Optimización mixto entera no lineal multi-objetivo basada en enjambre de partículas

Lucía Damiani[1*], Mariano Frutos[2], Aníbal M. Blanco[1]

[1]Planta Piloto de Ingeniería Química, Universidad Nacional del Sur (PLAPIQUI-CONICET), 8000 Bahía Blanca, Buenos Aires, Argentina.
[2]Departamento de Ingeniería, Universidad Nacional del Sur e IIESS UNS-CONICET 8000 Bahía Blanca, Argentina.
*ldamiani@plapiqui.edu.ar

Resumen. En este trabajo se presenta una herramienta de optimización para la resolución de problemas mixto entero no lineales multi-objetivo. El algoritmo se basa en la metaheurística de enjambre de partículas (PSO). Como PSO está diseñado para aplicarse a problemas continuos sin restricciones, para poder abordar problemas restringidos se le incorporó una técnica basada en el total de las violaciones a las restricciones de cada partícula. Adicionalmente, para tratar variables binarias, se anexó a la herramienta el método "Angle Modulation", el cual agrega cuatro variables continuas adicionales y con ellas establece, a través de una función trigonométrica, los valores de todas las variables binarias del problema. Finalmente, para abordar problemas multi-objetivo, se incorporó una metodología para identificar el frente de Pareto. El algoritmo desarrollado se probó sobre diferentes funciones benchmark de dos y tres objetivos, obteniéndose resultados factibles y muy similares a los reportados en la literatura.

Palabras Claves: optimización, PSO, MINLP-MO

1 Introducción

En las ciencias e ingenierías se pueden encontrar una gran variedad de problemas que pueden representarse matemáticamente a través de modelos de optimización mixto entera no lineal multi-objetivo (MINLP-MO). En este tipo de problemas, Ec. (1), se intenta hallar la mejor combinación de variables continuas ($\boldsymbol{x}$) y binarias ($\boldsymbol{y}$) que maximice una dada cantidad (NFO) de criterios de desempeño o funciones objetivo ($\boldsymbol{f(.)}$) de acuerdo a ciertas restricciones de igualdad ($\boldsymbol{h(.)}$) y desigualdad ($\boldsymbol{g(.)}$). Asimismo, en el caso de las variables continuas, también se debe considerar sus límites superiores ($\boldsymbol{x^{up}}$) e inferiores ($\boldsymbol{x^{lo}}$).

$$\begin{aligned} &Minimizar\ \boldsymbol{f}(\boldsymbol{x},\boldsymbol{y}) = (f_1(\boldsymbol{x},\boldsymbol{y}), \dots, f_k(\boldsymbol{x},\boldsymbol{y}), \dots\ f_{NFO}(\boldsymbol{x},\boldsymbol{y})) \\ &Sujeto\ a: \\ &\quad \boldsymbol{h}(\boldsymbol{x},\boldsymbol{y}) = 0;\ \boldsymbol{g}(\boldsymbol{x},\boldsymbol{y}) \leq 0;\ \boldsymbol{x^{lo}} - \boldsymbol{x} \leq 0;\ \boldsymbol{x} - \boldsymbol{x^{up}} \leq 0;\ \boldsymbol{y} \in (0,1) \end{aligned} \quad (1)$$

Los problemas que presentan no linealidades en sus funciones objetivo o en sus restricciones pueden introducir no convexidades que ocasionan regiones factibles irregulares. Esto conduce a múltiples óptimos locales, dificultando encontrar la mejor solución o,

incluso, simplemente una factible. Asimismo, si además de involucrar variables continuas (aquellas que pueden tomar cualquier valor entre dos cotas), el problema posee variables binarias (las que pueden ser solo igual a cero o a uno), se imponen restricciones de integralidad que conllevan una "complejidad combinatoria" debido a la gran cantidad de posibilidades que se pueden enumerar, incluso para un número modesto de decisiones. Por todas estas razones, este tipo de problemas es mucho más complejo de resolver que los que solo comprenden la versión continua y lineal.

Habitualmente en la práctica, también es necesario optimizar dos o más funciones objetivo en simultáneo las cuales, en general, se encuentran en conflicto entre sí. Esto ocasiona que una única solución no pueda mejorar a la vez todos los objetivos considerados y la mejora en uno de ellos se de en detrimento de los restantes. Esta característica acrecienta aún más los esfuerzos computacionales requeridos para encontrar soluciones de compromiso que resuelvan esta clase de problemas.

Por todas estas razones, la resolución de problemas de optimización MINLP-MO en tiempos de cómputo compatibles con aplicaciones prácticas es uno de los mayores desafíos dentro de la matemática aplicada moderna. Existen numerosas herramientas, libres y comerciales, para programar y resolver modelos MINLP. Las herramientas comerciales GAMS, gProms y Matlab, por ejemplo, proporcionan entornos de programación amigables y solvers muy eficientes, pero poseen costos elevados en concepto de licencias de software. Existen plataformas no-comerciales con buenas prestaciones para ciertos tipos de modelos (LP, NLP, MILP), pero no abundan los algoritmos MINLP generales en un entorno de programación lo suficientemente flexible para adaptarlo a aplicaciones diversas. Respecto de la optimización MO, habitualmente se requiere una programación ad-hoc en la plataforma de desarrollo adoptada. Por todo lo anterior, el objetivo principal de este trabajo es presentar una implementación propia de un solver MINLP-MO para abordar diferentes problemas de investigación y transferencia a desarrollar en el ámbito de nuestro grupo de investigación.

2 Herramienta de optimización MINLP-MO

Existe una gran cantidad de algoritmos de optimización numérica para resolver problemas MINLP-MO. Una clasificación básica se establece entre los determinísticos y los metaheurísticos. La optimización determinística es completamente predictiva, es decir, si se especifican los mismos puntos iniciales de las variables de entrada del algoritmo, éste convergerá siempre a la misma solución. Su implementación es matemáticamente sofisticada, y bajo ciertas condiciones, pueden garantizar la convergencia al óptimo local.

Por otro lado, se encuentran las metaheurísticas, las cuales cuentan con elementos aleatorios para guiar la exploración del espacio de soluciones, por lo cual pueden no necesariamente converger exactamente a la misma solución mediante la misma secuencia de cálculo. A diferencia de los métodos determinísticos, cuentan con la gran ventaja de no necesitar puntos iniciales próximos a los óptimos ni verificar propiedades matemáticas especiales, tornándolas muy populares para resolver problemas mixto-entero no lineales de tipo general [1].

En particular, en este trabajo se adoptó la metaheurística basada en el enjambre de partículas (PSO, por sus siglas en inglés) debido principalmente a: (i) un buen rendimiento reportado para una gran variedad de problemas; (ii) su relativa sencillez de implementación; (iii) su flexibilidad para anexarle nuevas técnicas que amplíen su campo de aplicación.

3.1 Optimización continua por Enjambre de Partículas (PSO)

La optimización continua por enjambre de partículas es un método propuesto originalmente por [2], basándose en la conducta de comunidades de organismos vivos, como por ejemplo los enjambres de las abejas en la búsqueda de alimento.

El PSO parte de una población o enjambre (N) de soluciones candidatas denominadas "partículas" que se van moviendo por el espacio de búsqueda de cierta dimensión (D) para encontrar mejores soluciones. La búsqueda se realiza principalmente en base a dos ecuaciones matemáticas sencillas en las cuales se va actualizando, en cada iteración (k), la posición (**x**)(2) y la velocidad (**v**)(3) de cada partícula (i) del enjambre. A medida que la población se mueve, los individuos van registrando la mejor posición encontrada por cada uno de ellos, a la que se denomina "mejor local" ($\boldsymbol{p}$), así como la mejor lograda por todos ellos, llamada "mejor global" ($\boldsymbol{q}$). Al hablar de la "mejor solución" se hace referencia a aquella que logra el menor valor de la función objetivo ($\boldsymbol{f}$) si se trata de un problema de minimización del objetivo (y viceversa en el caso de maximización).

$$x_{ij}^{k+1} = x_{ij}^{k} + v_{ij}^{k+1} t^{k} \quad \forall\, i \in [1, N],\ \forall\, j \in [1, D] \tag{2}$$

$$v_{ij}^{k+1} = w^{k} v_{ij}^{k} + c_1 r_{1_{ij}} \left(p_{ij}^{k} - x_{ij}^{k}\right) + c_2 r_{2_{ij}} \left(q_j - x_{ij}^{k}\right) \forall\, i \in [1, N],\ \forall\, j \in [1, D] \tag{3}$$

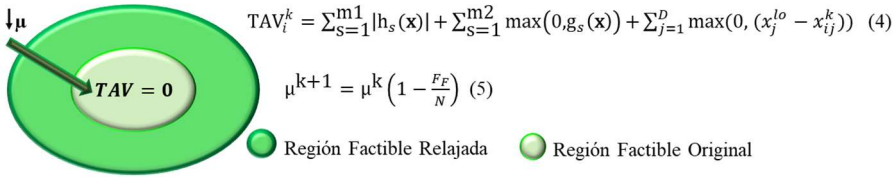
Finalmente, cuando se cumple el criterio de terminación impuesto (típicamente un número máximo de iteraciones, k_{max}), el algoritmo devuelve la mejor solución ($\boldsymbol{q}$) encontrada hasta ese momento.

3.2 Tratamiento de restricciones

El PSO básico [2] sólo tiene en cuenta los limites superiores (x_j^{up}) e inferiores (x_j^{lo}) de cada dimensión (j) del problema, pero no cuenta con un tratamiento explícito para manipular restricciones de igualdad ($\boldsymbol{h}(.)$) y de desigualdad ($\boldsymbol{g}(.)$). Para poder manipular estos tipos de restricciones se incorporó al PSO descrito por (1) y (2) la técnica propuesta en [3] basada en el concepto de violación absoluta de las restricciones. La misma se adoptó por ser bastante general, no requerir la reformulación o reducción de restricciones y ser bastante independiente de la selección de nuevos parámetros, lo que la vuelve mucho más práctica que otros métodos populares.

La metodología elegida consiste en agrandar temporalmente la región factible original, obteniendo una zona más relajada (Fig. 1). Esta región relajada se va reduciendo progresivamente a medida que las partículas comienzan a ingresar a ella hasta convertirse nuevamente en la región factible del problema original. Para ello, en cada iteración

(k) se debe calcular el valor del total de violaciones a las restricciones (TAV por sus siglas en inglés) por parte de cada partícula i (4), y el valor de relajación μ (5), el cual establece si una solución es temporalmente factible (cuando $\mathrm{TAV}_i^k < \mu^k$). Cabe mencionar que, en este trabajo, los límites de cada variable (límites de caja) se consideraron como restricciones de desigualdad y se tratan según el mismo procedimiento. En (5) F_F representa el número de individuos factibles en una dada iteración.



$$\mathrm{TAV}_i^k = \sum_{s=1}^{m1} |h_s(\mathbf{x})| + \sum_{s=1}^{m2} \max(0, g_s(\mathbf{x})) + \sum_{j=1}^{D} \max(0, (x_j^{lo} - x_{ij}^k)) \quad (4)$$

$$\mu^{k+1} = \mu^k \left(1 - \frac{F_F}{N}\right) \quad (5)$$

Fig. 1. Relajación de la región factible para el tratamiento de restricciones

Dado que ahora se tienen en cuenta las restricciones, cambia la forma de determinar cuál es el mejor global y local en cada iteración. Según el criterio de factibilidad, se considera que:

(i) Una solución factible ($\mathrm{TAV}_i^k < \mu^k$) es preferible a una infactible ($\mathrm{TAV}_i^k > \mu^k$);

(ii) Entre dos soluciones factibles ($\mathrm{TAV}_i^k < \mu^k$), aquella que alcanzó un mejor valor de la función objetivo (f_i) es la preferida;

(iii) Entre dos soluciones infactibles ($\mathrm{TAV}_i^k > \mu^k$), se selecciona la de menor TAV_i.

Esta técnica favorece la exploración del espacio de búsqueda al admitir violaciones transitorias a las restricciones mientras conduce a la población hacia la región factible de manera progresiva. La implementación de esta prestación se testeó sobre diversas funciones benchmark que difieren en su complejidad de resolución debido a la presencia de diversos tipos de no linealidad y restricciones, así como de distintos números de ecuaciones y variables [4]. Los resultados resultantes de optimizar este set de problemas demuestran un buen rendimiento general en todos ellos, alcanzándose el óptimo global en la mayoría de los casos y subóptimos factibles en el resto.

3.3 Tratamiento de variables binarias

Con el fin de ampliar aún más la gama de problemas que podrían resolverse con esta herramienta, al PSO descrito anteriormente se le incorporó la metodología "Angle Modulation" [5] para tratar variables binarias. La misma fue seleccionada principalmente porque permite trabajar con la metaheurística sin necesidad de realizar modificaciones especiales al algoritmo. Se trata de una técnica sencilla y fácil de anexar al PSO y para la cual se han reportado buenos resultados en distintas aplicaciones.

Básicamente este método consiste en sumar 4 variables continuas (a, b, c, d) al problema a optimizar las cuales son coeficientes de una función trigonométrica simple $G(.)$ (6). Dependiendo del valor que tome esta función, el valor de cada variable binaria (j) será igual a uno o cero según el criterio (7).

$$G(\mathrm{I}) = \sin(2\pi(I - a) \times b \times \cos(2\pi \times c(I - a))) + d \quad (6)$$

$$\left.\begin{matrix}\text{Si } G(I_i) > 0 \ \rightarrow y_i = 1 \\ Si\ G(I_i) \leq 0 \ \rightarrow y_i = 0\end{matrix}\right\} \forall\ i\ \in [1, n_b] \quad (7)$$

Los coeficientes $a,\ b,\ c, d$, manipulados como variables por el PSO, controlan la forma de la función sinusoidal $G(I)$, mientras que I es un intervalo que se divide en tantos subintervalos regulares como cantidad de variables binarias (n_b) sean requeridas.

De esta manera, la técnica "Angle Modulation" no solo resulta muy compatible con la mecánica de optimización del PSO al manipular un número pequeño de variables continuas adicionales, sino que también vuelve más sencillo el problema a resolver, ya que evita la complejidad combinatoria correspondiente al problema binario original. Cable aclarar que esta técnica proporciona una suerte de "memoria" en la búsqueda, pero no asegura una exploración exhaustiva del árbol de opciones al estilo "Branch & Cut" con lo cual es imposible asegurar la convergencia a soluciones globalmente óptimas.

3.4 Tratamiento de objetivos múltiples

Para extender aún más las prestaciones de la herramienta con el objeto de optimizar una amplia gama de problemas de diferentes características, también se incorporó al PSO una técnica para tratar problemas multi-objetivo. Si bien existe una gran variedad de métodos para resolver esta clase de problemas [6], en este trabajo se implementó una técnica que detecta y expone el frente de Pareto. Esta metodología consiste en que el algoritmo identifique aquellas soluciones pertenecientes a la frontera de Pareto, la cual representa un conjunto de soluciones donde cada una es mejor que las restantes en al menos uno de los objetivos optimizados. Este frente está representado por una curva en caso de problemas de optimización bi-objetivo y por una superficie en caso de problemas tri-objetivo. Con esta técnica se reduce la subjetividad y arbitrariedad en el tratamiento de los diferentes objetivos, a diferencia de otros métodos populares en los cuales se deben asignar prioridades. En este caso el tomador de decisión tiene a su disposición diversas soluciones y decide cuál de ellas aceptar de acuerdo a su conocimiento e intuición, al priorizar determinado criterio por sobre los demás [7].

Para poder establecer el frente de Pareto, es necesario primero definir los términos "soluciones dominadas" y "soluciones no dominadas", los cuales se utilizan para ordenar las distintas soluciones encontradas. Se dice que una solución S_1 domina a otra solución S_2 si se satisfacen dos condiciones (en problemas de minimización de todos los objetivos):

1. La solución S_1 no es peor que S_2 en todos los objetivos (f_k). Es decir, se debería cumplir que: $f_k(S_1) \leq f_k(S_2)$ para todo $k = \{1, \dots, N_{FO}\}$.
2. La solución S_1 es estrictamente mejor que S_2 al menos en un objetivo (f_k). Esto quiere decir que $f_k(S_1) < f_k(S_2)$ para al menos un objetivo $k = \{1, \dots, N_{FO}\}$.

En otras palabras, esta definición indica que la solución S_1 se considera un óptimo de Pareto si es igualmente buena en todos los objetivos y mejor en al menos uno de ellos. De esta manera, entre dos soluciones existen tres posibilidades:

- S_1 domina a S_2;
- S_1 es dominada por S_2;
- S_1 y S_2 son no dominadas entre sí.

A medida que se identifican las soluciones no dominadas, éstas se almacenan en un archivo externo (A). No obstante, se debe limitar el tamaño de este archivo para que no consuma demasiada memoria y disminuya la velocidad de la optimización con excesivas operaciones de almacenamiento. Si bien se programó que la dimensión de A sea establecida por el usuario, se requiere un criterio para determinar qué soluciones se descartan si el archivo está completo. En particular, para ello se utilizó la metodología propuesta por [8], la cual ordena las soluciones del archivo externo según la distancia de hacinamiento que existe entre ellas.

Para determinar la distancia de hacinamiento (dh) se inicializa su valor en cero para todas las soluciones presentes en A. Posteriormente, se clasifica el conjunto de soluciones según valores de función objetivo en orden ascendente. Luego, se establece el valor de dh como la distancia promedio entre las dos soluciones vecinas. A las soluciones límite, que son aquellas que tienen el menor y mayor valor de la función objetivo, se les asigna, arbitrariamente, un valor infinito en la distancia de hacinamiento para que siempre queden seleccionadas. Este proceso se realiza para cada función objetivo del problema de optimización. Finalmente, se calcula el valor de la distancia de hacinamiento de cada solución, como la suma de todas las dh de cada función objetivo.

Una vez obtenida la distancia de hacinamiento para cada partícula, se las ordena de mayor a menor en el archivo externo A. Si resulta que éste supera el límite de partículas establecido, entonces se eliminan tantas soluciones de la parte inferior de A como sea necesario para cumplir con la dimensión especificada para éste. De esta manera, sólo permanecen las partículas que mayor diversidad aportan, en cuanto a su valor en las funciones objetivo. El pseudocódigo para el cálculo de la distancia de hacinamiento se presenta en la Fig. 2.

Cabe mencionar que, con esta metodología, cambia la forma de determinar el mejor global ($\boldsymbol{q}$) del PSO, que antes era un valor único, y ahora puede ser igual a cualquiera de las soluciones presentes en A. Entonces, para establecer $\boldsymbol{q}$, siguiendo la propuesta de [8], a cada partícula se le asignó aleatoriamente cualquiera de las soluciones correspondientes al 10% de la parte superior de archivo externo (una vez que éste ya se ordenó de mayor a menor dh). De esta manera, se promueve el movimiento del enjambre hacia los puntos más dispersos entre sí.

En cuanto al mejor local ($\boldsymbol{p}$), sólo es necesario comparar la nueva solución encontrada en esa iteración ($\boldsymbol{x}$) con la guardada en $\boldsymbol{p}$. Si $\boldsymbol{x}$ domina a $\boldsymbol{p}$, entonces se actualiza el mejor local, si no, mantiene su valor.

1) Se obtiene el número total de soluciones no dominadas en el repositorio externo A.

$n_A = len(A)$

2) Se inicializa la distancia de hacinamiento dh de cada solución s en cero.

$dh[S_i] = 0.$

3) Se calcula la distancia de hacinamiento de cada solución.

Para cada función objetivo $k = \{1, \dots, N_{FO}\}$:

Se ordenan las soluciones de mayor a menor valor de la función objetivo k y se calcula la distancia de cada solución según el objetivo k:

Para cada k desde 1 hasta N_{FO}:

$S = sort(S, k)$

Para cada S_i desde 1 hasta n_A:

$dh[S_i] = dh[S_i] + dh_k[S_{i+1}] - dh_k[S_{i-1}]$

Se asigna a la distancia de hacinamiento de las soluciones límite un valor infinito.

$dh[S_0] = dh[S_{n_A}] = \infty$

Fig. 2. Pseudocódigo del cálculo de la distancia de hacinamiento

3.5 Pseudocódigo PSO MINLP-MO: Implementación propia

La implementación propia del algoritmo con los elementos descriptos en las secciones anteriores queda definida según el diagrama de la Fig. 3. En la Fig. 4 se proporciona el pseudocódigo correspondiente. A modo de resumen el optimizador se basa en un PSO básico [2], al cual se le adicionó: (i) el cálculo de "TAV" [3] para permitir la inclusión de restricciones; (ii) la técnica "Angle Modulation" [5] para manipular variables binarias y (iii) una metodología basada en la distancia de hacinamiento para identificar el frente de Pareto [8] a fin de optimizar problemas multi-objetivo. Como plataforma de desarrollo se adoptó el lenguaje de código abierto Python. Se seleccionó esta alternativa por su enorme popularidad en computación científica, gran disponibilidad de complementos, versatilidad, agilidad de lectura, flexibilidad y calidad de sintaxis.

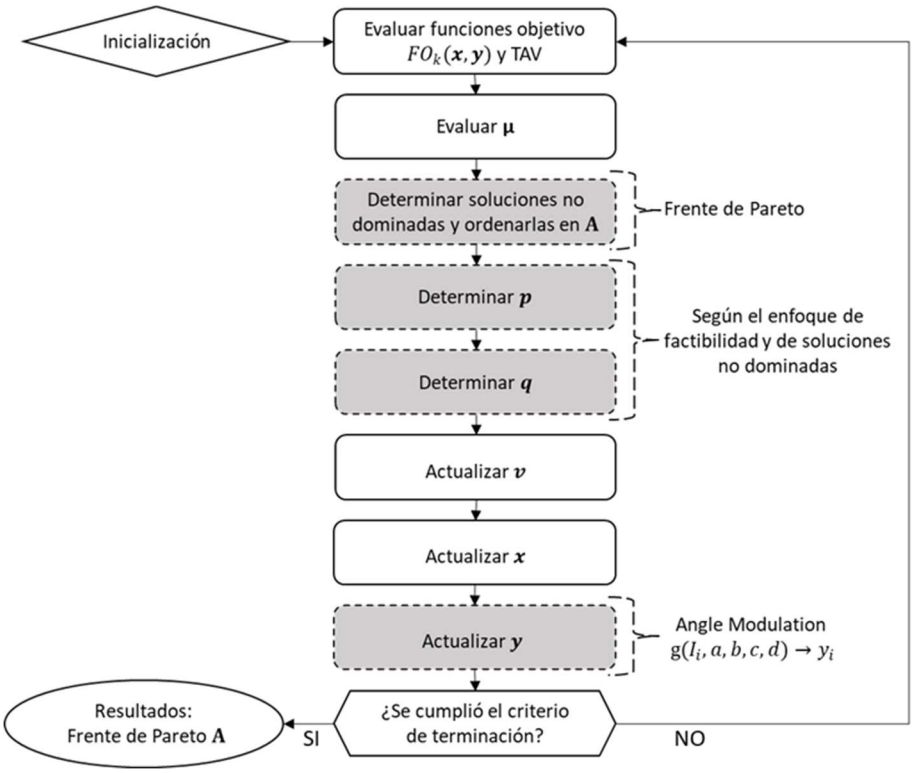


Fig 3. Esquema del PSO con variables continuas, binarias y multi-objetivo

1. Inicialización de cada partícula del enjambre:

   a. Inicializar la posición aleatoriamente: $x_{ij}^0 = U\left(x_{ij}^{lo}, x_{ij}^{up}\right)$ donde $x_{ij}^0 = [a, b, c, d, x_1, \dots, x_D]$

   b. Calcular $g(I_i) = \sin(2\pi(I_i - a) \times b \times \cos(A)) + d$, donde A= $2\pi \times c(I_i - a)$.

   c. Determinar las variables binarias : $\begin{Bmatrix} Si\ g(I_i) > 0 \ \rightarrow y_i = 1 \\ Si\ g(I_i) \leq 0 \ \rightarrow y_i = 0 \end{Bmatrix} \forall\ i\ \in [1, n_b]$

   d. Inicializar la velocidad en cero: $v_{ij}^0 = 0$

   e. Inicializar la mejor posición local: $p_{ij}^0 = x_{ij}^0$.

   f. Calcular el valor de la función$f_{N_{obj}}(x_i)$ y de $TAV_i(x_i)$.

   g. Determinar las soluciones no dominadas del óptimo de Pareto: S_i.

   h. Ordenar S_i según su dh en orden decreciente.

   i. Almacenar S_i en el archivo externo A.

   j. Determinar el mejor global: $q_i = U(10\%A)$.

   k. Calcular el valor de relajación inicial: $\mu^0 = Promedio\,(\boldsymbol{TAV})$.

2. Repetir hasta que se cumpla el/los criterios de terminación elegidos:

   a. Actualizar la velocidad de cada partícula:

   $$v_{ij}^{k+1} = w^k v_{ij}^k + c_1 r_{1_{ij}}\left(p_{ij}^k - x_{ij}^k\right) + c_2 r_{2_{ij}}\left(q_j - x_{ij}^k\right)$$

   b. Actualizar la posición de cada partícula:

   $$x_{ij}^{k+1} = x_{ij}^k + v_{ij}^{k+1}$$

   c. Calcular $g(I_i) = \sin(2\pi(I_i - a) \times b \times \cos(A)) + d$, donde A= $2\pi \times c(I_i - a)$.

   d. Determinar las variables binarias : $\begin{Bmatrix} Si\ g(I_i) > 0 \ \rightarrow y_i = 1 \\ Si\ g(I_i) \leq 0 \ \rightarrow y_i = 0 \end{Bmatrix} \forall\ i\ \in [1, n_b]$

   e. Evaluar el valor de las funciones objetivo $f_{N_{obj}}\left(x_i^{k+1}\right)$ y de $TAV_i\left(x_i^{k+1}\right)$.

   f. Determinar las soluciones no dominadas del óptimo de Pareto: S_i.

   g. Ordenar S_i según su dh en orden decreciente.

   h. Almacenar S_i en el archivo externo A.

   i. Evaluar el valor de relajación: $\mu^{k+1} = \mu^k\left(1 - \frac{F_F}{N}\right)$,

   j. Actualizar el mejor local: $p_{ij} = x_{ij}^{k+1}$.

   k. Actualizar el mejor global: $q_i = U(10\%A)$.

3. Frente Pareto = A.

Fig. 4. Pseudocódigo del PSO con variables continuas, binarias y multi-objetivo

4 Resultados

Como se comentó anteriormente, en [4] se testeó el PSO continuo con el tratamiento de restricciones basado en TAV, optimizando diversas funciones benchmark con diferentes números de ecuaciones, variables y no linealidades, lográndose identificar el óptimo global en la mayoría de los casos estudiados y buenas soluciones factibles en los restantes.

En este trabajo nos limitaremos el testeo del algoritmo en lo que respecta a la identificación de la frontera de Pareto. Para ello se tomaron ocho problemas benchmark pequeños, los cuales cuentan con dos o tres funciones objetivo y distintas restricciones, empleados típicamente en estudios de problemas multi-objetivo. En particular, el algoritmo se probará sobre cuatro problemas con dos funciones objetivo: MO1 [9] (Fig. 5), MO2 [10] (Fig. 6), MO3 [11] (Fig. 7) y MO4 [11] (Fig. 8); y otros cuatro con tres funciones objetivo: MO5 [12] (Fig. 9), MO6 [12] (Fig. 10), MO7 [13] (Fig. 11) y MO8 [13] (Fig. 12). Para todos ellos se presenta la correspondiente formulación y una comparación gráfica entre el frente de Pareto reportado en la literatura para el problema en cuestión, y el hallado por el PSO propio con el fin de establecer la calidad de la solución encontrada.

En todos los casos, se utilizaron los siguientes parámetros del PSO: cantidad de partículas (N) = 100, constante de aceleración cognitiva (c_1) = 1,5, constante de aceleración social (c_2) = 1,5, peso de inercia (w) = 0,75, iteraciones máximas ($k_{máx}$) = 5000 y la dimensión del archivo externo A (dim_A) = 100. Estos valores se tomaron de Damiani y col. (2020), quienes testearon el PSO con estos parámetros sobre diferentes funciones benchmark de distinta complejidad, reportando en todos los casos buenos rendimientos.

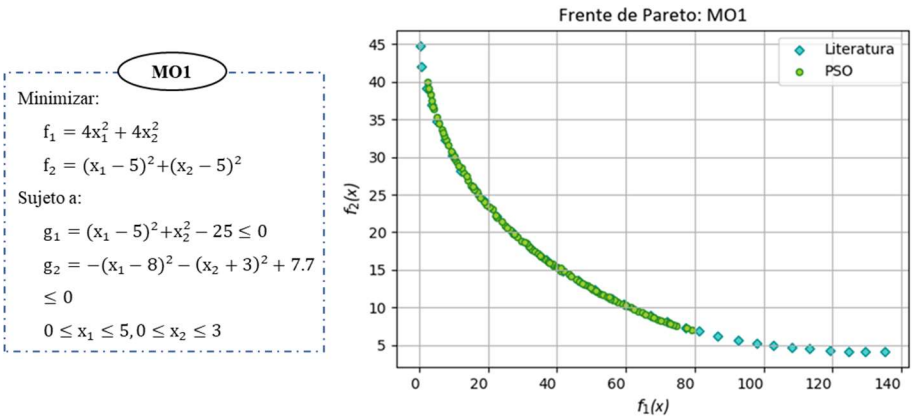


Fig. 5. Frente de Pareto MO1 literatura (rombos celestes) vs PSO propio (círculos verdes)

MO2

Minimizar:

$f_1 = 2 + (x_1 - 2)^2 + (x_2 - 1)^2$

$f_2 = 9x - (x_2 - 1)^2$

Sujeto a:

$g_1 = x_1^2 + x_2^2 - 225 \leq 0$

$g_2 = x_1 - 3x_2 + 10 \leq 0$

$-20 \leq x_1, x_2 \leq 20$

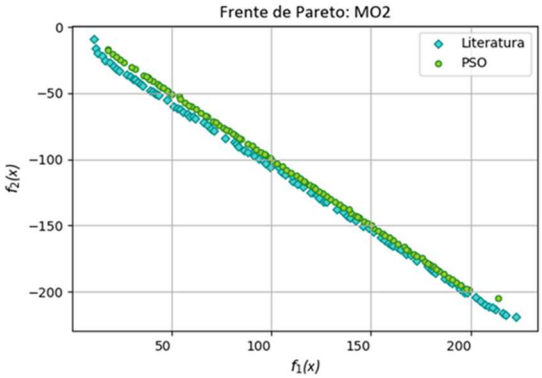


Fig. 6. Frente de Pareto MO2 literatura (rombos celestes) vs PSO propio (círculos verdes)

MO3

Minimizar:

$f_1 = x_1^2$

$f_2 = (x_1 - 2)^2$

Sujeto a:

$-100 \leq x_1 \leq 100$

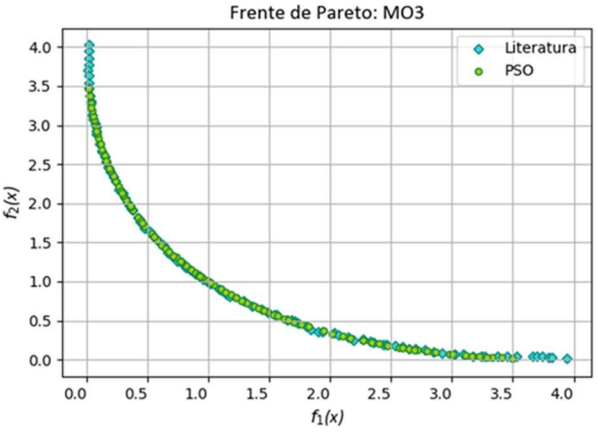


Fig. 7. Frente de Pareto MO3 literatura (rombos celestes) vs PSO propio (círculos verdes)

MO4

Minimizar:

$$f_1 = \begin{cases} -x_1 \ \ si\ x_1 \leq 1 \\ x_1 - 2 \ \ si\ 1 < x_1 \leq 3 \\ 4 - x_1 \ si\ 3 < x_1 \leq 4 \\ x_1 - 4 \ si\ x_1 > 4 \end{cases}$$

$f_2 = (x_1 - 5)^2$

Sujeto a:

$-5 \leq x_1 \leq 10$

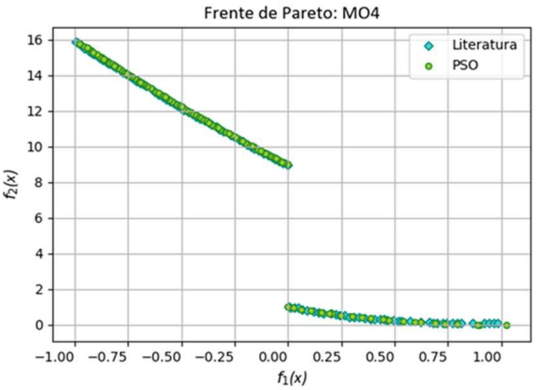


Fig. 8. Frente de Pareto MO4 literatura (rombos celestes) vs PSO propio (círculos verdes)

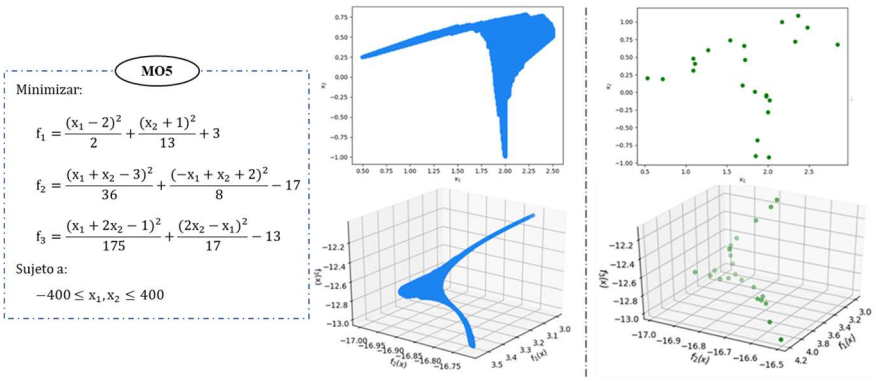


Fig. 9. Frente de Pareto MO5 literatura (izquierda) vs PSO propio (derecha)

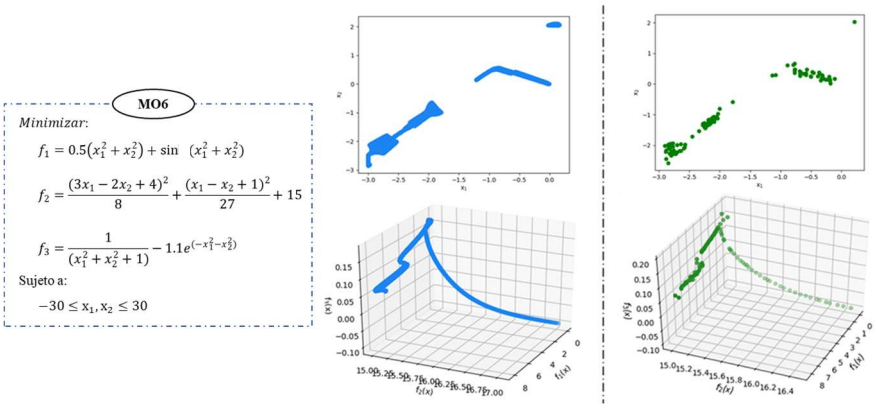


Fig. 10. Frente de Pareto MO6 literatura (izquierda) vs PSO propio (derecha)

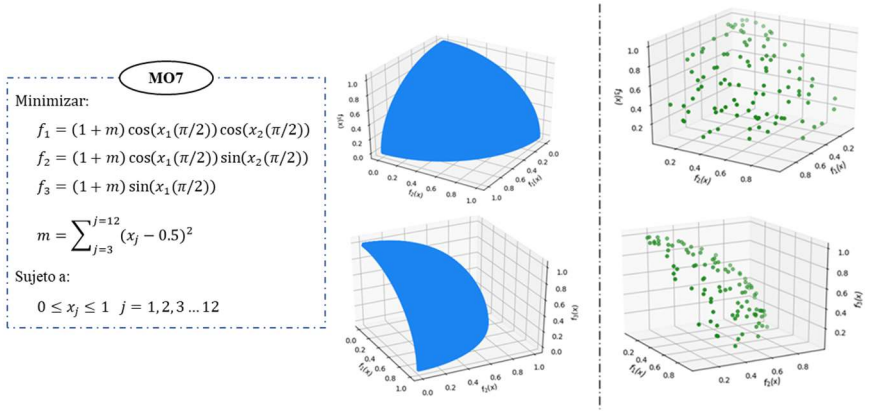


Fig. 11. Frente de Pareto MO7 literatura (izquierda) vs PSO propio (derecha)

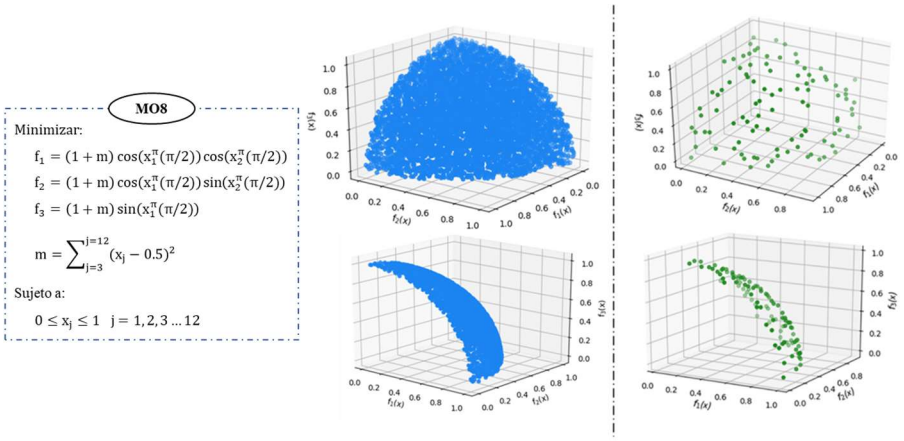


Fig. 12. Frente de Pareto MO8 literatura (izquierda) vs PSO propio (derecha)

Como puede observarse en las figuras anteriores, en todos los casos de testeo propuestos, los resultados obtenidos con la herramienta desarrollada son muy simulares a la solución reportada. Asimismo, cabe mencionar que todas las soluciones son factibles. Esto se concluye ya que los valores de TAV de cada frontera de Pareto (datos no mostrados) son siempre igual a cero, es decir, no se viola ninguna restricción. Los experimentos se realizaron en una computadora Intel(R) Core (TM) i7-3537U CPU @ 2.00GHz 2.50GHz con una memoria RAM de 8GB, demandando, en todos los casos, unos pocos minutos de cómputo para completar el número de iteraciones adoptado.

5 Conclusiones

En este trabajo se implementó un optimizador continúo basado en PSO. Para extender sus prestaciones, se incorporó al algoritmo básico una técnica para manipular restricciones, otra para tratar variables binarias y un método para optimizar multi-objetivos.

El algoritmo se utilizó en este trabajo para optimizar diferentes problemas benchmark de dos o tres funciones objetivo, con diferentes restricciones y características. Los resultados obtenidos indican que la herramienta demuestra un buen desempeño general, ya que se alcanzan soluciones factibles y similares a las reportadas en la literatura. Si bien los ejemplos seleccionados son de pequeña dimensión, permiten comprobar el buen funcionamiento de la implementación.

Por estas razones, sumado a evidencia generada en trabajos previos [4] [14], entendemos que el PSO MINLP-MO desarrollado cumple con el propósito de resolver problemas mixto enteros no-lineales multi-objetivo resultantes de aplicaciones de la ingeniería en sus diversas ramas. Es importante destacar que con este solver no se pretende competir con plataformas comerciales de modelamiento y optimización. Lo que se busca principalmente es disponer de un instrumento accesible para desarrollar proyectos de investigación y transferencia, que proporcione soluciones de aceptable eficiencia y no requiera presupuestos para adquisición y mantenimiento de licencias comerciales

de software lo que, en la práctica, suele resultar una restricción importante para transferir y desarrollar colaborativamente.

6 Referencias

1. Woon, S. F., & Rehbock, V. (2010). A critical review of discrete filled function methods in solving nonlinear discrete optimization problems. Applied mathematics and computation, 217(1), 25-41.
2. Kennedy, J., Eberhart, R. (1995). Particle swarm optimization (PSO). In Proc. IEEE International Conference on Neural Networks, Perth, Australia (pp. 1942-1948).
3. Zhang, H., & Rangaiah, G. P. (2012). An efficient constraint handling method with integrated differential evolution for numerical and engineering optimization. Computers & Chemical Engineering, 37, 74-88.
4. Damiani, L., Diaz, A. I., Iparraguirre, J., & Blanco, A. M. (2020). Accelerated particle swarm optimization with explicit consideration of model constraints. Cluster Computing, 23(1), 149-164.
5. Pampara, G., Franken, N., & Engelbrecht, A. P. (2005). Combining particle swarm optimisation with angle modulation to solve binary problems. In 2005 IEEE congress on evolutionary computation (Vol. 1, pp. 89-96). IEEE.
6. Ngatchou, P., Zarei, A., & El-Sharkawi, A. (2005). Pareto multi objective optimization. In Proceedings of the 13th International Conference on, Intelligent Systems Application to Power Systems (pp. 84-91). IEEE.
7. Selçuklu, S. B., Coit, D. W., & Felder, F. A. (2020). Pareto Uncertainty Index for Evaluating and Comparing Solutions for Stochastic Multiple Objective Problems. European Journal of Operational Research.
8. Raquel, C. R., & Naval Jr, P. C. (2005). An effective use of crowding distance in multiobjective particle swarm optimization. In Proceedings of the 7th annual conference on Genetic and evolutionary computation (pp. 257-264).
9. Binh, T. T., & Korn, U. (1997). Multiobjective evolution strategy for constrained optimization problems. In Proceedings of the 15th IMACS World Congress on Scientific Computation, Modelling and Applied Mathematics, Berlin (Vol. 357, p. 362).
10. Chankong, V., & Haimes, Y. Y. (2008). Multiobjective decision making: theory and methodology. Courier Dover Publications.
11. Schaffer, J. David (1984). Multiple Objective Optimization with Vector Evaluated Genetic Algorithms. Proceedings of the First Int. Conference on Genetic Algortihms, Ed. G.J.E Grefensette, J.J. Lawrence Erlbraum (PhD). Vanderbilt University.
12. Coello, C. A. C., Lamont, G. B., & Veldhuizen, D. A. V. (2007). MOEA Test Suites. Evolutionary Algorithms for Solving Multi-Objective Problems: Second Edition, 175-232.
13. Kaveh, A., & Mahdavi, V. R. (2019). Multi-objective colliding bodies optimization algorithm for design of trusses. Journal of Computational Design and Engineering, 6(1), 49-59.
14. Damiani, L. (2021). Planeamiento Óptimo del Manejo Integrado de Malezas. Tesis de Doctorado en Ingeniería. Universidad Nacional del Sur.