BREVE REVISIÓN SOBRE HEURÍSTICAS INSPIRADAS EN LA NATURALEZA PARA OPTIMIZACIÓN MONO Y MULTI-OBJETIVO

Michel Fernández[a], Teresa Pérez-Sosa[b], Ramón Quiza[c]

[a] *Centro Provincial de Ingeniería Clínica y Electromedicina, Matanzas*
Callejón de Quintanales Final, Matanzas 40100, Cuba, Email: michelfernandez.mtz@infomed.sld.cu

[b] *Centro de Estudio de Fabricación Avanzada y Sostenible (CEFAS), Universidad de Matanzas*
Autopista a Varadero km 3½, Matanzas 44740, Cuba, Email: teresa.perez@umcc.cu

[b] *Centro de Estudio de Fabricación Avanzada y Sostenible (CEFAS), Universidad de Matanzas*
Autopista a Varadero km 3½, Matanzas 44740, Cuba, Email: ramon.quiza@umcc.cu

Resumen: Las heurísticas inspiradas en la naturaleza juegan un papel crucial en la optimización actual, tanto para la optimización mono-objetivo como para la multi-objetivo, ya que ellas evitan los requisitos de continuidad, derivabilidad y unimodalidad, requeridos por otras técnicas, tales como los métodos de optimización analíticos o numéricos. Este trabajo presente una breve revisión de algunas heurísticas, recientemente publicadas, para ser usadas en optimización mono y multi-objetivo. Estas son resumidas teniendo en cuanta sus ventajas y limitaciones. También se incluyen sus respectivos diagramas de bloque. Como conclusión principal del trabajo se llega a que una gran variedad de heurísticas de optimización ha sido reportada en la literatura contemporánea y que, consecuentemente, su uso fiable debe ser precedido por una labor de implementación y comparación de su eficiencia para los problemas considerados.

Palabras claves: Heurística inspiradas en la naturaleza; Optimización.

Abstract: Nature inspired heuristics play a key role in current optimization for both single- and multi-optimization, as they avoid the requisites of continuity, differentiability and unimodality, required by other techniques, such as analytic or numeric optimization method. This work presents a brief review of some recently published heuristics to be used in single- and multi-objective optimization. They are summarized by take into account their advantages and shortcomings. Their respective block diagrams are also depicted. As a main conclusion of the work, arises the high variety of optimization heuristics which have been reported in up-to-date literature and, consequently, their reliable use should be preceded by a work of implementation and comparison on base of their performance for the considered problem.

Keywords: Nature inspired heuristics; Optimization.

1. Introducción.

La optimización, hoy en día, juega un papel clave no sólo como asunto teórico de las matemáticas, sino también como herramienta de aplicación práctica en las diversas ramas de la ingeniería [1].

Aplicada en diversos ámbitos, permite obtener productos, procesos o sistemas más efectivos y competitivos [2].

Un problema de optimización mono-objetivo es aquel que, sin pérdida de generalidad, consiste en minimizar la función escalar f: $\Omega \subset \mathbb{R}^n \rightarrow \mathbb{R}$:

$$f(x_1, \ldots, x_n) \; ; \tag{1}$$

donde las variables de decisión están definidas para los intervalos:

$$l_i \leq x_i \leq u_i, \quad i = 1, \ldots, n; \tag{2}$$

y están sujetas a las restricciones:

$$g(x_1, \ldots, x_n) \leq 0, \quad i = 1, \ldots, p \; ; \text{ y} \tag{3}$$

$$h(x_1, \ldots, x_n) = 0, \quad i = 1, \ldots, q \; . \tag{4}$$

donde n es la cantidad de variables de decisión, p es el número de restricciones de desigualdad y q el de restricciones de igualdad, mientras que l_i y u_i son los límites inferiores y superiores de las variables de decisión [3].

Por su parte, un problema de optimización multi-objetivo consiste en minimizar la función vectorial $\mathbf{f}$: $\Omega \subset \mathbb{R}^n \rightarrow \mathbb{R}^m$:

$$\mathbf{f}(\mathbf{x}) = \begin{bmatrix} f_1(x_1, \ldots, x_n) \\ \vdots \\ f_m(x_1, \ldots, x_n) \end{bmatrix}; \tag{5}$$

sujeta a restricciones similares a la de la optimización mono-objetivo [4].

A diferencia de la optimización mono-objetivo, en la multi-objetivo, no existe necesariamente un óptimo global, dada la no existencia de una relación de ordenamiento o comparación entre vectores. En su lugar, se utiliza el concepto de dominancia, el cual establece que un vector, $\mathbf{u} \in \mathbb{R}^m$, domina a otro vector, $\mathbf{v} \in \mathbb{R}^m$ (y se denota $\mathbf{u} \preceq \mathbf{v}$), si y sólo si, $\mathbf{u}$ no es peor que $\mathbf{v}$ en todos los objetivos ($u_i \leq v_i, \forall i \in \{1, \ldots, m\}$); y, además, es estrictamente mejor en, al menos, uno de los objetivos ($\exists k \in \{1, \ldots, m\} \mid u_i < v_i$) [5].

Basado en este concepto de dominancia, se dice que $\mathbf{u}^*$ es una solución no dominada, u óptimo de Pareto, si no existe, dentro del espacio de búsqueda considerado, ninguna otra solución que lo domine ($\neg\exists\, \mathbf{u}' \in \Omega \mid \mathbf{u}' \preceq \mathbf{u}^*$). Se llama, entonces, conjunto de Pareto, $\mathcal{P}$, al conjunto de todas las soluciones no dominadas para un problema dado ($\mathcal{P} := \{\mathbf{x} \in \Omega \mid \neg\exists\, \mathbf{x}' \in \Omega : \mathbf{x}' \preceq \mathbf{x}^*\}$), y frontera de Pareto, $\mathcal{F}$, a la imagen de dicho conjunto de Pareto ($\mathcal{F} := \mathbf{f}(\mathbf{x}) \mid \mathbf{x} \in \mathcal{P}$) [6]. Estas soluciones no dominadas son óptimas en el sentido más amplio de que ninguna otra, en el espacio de búsqueda considerado, es capaz de mejorar, con respecto a cualquiera de ellas, un objetivo, sin empeorar otro [7].

Para la solución de los problemas de optimización, especialmente aquellos que corresponden a situaciones reales de la práctica industrial, se utilizan las llamadas heurísticas inspiradas en la

naturaleza. Estas técnicas, tienen un grupo de ventajas sobre las herramientas numéricas tradicionales, entre las que se destacan [8]:

- Son capaces de realizar búsquedas en todo el espacio de variables, llegando al óptimo global, es decir, son capaces de resolver problemas multimodales.
- Son robustos, es decir, no dependen de la naturaleza específica del problema tratado.
- No requieren los gradientes de las funciones objetivos, por lo que son capaces de resolver problemas altamente no lineales, con espacios de búsqueda discontinuos o no derivables.
- Tienen una componente estocástica, lo que amplía la diversidad de las soluciones obtenidas.

En el presente trabajo se realiza una revisión bibliográfica de algunas de las heurísticas inspirada en la naturaleza más populares reflejadas en la literatura contemporánea.

2. Heurísticas para optimización mono-objetivo.

2.1. Algoritmo genético

Los algoritmos genéticos (*genetic algorithms*, GA) son una de las heurísticas más populares para resolver problemas de optimización. Al igual que otros algoritmos evolutivos, están inspirados en la evolución de las especies biológicas. Sin embargo, la principal característica distintiva de los GA es la codificación de la información de cada solución individual en una cadena (llamada cromosoma) [9]. Esta codificación hace que el algoritmo sea independiente del problema, lo que le confiere robustez.

Los GA utilizan un conjunto de soluciones (conocidas como población) que se evalúan y procesan en paralelo. Se crea una población aleatoria al inicio del algoritmo (ver Pseudocódigo 1) y luego se evalúa mediante el uso de la función objetivo predefinida. Posteriormente, se aplican los operadores que realizan la selección, el cruzamiento y la mutación, que permiten obtener una nueva población a partir de la actual. Este proceso se repite hasta lograr cualquiera de las condiciones de parada (principalmente, que se alcance un número de iteraciones predefinidas, o un nivel de convergencia dado en las soluciones). Todos los operadores incorporan algún comportamiento aleatorio, lo que garantiza la capacidad de encontrar la solución óptima global incluso cuando no está incluida en la población inicial [10].

Pseudocódigo 1. Algoritmo genético

```
INICIO ga
   crear una población inicial de forma aleatoria
   MIENTRAS alcance la condición de parada
      evaluar la población
      crear la nueva población (selección + cruzamiento + mutación)
   FIN MIENTRAS
FIN ga
```

Se han propuesto varios enfoques para cada operador. La selección puede llevarse a cabo, entre otros, mediante el método de la ruleta, el de torneo o el de clasificación. Los enfoques principales para el entrecruzamiento son el de punto único, el multipunto, el uniforme, el semi-uniforme, el

parcialmente emparejados y el basado en heurística. Finalmente, los método de mutación más usado son el uniforme, el no uniforme, el gaussiano y el supervisado [3].

Algunos conceptos como el elitismo, que garantiza la supervivencia de las mejores soluciones en la próxima población, también se han incorporado para mejorar el desempeño de los GA [11].

2.2. Recocido simulado

El recocido simulado (*simulated annealing*, SA) es otra popular heurística de optimización. Se basa en el proceso de enfriamiento de los metales [12]. En el proceso de recocido metalúrgico, el enfriamiento lento tiene como objetivo obtener un estado de energía mínima global en el metal, lo que proporciona un estado estructural estable y evita los estados meta-estables con mayor energía. De manera similar, el método de recocido simulado se enfoca en alcanzar el óptimo global de una función matemática, evitando los óptimos locales [13].

El SA trabaja con un único punto de solución, que se selecciona aleatoriamente. También se inicializa un parámetro del algoritmo, llamado temperatura, que determina la probabilidad de moverse de un estado a otro (ver Pseudocódigo 2).

Pseudocódigo 2. Recocido simulado

```
INICIO sa
   elegir una solución inicial aleatoria y una temperatura inicial
   MIENTRAS alcance la condición de fin de ciclo
      crear una solución a través de una distribución gaussiana
      SI (nueva solución es mejor que la anterior)
         aceptar nueva solución
      SINO SI (la probabilidad asociada es mayor que un valor elegido al azar)
         aceptar nueva solución
      SINO
         mantener solución anterior
      FIN SI
      actualizar con la mejor solución y la temperatura
   FIN MIENTRAS
END sa
```

El proceso de optimización se lleva a cabo en un ciclo, que finaliza cuando se logra alguna de las condiciones de parada: generalmente, cuando el parámetro de temperatura alcanza algún valor prescrito o después de un número máximo de iteraciones. En cada iteración, se intenta una nueva solución, que se mueve, desde la existente, una distancia aleatoria con una distribución gaussiana. Si la nueva solución es mejor (es decir, tiene un mejor valor de función objetivo), se acepta. Por el contrario, si es peor, la probabilidad, $p = \exp(-\Delta f/T)$, se calcula y se compara con un número aleatorio, r. Si $p > r$, la nueva solución es aceptada (incluso siendo peor que la anterior). De lo contrario, se conserva la vieja solución. En cada iteración, el valor óptimo global obtenido en todo el proceso se actualiza (si corresponde). El valor de la temperatura también se actualiza [2].

2.3 Optimización de enjambre de partículas

La optimización de enjambre de partículas (*particle swarm optimization*, PSO) está inspirada en el comportamiento cooperativo de algunas especies animales, como las aves y los peces. A pesar

de algunas deficiencias, como la convergencia a los óptimos locales y las limitaciones relacionadas con la invariabilidad de la transformación [14], el PSO es una heurística de optimización muy bien reputada, que se ha aplicado ampliamente para resolver problemas prácticos [3].

En la PSO, un conjunto de soluciones se mueve en el espacio de la variable de decisión, actualizando su posición a través de sus respectivos parámetros de velocidad (ver Pseudocódigo 3). Por otro lado, la velocidad depende de tres factores: la tendencia de una partícula a mantener su velocidad (comportamiento inercial), la atracción de la partícula a su mejor posición particular lograda (comportamiento cognitivo) y la atracción a la mejor posición general alcanzada por toda la población (comportamiento social) [15].

Pseudocódigo 3. Optimización de enjambre de partículas

```
INICIO pso
   generar una posición y velocidad aleatoria
   MIENTRAS alcance las condiciones de parada
      evaluar población
      elegir la mejor posición para cada partícula y la mejor posición global
      actualizar velocidad
      actualizar posición
   FIN MIENTRAS
FIN pso
```

2.4. Entropía cruzada

El método de entropía cruzada (*cross-entropy method*, CE) es una heurística basada en población, que resuelve los problemas de optimización al transformarlos en problemas estocásticos asociados con eventos con una probabilidad muy pequeña, utilizando alguna técnica de minimización de la varianza [16]. La base de la CE es la construcción de una secuencia aleatoria de soluciones que converge probabilísticamente a una solución óptima o casi óptima [17].

Pseudocódigo 4. Método de entropía cruzada.

```
INICIO ce
   inicializar media y varianza
   MIENTRAS condición de parada sea alcanzada
      recalcular media y varianza a partir de la población élite
      generar una población de trabajo aleatoria y Gaussian (media y varianza)
      evaluar la población de trabajo
      extraer la población elite de la población de trabajo
   FIN MIENTRAS
FIN ce
```

El algoritmo de CE [18] comienza con la inicialización de la media y la varianza de la distribución que debe utilizarse para generar la población activa. Esta inicialización tiene un componente estocástico. Después de eso, se realiza un bucle hasta alcanzar las condiciones de parada consideradas, ya sea al alcanzar el número máximo de iteraciones o al obtener una convergencia de las soluciones. En cada iteración, la media y la varianza se actualizan a partir de la llamada población de élite, que está compuesta por los mejores individuos de la población trabajadora (ver Pseudocódigo 4).

3. Heurísticas para optimización multi-objetivo

3.1 Algoritmo genético por ordenamiento no dominado

El algoritmo genético por ordenamiento no nominado (*nondominated sorting genetic algorithm*, NSGA) es una de las heurísticas más populares para optimización multi-objetivo. Propuesto en 1994 por Srinivas y Deb [19], ha sido mejorado en dos ocasiones, dando lugar al NSGA-II, en 2002 [20], y al NSGA-III, en 2014 [21].

El NSGA-III (ver Pseudocódigo 5) se basa en un ciclo de N iteraciones, en cada una de las cuales, se crea una nueva población, Qt, a partir de la población actual, Pt, aplicando los operadores de selección, cruzamiento y mutación. Luego, se unen ambas poblaciones y se ordenan, según el criterio de no dominancia. Finalmente, se crea una población St, adicionando las diferentes fronteras de dominancia, de menor a mayor, hasta alcanzar o superar el tamaño de población Z. Si la cantidad de elementos en St exceden a Z, entonces de la última capa adicionada, se seleccionan los K = Z – |St| elementos según un algoritmo de nicho.

Pseudocódigo 5. Algoritmo genético por ordenamiento no dominado - III

```
INICIO nsga3
   Crear población inicial aleatoria
   PARA t = 1 … N
      generar población hija mediante selección, cruzamiento y mutación
      unir ambas poblaciones y ordenarlas por no dominancia
      extraer nueva población hija, a partir de las mejores soluciones
   FIN MIENTRAS
FIN nsga3
```

Una característica muy importante, en esta heurística es el ordenamiento por dominancia, que es de orden $O(mZ^2)$. También se destaca el algoritmo de nicho, que se basa en la normalización sobre un plano de referencia en el espacio de objetivos.

3.2. Optimización multi-objetivo por enjambre de partículas

La optimización multi-objetivo por enjambre de partícula (*multi-objective particle swarm optimization*, MOPSO), mantiene muchas de las características del esta heurísticas para problemas mono-objetivos [22]. La misma (ver Pseudocódigo 6) se basa en el uso de una población de partículas y un repositorio donde se almacenan las soluciones no dominadas. Dicho repositorio tiene dos componentes principales: el controlador de repositorio, cuya función es decidir cuándo cierta solución debe ser adicionada al repositorio o no, y la rejilla, que se encarga de producir fronteras de Pareto bien distribuidas.

A través de un bucle de *N* iteraciones, el algoritmo va actualizando las posiciones y velocidades de cada partícula, para lo cual utiliza no sólo su posición actual sino la mejor posición previamente alcanzada y la del repositorio. También, en cada iteración, se actualiza el repositorio agregando las soluciones no dominadas. Luego de cada adición, las soluciones no dominadas son eliminadas del repositorio. Si aun así excede el tamaño prestablecido, se aplica el criterio de priorizar aquellas soluciones localizadas en las áreas menos pobladas.

Pseudocódigo 6. Optimización multi-objetivo por enjambre de partículas

```
INICIO mopso
   inicializar la población
   inicializar las velocidades de cada partícula
   evaluar cada partícula
   almacenar las partículas no dominadas en el repositorio
   generar hipercubos en el espacio de búsqueda y localizar las partículas
   inicializar la memoria de cada partícula
   PARA t = 1 … N
      calcular la nueva velocidad de cada partícula
      calcular la nueva posición de cada partícula
      mantener las variables dentro de sus límites prestablecidos
      evaluar cada partícula
      actualizar el contenido del repositorio
      actualizar las mejores posiciones de cada partícula
   FIN PARA
FIN mopso
```

3.3. Entropía cruzada simple para optimización multiobjetivo

La entropía cruzada simple para optimización multiobjetivo (*simple multi-objective cross-entropy*, SMOCE) [23], se basa en un desarrollo de la propuesta de Beruvides et al. [24], la cual, a su vez, está inspirada en el método de entropía cruzada para optimización multi-objetivo de Bekker y Aldrich [25].

Pseudocódigo 7. Entropía cruzada simple para optimización multiobjetivo

```
INICIO smoce
   crear una población élite vacía
   PARA t = 1 … N
      SI (población de élite vacía)
         crear una población de trabajo inicial aleatoria
      SINO
         crear una población de trabajo a partir de la de élite
      FIN SI
      evaluar y clasificar la población de trabajo
      actualizar la población de élite a partir de la de trabajo
   FIN PARA
   extraer el conjunto de Pareto
FIN smoce
```

El núcleo de SMOCE es la población de trabajo, compuesta por Z soluciones y sus respectivas funciones objetivos (ver Pseudocódigo 7). El algoritmo comienza creando una población de élite vacía. El proceso evolutivo ocurre en un único bucle de t iteraciones. En cada uno de ellos, se crea una nueva población de trabajo a partir de la población de élite. Si ésta está vacía (lo que ocurre en la primera iteración), la población de trabajo se crea, aleatoriamente, con una distribución uniforme restringida a los límites inferior y superior de cada variable de decisión. En caso contrario, la población de élite se agrupa en mD clases (donde m es la cantidad de objetivos y D, el número de intervalos en que se divide cada uno de ellos), dadas por igual número de cuboides homogéneos, en los que se divide el espacio de objetivos. A partir de cada clase se crea una cantidad de soluciones para la nueva población de trabajo, proporcional a la cantidad de

elementos en cada clase, y utilizando una distribución gaussiana acotada, con media y varianza calculadas a partir de dichos elementos.

La evaluación de la población incluye la penalización, con un factor Γ dado, para el incumplimiento de las restricciones, sean de desigualdad o de igualdad. Una vez evaluadas, las soluciones son ordenadas según el criterio de dominancia y, a partir de ellas, se incluyen las ⌊αZ⌋ ($0 < \alpha < 1$) mejores soluciones. Una vez adicionadas, la población de élite es filtrada para dejar sólo las no dominadas y, en el caso de aún superar el número ⌊αZ⌋, aplicar un criterio de aglomeramiento basado en el hipervolumen. Una vez concluido el bucle, se extrae el conjunto de Pareto de la población de élite.

5. Conclusiones.

Las heurísticas incluidas en la revisión, abarcan tanto la optimización mono-objetivo como la multi-objetivo. En ambos casos, se mostraron las principales características de las mismas. La revisión debe ser enriquecida, en un futuro, adicionando tanto otras heurísticas clásicas de amplia popularidad, como nuevos algoritmos emergentes. Dicha revisión debe ser la base para la implementación de una librería de clases, en software libre, que permita su uso para la solución de problemas de ingeniería.

Referencias.

[1] X.-S. Yang and S. Koziel, Eds., *Computational optimization and applications in engineering and industry* (Studies in Computational Intelligence 359). Berlin (Germany): Springer-Verlag, 2011, p.^pp. Pages.

[2] X. S. Yang, S. Koziel, and L. Leifsson, "Computational optimization, modelling and simulation: Past, present and future," *Procedia Computer Science,* vol. 29, pp. 754-758, 2014.

[3] S. Mirjalili, *Evolutionary algorithms and neural networks: Theory and applications*. Cham (Switzerland): Springer, 2019.

[4] C. A. Coello, G. B. Lamont, and D. A. Veldhuizen, *Evolutionary algorithms for solving multi-objective problems*, 2nd ed. New York: Springer, 2007.

[5] S. Deshpande, L. T. Watson, and R. A. Canfield, "Pareto front approximation using a hybrid approach," *Procedia Computer Science,* vol. 18, pp. 521-530, 2013.

[6] S. Mirjalili, S. Saremi, S. M. Mirjalili, and L. S. Coelho, "Multi-objective grey wolf optimizer: A novel algorithm for multi-criterion optimization," *Expert Systems with Applications,* vol. 47, pp. 106-119, 2016/04/01/ 2016.

[7] H. A. Abbass, R. Sarker, and C. Newton, "PDE: A Pareto–frontier differential evolution approach for multi objective optimization problems," in *Congress on Evolutionary Computation*, Piscataway, NJ (U.S.A.), 2001, pp. 971-978.

[8] X.-S. Yang, Ed., *Nature-inspired algorithms and applied optimization* (Studies in Computational Intelligence 744). 2018, p.^pp. Pages.

[9] P. Punia and M. Kaur, "Various genetic approaches for solving single and multi-objective optimization problems: A review," *International Journal of Advanced Research in Computer Science and Software Engineering,* vol. 3, pp. 1014-1020, 2013.

[10] M. Affenzeller, S. Wagner, S. Winkler, and A. Beham, *Genetic algorithms and genetic programming: Modern concepts and practical applications*. Boca Raton, FL (USA): Chapman and Hall/CRC 2018.

[11] J. A. Rojas Cruz and A. G. C. Pereira, "The elitist non-homogeneous genetic algorithm: Almost sure convergence," *Statistics and Probability Letters,* vol. 83, pp. 2179-2185, 2013/10/01/ 2013.

[12] R. E. Haber, R. Haber-Haber, A. Jiménez, and R. Galán, "An optimal fuzzy control system in a network environment based on simulated annealing. An application to a drilling process," *Applied Soft Computing,* vol. 9, pp. 889-895, 2009/06/01/ 2009.

[13] P. Siarry, Ed., *Metaheuristics*. Cham (Switzerland): Springer, 2016, p.^pp. Pages.

[14] M. R. Bonyadi and Z. Michalewicz, "Particle swarm optimization for single objective continuous space problems: A review," *Evolutionary Computation,* vol. 25, pp. 1-54, 2017.

[15] S. Fidanova, Ed., *Recent advances in computational optimization (Results of the Workshop on Computational Optimization WCO 2016)* (Studies in Computational Intelligence 717). Cham (Switzerland): Sprimger, 2018, p.^pp. Pages.

[16] R. E. Haber, R. M. Del Toro, and A. Gajate, "Optimal fuzzy control system using the cross-entropy method. A case study of a drilling process," *Information Sciences,* vol. 180, pp. 2777-2792, 2010.

[17] G. Beruvides, F. Castaño, R. E. Haber, R. Quiza, and A. Villalonga, "Coping with complexity when predicting surface roughness in milling processes: Hybrid incremental model with optimal parametrization," *Complexity,* 2017.

[18] P. T. De Boer, D. P. Kroese, S. Mannor, and R. Y. Rubinstein, "A Tutorial on the cross-entropy method," *Annals of Operations Research,* vol. 134, pp. 19-67, 2005.

[19] N. Srinivas and K. Deb, "Muiltiobjective optimization using nondominated sorting in genetic algorithms," *Evolutionary Computation,* vol. 2, pp. 221-248, 1994.

[20] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation,* vol. 6, pp. 182-197, 2002.

[21] K. Deb and J. Jain, "An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, Part I: Solving problems with box constraints," *IEEE Transactions on Evolutionary Computation,* vol. 18, pp. 577-601, 2014.

[22] C. A. C. Coello, G. T. Pulido, and M. S. Lechuga, "Handling multiple objectives with particle swarm optimization," *IEEE Transactions on Evolutionary Computation,* vol. 8, pp. 256-279, 2004.

[23] R. E. Haber, G. Beruvides, R. Quiza, and A. Hernández, "A simple multi-objective optimization based on the cross-entropy method," *IEEE Access,* vol. 5, pp. 22272-22281, 2017.

[24] G. Beruvides, R. Quiza, and R. E. Haber, "Multiobjetive optimization based on an improved cross-entropy method. A case study of a micro-scale manufacturing process," *Information Sciences,* vol. 334-335, pp. 161-173, 2016.

[25] J. Bekker and C. Aldrich, "The cross-entropy method in multi-objective optimisation: An assessment," *European Journal of Operational Research,* vol. 211, pp. 112-121, 2011.