XLI CILAMCE
XLI Ibero-Latin American Congress on Computational Methods in Engineering
16-19 November, 2020 FOZ DO IGUAÇU PARANÁ - BRAZIL

Optimización de armaduras con un algoritmo de evolución diferencial a través de control de parámetros

Oscar. Contreras-Bejarano[1], Jesús D. Villalba-Morales[2],

[1,2]*Dept. Ingeniería Civil, Pontificia Universidad Javeriana*
Bogotá, Colombia
contreras.o@javeriana.edu.co, jesus.villalba@javeriana.edu.co

Abstract. In recent decades, numerical tools have been to improve the design process of civil structures. Among them, it is possible to find the evolutionary algorithms, which are based on analogies with the evolutionary process. Among the main limitations observed in the literature for its implementation is the need to enter the algorithm parameters for a problem in study by the user. There exist some parameters own of each algorithm that control the success level. This paper presents the application of an adaptive differential evolution algorithm (ADEA) to the optimization of planar steel trusses. In ADEA is necessary that the user to set the population size and the coefficients c_F and c_{CR}. To show the performance of the proposed methodology, two cases obtained in the literature are presented for comparison, which is done in terms of quality of the objective function, robustness and convergence. It is expected that with this type of work, the results required in structural optimization problems can be improved through implementing the differential evolution algorithm.

Keywords: algoritmo adaptativo, armadura, control de parámetros, evolución diferencial, optimización.

1 Introducción

Las técnicas de evolución computacional han sido utilizadas en diferentes problemas relacionados con procesos de optimización. Las denominadas heurísticas son algoritmos que poseen una secuencia dada de elementos de acuerdo con un conjunto de reglas fijas, a menudo estas técnicas producen soluciones con un razonable marco de tiempo y se consideran lo suficientemente buenas para el manejo del problema en cuestión, pero usualmente no generan soluciones cercanas al óptimo. Por otro lado las metaheurísticas operan con un mayor nivel de flexibilidad que las heurísticas, y por lo general proporcionan soluciones cercanas a la óptima [1]. El teorema "No Free Lunch" afirma que todas las heurísticas son igualmente eficientes cuando se mide un rendimiento sobre todos los posibles problemas [2], y considerando que las metaheurísticas están, por definición, destinadas a ser aplicadas a una variedad de problemas, esto significa que las implementaciones deben ser fáciles de usar y extensibles para los usuarios que provienen de diferentes campos de aplicación [3] se utiliza la evolución diferencial (DEA) como algoritmo de estudio, este es uno de los algoritmos de búsqueda estocástica más utilizados en evolución computacional [4]. Esta técnica basada en la población [2] es un algoritmo simple pero eficiente para la optimización numérica [5] inspirado en el concepto biológico de la selección natural en donde se permite que las personas con buenos valores de condición física sobrevivan de generación en generación, intentando mejorar iterativamente este valor a través de operaciones de adaptación [6]. Esta técnica fue desarrollada inicialmente por Storn y Price [7]. Este algoritmo simula la evolución de individuos utilizando un conjunto predefinido de operadores. Comúnmente se utilizan dos tipos de operadores, selección y operadores de búsqueda. De esta última los más utilizados son la mutación y la recombinación [8]. Dentro del proceso del algoritmo de optimización se presentan dos fases, la primera hace referencia a la exploración en el espacio de búsqueda y la segunda es la explotación, la cual se da después que encontrar un punto potencialmente óptimo. Aunque el DEA ha tenido éxito en diversos campos, puede tener una baja eficiencia en la explotación de las soluciones [5] dado que los algoritmos evolutivos son demasiados sensibles a los valores iniciales de los parámetros intrínsecos, el uso de valores iniciales

fijos para dichos parámetros puede comprometer el éxito de la resolución de los algoritmos evolutivos [4]. La relación caótica entre los valores iniciales de los parámetros de los algoritmos evolutivos y los éxitos de resolución de problemas de los algoritmos relevantes sigue siendo un área de investigación activa ya que no existe un método de configuración de parámetros analíticos que establezca los mejores parámetros del algoritmo de búsqueda evolutiva para resolver [4]. Este trabajo se enfoca en la implementación un ADEA para el control de los parámetros iniciales aplicado a problemas de optimización del peso de armaduras.

2 Algoritmo de evolución diferencial adaptativo

A continuación, se presenta el ADEA [9]. Dentro de dichos parámetros iniciales los factores de mutación y cruce representan las condiciones con las cuales el DEA puede hallar una mejor solución. Una magnitud alta del factor de mutación implica un cambio importante en la siguiente generación, por otro lado, un valor alto del factor de cruce afecta la etapa de combinación aumentando la probabilidad de elegir el individuo obtenido en la etapa de mutación. Por tal motivo para la etapa de exploración, el factor de mutación se incrementa mientras que el factor de cruce decrece, caso contrario en la etapa de explotación, en donde el factor de mutación decrece mientras el factor de cruce aumenta. La Tab. 1 presenta la secuencia lógica de pasos a seguir para la obtención de una solución. Para el caso del DEA se omiten los pasos del 2 al 12.

Tabla 1a. Algoritmo de evolución diferencial con dos niveles de parámetros adaptativos

Pasos	Acciones
Paso 1	Generar la población inicial (X_{ij}).
Paso 2	Dividir la población inicial en grupos de igual número de individuos. For i=1:NP
Paso 3	Generar el costo de cada individuo.
Paso 4	Ordenar la posición del vector de costo del mejor al peor por cada grupo (f_i).
Paso 5	Calcular las distancias euclidianas del mejor individuo a los demás individuos.
Paso 6	Ordenar de manera descendente las distancias del paso 5 (del más cercano al más lejano). Suponga que el ranking de la distancia del individuo i se denota como d_i. En donde $i = 1, 2, \dots , NP$
Paso 7	Generar el indicador de etapa de optimización (IOS). $$IOS = \sum_{i=1}^{NP} \lvert f_i - d_i \rvert \quad (1)$$
Paso 8	Generar el IOS mínimo (IOS_{min}) y máximo (IOS_{max}). si las dos clasificaciones para cada individuo son exactamente iguales (es decir, $f_i = d_i$ para cada i), entonces los mejores individuos también están más cerca del mejor individuo, y el IOS tiene su valor mínimo $$IOS_{min} = \sum_{r=1}^{NP} \lvert r - r \rvert = 0$$ $$IOS_{max} = \begin{cases} \frac{NP^2}{2}, \ si\ NP\ es\ par \\ \frac{(NP+1)(NP-1)}{2}, \ si\ NP\ es\ impar \end{cases} \quad (2)$$
Paso 9	Generar el IOS normalizado ($\overline{IOS}$). $$\overline{IOS} = \frac{IOS - IOS_{min}}{IOS_{max} - IOS_{min}} \quad (3)$$
Paso 10	Generar el estado de optimización (ϕ). $$\phi = \begin{cases} S_1, \ si\ Rand(0,1) < \overline{IOS} \\ S_2, \ de\ otra\ forma \end{cases} \quad (4)$$
Paso 11	Generar los factores de mutación (F_p) y cruce (CR_p) a nivel de la población.

Tabla 1b. Algoritmo de evolución diferencial con dos niveles de parámetros adaptativos

$$F_p^g = \begin{cases} F_p^{g-1} + c_f * \Delta F_p, & si\ \phi = S_1 \\ F_p^{g-1} - c_f * \Delta F_p, & si\ \phi = S_2 \end{cases} \quad (5)$$

$$CR_p^g = \begin{cases} CR_p^{g-1} - c_{cr} * \Delta CR_p, & si\ \phi = S_1 \\ CR_p^{g-1} + c_{cr} * \Delta CR_p, & si\ \phi = S_2 \end{cases} \quad (6)$$

$$En\ donde:$$

$$\Delta F_p, \Delta CR_p = \begin{cases} \frac{IOS - IOS_{min}}{IOS_{max} - IOS_{min}}, & si\ \phi = S_1 \\ \frac{IOS_{max} - IOS}{IOS_{max} - IOS_{min}}, & si\ \phi = S_2 \end{cases} \quad (7)$$

Paso 12 Generar los factores de mutación (F_i) y cruce (CR_i) a nivel de individuos.

$$F_i^g = \begin{cases} F_p^g + \Delta F_i, & si\ \left(f_i > \frac{NP}{2}\right) \wedge \left(d_i > \frac{NP}{2}\right) \\ F_p^g - \Delta F_i, & si\ \left(f_i < \frac{NP}{2}\right) \wedge \left(d_i < \frac{NP}{2}\right) \\ F_p^g, de\ otra\ forma \end{cases} \quad (8)$$

$$CR_i^g = \begin{cases} CR_p^g - \Delta CR_i, & si\ \left(f_i > \frac{NP}{2}\right) \wedge \left(d_i > \frac{NP}{2}\right) \\ CR_p^g + \Delta CR_i, & si\ \left(f_i < \frac{NP}{2}\right) \wedge \left(d_i < \frac{NP}{2}\right) \\ F_p^g, de\ otra\ forma \end{cases} \quad (9)$$

En donde:

$$\Delta F_i, \Delta CR_i = \begin{cases} \frac{(f_i + d_i) - NP}{2NP}, & si\ \left(f_i > \frac{NP}{2}\right) \wedge \left(d_i > \frac{NP}{2}\right) \\ \frac{NP - (f_i + d_i)}{2NP}, & si\ \left(f_i < \frac{NP}{2}\right) \wedge \left(d_i < \frac{NP}{2}\right) \end{cases} \quad (10)$$

Paso 13 Generar la población mutada (V_{ij}).

$$V_{ij} = X_{best} + F_i(X_{r1} - X_{r2}) \quad (11)$$

Paso 14 Generar la población combinada (U_{ij}).

$$U_{ij} = \begin{cases} V_{ij}, si\ Rand(0,1) \leq CR\ or\ j = j_{rand} \\ X_{ij},\ de\ otra\ forma \end{cases} \quad (12)$$

Paso 15 Seleccionar los individuos con mejor costo para conformar la población de la siguiente generación.

$$X_{ij}^{g+1} = \begin{cases} U_{ij}, si\ f(U_{ij}) \leq f(X_{ij}) \\ X_{ij},\ de\ otra\ forma \end{cases} \quad (13)$$

End

Paso 16 Si las condiciones son satisfechas; terminar el algoritmo, de lo contrario volver al paso 2.

3 Metodología

Los algoritmos tanto DEA como ADEA son programados en el software Matlab con la lógica descrita en la Tab 1. Se realizó una búsqueda en la literatura con el fin de encontrar dos investigaciones similares en las que se comparen diferentes técnicas de optimización en armaduras planas. Los criterios de comparación se limitan a la sección transversal de las barras, el peso de la armadura y el número de análisis hechos del algoritmo. Se ejecutaron dichos programas considerando la ecuación 14 para la penalización estática del peso de la armadura.

$$Peso_{restringido} = Peso * \left(1 + \alpha * \frac{NRNS}{NRT}\right)^{\beta} \tag{14}$$

En donde NSRS corresponde al número de restricciones no satisfechas, NRT es el número de restricciones totales, y α y β son valores deterministas con los que se ajusta el algoritmo; los valores de α y β usados en este problema son de 10 y 9 respectivamente. El proceso de optimización del peso de las armaduras depende de la sección transversal de las barras que la conforman. Se utiliza el método matricial de la rigidez para conocer el desplazamiento máximo y se usa la ecuación de pandeo de Euler para comparar el esfuerzo máximo actuante que es capaz de resistir cada barra de la armadura.

Minimizar:

$$w(A) = \sum_{i=1}^{n} A_i L_i \gamma_i \tag{15}$$

Sujeto a:

$$g_1 : (A_i, L_i) > 0 \tag{16}$$

$$g_2 : \delta_{max} - \delta_i > 0 \tag{17}$$

$$g_3 : \sigma_{max} - \sigma_i \tag{18}$$

En donde $w(A)$: Peso de la armadura, δ_i : Deformación en la barra i, σ_i : Esfuerzo axial en la barra i, A_i : Área transversal de la barra i, L_i : Longitud de la barra i, γ_i : Peso por unidad de volumen de la barra i, n : Número de barras en la armadura.

4 Ejemplos numéricos

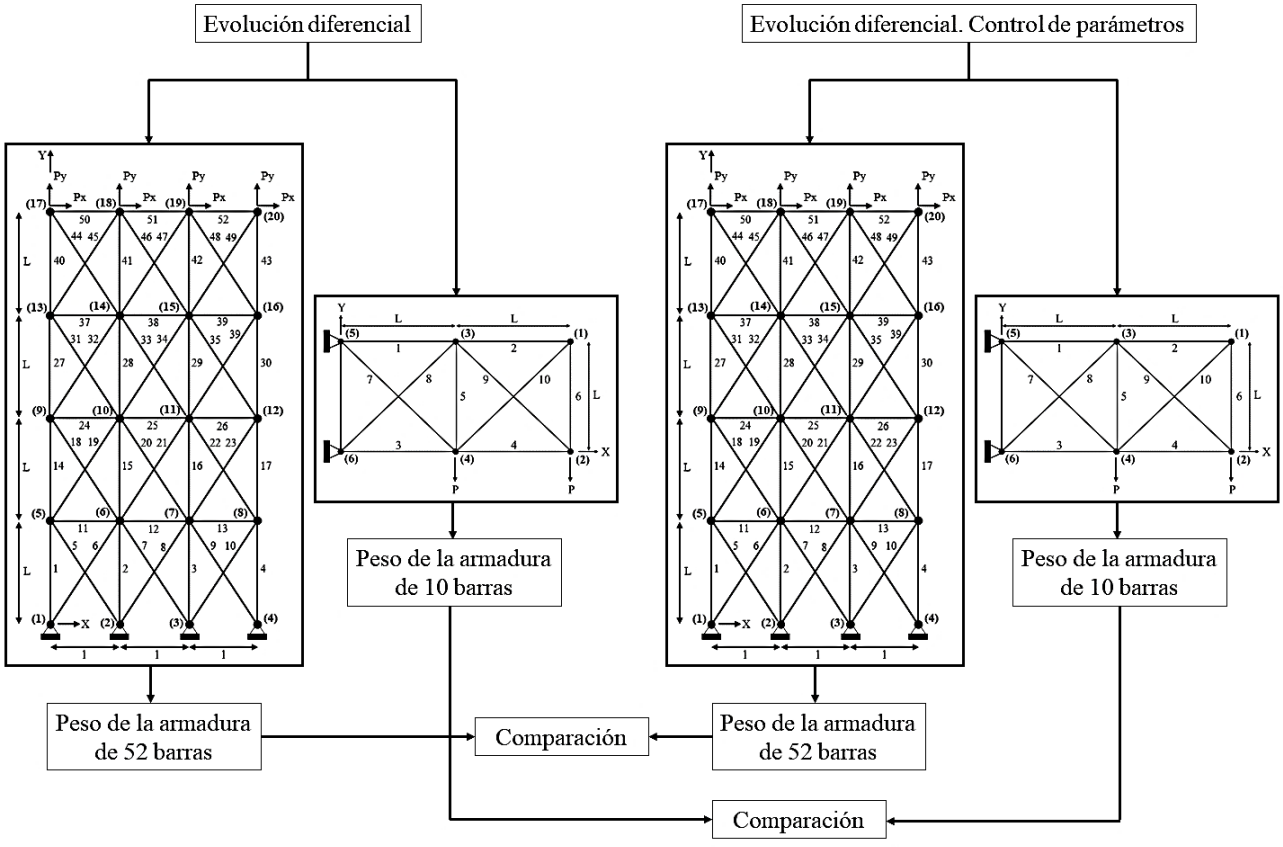

Figura 1. Armaduras de aplicación de los algoritmos

4.1 Armadura de 10 barras

Esta armadura presenta una optimización individual de cada una de las 10 barras, contempla 6 nodos, la longitud L equivale a 9.144 metros. La carga P está aplicada en los nodos 2 y 4 con una magnitud de 444.82 kN. Se considera una densidad de 2768 kg/m^3, un módulo de elasticidad de 68947.57 MPa, un desplazamiento máximo de ± 5.08 cm y un esfuerzo máximo a tracción o a compresión de 172.369 MPa. Las secciones transversales pueden considerar valores de área superiores a 64.516 mm^2. Se utilizan 100 ejecuciones y 100 poblaciones. Los valores de los coeficientes c_F y c_{CR} son 0.1 y 0.05 respectivamente.

4.2 Armadura de 52 barras

Las barras de esta armadura están subdivididas en 12 grupos; (1) A_1-A_4, (2) A_5-A_{10}, (3) A_{11}-A_{13}, (4) A_{14}-A_{17}, (5) A_{18}-A_{23}, (6) A_{24}-A_{26}, (7) A_{27}-A_{30}, (8) A_{31}-A_{36}, (9) A_{37}-A_{39}, (10) A_{40}-A_{43}, (11) A_{44}-A_{49} y (12) A_{50}-A_{52}. Se contemplan 20 nodos. La longitud L y l equivalen a 3 m y 2 m respectivamente. La carga Px y Py están aplicadas del nodo 17 al nodo 20 con una magnitud de 100 kN y 200 kN respectivamente. Se considera una densidad de 7860 kg/m^3, un módulo de elasticidad de 207 GPa, y un esfuerzo máximo a tracción o a compresión de 180 MPa. La armadura de 52 barras solo puede generar secciones transversales del siguiente conjunto de datos D = [71.613, 90.968, 126.451, 161.290, 198.064, 252.258, 285.161, 363.225, 388.386, 494.193, 506.451, 641.289, 645.160, 792.256, 816.773, 939.998, 1008.385, 1045.459, 1161.288, 1283.868, 1374.191, 1535.481, 1690.319, 1696.771, 1858.061, 1890.319, 1993.544, 2019.351, 2180.641, 2238.705, 2290.318, 2341.931, 2477.414, 2496.769, 2503.221, 2696.769, 2722.575, 2896.768, 2961.284, 3096.768, 3206.445, 3303.219, 3703.218, 4658.055, 5141.925, 5503.215, 599.988, 6999.986, 7419.340, 8709.660, 8967.724, 9161.272, 9999.980, 10322.560, 10903.204, 12129.008, 12838.684, 14193.520, 14774.164, 15806.420, 17096.740, 18064.480, 19354.800, 21612.860] (valores en mm^2), así mismo en esta armadura solo se considera una sola restricción relacionada al esfuerzo máximo. Se utilizan 50 ejecuciones y 700 poblaciones. Es evidente que el DEA para la armadura de 10 barras presenta un comportamiento similar respecto al resultado de optimización de otros algoritmos como HS o PSO, en cuanto a la implementación del ADEA, se encuentra un decrecimiento de 55.44% en la desviación estándar y una disminución en el peso al utilizar control de parámetros.

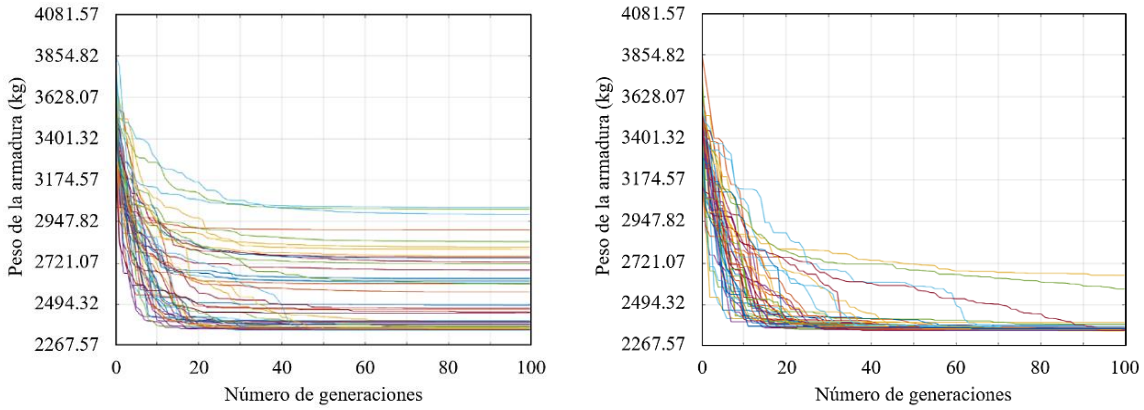

Figura 2. Convergencia del algoritmo de armadura de 10 barras. Evolución diferencial (izquierda) y del algoritmo de evolución diferencial con control de parámetros (derecha)

Cada una de las líneas de la figura 2 corresponde a una ejecución. Las ejecuciones son dependientes dado que se inicia una nueva ejecución con los valores obtenidos de la ejecución anterior. En la tabla 2 se presenta la comparación de los resultados obtenidos en este trabajo con los resultados de estudios comparables encontrados en la literatura.

Tabla 2. Algoritmo de evolución diferencial con dos niveles de parámetros adaptativos

A_i (cm^2)	HS [10]	PSO [11]	HPSO [11]	ADE	ADEA
A_1	194.516	215.829	198.09	190.21	193.66
A_2	0.658	0.710	0.645	0.65	0.65
A_3	146.516	149.529	149.464	172.09	167.03
A_4	98.516	99.835	97.955	103.40	111.25
A_5	0.658	23.5419	0.645	0.645	0.645
A_6	3.510	0.748	3.555	0.645	0.645
A_7	48.652	53.729	48.129	46.63	46.97
A_8	139.096	150.580	132.309	137.31	135.48
A_9	138.387	148.477	138.761	142.48	138.89
A_{10}	0.645	1.226	0.645	0.645	0.645
Peso (kg)	2293.82	2507.70	2295.20	2353.88	2351.52
No. Análisis	20000	150000	125000	10000	10000

Para la armadura de 52 barras ADEA presentó una convergencia gradual como se presenta en la Fig 3 y La desviación estándar tuvo una reducción de 53.61%.

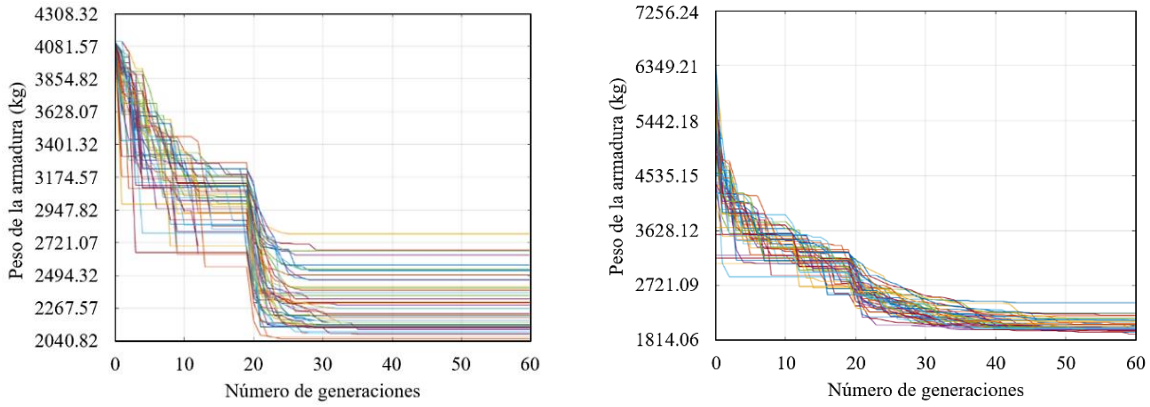

Figura 3. Convergencia del algoritmo de armadura de 52 barras. Evolución diferencial (izquierda) y del algoritmo de evolución diferencial con control de parámetros (derecha)

La tabla 3. Muestra una comparación de los resultados obtenidos para cada grupo de barras y los resultados de trabajos encontrados en la literatura. El aumento en el número de análisis genera mejores resultados, la desventaja radica en el aumento del costo computacional.

Tabla 3. Algoritmo de evolución diferencial con dos niveles de parámetros adaptativos

A_i (cm^2)	HS [12]	HPSO [13]	HHS [14]	ADE	ADEA
A_1-A_4	46.581	46.581	46.581	46.581	46.581
A_5-A_{10}	11.613	11.613	11.613	11.613	11.613
A_{11}-A_{13}	5.065	3.632	4.942	9.400	6.452
A_{14}-A_{17}	33.032	33.032	33.032	32.064	33.032
A_{18}-A_{23}	9.400	9.400	9.400	11.613	10.452
A_{24}-A_{26}	4.942	4.942	4.942	11.613	4.942
A_{27}-A_{30}	22.903	22.387	22.387	21.806	22.387
A_{31}-A_{36}	10.084	10.084	10.084	11.613	10.452
A_{37}-A_{39}	22.903	3.884	4.942	11.613	3.632
A_{40}-A_{43}	15.355	12.839	12.839	11.613	15.355
A_{44}-A_{49}	10.452	11.613	11.613	11.613	10.452
A_{50}-A_{52}	5.065	7.923	4.942	11.613	4.942
Peso (kg)	1906.8	1905.5	1902.6	2055.4	1931.6
No. Análisis	19104	150000	5300	35000	35000

5 Conclusiones

Se adaptan dos armaduras de 10 y 52 barras encontradas en la literatura, las cuales son sometidas a procesos de optimización mediante algoritmos como HS, PSO, HPSO, HHS, DE y DE adaptativo con control de parámetros. Los dos últimos trabajados en este artículo. Tanto el algoritmo de evolución diferencial como el algoritmo adaptativo con control de parámetros analizan la totalidad de ejecuciones definidas por el usuario con cero restricciones no satisfechas, esto evidencia que ambos algoritmos llegan a la convergencia.

Se aplica el algoritmo de evolución diferencial encontrando una similitud en los resultados de las secciones transversales de otras metodologías de optimización tanto para la armadura de 10 barras como para la de 52 barras. Es evidente la reducción de alrededor del 50% en la desviación estándar utilizando el control de parámetros en el algoritmo de evolución diferencial, esto permite disminuir la incertidumbre en los resultados encontrados.

El proceso de convergencia en el algoritmo de evolución diferencial tiende a encontrar un óptimo más rápido que el algoritmo de evolución diferencial con control de parámetros, esto indica que la inclusión de esta última técnica potencializa la exploración en el espacio de búsqueda al inicio del algoritmo, posteriormente disminuye su capacidad de exploración al tiempo que incrementa la explotación del mismo, por tal motivo el óptimo teniendo en cuenta control de parámetros siempre tiende a un mismo punto a diferencia el algoritmo de evolución diferencial original en donde tiende a poseer un comportamiento asintótico.

Deben realizarse más simulaciones numéricas con otras armaduras con el objetivo de entender la rápida convergencia del algoritmo, lo cual lo está afectando en la obtención de una mejor solución de diseño.

Authorship statement. The authors hereby confirm that they are the sole liable persons responsible for the authorship of this work, and that all material that has been herein included as part of the present paper is either the property (and authorship) of the authors, or has the permission of the owners to be included here.

6 Referencias

[1] R. G. Rakotonirainy and J. H. Van Vuuren, "Improved metaheuristics for the two-dimensional strip packing problem," *Appl. Soft Comput. J.*, vol. 92, 2020.

[2] R. L. Becerra and C. A. C. Coello, "Cultured differential evolution for constrained optimization," *Comput. Methods Appl. Mech. Eng.*, vol. 195, pp. 4303–4322, 2006, doi: 10.1016/j.cma.2005.09.006.

[3] J. Gmys, T. Carneiro, N. Melab, E. Talbi, and D. Tuyttens, "A comparative study of high-productivity high-performance programming languages for parallel metaheuristics," *Swarm Evol. Comput.*, vol. 57, no. October 2019, 2020.

[4] P. Civicioglu and E. Besdok, "Bernstain-search differential evolution algorithm for numerical function optimization," *Expert Syst. Appl.*, vol. 138, 2019.

[5] W. Gong, Z. Cai, and D. Liang, "Engineering optimization by means of an improved constrained differential evolution," *Comput. Methods Appl. Mech. Eng.*, vol. 268, pp. 884–904, 2014.

[6] I. M. Ali, D. Essam, and K. Kasmarik, "Novel binary differential evolution algorithm for knapsack problems," *Inf. Sci. (Ny).*, vol. 542, pp. 177–194, 2021.

[7] R. Storn and K. Price, "Differential Evolution - A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces," *J. Glob. Optim.*, vol. 11, no. 4, pp. 341–359, 1997, doi: 10.1023/A:1008202821328.

[8] V. P. Plagianakos, D. K. Tasoulis, and M. N. Vrahatis, "A Review of Major Application Areas of Differential," *Adv. Differ. Evol.*, vol. 143, pp. 197–238, 2008.

[9] W. Yu *et al.*, "Differential Evolution With Two-Level Parameter Adaptation," *IEEE Trans. Cybern.*, vol. 44, no. 7, pp. 1080–1099, 2014.

[10] K. S. Lee and Z. W. Geem, "A new structural optimization method based on the harmony search algorithm," *Comput. Struct.*, vol. 82, no. 9–10, pp. 781–798, 2004, doi: 10.1016/j.compstruc.2004.01.002.

[11] L. J. Li, Z. B. Huang, F. Liu, and Q. H. Wu, "A heuristic particle swarm optimizer for optimization of pin connected structures," *Comput. Struct.*, vol. 85, no. 7–8, pp. 340–349, 2007, doi: 10.1016/j.compstruc.2006.11.020.

[12] K. S. Lee, Z. W. Geem, S. H. O. Lee, and K. W. Bae, "The harmony search heuristic algorithm for discrete structural optimization," *Eng. Optim.*, vol. 37, no. 7, pp. 663–684, 2005, doi: 10.1080/03052150500211895.

[13] L. J. Li, Z. B. Huang, and F. Liu, "A heuristic particle swarm optimization method for truss structures with discrete variables," *Comput. Struct.*, vol. 87, no. 7–8, pp. 435–443, 2009, doi: 10.1016/j.compstruc.2009.01.004.

[14] M. Y. Cheng, D. Prayogo, Y. W. Wu, and M. M. Lukito, "A Hybrid Harmony Search algorithm for discrete sizing optimization of truss structure," *Autom. Constr.*, vol. 69, pp. 21–33, 2016, doi: 10.1016/j.autcon.2016.05.023.