of a Class of Genetic Adaptive Systems. University of Michigan (1975)
21. Li, X., Tang, K., Omidvar, M.N.: Benchmark Functions for the CEC 2013 Special Session and Competition on Large-Scale Global Optimization. Technical Report, Evolutionary Computation and Machine Learning Group, RMIT University (2013)
22. C.T., Yue, K.V., Price, P.N.: Problem Definitions and Evaluation Criteria for the CEC 2020 Special Session and Competitionon Single Objective Bound Constrained Numerical Optimization. Technical Report. Rep., Zhengzhou University and Nanyang Technological University (2019)
23. Awad, M.Z., Ali, P.N., Suganthan, J.J.: Problem Definitions and Evaluation Criteria for the CEC 2017 Special Session and Competition on Single Objective Real-Parameter Numerical Optimization. Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou China and Technical Report, Nanyang Technological University, Singapore (2016)
24. Johnson, J.M., Rahmat-Samii, V.: Genetic Algorithms in Engineering Electromagnetics. IEEE Antennas and Propagation Magazine, 39(4), pp. 7–21 (1997) doi: 10.1109/74.632992.
25. Back, T.: Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms. Oxford University Press (1996)

26. Herrera, F., Lozano, M., Sánchez, A.M.: A Taxonomy for the Crossover Operator for Real-Coded Genetic Algorithms: An Experimental Study. International Journal of Intelligent Systems, 18(3), pp. 309–338 (2003) doi: 10.1002/int.10091.
27. Singh, P., Dwivedi, P., Kant, V.: A Hybrid Method Based on Neural Network and Improved Environmental Adaptation Method Using Controlled Gaussian Mutation with Real Parameter for Short-Term Load Forecasting. Energy, 174(C), pp. 460–477 (2019) doi: 10.1016/j.energy.2019.02.141.
28. Srinivas, M., Patnaik, L.M.: Genetic Algorithms: A Survey. Computer, 27(6), pp. 17–26 (1994) doi: 10.1109/2.294849.
29. Alphabet Google Colaboratory: https://colab.research.google.com/ (2021)
30. Zhang, Q.L.: Metrics for Meta-Heuristic Algorithm Evaluation. In: IEEE Computer Society. 2, pp. 1241–1242 (2003) doi: 10.1109/ICMLC.2003.1259677.