REVISIÓN CONCISA SOBRE HEURÍSTICAS PARA OPTIMIZACIÓN MONO-OBJETIVO

Ing. Michel Fernández González[1], Dr.C. Ramón Quiza Sardiñas[2]

1. Estudiante de Maestría en Ingeniería Asistida por Computadora, Universidad de Matanzas
Autopista a Varadero km 3½, Matanzas 44740, Cuba.
Email: michelbarza2017@gmail.com

2. Centro de Estudio de Fabricación Avanzada y Sostenible (CEFAS), Universidad de Matanzas
Autopista a Varadero km 3½, Matanzas 44740, Cuba.
Email: ramon.quiza@umcu.cu

Resumen

El objetivo que se propone esta revisión concisa de algunos de los métodos de heurística propuestos en esta monografía es precisamente mostrar técnicas de avanzadas usadas en la solución de problemas de optimización multiobjetivo. Demostrando en las técnicas propuestas la viabilidad de cada propuesta gracias a su fácil implementación y la alta fiabilidad de los resultados propuestos, teniendo como debilidad el costo computacional de las técnicas producto a la ejecución de la cantidad de ciclos que se realizan en las mismas. Cada técnica está inspirada en la observación de comportamientos estudiado por parte de los científicos en la naturaleza. Cada uno de estos métodos ha demostrado su eficiencia y su eficacia, lo que los hace novedoso y resolutivos.

***Palabras claves**: Heurística; Algoritmo; Optimización*

CD Monografías 2018

ISBN: 978-959-16-4235-6

1. Introducción

Las técnicas de optimización han alcanzado una enorme relevancia no sólo en la ciencia sino también en la industria contemporánea (Haber *et al*. 2017). La aplicación de estas herramientas permite mejorar notablemente los procesos productivos mejorando su eficiencia técnica y económica (Beruvides *et al*. 2016).

Las heurísticas de optimización son técnicas no basadas en gradiente, inspiradas en procesos y sistemas naturales, tanto biológicos como físicos. Sus características permiten resolver problemas de optimización cuyas funciones objetivos y restricciones no cumplan los requisitos de continuidad, suavidad y unimodalidad, exigidos por otras técnicas matemáticas, tales como las analíticas y las numéricas (Xing y Gao 2014). El presente trabajo realiza una revisión bibliográfica concisa sobre algunas de las heurísticas de optimización monobjetivo más recientes.

2. Desarrollo

2.1 Algoritmo Genético

Los algoritmos genéticos son una de las heurísticas sin gradiente más populares para resolver problemas de optimización. 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 cada información de solución individual en una cadena (llamada cromosoma) (Punia y Kaur 2013). Esta codificación hace que el algoritmo sea independiente del problema, por lo que pueden considerarse de naturaleza robusta.

Pseudocódigo 1: Algoritmo Genético General

```
BEGIN ag
   crea una población inicial de forma aleatoria
   WHILE si alcanza la condición de parada
      evalúa población
      crear la nueva población (selección + entrecruzamiento + mutación)
   END WHILE
END ag
```

Los AG 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 de aptitud predefinida (t. objetivo, meta o función objetivo). Posteriormente, se solicita a los operadores que realicen la selección, entrecruzamiento y mutación, que permitan obtener una nueva población de la actual. Este proceso se repite hasta lograr cualquiera de las condiciones de parada. Todos los operadores incorporan algún comportamiento aleatorio, lo que garantiza la capacidad de

ISBN: 978-959-16-4235-6

encontrar la solución óptima global incluso cuando no está incluida en la población inicial (Affenzeller et al. 2009).

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 la clasificación. Los enfoques principales para el entrecruzamiento son de punto único, multipunto, uniforme, semi-uniforme, parcialmente emparejados y basados en heurística. Finalmente, el método de mutación más usado es el uniforme, no uniforme, gaussiano y supervisado (Mirjalili 2019).

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 la AG (Rojas Cruz et al. 2013).

2.2 Recocido Simulado

El recocido simulado (RS) es otra heurística popular de optimización sin gradientes. Se basa en el proceso de enfriamiento de metales (Haber et al. 2009). En el proceso de recocido metalúrgico, el enfriamiento lento tiene como objetivo obtener un estado de energía mínima global en un 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 (Siarry 2016).

RS 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

```
BEGIN rs
   elige una solución inicial aleatoria, una temperatura inicial
   WHILE alcance la condición de fin de ciclo
      crear una solución a través de una distribución gaussiana
      IF (nueva solución es mejor que la anterior)
         aceptar nueva solución
      ELSE IF (la probabilidad asociada es mayor que el valor elegido al azar)
         aceptar nueva solución
      ELSE
         mantener solución anterior
      END
      actualizar con la mejor solución y temperatura
   END WHILE
END rs
```


ISBN: 978-959-16-4235-6

El proceso de optimización se lleva a cabo en un ciclo, que finaliza cuando se logran algunas condiciones: generalmente, cuando el parámetro de temperatura alcanza algún valor prescrito y después de un número máximo de iteración. 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 de aptitud), se acepta. Por el contrario, si es peor, la probabilidad, p = exp (□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 (Yang 2014).

2.3 Optimización de Enjambre de Partículas

La optimización del enjambre de partículas (OEP) 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 (Bonyadi y Michalewicz 2017), el enjambre de partículas OEP es una heurística de optimización muy bien reputada, que se ha aplicado ampliamente para resolver problemas prácticos (Mirjalili 2019).

Pseudocódigo 3: Optimización de Enjambre de Partículas

```
BEGIN oep
   generar una posición y velocidad aleatoria
   WHILE alcance las condiciones de parada
      evaluar población
      elegir la mejor posición para cada partícula y la mejor posición global
      actualizar velocidad
      actualizar posición
   END WHILE
END oep
```

En la OEP, 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 particulasr lograda (comportamiento cognitivo) y la atracción a la mejor posición general alcanzada por toda la población (comportamiento social) (Krzeszowski et al. 2018).

ISBN: 978-959-16-4235-6

2.4 Entropía Cruzada

El método de entropía cruzada (EC) es una heurística basada en la población que resuelve los problemas de optimización al transformarlos en problemas estocásticos asociados con una probabilidad muy pequeña utilizando alguna técnica de minimización de la varianza (Haber et al. 2010). La base de la EC es la construcción de una secuencia aleatoria de soluciones que converge probabilísticamente a una solución óptima o casi óptima (Beruvides et al. 2017).

El algoritmo EC (De Boer et al., 2005) 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).

Pseudocódigo 4: Método de Entropía Cruzada.

```
BEGIN ec
   inicializar media y varianza
   WHILE 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
   END WHILE
END ec
```

2.5 Optimizador de Lobo Gris

El optimizador de lobos grises (OLG) es una heurística basada en la población, inspirada en el comportamiento de las manadas de lobos grises. La característica más notable de este algoritmo es la división de las soluciones en varias clases de dominación (alfa, beta, delta y omega), que imitan la jerarquía social en las manadas de lobos.

ISBN: 978-959-16-4235-6

Pseudocódigo 5: Algoritmo de Lobo Gris.

```
BEGIN olg
   crear una posición inicial aleatoria
   evaluar la población en la posición dada
   seleccionar las tres mejores soluciones (α, β y δ)
   WHILE alcance la condición de parada
        actualizar la posición de cada solución
        evaluar la población en la nueva posición
        actualizar los parámetros del algoritmo
        actualizar las tres mejores soluciones (α, β y δ)
   END WHILE
END olg
```

En el algoritmo OLG (ver Pseudocódigo 5), una población inicial de soluciones (posiciones) se crean y evalúan al azar. Las tres posiciones con mejor condición de aptitud se seleccionan como alfa, beta y delta, respectivamente. Luego, se produce un bucle hasta alcanzar un número de iteraciones prescritas (condición de parada). En cada iteración, las posiciones se actualizan considerando las posiciones de los individuos alfa, beta y delta. Después de eso, se evalúan y se seleccionan nuevas posiciones alfa, beta y delta (Mirjalili et al. 2014).

2.6 Algoritmo de Optimización de Ballena

El algoritmo de optimización de ballenas (AOB) es otro algoritmo estocástico basado en la población. AOB está inspirada en el método de caza de las ballenas jorobadas, que se conoce como método de alimentación con red de burbujas. Este comportamiento de forrajeo se basa en rodear la mancha de kril o peces pequeños cerca de la superficie, expulsando burbujas de aire en una trayectoria en espiral (Mirjalili y Lewis 2016). El AOB (ver Pseudocódigo 6) se basa en crear una población aleatoria inicial de agentes de búsqueda. Luego de evaluado, entonces se elige el mejor agente de búsqueda. Luego, se realiza un bucle hasta alcanzar un número de iteración máximo prescrito (condición de parada). En cada iteración, la posición de cada agente de búsqueda se actualiza, con la misma probabilidad, ya sea hacia el mejor agente de búsqueda (encogiendo el mecanismo del círculo) o siguiendo el movimiento en forma de espiral, que es característico de las ballenas jorobadas. Además, en el mecanismo de encogimiento, si el parámetro A está fuera del intervalo [–1, 1], la posición del agente de búsqueda se actualiza hacia un agente de búsqueda creado aleatoriamente en lugar del mejor agente de búsqueda elegido previamente. Después de completar la actualización de la población de agentes de búsqueda, se evalúan y el mejor agente de búsqueda también se actualiza si hay una mejor solución en la nueva población (Mafarja y Mirjalili 2017).

ISBN: 978-959-16-4235-6

Pseudocódico 6: Algoritmo de Optimización de Ballena

```
BEGIN aob
   crear una población aleatoria inicial de agentes
   evaluar la población
   seleccionar el mejor agente
   WHILE alcance la condición de parada
      FOR EACH agente_de_busqueda
         actualizar parámetros del algoritmo (a, A, C)
         generar valores aleatorios p, l
         IF (p < 0.5)
            IF (|A| < 1)
               actualizar agentes hacia el mejor agente de búsqueda
            ELSE
               actualizar posición del agente hacia el agente aleatorio
            END IF
         ELSE
            actualizar el agente siguiendo en la forma del movimiento espiral
         END IF
      END FOR
      evaluar la población
      actualizar con el mejor agente
   END WHILE
END WOA
```

2.7 Algoritmo de Optimización de Saltamontes

El algoritmo de optimización saltamontes (AOS) se basa en el comportamiento de los enjambres de saltamontes. En AOS, el proceso de optimización se lleva a cabo en dos etapas: una fase de exploración donde se identifican las regiones más prometedoras, y una fase de explotación donde se encuentra una solución precisa y casi óptima. Este comportamiento está modelado por el parámetro c, que disminuye a través del proceso de optimización (Saremi et al. 2017).

El algoritmo AOS se resume en el Pseudocódigo7. En primer lugar, se crea una población de saltamontes, que establece sus posiciones aleatoriamente a través del espacio de búsqueda. Después de evaluar cada saltamontes, se selecciona el mejor. Luego, se lleva a cabo un ciclo hasta alcanzar un número de iteración prescrito (condición de parada). En cada iteración, el parámetro c se actualiza y, para cada saltamontes en el enjambre, la distancia a todos los demás saltamontes se normaliza en el intervalo [1, 4] y la posición se actualiza teniendo en cuenta no solo estas distancias (interacción social) sino también la dirección del viento, que se asume que es hacia el objetivo (es decir, hacia la mejor posición del saltamontes). Finalmente, se evalúa a toda la población y se actualiza el mejor saltamontes si se genera una mejor solución.

ISBN: 978-959-16-4235-6

Pseudocódigo 7: Algoritmo de Optimización de Saltamontes

```
BEGIN aos
   crear de manera aleatoria un enjambre inicial
   evaluar la población
   seleccionar el mejor saltamontes
   WHILE alcance la condición de parada
      actualizar el parámetro c
      FOR EACH saltamontes
         normalizar las distancias entre saltamontes en el intervalo [1, 4]
         actualizar la posición del saltamontes
         IF (nueva posición fuera de los limites)
            regresar al saltamontes anterior
         END IF
      END FOR
      evaluar la población
      actualizar con el mejor saltamontes
   END WHILE
END aos
```

2.8 Algoritmo de Enjambre de Salpas

El algoritmo de enjambre de salpas (AES) está inspirado en el comportamiento de las salpas, que se caracterizan por formar colonias con forma de cadena. La AES se basa en la clasificación de las soluciones (salpas) en función de su aptitud. La mejor solución es el líder de la cadena y su posición se actualiza considerando solo la fuente de alimento (idealmente, es la solución; sin embargo, como se desconoce, se usa la solución mejor lograda). Las otras soluciones se consideran como seguidoras y su posición se actualiza teniendo en cuenta la solución de la salpa anterior en la cadena (Mirjalili et al 2017). El algoritmo se muestra en el Pseudocódigo 8. Después de crear un enjambre inicial de salpas aleatorio, se ejecuta un bucle hasta alcanzar la condición de parada (número de iteraciones máximas prescritas). En cada iteración, la población de salpas es evaluada y clasificada. Luego, las posiciones se actualizan como se explicó anteriormente. Las nuevas posiciones calculadas se modifican si están fuera de los límites. El algoritmo depende de un solo parámetro, c_1, lo que lo hace muy simple y fácil de implementar

ISBN: 978-959-16-4235-6

Pseudocódigo 8: Algoritmo de Enjambre de Salpa

```
BEGIN ssa
   crear un enjambre inicial de salpas de manera aleatoria
   WHILE alcance condición de parada
      evaluar la población
      ordenar la población
      seleccionar la mejor salpa
      actualizar el parámetro c1
      FOR EACH salpa
         IF (salpa es la mejor)
            actualizar posición considerando solo la Fuente de alimentación
         ELSE
            actualizar la pos. considerando solo la pos. de la salpa anterior
         END IF
         IF (nueva posición fuera de los límites)
            corregir la posición de la salpa
         END IF
      END FOR
   END WHILE
END ssa
```

3. Conclusiones

Las heurísticas propuestas en la monografía ponen de manifiesto la inventiva científica moderna la cual es capaz de aplicar conocimiento tomado de la observación de fenómenos naturales y adaptarlos a las necesidades actuales de la ciencia y la técnica de manera original, creando métodos de trabajos relativamente sencillos de implementar y con gran potencia resolutiva. Cada uno de estos métodos presenta sus peculiaridades en cuanto a sus fortalezas y sus debilidades, pero todos en su conjunto contienen un algoritmo potente con un costo computacional que en ocasiones puede llegar a ser alto pero que aún así superan a otros métodos de optimización más complejos y de algoritmos más engorrosos.

ISBN: 978-959-16-4235-6

Bibliografía

Affenzeller, M.; Wagner, S.; Winkler, S.; Beham, A. (2009). *Genetic Algorithms and Genetic Programming: Modern Concepts and Practical Applications*. Boca Raton, FL (USA): CRC Press. ISBN 978-1-1381-1427-2.

Beruvides, G.; Quiza, R.; Haber, R.E. (2016). "Multi-objective optimization based on an improved cross-entropy method: A case study of a micro-scale manufacturing process". *Information Sciences*, 334-335 pp. 161-173. DOI: 10.1016/j.ins.2015.11.040.

Bonyadi, M.R.; Michalewicz, Z. (2017). "Particle swarm optimization for single objective continuous space problems: A review". *Evolutionary Computation* 25(1): pp. 1-54. DOI: 10.1162/EVCO_r_00180.

Chouhan, S.S.; Kaul, A.; Singh, U.P. (2018). "Soft computing approaches for image segmentation: a survey". *Multimedia Tools and Applications* 77 (21) pp. 28483- 28537. DOI: 10.1007/s11042-018-6005-6.

De Boer, P.T.; Kroese, D.P.; Mannor, S.; Rubinstein, R.Y. (2005). "A Tutorial on the cross-entropy method". *Annals of Operations Research* 134 pp. 19-67. DOI: 10.1007/s10479-005-5724-z.

Haber, R. E., Beruvides, G., Quiza, R.; Hernández, A. (2017). "A simple multi-objective optimization based on the cross-entropy method". *IEEE Access* 5(1) pp. 22272-22281. DOI: 10.1109/ACCESS.2017.2764047.

Haber, R.E.; Del Toro, R.M.; Gajate, A. (2010). "Optimal fuzzy control system using the cross-entropy method. A case study of a drilling process". *Information Sciences*, 180 pp. 2777-2792. DOI: 10.1016/j.ins.2010.03.030.

Haber, R.E.; Haber, R.; Jiménez, A.; Galán, R. (2009). "An optimal fuzzy control system in a network environment based on simulated annealing. An application to a drilling process". *Applied Soft Computing* 9 (3) pp. 889-895. DOI: 10.1016/j.asoc.2008.11.005.

Krzeszowski, T.; Wiktorowicz, K.; Przednowek, K. (2018). "Comparison of selected fuzzy PSO algorithms". In: Fidanova, S. (ed.). *Recent Advances in Computational Optimization Results of the Workshop on Computational Optimization WCO* 2016. Cham (Switzerland): Springer. ISBN: 978-3-319-59861-1.

Mafarja, M.M.; Mirjalili, S. (2017). "Hybrid whale optimization algorithm with simulated annealing for feature selection". *Neurocomputing* 260 pp. 302-312. DOI: 10.1016/j.neucom.2017.04.053.

ISBN: 978-959-16-4235-6

Mirjalili, S. (2019). *Evolutionary Algorithms and Neural Networks*. Cham (Switzerland): Springer. ISBN: 978-3-319-93024-4.

Mirjalili, S.; Gandomi, A.H.; Mirjalili, S.Z.; Saremi, S.; Faris, H.; Mirjalili, S.M. (2017). "Salp swarm algorithm: A bio-inspired optimizer for engineering design problems". *Advances in Engineering Software*, 114 pp. 163-191. DOI: 10.1016/j.advengsoft.2017.07.002.

Mirjalili, S.; Lewis, A. (2016). "The whale optimization algorithm". *Advances in Engineering Software* 95 pp. 51-67. DOI: 10.1016/j.advengsoft.2016.01.008.

Mirjalili, S.; Mirjalili, M.S.; Lewis, A. (2014). "Grey wolf optimizer". *Advances in Engineering Software* 69 pp. 46-61. DOI: 10.1016/j.advengsoft.2013.12.007.

Punia, P.; Kaur, M. (2013). "Various genetic approaches for solving single and multi-objective optimization problems: A review". *International Journal of Advanced Research in Computer Science and Software Engineering* 7 (3) pp. 1014-1020.

Rojas Cruz, J.A.; Pereira, A.G.C. (2013). "The elitist non-homogeneous genetic algorithm: Almost sure convergence". *Statistics and Probability Letters* 83 (10) pp. 2179-2185. DOI: 10.1016/j.spl.2013.05.025.

Saremi, S.; Mirjalili, S.; Lewis, A. (2017). "Grasshopper optimisation algorithm: Theory and application". *Advances in Engineering Software* 105 pp. 30-47. DOI: 10.1016/j.advengsoft.2017.01.004.

Siarry, P. (ed.) (2016). *Metaheuristics*. Cham (Switzerland): Springer. ISBN: 978-3-319-45401-6.

Xing, B.; Gao, W.-J. (2014). *Innovative Computational Intelligence: A rough guide to 134 clever algorithms* (Vol. 62). Cham (Switzerland): Springer.

Yang, X.-S. (2014). *Nature-Inspired Optimization Algorithms*. London (UK): Elsevier. ISBN: 978-0-12-416743-8.

ISBN: 978-959-16-4235-6